Tuning of the Cellular Automaton parameters for the Phase 1 track seeding at the CMS experiment

Yalovetzky, Romina.
FCEyN, UBA, ARGENTINA

rominayal@gmail.com

Supervisor: Dr. Pantaleo, Felice.

Abstract

This report is intended to give an overview of my contributions to the project I was working on, during my Summer Student Internship at CERN in the period of June to August 2017. I worked under the supervision of Dr. Pantaleo, Felice at the CMS collaboration.

0.1. Motivation

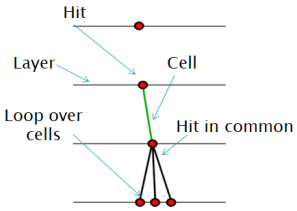
Since the Phase-1 detector has a fourth layer, one more than the original one, the approach of track seeding would be to extend the existing algorithm that have been used until 2016 which implements the tecnique called Cellular Automaton (CA).

The aim of this work is to develop a criteria for the longitudinal plane compatibility for the alignment of 4 hits for any value of the transverse momentum in the range of [0, 10] GeV. The most important part of this is to propuse a method that could be implemeted for any geometry.

The method propused in this work is the tuning of the CA parameters.

0.1.1. Cellular Automaton

Before going deeply into the explanation of this tecnique, in the following graph it can be seen some important terms.



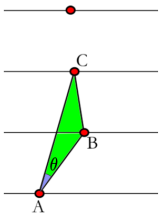
Graph 1: *The black lines represent the layers of the detector. It can be seen that a for a given layer there are multiple possible hits. A cell connects two hits and a cell is connected with another cell which has a hit in common. In the CA a loop is made in order to determinate between which cells the connection should be made.*

It is a local and parallelizable algorithm that works as follows: There is a container class that has a list of all the layers of the detector according to his geometry and all the possible pairings of layer given the configuration. A Cell is the building block of the Hit-Chain Maker using the CA. It is constructed starting from two hits that are compatible with the track seeding criteria. One of the most important parts of this is the Cell connection. This connection is to nd neighbors for each Cell. Two cells are said to be connected if they belong to dierent layer pairs and they have one hit in common. Also these 2 two hits must be aligned along the (ρ, z) plane and compatible with a circular trajectory coming from the beam-spot in the (x,y) plane.

In conclusion, in order to accomplish the seeding task the cell's connection must be made. This work propuse a method for the alignment along the (ρ, z) plane based on the longitudinal plane compatibility to be implemented for any geometry.

0.1.2. Longitudinal plane compatibility

In the longitudinal plane (ρ, z) the criteria to connect two cells is based on the angle shown in the Graph 2.



Graph 2: *Representation of the angle θ which is one of the angles of the triangle of the sides made by the connection between the hits. Extracted from [1]*

The area inside the triangle of the sides made by the connec-

tion between the hits can be written as the following formula:

$$A = z_a(\rho_b - \rho_c) + z_b(\rho_c - \rho_a) + z_c(\rho_a - \rho_b) \quad (1)$$

Extracted from [1]

It can be demostrated that the angle θ can be approximated for smalls values by:

$$tan(\theta) = \frac{2A}{dist^2(A,C)} \approx \theta \quad (2)$$

Extracted from [1]

There is a threshold value for this angle for a given momentum that could be implemented in the seeding criteria. The main purpose of this work is to find this value. This angle is called θ_{cut} and it is an external parameter that depends on the geometry of the detector and also on the transverse momentum (p_t), the initial ρ position (ρ_0), the initial z position (z_0) and the pseudorapidity (η).

The objective is to determinate a criteria that could state which is the θ_{cut} for every triplet and different conditions of $p_t, \rho_0,\ z_0$ and η. Also, it is important that this criteria could work for any geometry. Some analysis of the dependency of this angle with some physical magnitudes was also required and some analysis will be shown.

0.1.3. The method

To begin with, in order to find the θ_{cut} it is needed to calculate the hits in every layer of every possible combinations given a geometry of the detector. The first step to do this is to simulate the trajectory of the particle inside the detector. Then, the following step is to find the intersection between the trajectory and the layers.

Since the analytical approach had shown his limitations to achieve the proposed goal, it have been proposed a numerical method. In the first place, the Runge-Kutta method was considered to be the right one to propagate the trajectory of the particle and then to find the intersection with the layer. Nevertheless, it has shown some complications in order to solve the differential equations of motion. That is why it has been proposed a method that combines a numerical and analytical solution.

