DELPHES
Status and Plans

Michele Selvaggi

CERN

ECFA 02/02/2021

Detector Simulation

- Full simulation (GEANT):
  - **simulates** all **particle-detector interaction** (e.m/hadron showers, nuclear interaction, brem, conversions)

10^2-10^3 s/ev

- Experiment Fast Simulation (ATLAS, CMS ..)
  - **simplify** geometry, smear at the level of detector hits, frozen showers

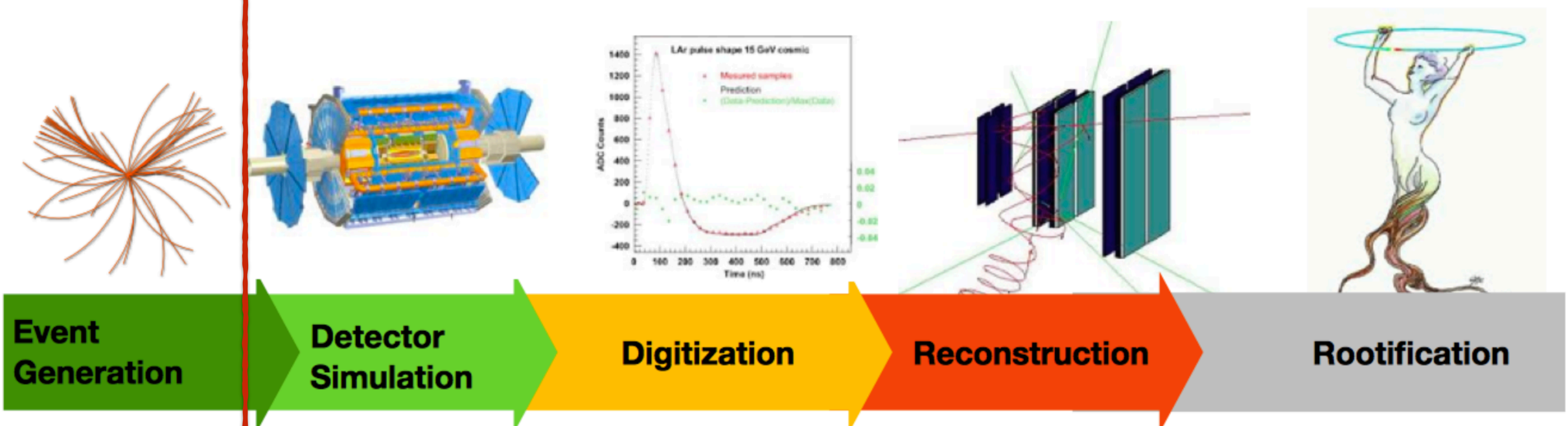
10-10^2 s/ev

- Parametric simulation (**Delphes**, PGS):
  - **parameterise detector response** at the particle level(efficiency, resolution on tracks, calorimeter objects)
  - reconstruct **complex objects** and observables(use particle-flow, jets, missing ET, pile-up ..)

10^{-2} - 10^{-1} s/ev

- Ultra Fast (ATOM, TurboSim):
  - from parton to detector object (smearing/lookup tables)

MonteCarlo EvGen



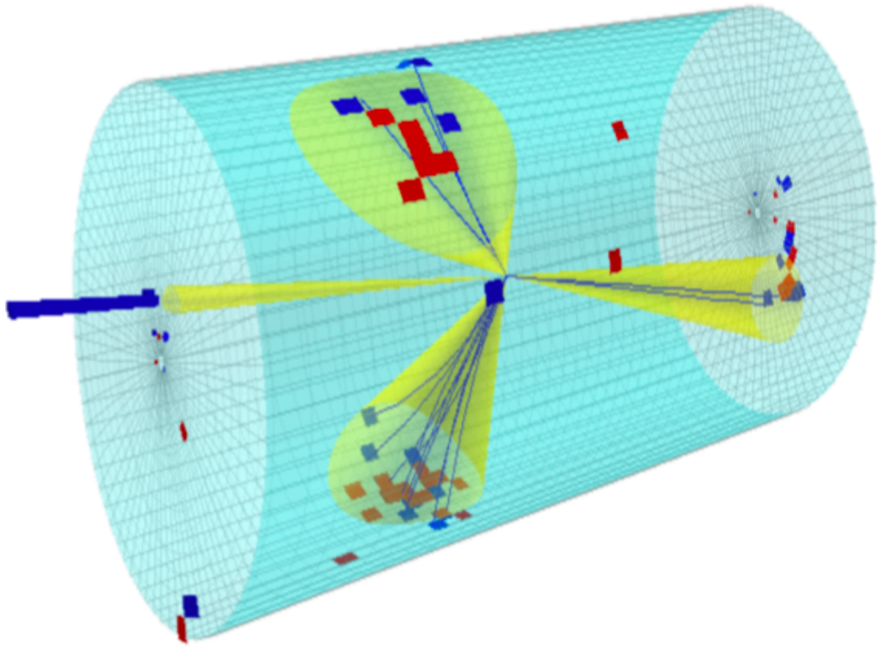
Delphes FastSim

courtesy of Andreas Salzburger

Delphes in a nutshell

- **Delphes** is a modular framework that simulates the response of a **multipurpos detector** in a parameterised fashion

- Includes:
  - pile-up
  - charged particle propagation in B field
  - EM/Had calorimeters
  - particle-flow



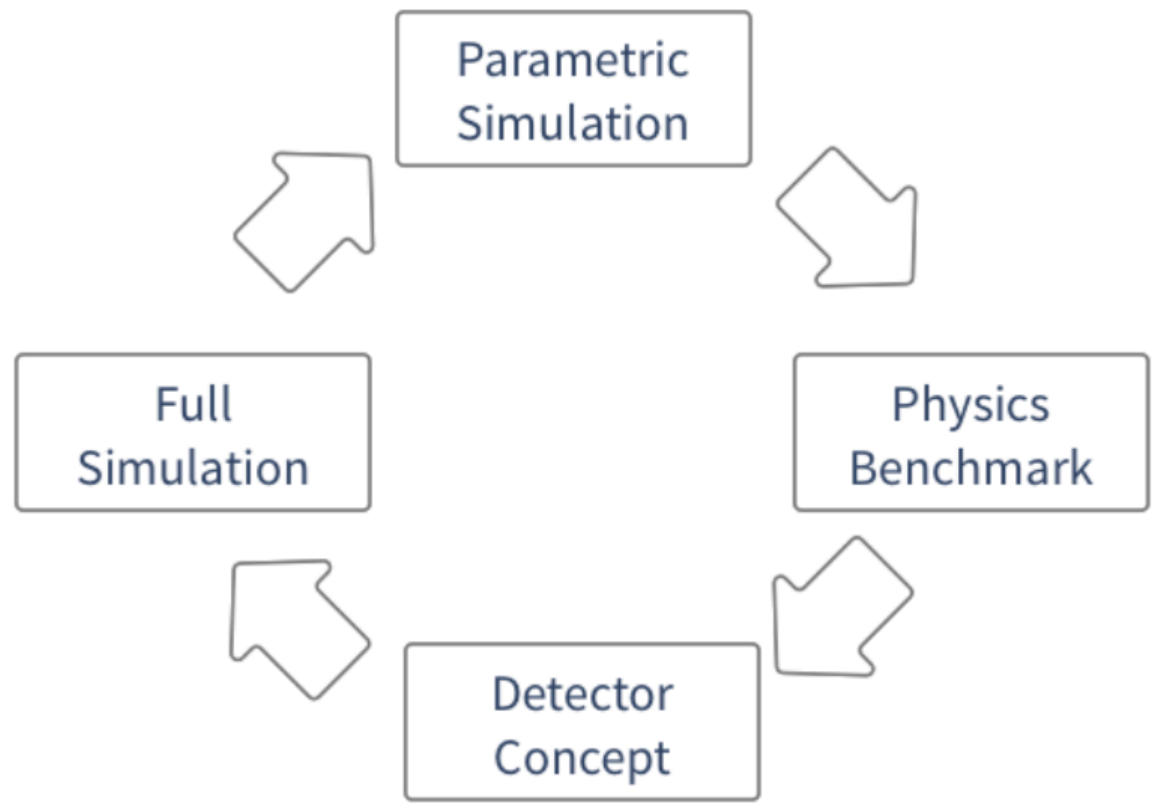
- Provides:
  - leptons, photons, neutral hadrons
  - jets, missing energy
  - heavy flavour tagging

- designed to deal with hadronic environment
- well-suited also for e+e- studies
- detector cards for: CMS (current/PhaseII) - ATLAS - LHCb - FCC-hh - ILD - CEPC - FCCee (IDEA/CLD)

Introductory remarks

Why fast parametric detector simulation?

- Easily scan detector parameters
- Reverse engineer detector that maximises performance
- Preliminary sensitivity studies for key physics benchmarks



→ paradigm adopted in the context of FCC studies

Tracking

Delphes fast simulation

pattern recognition
track fitting

Full Simulation

-most accurate
-least flexible
-does not allow for change of geometry

parameterisation

(d_0, d_z, p, ctg θ, φ)

Delphes

TkLayout

-standalone tool
-predicts tracking performance for a given geometry
-allows for quick turnaroud
- no further implementation needed in Delphes
-requires intermediate step

FastTrackCovariance

-(for now) standalone tool
-can be plugged into Delphes
-predicts performance given geometry
-allows for quick turnaroud
-not implemented in Delphes yet

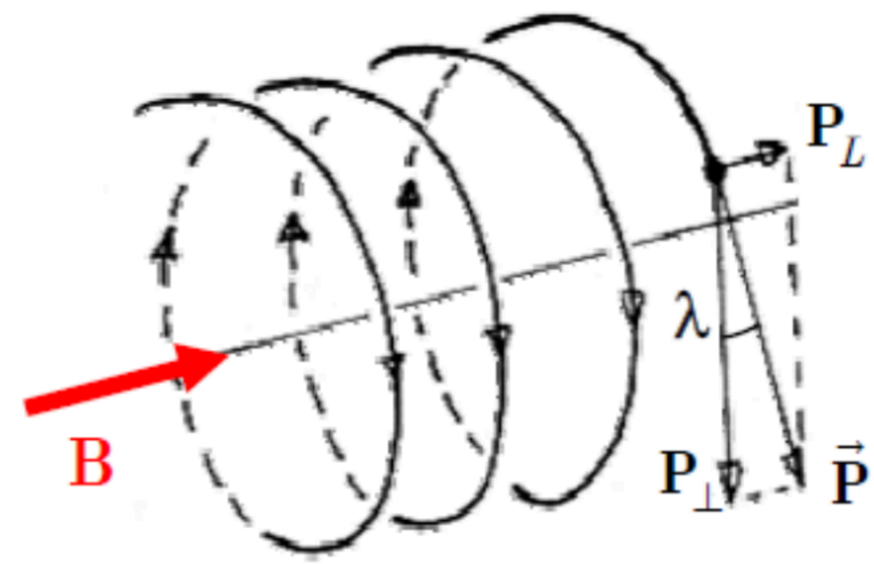
Charged Particle parameterisation

- Charged and neutral particles are propagated in B field until they reach calorimeters

