

# Instructions for the clustering tutorial

## Setup and run instructions Instructions

1. Coding is to be done in **serc19\_ecal\_clustering.C**
2. You need to have CLHEP installed → Take the one inside G4 installation
3. Then source like I have sourced (with your correct paths)

```
source
/usr/local/Caskroom/miniforge/base/envs/geant4env/share/Geant4/geant4make/geant4make.sh
export
DYLD_LIBRARY_PATH=/Users/shilpi2015/geant4/geant4-v11.2.0-install
/lib:$DYLD_LIBRARY_PATH
```

6. Correct the paths in the Makefile
7. Make [Do it everytime you change **serc19\_ecal\_clustering.C** ]
8. To run: **./serc19\_ecal\_clustering inputfiles [where inputfiles is a list of files from geant4 that you want to run on]**

## Hints for writing various clustering algorithms

First step of:

- (a) Adding Gaussian noise to each hit
- (b) Ordering hit in decreasing magnitude of energy

(Above task done by **fill\_digipos\_array** and the result is stored in **digihitpos**)

Now use **digihitpos** which is a **Hep3Vector** whose elements are (energy, theta, phi) of that hit

1. **Clustering algorithm that collects even those hits around the seed are higher in energy compared to the ones in the immediate 3x3 ( i.e. touching the sides) around the seed**
  - a. Loop over all the hits, if a hit crosses the seedThreshold (0.5GeV), start clustering hits around it.
  - b. Consider only those hits around it which are within 3x3 around it OR around the hit which is its 3x3 neighbour.
  - c. The above concept is shown below. Color coding: **RED is highest energy, then orange, then yellow, then green, then blue**
  - d. Although the cells in black boundary are not immediate neighbour or RED still they should get collected as a part of the algorithm

e.

	A	B	C	D	E
1				Yellow	Green
2				Yellow	Orange
3		Green	Yellow	Orange	Yellow
4		Cyan	Red	Green	
5		Orange	Orange	Yellow	

NOTE: In above, Although there is rise in energy if we move from D2 to E2, the algorithm should still collect this hit. This is not a good step because if there are two different photons nearby, this algorithm will make it one photon. So this is not optimum. The next algorithm should treat this malady

2. **Clustering algorithm that is similar to above but DOES NOT** collect those hits around the seed are higher in energy compared to the ones in the immediate 3x3 ( i.e. touching the sides) around the seed. I.e. if we are talking about the above scenario then only the following are clustered

	A	B	C	D	E
1					
2					
3		Green	Yellow	Orange	
4		Cyan	Red	Green	
5		Orange	Orange	Yellow	

3.

#### 4. EM algorithm

- a. First make a list of seeds
- b. If the two seeds lie close to each other (within 3x3 neighbour, i.e. touching the sides) then remove the one with lower energy as seed (i.e. dont consider that one as seed)
- c. Number of seeds == number of particles == number of gaussians
- d. Make a list of 3x3 clusters around it (i.e. same as island algorithm)
- e. Now consider seeds as Gaussian and use the expectation algorithm i.e.

- i. For every cells calculate (consider only those gaussians for that hit which are a part of that gaussian):

```
double arg_the = (cell_theta - mu_theta[ipart])/sigma_theta;  
double arg_phi = (cell_phi - mu_phi[ipart])/sigma_phi;  
  
//contrib[icell][ipart] = amp[ipart]* TMath::Exp(-0.5*arg_the*arg_the - 0.5*arg_phi*arg_phi);  
double tmpcont = amp[ipart]* TMath::Exp(-0.5*arg_the*arg_the - 0.5*arg_phi*arg_phi);
```

- ii. Then Get the fractional contribution from each gaussian, i.e. fractional = Contribution from that Gaussian to the energy/(Sum over all the Gaussians)
- f. Now do the maximization part:
    - i. Loop over all the gaussians.
    - ii. For each gaussian, recalculate its amp (i.e. its amplitude) and its position (i.e. mu\_theta and mu\_phi) in the above