It has been taken the solution of the analytical trajectory of a charged particle in a magnetic field:

$$\rho(z) = \sqrt{x(z)^2 + y(z)^2} - \rho_0 \quad (3)$$

$$x(z) = Rcos(\frac{\omega m(z - z_0)}{p_z}) - R \quad (4)$$

$$y(z) = Rsin(\frac{\omega m(z - z_0)}{p_z}) \quad (5)$$

With $p_z = p_t sinh(\eta)$, $\omega = \frac{qB}{m}$ and $R = \frac{p_t}{Bq}$ where B is the magnetic field, q the charge and m the mass.

It was implemented a numerical method to find the intersection between the particle and the layer called the Newton method. It has the characteristic that it can be setted the precision desired: The minimum distance between two z positions taken in order to find the root.

This method is implemented in Python and it has been made for any geometry with the following distribution of layers. It has layers with the shape of a plane and there are two types of them: barrel and forward layers. They are located in a fixed ρ for the first case and a fixed z for the second one. For now one, it is going to be refer to end-cap layers as BLayers and for forward layers as Flayers.

There is a function called `geometry` that given the geometry of the detector it gives as an output 4 vectors. The geometry must be writen in matrices called layers that refer to every combination of 3 layers of the detector. Every matrix has 3 rows of 4 components, each one refer to a single layer as it can be seen in the Fig 11 in the section 3. The 4 components are: ρ position of the layer if it is a BLayer or z position of the layer if it is a FLayer, ρ_{min} if it is a FLayer or z_{min} if it is a BLayer, ρ_{max} if it is a FLayer or z_{max} if it is a BLayer, and a number that shows if it is a BLayer or Flayer: 1 if it is BLayer or 0 if it is FLayer.

This function takes this as an input and gives as an output 4 vectors: rhosArray that has 3 components, the non zero are the ones that refer to the positon of the BLayer, zArray also of 3 components and the non zero refer to the position of the FLayer. rholimArray is made by 6 components, each non zero pair show the lower and upper limit of every BLayer and zlimArray that works as rholimArray but each pair refer to the limits of the FLayer.

With this 4 vectors it is possible to find the solution of the intersections between every layer of the combination and the trajectory of the particle. This step is different depending on the type of layer: Barrel (B) or Forward (F). If it is a BLayer the Newton method is implemented and if it is a FLayer the replacement in the trajectory is executed. It can be seen in Graph 12 in the section 4.

The Newton method has the property that it can settled the precision by adjusting the minimum distance between two following roots the function that it is being computed the roots. It was settled to be $1.5{*}10^{-3}$.

After doing this all the possible intersections between the

trajectory of the particle and each layer of every combination have been computed. Nevertheless, some of these intersections could occur outside the detector. If one hit of the triplet is outside the detector that triplet should be discarded. That is why we have in the code a function that takes every triplet and analyses if every hit of that triplet is inside the detector. This is done in different ways depending if it is a BLayer of a Flayer. It also takes the z limits and the ρ limits from the function geometry. It one hit is outside the detector, that triplet is discarded. This process is repeated for every combination. The way that triplet is discarded is by replacing the intersection (ρ or z in which corresponds) by a fix number (called val) that we are sure that can not be a intersection point. This can be seen in Graph 13 in the section 4.

The θ angle is not calculated for that triplet that it is not inside the detector. In contrast, if every hit of the triplet is in the detector the angle is calculated. This is executed as it is seen in Graph 14 in the section 4.

The challenge in this final step was to develop a method that could discard the triplets that have a hit outside the detector since for doing that we should take into account triplets instead of hits in every of the 3 layers.

In order to find the most accurate method to find θ_{cut} some analysis have been made for the geometry of the tracker of the CMS collaboration.

1. Results and discussion

The current CMS geometry provides a four-hit coverage instead of three-hit as it was in the original detector. In the software used for event reconstruction a list of seeding layers is passed via a file. There are special combinations of layers for this geometry with end-cap pixels (BPix) and forward pixels (FPix).