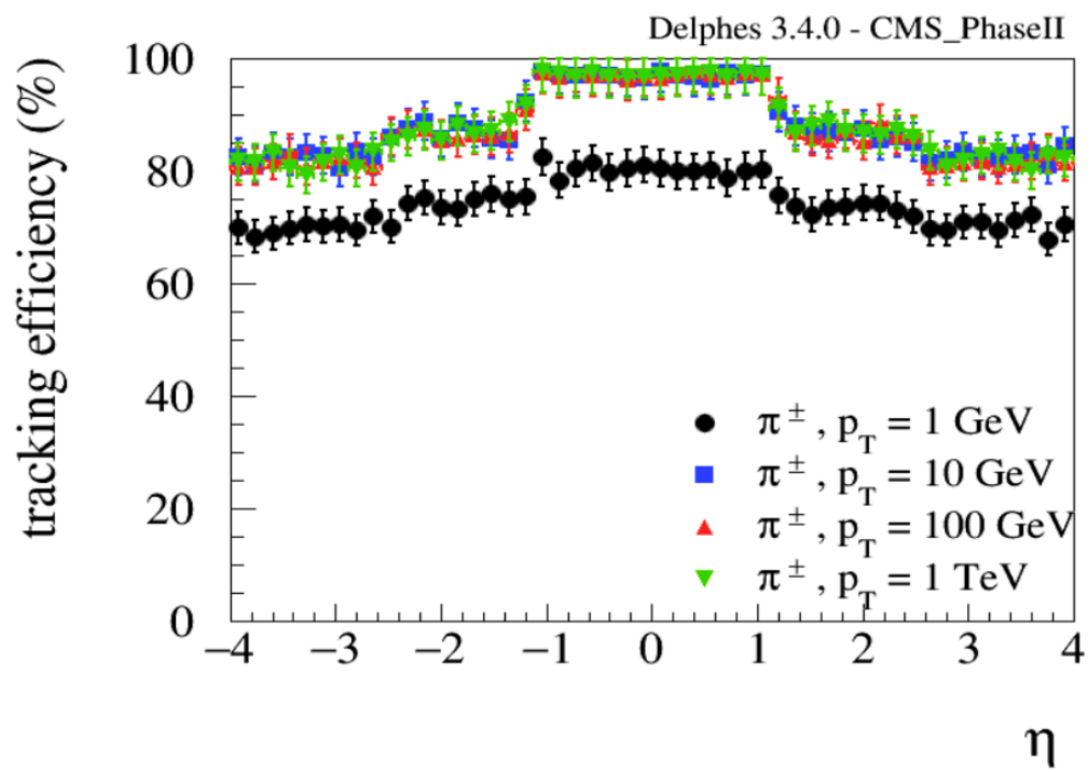
- Propagation parameters:
  - magnetic field B
  - Radius and half-length (R_{max}, z_{max})

- **Efficiency** and **resolution** depends on:
  - particle ID (electron, muon or charged hadron)
  - particle 4-momentum

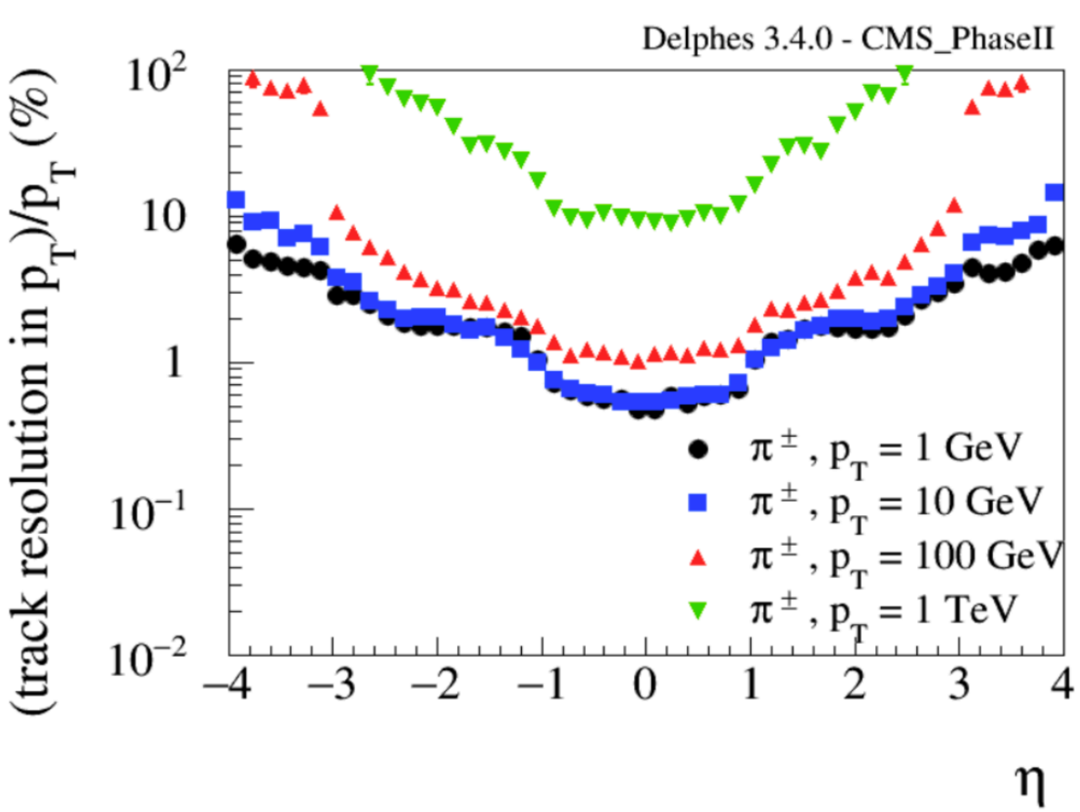
- **TrackSmearing** module allows for **diagonal smearing of (d_0, d_z, p, ctg θ, φ)**