It is consider these combinations but only the ones of the side of positive z since the behaviour would be the same in the other side. These are the following:

- 1. BPix1 + BPix2 + BPix3
- 2. BPix2 + BPix3 + BPix4
- 3. BPix2 + BPix3 + FPix1+
- 4. BPix1 + BPix2 + FPix1+
- 5. BPix2 + FPix1+ + FPix2+
- 6. BPix1 + FPix1+ + FPix2+
- 7. FPix1+ + FPix2+ + FPix3+

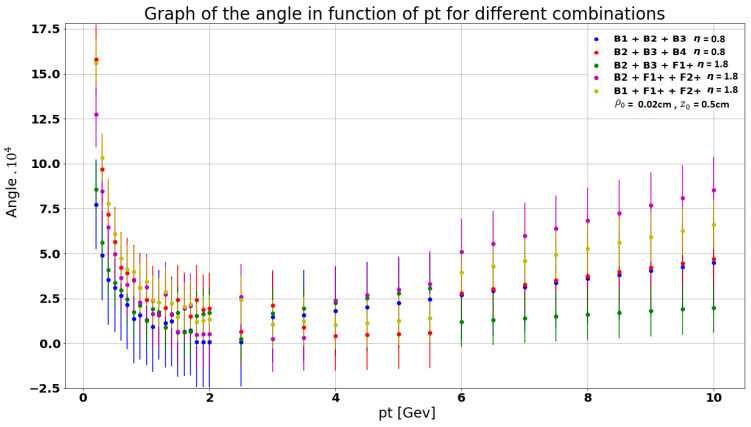
List 1: Layers combinations of the Phase-1 pixel detector.

The following results show the dependency of θ, calculated by the method already explained, with the following parameters: p_t, ρ_0, z_0) and η. It have been consider a range of momentum from 0.2 GeV to 10 GeV, the beam spot of the CMS tracker that is a cylindre of 20cm long in z and a diameter of 6mm and values of η from 0.1 to 3. This analysis is made for the combinations shown in List 1.

The error bar is calculated by doing the error propagation from equation 2. It depends on the hits, the intersection between the trajectory of the particule and the layer, and also on the momentum. In the following graphs the error bar is calculated for any of the combinations of List 1 but taking the bigest value of the position of the hits in the combination and the bigest momentum considered, 10 GeV. As a consequence, the error bar is the same for any of the combinations considered. This error could be improved by calculating it for every hit and momentum. By doing this, the error would decrease for some of the dots.

1.1. The dependancy with pt

Fixing ρ_0 in 0.02cm, z_0 in 0.5cm and also η in 0.8 and 1.8 depending on the combination it can be plotted θ as function of p_t. It is seen that for $\eta = 0{,}8$ there are intersections only between the trajectory of the particle and the layers of combinations 1 and 2 of List 1. In contrast, for $\eta = 1.8$ the intersections ocurr for combinations 3, 5 and 6. It is important to notice the absence of intersections for both values of η for any momentum in the range of 0.2 GeV and 10 GeV for the combinations 4 and 7.



Graph 3: *Graph of the angle in function of pt for different combinations for $\rho_0 = 0.02cm$, $z_0 = 0.5cm$ and $\eta = 0.8$ for some of them and $\eta = 1.8$ for the rest.*

Moreover, Graph 3 shows the dependency between the θ and p_t that was expected. There is an asymptotic behaviour in the vecinity of 0 GeV and then it decreases and from 2 GeV ahead

the behaviour is increasing.

It is seen that for values of p_t less than 2 GeV the overlap between the curves for the combinations is higher than the one for higher values. Even though the overlap decreses, there is still an overlap between some of the combinations. It is stated that there are some discontinuities in this behaviour in an interval centred in 6 GeV.

It must be highlighted that the unit of θ is arbitrary and it is the one used in the algorithm that currently is being used but it is multiply by a factor of 10^4. Those discontinuities are at least in an θ of 2.5 and the combination that sweeps less angles do this from 0 to 7.5.

In order to discuss deeply the relationship between the existance of triplets inside the detector for a given combination and values of momentum in the range of 0.2 GeV and 10 GeV an analysis of the dependancy between the angle and the pseudorapidity must be made.

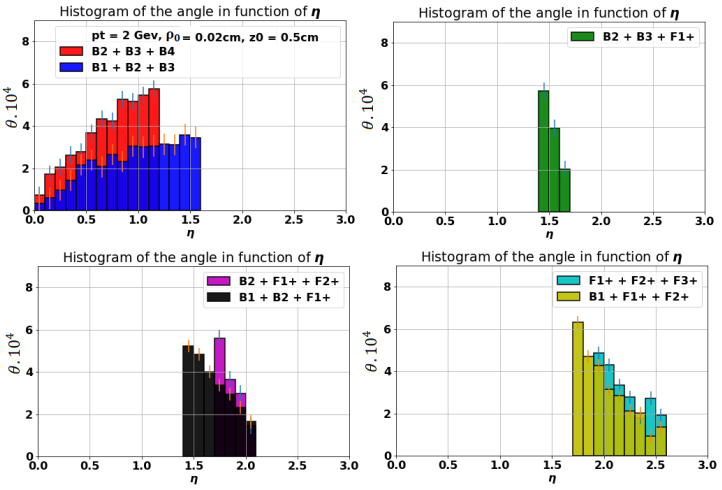
1.2. The dependancy with pseudorapidity

It is plotted the angle in function of η for different combinations for fixed values of $p_t = 2$ GeV, $\rho_0 = 0.02$cm and $z_0 = 0.5$cm.

In the following Graph 4 it can be seen the angle in function of η for a pair of combinations in each graph in histograms. The first conclusion that it can reached is: For these fixed values of p_t, ρ_0 and z_0 there is not an angle for any value of η in the range of 0.1 and 3. As it can be seen, the existance of values of θ for a given η depends on the combination considered.

Whereas for the first two combinations (end-cap layers only) the dependancy between θ and η is increasing, for the rest (involve forward layer) is decreasing. This affirmation is surprising and it was not expected.

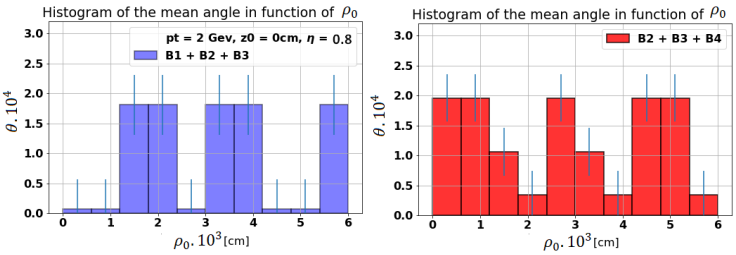
For the two combinations that involve only barrel layers the minimum value of η is around 1 10^4 whereas for the rest of the combinations that involve at least one forward layer the values of the pseudorapidity the minimum θ is higher than 1 10^4. These values of η correspond to an intersection of the trajectory of the particle given the fixed parameters and the three layers that constitute the combination. This behaviour expresses what it was expected. If the combination involves a forward layer the initial pseudorapidy for which there is an angle is higher. Nevertheless, the maximum value of θ is around 6 10^4 for all the combinations even though the corresponding value of η changes depending on the combination.



Graph 4: *Histograms of the angle in function of η for different combinations with $\rho_0 = 0.02cm$, $z_0 = 0.5cm$ and $p_t = 2$ GeV.*

1.3. The dependancy with spatial initial conditions

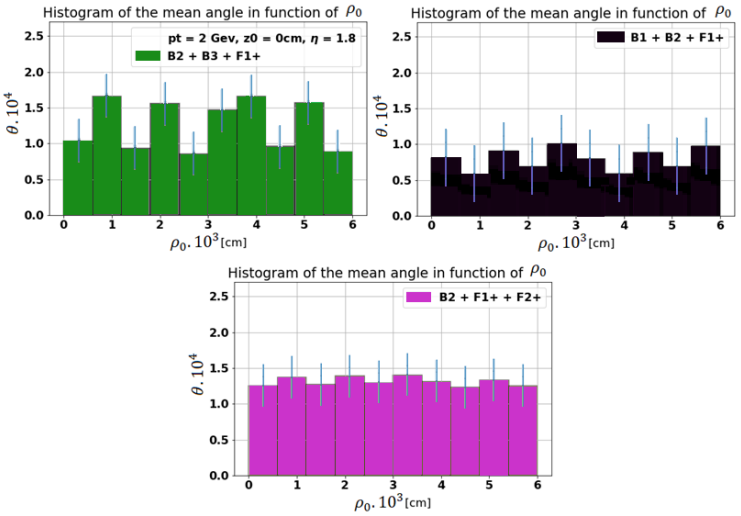
Another dependency of interest is the one with the spatial initial conditions: ρ_0 and z_0. To begin with, it can be seen in the following graph the relationship between θ and ρ_0 taking $z_0 = 0cm$, $pt = 2GeV$ and $\eta = 0.8$.