Tracking parameterisation

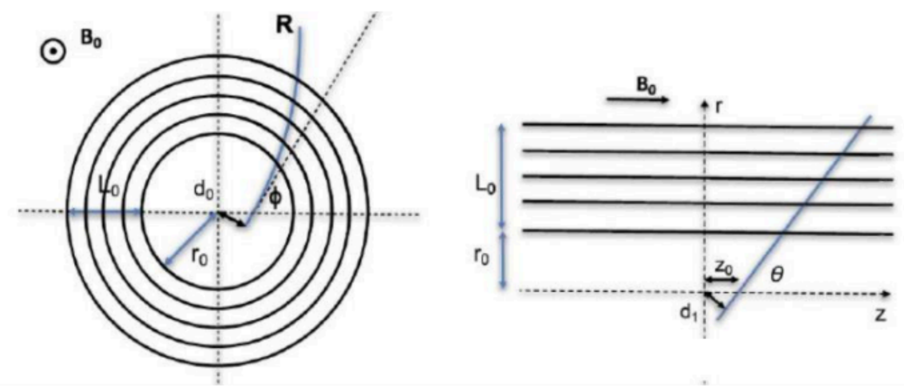


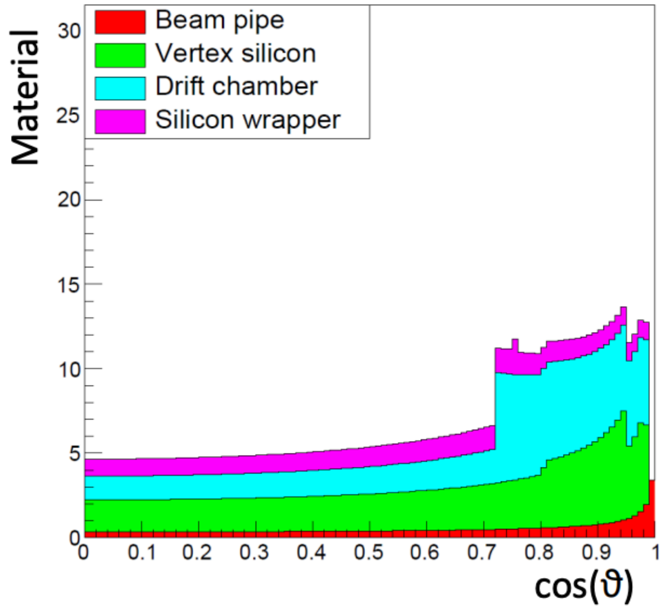

```
####################################
# Charged hadron tracking efficiency
####################################

module Efficiency ChargedHadronTrackingEfficiency {
  ## particles after propagation
  set InputArray  ParticlePropagator/chargedHadrons
  set OutputArray chargedHadrons
  # tracking efficiency formula for charged hadrons
  set EfficiencyFormula {
      (pt <= 0.2) * (0.00) + \
          (abs(eta) <= 1.2) * (pt > 0.2 && pt <= 1.0) * (pt * 0.96) + \
          (abs(eta) <= 1.2) * (pt > 1.0) * (0.97) + \
          (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 0.2 && pt <= 1.0) * (pt*0.85) + \
          (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 1.0) * (0.87) + \
          (abs(eta) > 2.5 && abs(eta) <= 4.0) * (pt > 0.2 && pt <= 1.0) * (pt*0.8) + \
          (abs(eta) > 2.5 && abs(eta) <= 4.0) * (pt > 1.0) * (0.82) + \
          (abs(eta) > 4.0) * (0.00)
  }
}
```




```
set ResolutionFormula {    (abs(eta) >= 0.0000 && abs(eta) < 0.2000) * (pt >= 0.0000 && pt < 1.0000) * (0.00457888) + \
   (abs(eta) >= 0.0000 && abs(eta) < 0.2000) * (pt >= 1.0000 && pt < 10.0000) * (0.004579 + (pt-1.000000)* 0.000045) + \
   (abs(eta) >= 0.0000 && abs(eta) < 0.2000) * (pt >= 10.0000 && pt < 100.0000) * (0.004983 + (pt-10.000000)* 0.000047) + \
   (abs(eta) >= 0.0000 && abs(eta) < 0.2000) * (pt >= 100.0000) * (0.009244*pt/100.000000) + \
   (abs(eta) >= 0.2000 && abs(eta) < 0.4000) * (pt >= 0.0000 && pt < 1.0000) * (0.00505011) + \
   (abs(eta) >= 0.2000 && abs(eta) < 0.4000) * (pt >= 1.0000 && pt < 10.0000) * (0.005050 + (pt-1.000000)* 0.000033) + \
   (abs(eta) >= 0.2000 && abs(eta) < 0.4000) * (pt >= 10.0000 && pt < 100.0000) * (0.005343 + (pt-10.000000)* 0.000043) + \
   (abs(eta) >= 0.2000 && abs(eta) < 0.4000) * (pt >= 100.0000) * (0.009172*pt/100.000000) + \
   (abs(eta) >= 0.4000 && abs(eta) < 0.6000) * (pt >= 0.0000 && pt < 1.0000) * (0.00510573) + \
   (abs(eta) >= 0.4000 && abs(eta) < 0.6000) * (pt >= 1.0000 && pt < 10.0000) * (0.005106 + (pt-1.000000)* 0.000023) + \
   (abs(eta) >= 0.4000 && abs(eta) < 0.6000) * (pt >= 10.0000 && pt < 100.0000) * (0.005317 + (pt-10.000000)* 0.000042) + \
   (abs(eta) >= 0.4000 && abs(eta) < 0.6000) * (pt >= 100.0000) * (0.009077*pt/100.000000) + \
   (abs(eta) >= 0.6000 && abs(eta) < 0.8000) * (pt >= 0.0000 && pt < 1.0000) * (0.00578020) + \
   (abs(eta) >= 0.6000 && abs(eta) < 0.8000) * (pt >= 1.0000 && pt < 10.0000) * (0.005780 + (pt-1.000000)* -0.000000) + \
   (abs(eta) >= 0.6000 && abs(eta) < 0.8000) * (pt >= 10.0000 && pt < 100.0000) * (0.005779 + (pt-10.000000)* 0.000038) + \
   (abs(eta) >= 0.6000 && abs(eta) < 0.8000) * (pt >= 100.0000) * (0.009177*pt/100.000000) + \
   (abs(eta) >= 0.8000 && abs(eta) < 1.0000) * (pt >= 0.0000 && pt < 1.0000) * (0.00728723) + \
   (abs(eta) >= 0.8000 && abs(eta) < 1.0000) * (pt >= 1.0000 && pt < 10.0000) * (0.007287 + (pt-1.000000)* -0.000031) + \
   (abs(eta) >= 0.8000 && abs(eta) < 1.0000) * (pt >= 10.0000 && pt < 100.0000) * (0.007011 + (pt-10.000000)* 0.000038) + \
   (abs(eta) >= 0.8000 && abs(eta) < 1.0000) * (pt >= 100.0000) * (0.010429*pt/100.000000) + \
   (abs(eta) >= 1.0000 && abs(eta) < 1.2000) * (pt >= 0.0000 && pt < 1.0000) * (0.01045117) + \
   (abs(eta) >= 1.0000 && abs(eta) < 1.2000) * (pt >= 1.0000 && pt < 10.0000) * (0.010451 + (pt-1.000000)* -0.000051) + \
   (abs(eta) >= 1.0000 && abs(eta) < 1.2000) * (pt >= 10.0000 && pt < 100.0000) * (0.009989 + (pt-10.000000)* 0.000043) + \
```

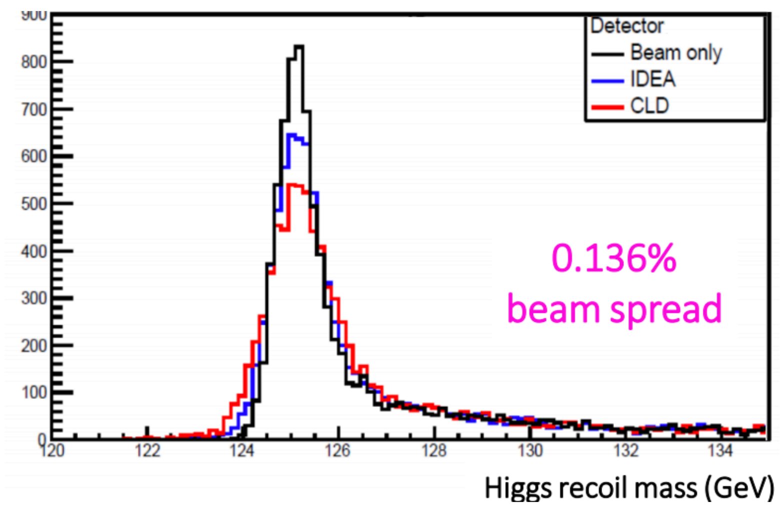
Fast Tracking Simulation





Track Smearing

- Simple tracker geometry implementation, **including material**
- Computes **full covariance matrix** (in present Delphes we have "diagonal" smearing in the 5 tracking parameters)
- Can be used for studying impact of material and realistic **HF tagging** simulation

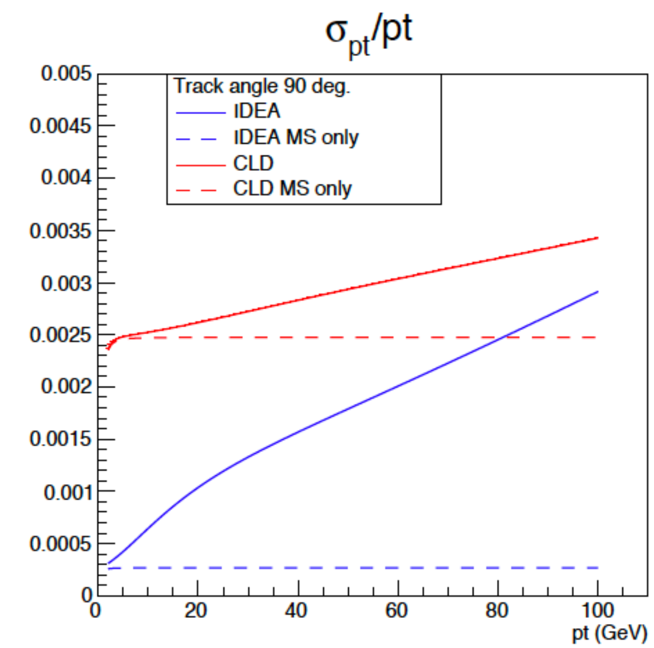


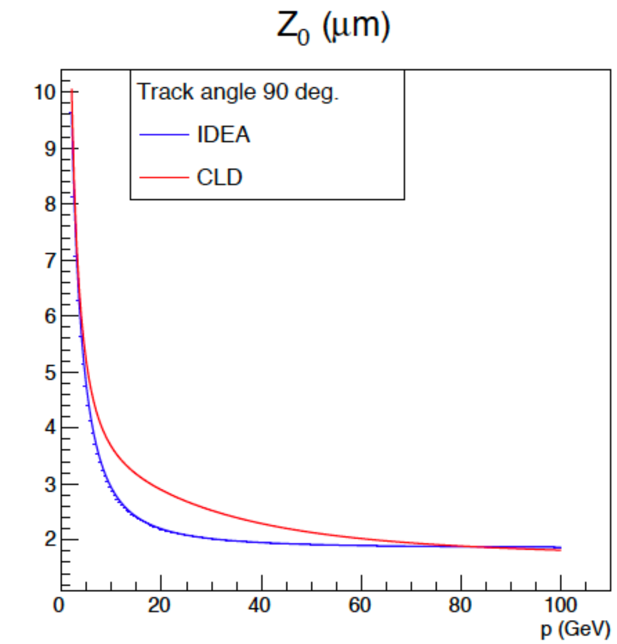
F. Bedeschi

FCC-ee TrackCovariance

DELPHES fast simulation

```
#######################################
# Smearing for charged tracks
#######################################

module TrackCovariance TrackSmearing {

    set InputArray TrackMergerPre/tracks
    set OutputArray tracks

    ## minimum number of hits to accept a track
    set NMinHits 6

    ## magnetic field
    set Bz $B

    ## uses https://raw.githubusercontent.com/selvaggi/FastTrackCovariance/master/GeoIDEA_BASE.txt
    set DetectorGeometry {

  # barrel  name      zmin    zmax   r        w (m)      X0        n_meas   th_up (rad) th_down (rad)    reso_up (m)   reso_down (m)  flag

  1        PIPE      -100    100    0.015    0.001655  0.2805    0        0          0                0             0              0
  1        VTXLOW    -0.12   0.12   0.017    0.00028   0.0937    2        0          1.5708           3e-006        3e-006         1
  1        VTXLOW    -0.16   0.16   0.023    0.00028   0.0937    2        0          1.5708           3e-006        3e-006         1
  1        VTXLOW    -0.16   0.16   0.031    0.00028   0.0937    2        0          1.5708           3e-006        3e-006         1
  1        VTXHIGH   -1      1      0.32     0.00047   0.0937    2        0          1.5708           7e-006        7e-006         1
  1        VTXHIGH   -1.05   1.05   0.34     0.00047   0.0937    2        0          1.5708           7e-006        7e-006         1
```





TrackCovariance module

- **Requires:**
  - Geometry input
  - Magnetic field

F. Bedeschi

Identification/ Fakes

- (Mis-)Identification maps can be defined both:
  - at the particle level (IdentificationMap)
  - at the jet level (JetFakeParticle)

```
# --- pions ---

add EfficiencyFormula {211} {211} {       (eta <= 2.0)                                    * (0.00) +
                                          (eta > 2.0  && eta <= 5.0) *       (pt < 0.8) * (0.00) +
                                          (eta > 2.0  && eta <= 5.0) *       (pt >= 0.8)* (0.95) +
                                          (eta > 5.0)                                     * (0.00)}

add EfficiencyFormula {211} {-13} {       (eta <= 2.0)                                    * (0.00) +
                                          (eta > 2.0  && eta <= 5.0) *       (pt < 0.8) * (0.00) +
                                          (eta > 2.0  && eta <= 5.0) *       (pt >= 0.8)* (0. 05 +
                                          (eta > 5.0)                                     * (0.00)}
```

id

fake

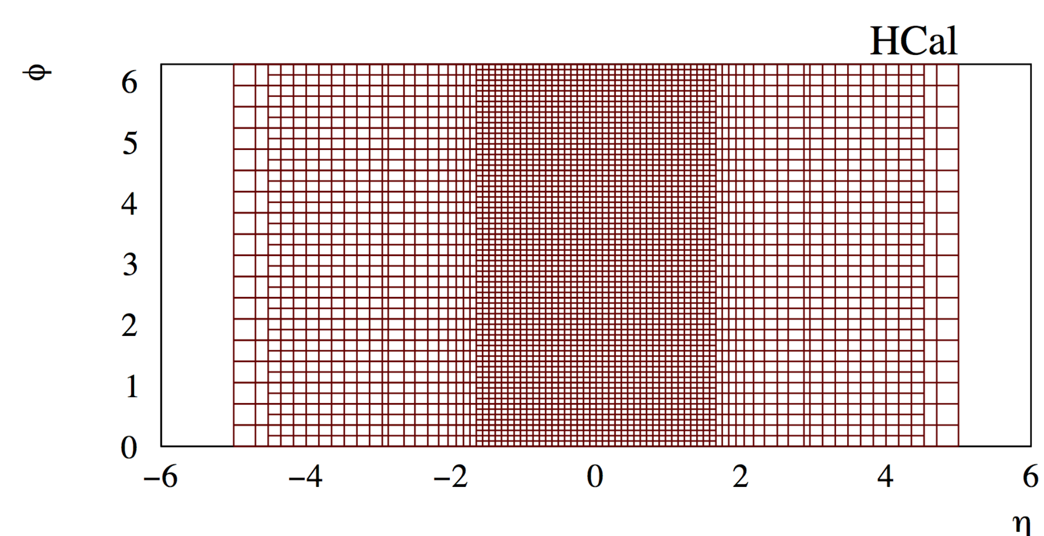
Calorimetry

- ECAL/HCAL **segmentation** specified in (η,ϕ) coordinates
- Particles that reach calorimeters **deposits fixed fraction of energy** in f_{EM} (f_{HAD}) in ECAL(HCAL)

particles	f_{EM}	f_{HAD}
$e\ \gamma\ \pi^0$	1	0
Long-lived neutral hadrons (K^0_S, Λ^0)	0.3	0.7
$\nu\ \mu$	0	0
others	0	1

- Particle **energy** and **position** is **smeared** according to the calorimeter it reaches

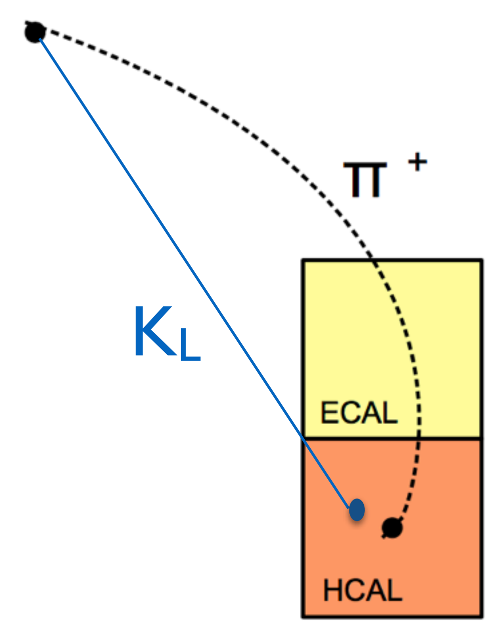
$$\left(\frac{\sigma}{E}\right)^2 = \left(\frac{S(\eta)}{\sqrt{E}}\right)^2 + \left(\frac{N(\eta)}{E}\right)^2 + C(\eta)^2$$



Particle-Flow

- Given **charged track** hitting **calorimeter cell**:
  - is deposit more compatible with **charged only** or **charged + neutral** hypothesis?
  - how to **assign momenta** to resulting components?

- We have two measurements (E_{trk}, σ_{trk}) and (E_{calo}, σ_{calo})
- Define $E_{Neutral} = E_{calo} - E_{trk}$



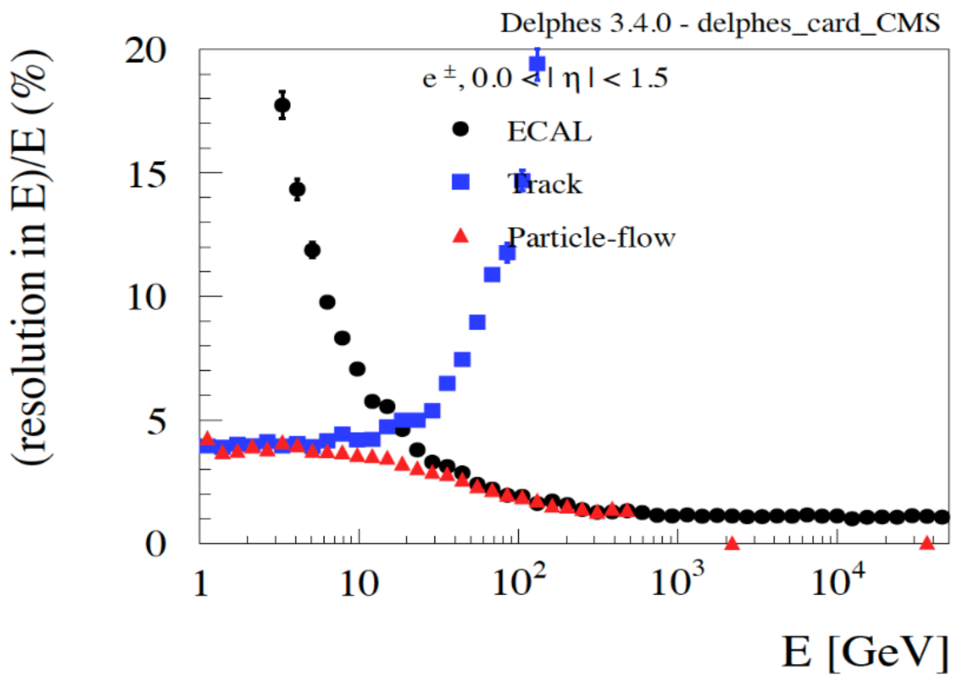
Algorithm:

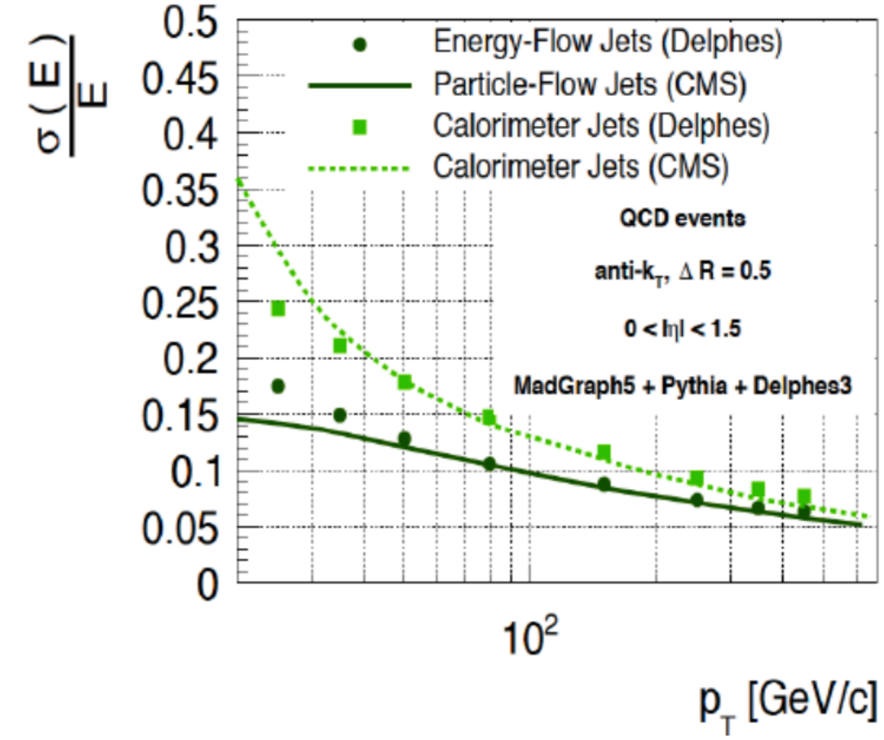
- If $E_{neutral}/\sqrt{(\sigma_{calo}^2 + \sigma_{trk}^2)} > S$:
  → create PF-neutral particle + PF-track

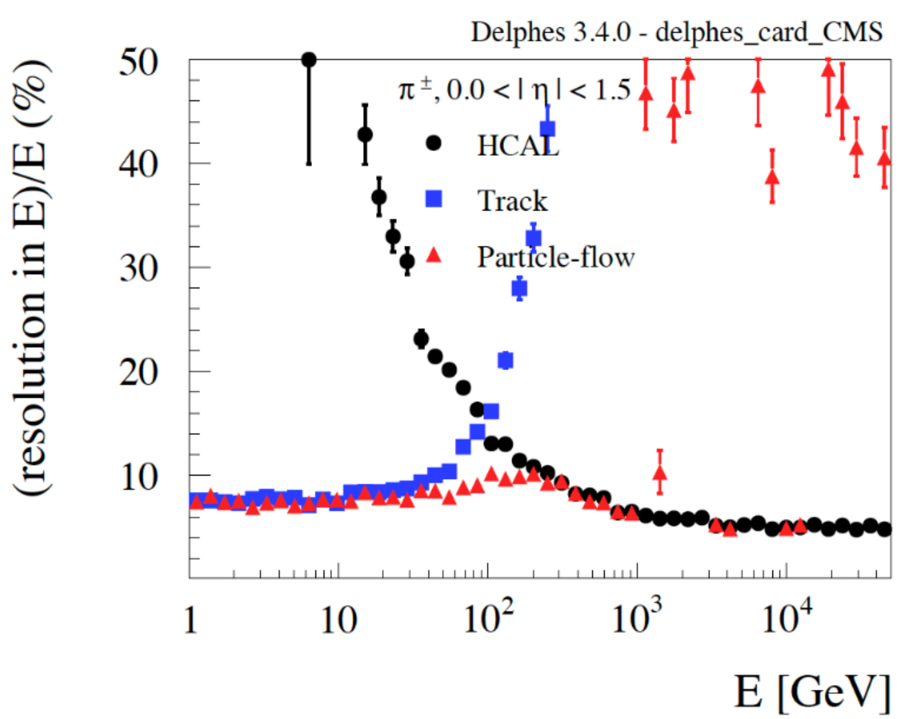
- Else:
  create PF-track and **rescale momentum** by combined calo+trk measurement