Graph 5: *Histogram of the angle in function of ρ_0 for different combinations with $z_0 = 0cm$ $pt = 2Gev$ and $\eta = 0.8$.*

As it is seen in Graph 5, not for any combination from List 1 there is a θ for a given value of ρ_0. For some values, given the rest of the conditions fixed in the values said, there are not triplets in some combinations. In this case, only for the first 2 combinations of List 1 there are triplets.

Moreover, it seems that there is not a clear tendency in this behaviour. Although for some different values of ρ_0 the angle takes the same value and for other different values of ρ_0 it takes some different values that is the same among all of them, it is not possible to associate a clear behaviour to relate the angle and ρ_0.

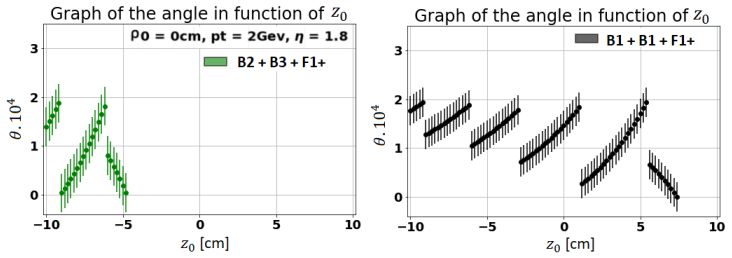


Graph 6: *Histogram of the angle in function of rho$_0$ for different combinations with $z_0 = 0$, $pt = 2Gev$ and $\eta = 1.8$.*

If the value of η is changed to 1.8 there are some values of θ for some of the combinations that previusly did not have triplets. These ones have the property to have a forward layer. Taking into account the error bar, for each combination, the value of θ remains equal for all the values of ρ_0.

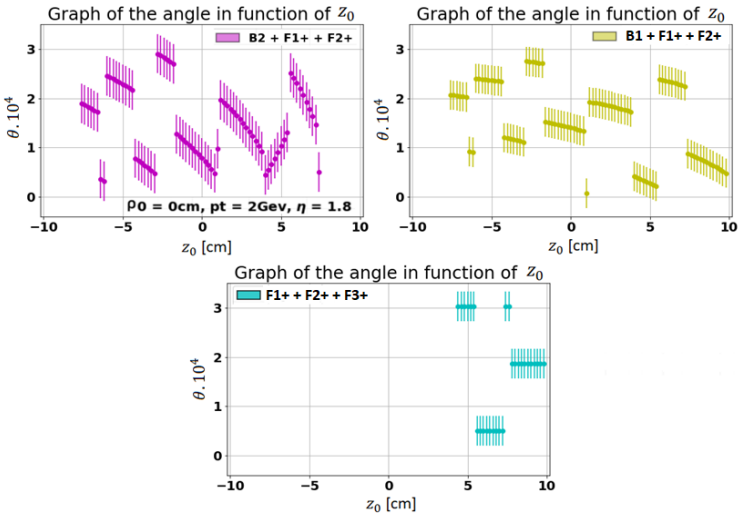
A characteristic that it is present in all the graphs shown is that there is a θ corresponding to each the values of ρ_0. It means that for any position in the ρ axis inside the beam sport there is a triplet for the combinations shown. Nevertheless, there are two combinations that did not have not a single triplet for the fixed values of z_0, p_t and η: 6. B1 + F1+ + F2+ and 7. F1+ + F2+ + F3+. This property is opposite to one saw in Graph 4, the angle in function of the pseudorapidity, in which there were theta for any value of η for some combinations and none for some others given the rest of the parameters fixed.

As regards, the spatial initial condition z_0. It have been taken $\eta = 1.8$, $\rho_0 = 0$cm and $p_t = 2GeV$. In Graph 7 it can be seen the dependancy between the angle and z_0 for some of the combinations in List 1. The ones that are not shown have none triplets inside the detector for these fixed conditions.