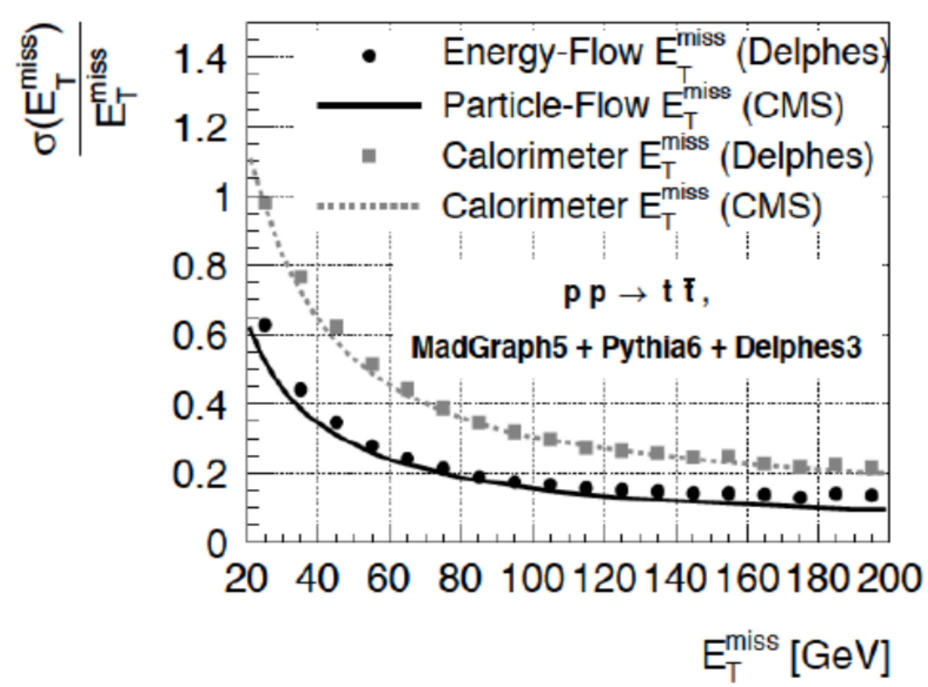
  - EM (had) deposit 100% in ECAL (HCAL)
  - No propagation in calorimeters
  - No clustering (topological) clustering, exploiting pre-defined grid

Particle-Flow







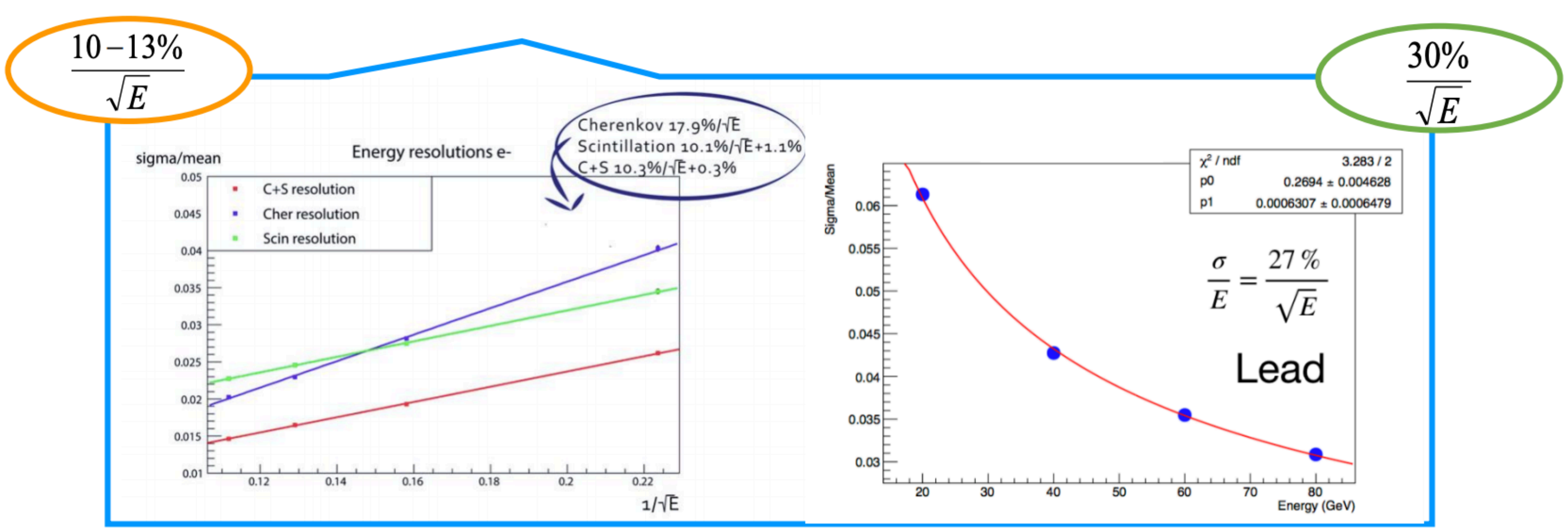


IDEA - (DR) Calorimetry

```
#############
# Calorimeter
#############
module DualReadoutCalorimeter Calorimeter {
  set ParticleInputArray ParticlePropagator/stableParticles
  set TrackInputArray TrackMerger/tracks
```

```
# Lists of the edges of each tower in eta and phi;
# each list starts with the lower edge of the first tower;
# the list ends with the higher edged of the last tower.
# Barrel:  deta=0.02 towers up to |eta| <= 0.88 ( up to 45°)
# Endcaps: deta=0.02 towers up to |eta| <= 3.0 (8.6° = 100 mrad)
# Cell size: about 6 cm x 6 cm
```

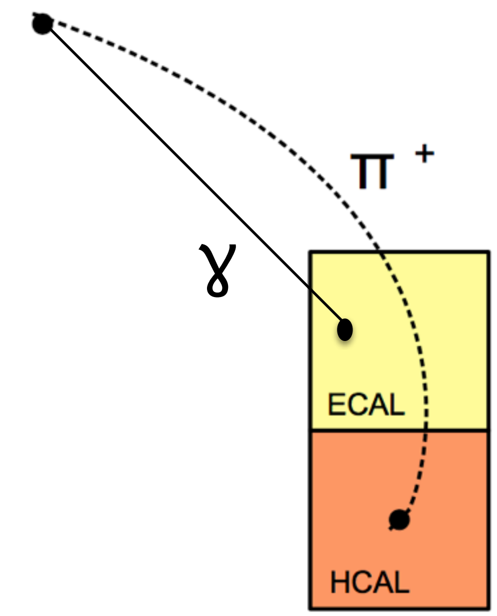
- If EM > 0 and E_{had} =0 → σ(EM)
  - e.g. γ
- If E_{had} > 0 → σ(had)
  - e.g. π^+ or (γ,π^+)



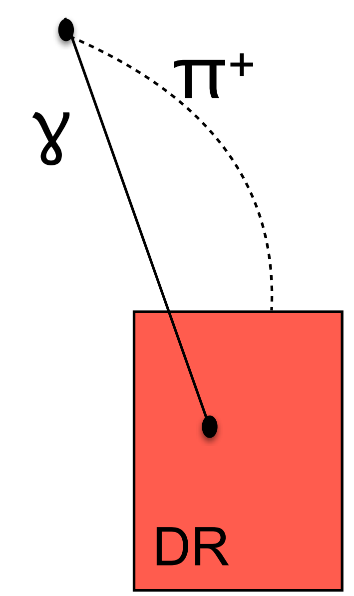
E. Fontanesi
L. Pezzotti
M. Antonello

Dual Readout Particle-Flow

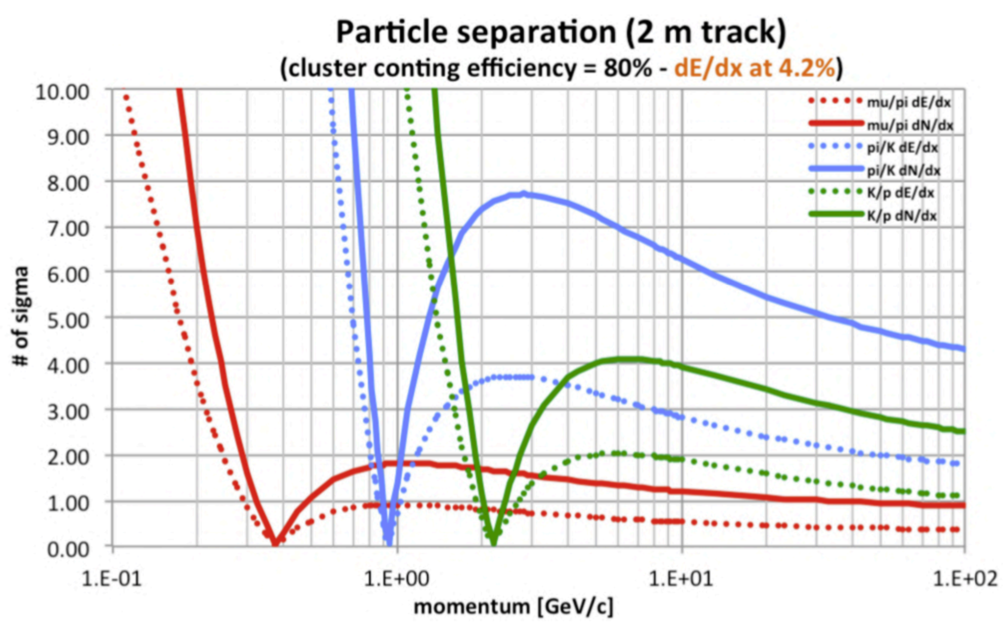
- Given **charged track** hitting **calorimeter cell**:
  - is deposit more compatible with **charged only** or **charged + neutral** hypothesis?
  - how to **assign momenta** to resulting components?
  - If **charged + neutral** how to associate particle ID to **charged** and **neutral** components, e.g **(γ,π^+)** or **(e+,K_L)** ?