Graph 7: *Graph of the angle in function of z_0 for different combination with $\rho_0 = 0cm$, $pt = 2GeV$ and $\eta = 1.8$. z takes negative values since the center of the reference system in z is taken in the middle of the beam spot.*

It can be seen that for all of these combinations there are some values of z_0 for which, given the rest of the parameters fixed, there is not a θ . What is in common among all the graphs plotted is that there is a clear tendency in the behaviour of θ corresponding to a small group of values of $_0$ and there are discontinuities because of that. Moreover, it is seen in Graph 7 and in the first graph in Graph 8 that for only one group of values of $_0$ for each graph separately there is an opposite behavior. While for most of the already mencioned group of values of $_0$ the angle is locally increasing for one group of $_0$ the behavior is decreasing. Also for the combination 7. F1+ + F2+ + F3+ the value of the angle is constant for some group of values of z_0.



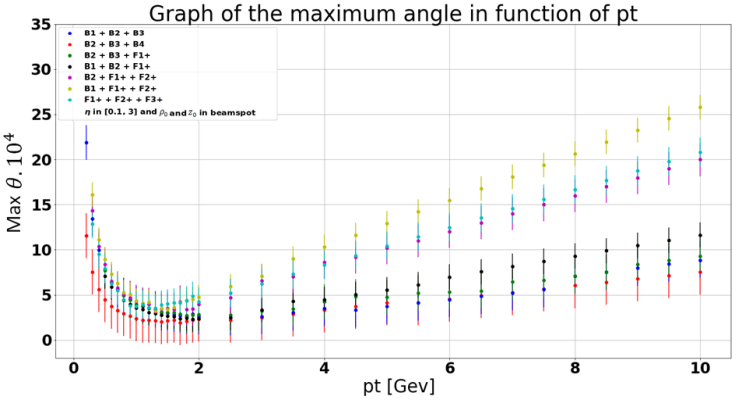
Graph 8: *Graph of the angle in function of z_0 for different combination with $\rho_0 = 0cm$, $pt = 2Gev$ and $\eta = 1.8$. The value of z takes negative values since the center of the reference system in z is taken in the middle of the beam spot.*

To sum up the analysis of the dependancy between the angle and the parameters, it is not possible to find a clear behaviour between the θ and any of the parameters: ρ_0, z_0 and η that applies for any possible combination of the actual detector. Nevertheless, it has been shown that for p_t there is a clear beha-

viour that is common in all the combinations considered as it is seen in Graph 3. As a consequence, the criteria to state which is the θ_{cut} should be based on this relationship between the angle and p_t. This θ_{cut} is for every triplet and different conditions of p_t, ρ_0, z_0 and η for each combination.

1.4. The maximum theta

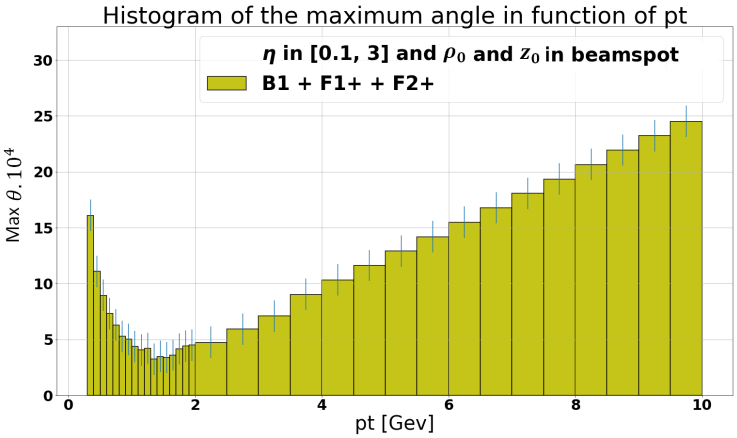
As the θ_{cut} it can be consider the maximum θ for any momentum and combination taking into account all the possible combinations of the values of the spatial conditions and pseudorapidity. In Graph 10 it is plotted the maximum value of θ in function of p_t taking all the possible combinations between the spatial initial conditions (inside the beam spot) and values of pseudorapidity between 0.1 and 3. It is done for all layer combinations from List 1.



Graph 9: *Graph of the θ_{max} in function of p_t for all the combinations of List 1 taking into account all the possible combinations of the values of ρ_0, z_0 and η*

It can be seen that effectivly it can be associated a maximum value of θ to a value of p_t. Moreover, it can be seen the same behaviour saw in Graph 3. For values of p_t less than 2 GeV the angle shows a decreasing behaviour and for values above 2 GeV it is increasing.

In order to see this in a more detailed way it is plotted the θcut in function of p_t for a single combination in Graph 10. It is shown that for small values of transverse momentum the θcut diverges that is why for those values there is not an angle in the graph. This is a consequence of the method that in the future should be improved. Moreover, it is stated that a mixture of an analitical and numerical approach is needed to accomplish the task.



Graph 10: *Graph of the θ_{max} in function of p_t for the combination B1+ + F1+ + F2+ taking into account all the possible combinations of the values of ρ_0, z_0 and η*

2. Conclusion

To sum up this project, the aim was to make the tuning of the CA parameters in order to implement the existing algorithm for any geometry. It has been studied the dependency of the angle θ with the spatial initial conditions, ρ_0 and $_0$, the pseudorapidity η and the transverse momentum p_t. What was observed is that it is not possible to find a clear behaviour of the angle in function of ρ_0, $_0$ or η. But it is possible for the dependency with p_t. That is why the proposed θ_{cut} is the maximum value of this angle for a p_t for a given geometry taking into account all the possible combinations of the parameters. What is remarkable is that the created algorithm takes the geometry as an input making it possible to be used for any detector.

3. References

[1] Pantaleo Felice,"New Track Seeding Techniques for the CMS Experiment".

4. Apendix

```
def solveGeometry(layers):
    for count in range(0,3):
        if layers[count][3] == 1: #its barrel
            rho = layers[count][0]
            namespace['rhosArray%d' % i].append(rho)
            namespace['zsArray%d' % i].append(0)
            zini = layers[count][1]
            zfin = layers[count][2]
            namespace['zlim%d' % i].append(zini)
            namespace['zlim%d' % i].append(zfin)
            namespace['rholim%d' % i].append(0)
            namespace['rholim%d' % i].append(0)
        else:
            namespace['rhosArray%d' % i].append(0)
            z = layers[count][0] #if not its a forward
            namespace['zsArray%d' % i].append(z)
            rhoini = layers[count][1]
            rhofin = layers[count][2]
            namespace['rholim%d' % i].append(rhoini)
            namespace['rholim%d' % i].append(rhofin)
            namespace['zlim%d' % i].append(0)
            namespace['zlim%d' % i].append(0)
    count += 1

layers0 = [0.0295 , 0, 0.2744, 1],[0.068 ,0, 0.2744, 1],[0.109 , 0, 0.2744, 1]  #1
layers1 = [0.068 , 0, 0.2744, 1],[0.109 , 0, 0.2744, 1], [0.160 , 0, 0.2744, 1] #1
layers2 = [0.068 , -0.2744, 0.2744, 1],[0.109 , -0.2744, 0.2744, 1], [0.295, 0.045, 0.161, 0] #2
layers3 = [0.0295 , 0, 0.2744, 1],[0.068 ,0, 0.2744, 1],[0.295, 0.045, 0.161, 0] #3
layers4 = [0.068 ,0, 0.2744, 1],[0.295, 0.045, 0.161, 0], [0.396, 0.045, 0.161, 0] #3
layers5 = [0.0295 ,0, 0.2744, 1],[0.295, 0.045, 0.161, 0],[0.396, 0.045, 0.161, 0] #4
layers6 = [0.295, 0.045, 0.161, 0],[0.396, 0.045, 0.161, 0],[0.516, 0.045, 0.161, 0] #4
```

Graph 11: *A part of the code that shows how the method is implemented for every detector's geometry.*

```
num_step_p_rho = 0.1*const.gev
num_step_eta = 0.1
num_step_rho_0 = 0.0005              # in meters
num_step_z_0 = 0.02                  # in meters
h = 1.5*10**-5                       # the required precision for distances in meters


for i in range (0,comb):
    for j in range (0, np.size(rhosArray0)):  #im going to find the intersection for every layer of the combination
        for p_rho in frange (0.2*const.gev, 10*const.gev, num_step_p_rho):
            for eta in frange (0.1, 3, num_step_eta):
                for rho_0 in frange (-0.003, 0.003, num_step_rho_0):
                    for z_0 in frange (-0.1, 0.1, num_step_z_0):
                        if  namespace['layers%d' % i][j][3] != 0:                        #it means that it is a barrel
                            rhoBL = namespace['rhosArray%d' % i][j]
                            zBL = solve(intersectionB, p_rho, eta, rho_0, z_0, h, rhoBL)
                            zBL_cm = zBL*100
                            namespace['zArray%d' % i][:][j].append(zBL_cm)
                            rhoBL_cm = rhoBL * 100
                            namespace['rhoArray%d' % i][:][j].append(rhoBL_cm)
                        else:                                                            #it means that it is a forward
                            zFL =  namespace['zsArray%d' % i][j]
                            zFL_cm = zFL*100
                            rhoFL = intersectionF(p_rho, eta, rho_0, z_0, zFL)
                            rhoFL_cm = rhoFL*100
                            namespace['rhoArray%d' % i][:][j].append(rhoFL_cm)
                            namespace['zArray%d' % i][:][j].append(zFL_cm)
                        momentum = p_rho/const.gev
                        namespace['momentumArray%d' % i][:][j].append(momentum)
                        z0 = z_0*100
                        namespace['z0Array%d' % i][:][j].append(z0)
                        rho0 = rho_0*100
                        namespace['rho0Array%d' % i][:][j].append(rho0)
                        namespace['etaArray%d' % i][:][j].append(eta)
```