- **DualReadoutCalorimeter** module in Delphes assumes we can always disentangle these two cases
  - Probably ok at FCC-ee since probability of overlap not so large (except for taus?)
  - Impact of granularity on performance was studied extensively by Elisa Fontanesi (see here)
  - See next talk Franco Bedeschi



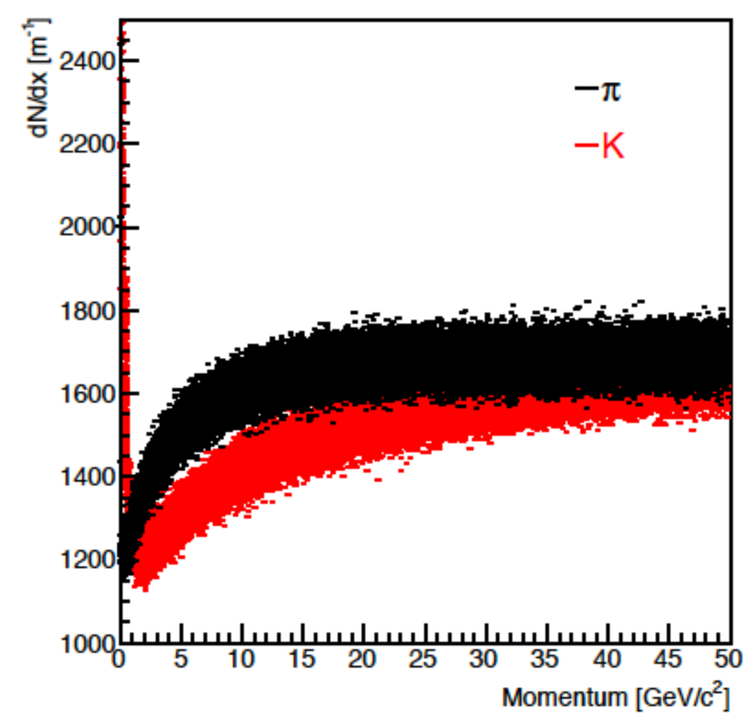
PID: ClusterCounting




```
###################
# Cluster Counting
###################

module ClusterCounting ClusterCounting {

  add InputArray TrackSmearing/tracks
  set OutputArray tracks

  set Bz $B

  ## check that these are consistent with DCHCANI/DCHNANO parameters in TrackCovariance module
  set Rmin $DCHRMIN
  set Rmax $DCHRMAX
  set Zmin $DCHZMIN
  set Zmax $DCHZMAX

  # gas mix option:
  # 0:  Helium 90% - Isobutane 10%
  # 1:  Helium 100%
  # 2:  Argon 50% - Ethane 50%
  # 3:  Argon 100%

  set GasOption 0

}
```

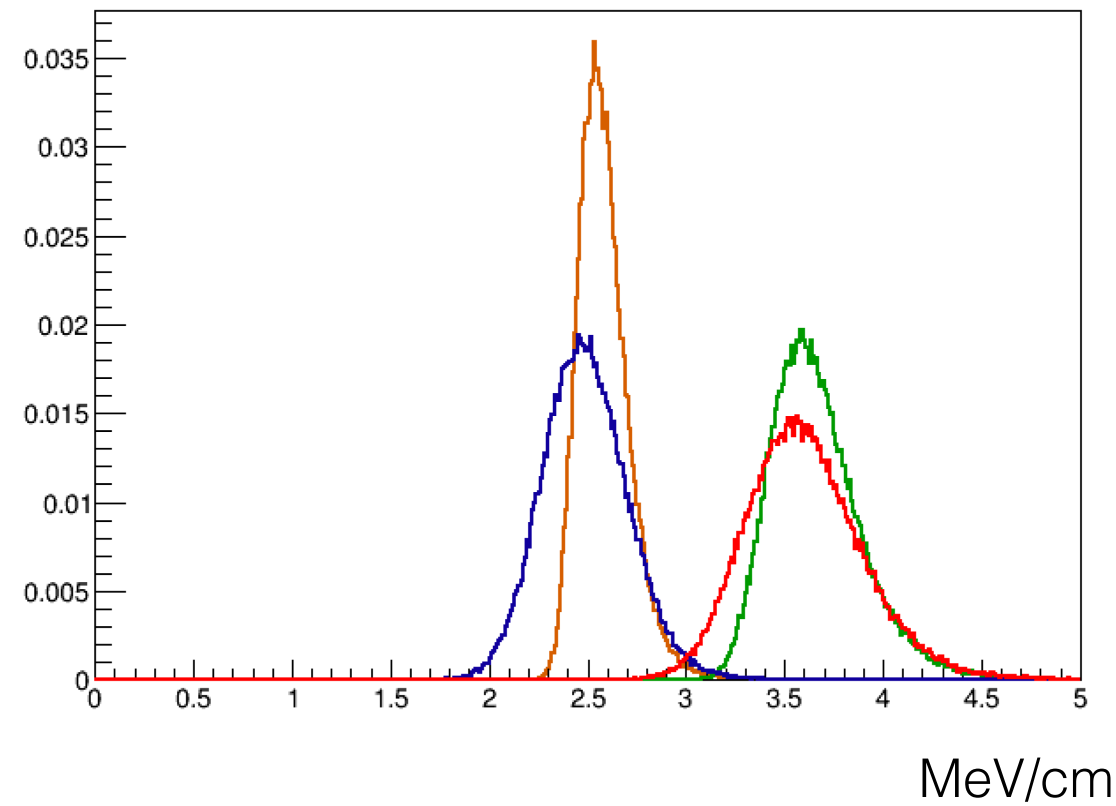
PID:

- Dndx method: implemented in same library that computes track parameters
- called by ClusterCounting module



PID: dEdX

```
module EnergyLoss EnergyLoss {
  add InputArray ChargedHadronMomentumSmearing/chargedHadrons
  add InputArray ElectronMomentumSmearing/electrons
  add InputArray MuonMomentumSmearing/muons


  # absolute resolution per measurement (normalized in MeV/cm)
  # CMS pixel detector performance is reproduceable with r = 0.4
  # dedicated dEdX detector can achieve r = 0.0 or below (i.e better than Landau)

  #set Resolution 0.4
  set Resolution 0.2

  # fraction of measurements to ignore when computing truncated mean
  # suggested range [0.4-0.6]

  set TruncatedMeanFraction 0.5

  # detector properties (active fraction = nhits*thickness/L)
  set Thickness 100E-6
  set ActiveFraction 0.0006666

  # Silicon properties, for other materials:
  # cf. http://pdg.lbl.gov/2014/AtomicNuclearProperties/properties8.dat

  set Z 14.
  set A 28.0855
  set rho 2.329

  # material polarisation correction parameters
  set a 0.1492
  set m 3.2546
  set x0 0.2015
  set x1 2.8716
  set I 173.0
  set c0 4.4355
```



10 measurements x100 um silicon

beta = 1 (Landau)

beta = 0.75 (Landau)

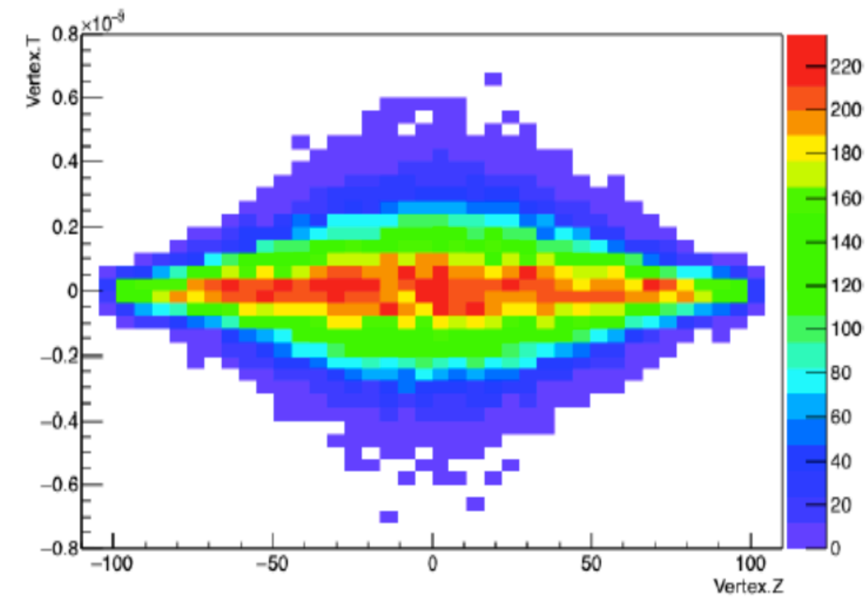
beta = 1 (LanGauss)

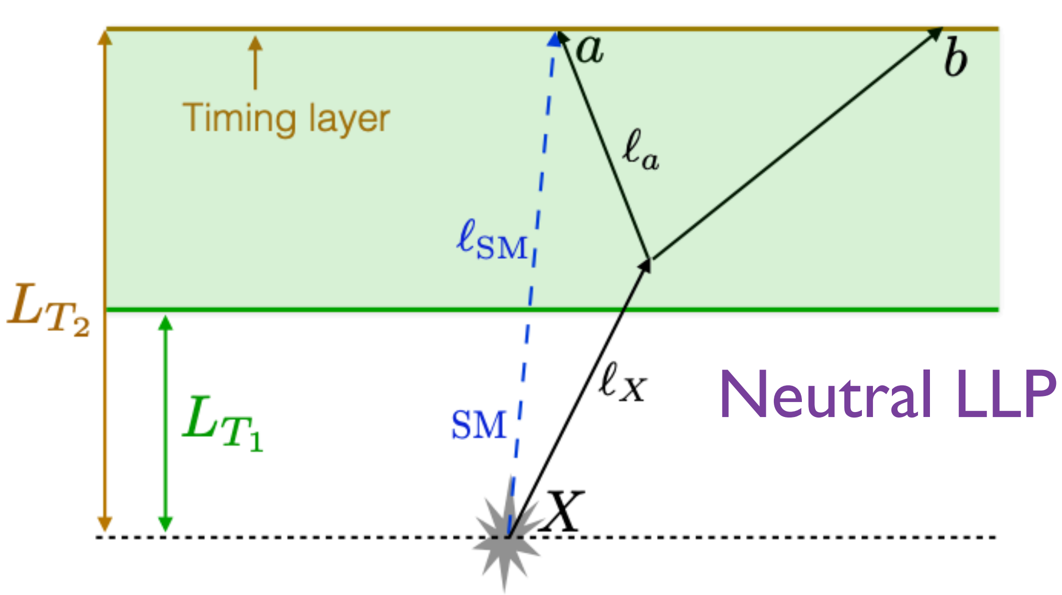
beta =0.75 (LanGauss)

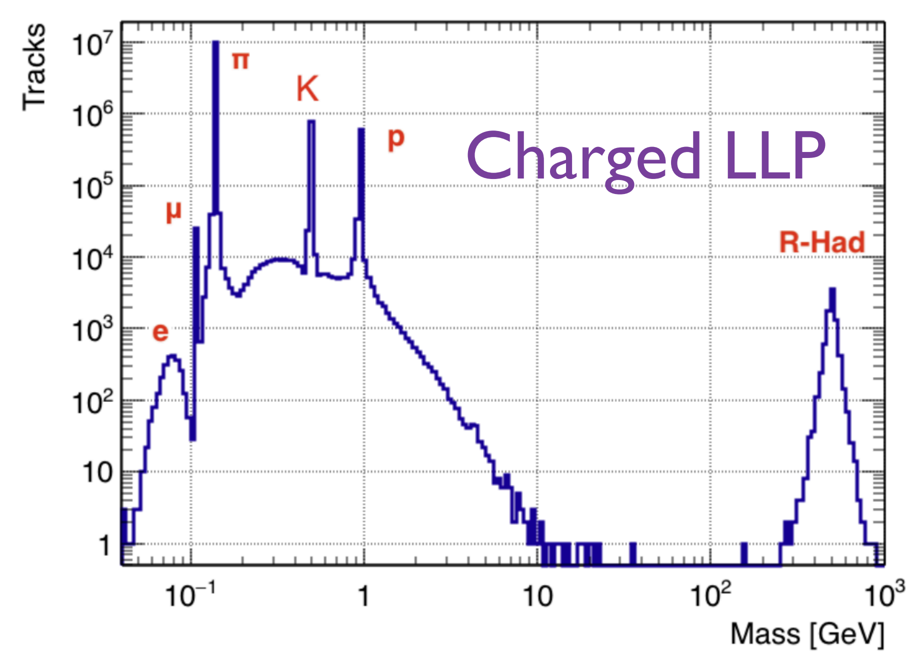
Timing detectors

- At the LHC, timing information can be used to disentangle hard vertex from pile-up, by **vertexing in 4D**
    - can this be used profitably in any way at FCC-ee?

- Timing can be used to measure TOF, and hence for **particle ID** (either SM or BSM long lived particles)

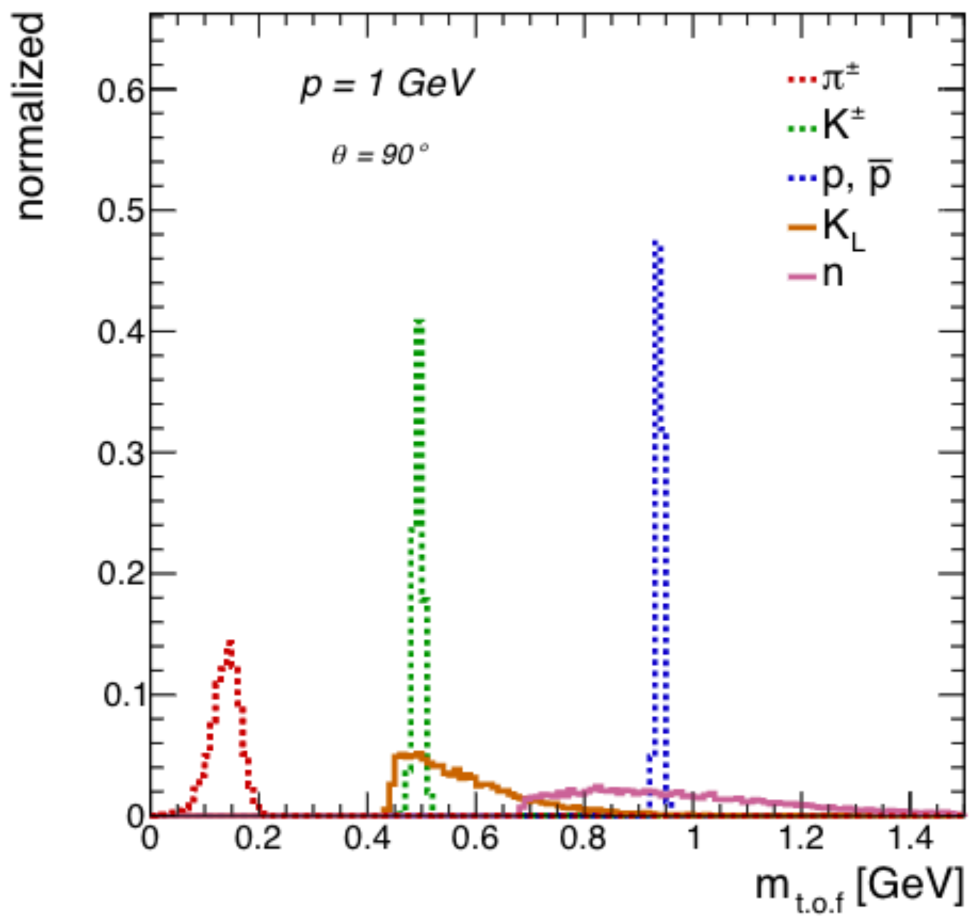






PID: Time of flight

- Implemented in the IDEA card for testing for both charged and neutrals:

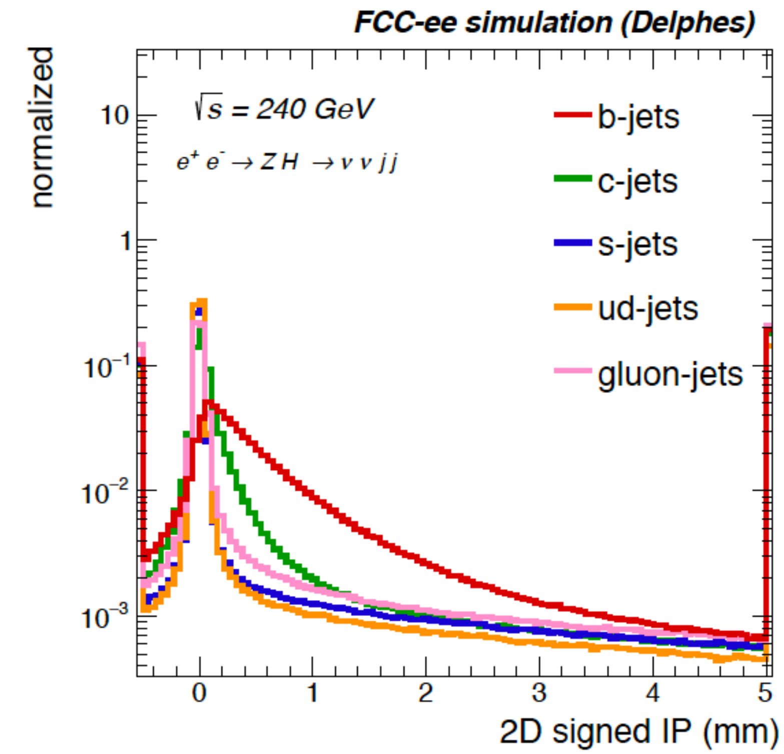



```
########################################
#   Time Smearing MIP
########################################

module TimeSmearing TimeSmearing {
  set InputArray ClusterCounting/tracks
  set OutputArray tracks

  # assume constant 30 ps resolution for now
  set TimeResolution {
                     (abs(eta) > 0.0 && abs(eta) <= 3.0)* 30E-12
                   }
}


########################################
#   Time Of Flight Measurement
########################################

module TimeOfFlight TimeOfFlight {
  set InputArray TimeSmearing/tracks
  set VertexInputArray TruthVertexFinder/vertices

  set OutputArray tracks

  # 0: assume vertex time tV from MC Truth (ideal case)
  # 1: assume vertex time tV = 0
  # 2: calculate vertex time as vertex TOF, assuming tPV=0

  set VertexTimeMode 0
}
```

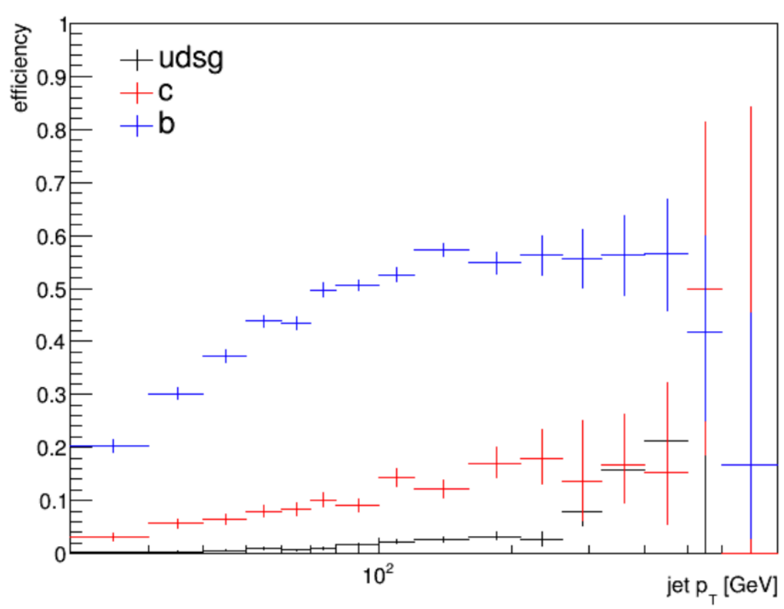
Heavy flavour Flavour Tagging

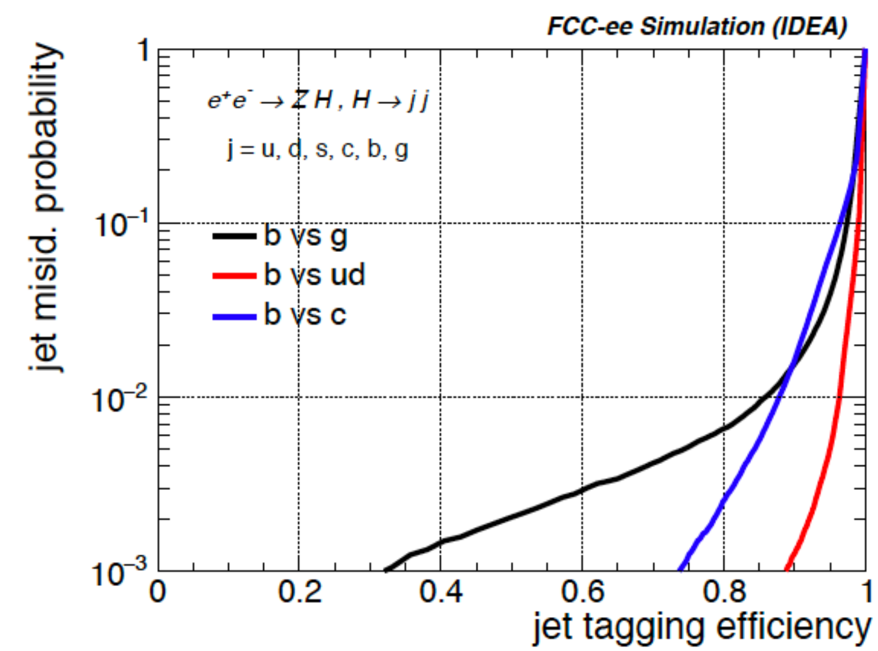
DELPHES fast simulation

- Track Counting B-Tagging:
  - parameterise **longitudinal** and **transverse impact parameter resolution (see previous slide)**
  - count number of tracks with significant displacement
  - no **secondary vertexing** is performed yet