Graph 12: *Part of the code that shows how the intersection between a given layer and the particle's trajectory is calculated: It is diveded in two cases depending on the kind of the layer: BLayer or FLayer.*

```
for i in range (0,7):
    for j in range (0, np.size(rhosArray0)):
        namespace['ZArray%d' % j] = []
        namespace['RhoArray%d' % j] = []
        namespace['MomentumArray%d' % j] = []
        namespace['EtaArray%d' % j] = []
        namespace['Rho0Array%d' % j] = []
        namespace['Z0Array%d' % j] = []
    for k in range (0, np.size(zArray0[:][0])):
        for j in range (0, np.size(rhosArray0)):
            if namespace['layers%d' % i][j][3] != 0:
                if namespace['zArray%d' % i][j][k] < 27.44 and namespace['zArray%d' % i][j][k] > -27.44:
                    namespace['ZArray%d' % j].append(namespace['zArray%d' % i][j][k])
                    namespace['RhoArray%d' % j].append(namespace['rhoArray%d' % i][j][k])
                    namespace['MomentumArray%d' % j].append(namespace['momentumArray%d' % i][j][k])
                    namespace['Z0Arrayy%d' % j].append(namespace['z0Array%d' % i][j][k])
                    namespace['Rho0Arrayy%d' % j].append(namespace['rho0Array%d' % i][j][k])
                    namespace['EtaArray%d' % j].append(namespace['etaArray%d' % i][j][k])
                else:
                    namespace['ZArray%d' % j].append(val)
                    namespace['RhoArray%d' % j].append(val)
                    namespace['MomentumArray%d' % j].append(val)
                    namespace['Z0Arrayy%d' % j].append(val)
                    namespace['Rho0Arrayy%d' % j].append(val)
                    namespace['EtaArray%d' % j].append(val)
            else:
                if namespace['layers%d' % i][j][3] == 0:
                    if namespace['rhoArray%d' % i][j][k] < 16.1 and namespace['rhoArray%d' % i][j][k] > 4.5:
                        namespace['ZArray%d' % j].append(namespace['zArray%d' % i][j][k])
                        namespace['RhoArray%d' % j].append(namespace['rhoArray%d' % i][j][k])
                        namespace['MomentumArray%d' % j].append(namespace['momentumArray%d' % i][j][k])
                        namespace['Z0Array%d' % j].append(namespace['z0Array%d' % i][j][k])
                        namespace['Rho0Array%d' % j].append(namespace['rho0Array%d' % i][j][k])
                        namespace['EtaArray%d' % j].append(namespace['etaArray%d' % i][j][k])
                    else:
                        namespace['ZArray%d' % j].append(val)
                        namespace['RhoArray%d' % j].append(val)
                        namespace['MomentumArray%d' % j].append(val)
                        namespace['Z0Array%d' % j].append(val)
                        namespace['Rho0Array%d' % j].append(val)
                        namespace['EtaArray%d' % j].append(val)
```

Graph 13: *Part of the code that exhibits how some hits are discarded if they are outside the detector.*

```
for k in range (0, np.size(zArray0[:][0])):            # for every triplet
    if ZAsarray0[k] != val and ZAsarray1[k] != val and ZAsarray2[k] != val:
        namespace['theta%d' % i] = theta(MomentumAsarray0[k], ZAsarray0[k], ZAsarray1[k], ZAsarray2[k], RhoAsarray0
        namespace['Theta%d' % i].append(namespace['theta%d' % i])
```

Graph 14: *Part of the code where θ is computed. It is done only for those hits who are inside.*