Can be used in conjunction with TrkCovariance module to build MVA HF jet tagger





Reconstruction performance

- Delphes is not fully parameteric, object reco. efficiency (e.g. Tracking) and Calorimeter performance is parameterised, BUT:

    - Tracking resolution , dE/dx
    - Particle Flow
    - Jet (anti-kT/Valencia exclusive)
    - Missing energy
    - HF-tagging

can be predicted (with all the caveats of a fast simulation model) …

Conclusion

- Delphes provides a **simple, highly modular framework** for performing fast detector simulation

- Can be used and configured for:
  - quick **phenomenological** studies
  - explore new detector geometries (tracking)
  - as an **alternative for full-sim** if accurately tuned

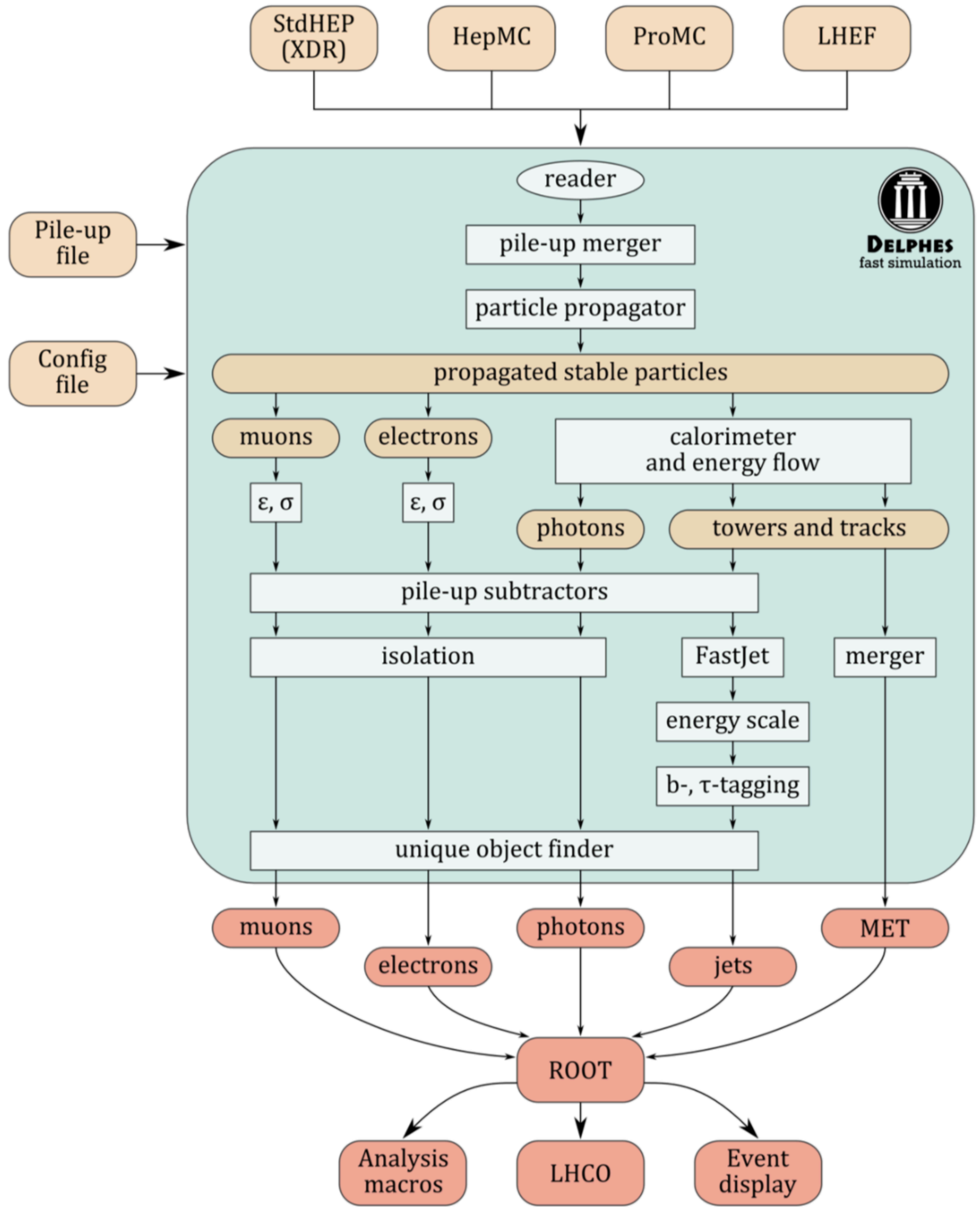
Backup

Modularity

- The modular system allows the user to configure a detector and schedule modules via a configuration file (.tcl), add modules, change data flow, alter output information

- Modules communicate entirely via exchange of collections (vectors) of universal objects (TObjArray of Candidate, 4-vector-like objects)

- Any module can access TObjArrays produced by other modules.



Run

- Install ROOT from root.cern.ch

- Clone Delphes from github.com/delphes

- Run Delphes:

```
>   ./configure
>   make
>   ./DelphesHepMC [detector_card] [output] [input(s)]
```

- Input formats: STDHEP, HepMC, ProMC, Pythia8

- Output: ROOT Tree

Configuration file

- Delphes configuration file is based on tcl scripting language
- This is where the detector parameters, the data-flow and the output content delphes root tree content are defined.
- Delphes provides tuned configurations for most existing detectors:
  - ATLAS, CMS, ILD, FCC, CEPC …

The order of execution of the various modules is configured in the execution path (usually defined at the beginning of the card):

```
set ExecutionPath {
    ParticlePropagator
    TrackEfficiency
    ...
    Calorimeter
    ...
    TreeWriter
}
```

Configuration file

```
module FastJetFinder FastJetFinder {

  set InputArray EFlowMerger/eflow
  set OutputArray jets

  # algorithm: 1 CDFJetClu, 2 MidPoint, 3 SIScone, 4 kt, 5 Cambridge/Aachen, 6 antikt
  set JetAlgorithm 5
  set ParameterR 0.8

  set ComputeNsubjettiness 1
  set Beta 1.0
  set AxisMode 4

  set ComputeTrimming 1
  set RTrim 0.2
  set PtFracTrim 0.05

  set ComputePruning 1
  set ZcutPrun 0.1
  set RcutPrun 0.5
  set RPrun 0.8

  set ComputeSoftDrop 1
  set BetaSoftDrop 0.0
  set SymmetryCutSoftDrop 0.1
  set R0SoftDrop 0.8

  set JetPTMin 20.0


}
```

Configuration file

```
module Calorimeter Calorimeter {

  set ParticleInputArray ParticlePropagator/stableParticles
  set TrackInputArray TrackMerger/tracks

  set TowerOutputArray towers
  set PhotonOutputArray photons

  set EFlowTrackOutputArray eflowTracks
  set EFlowPhotonOutputArray eflowPhotons
  set EFlowNeutralHadronOutputArray eflowNeutralHadrons

  ...

  # 10 degrees towers
  set PhiBins {}
  for {set i -18} {$i <= 18} {incr i} {
    add PhiBins [expr {$i * $pi/18.0}]
  }
  foreach eta {-3.2 -2.5 -2.4 -2.3 -2.2 -2.1 -2 -1.9 -1.8 -1.7 -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1 -0.9 -0.8
-0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8
1.9 2 2.1 2.2 2.3 2.4 2.5 2.6 3.3} {
    add EtaPhiBins $eta $PhiBins
  }

  ...
  set ECalResolutionFormula {
            (abs(eta) <= 1.5) * (1+0.64*eta^2) * sqrt(energy^2*0.008^2 + energy*0.11^2 + 0.40^2) +
            (abs(eta) > 1.5 && abs(eta) <= 2.5) * (2.16 + 5.6*(abs(eta)-2)^2) * sqrt(energy^2*0.008^2 +
energy*0.11^2 + 0.40^2) +
            (abs(eta) > 2.5 && abs(eta) <= 5.0) * sqrt(energy^2*0.107^2 + energy*2.08^2)}
```

input(s) candidates

output(s) candidates

13

Configuration file

Output collections are configured in the TreeWriter module

```
module TreeWriter TreeWriter {
# add Branch InputArray BranchName BranchClass
  add Branch Delphes/allParticles Particle GenParticle

  add Branch TrackMerger/tracks Track Track
  add Branch Calorimeter/towers Tower Tower

  add Branch Calorimeter/eflowTracks EFlowTrack Track
  add Branch Calorimeter/eflowPhotons EFlowPhoton Tower
  add Branch Calorimeter/eflowNeutralHadrons EFlowNeutralHadron Tower

  add Branch GenJetFinder/jets GenJet Jet
  add Branch GenMissingET/momentum GenMissingET MissingET

  add Branch UniqueObjectFinder/jets Jet Jet
  add Branch UniqueObjectFinder/electrons Electron Electron
  add Branch UniqueObjectFinder/photons Photon Photon
  add Branch UniqueObjectFinder/muons Muon Muon
  add Branch MissingET/momentum MissingET MissingET
  add Branch ScalarHT/energy ScalarHT ScalarHT
}
```