DELPHES
Fast Detector Simulation

Michele Selvaggi

CERN

PREFIT school 09/03/2020

Outline

- Introduction:
  - particle detectors
  - general principles, motivation, use cases …

- Overview
  - main modules:
    - tracking
    - calorimetry
    - particle flow
    - heavy flavour tagging

- Discussion
  - going beyond
    - realistic tracking,
    - PID with TOF, dEdX

- Practical use

Introduction

What is a particle?

Theorist answer:

A particle is an irreducible representation of the Poincare group …

ON UNITARY REPRESENTATIONS OF THE INHOMOGENEOUS LORENTZ GROUP*

By E. Wigner

(Received December 22, 1937)

1. Origin and Characterization of the Problem

It is perhaps the most fundamental principle of Quantum Mechanics that the system of states forms a *linear manifold*,[1] in which a unitary *scalar product* is defined.[2] The states are generally represented by wave functions[3] in such a way that φ and constant multiples of φ represent the same physical state. It is possible, therefore, to normalize the wave function, i.e., to multiply it by a constant factor such that its scalar product with itself becomes 1. Then, only a constant factor of modulus 1, the so-called phase, will be left undetermined in the wave function. The linear character of the wave function is called the superposition principle. The square of the modulus of the unitary scalar product (ψ, φ) of two normalized wave functions ψ and φ is called the transition probability from the state ψ into φ, or conversely. This is supposed to give the probability that an experiment performed on a system in the state φ, to see whether or not the state is ψ, gives the result that it is ψ. If there are two or more different experiments to decide this (e.g., essentially the same experiment,

What is a particle?

DELPHES fast simulation

Theorist answer:

A particle is an irreducible representation of the Poincare group …

ON UNITARY REPRESENTATIONS OF THE INHOMOGENEOUS LORENTZ GROUP*

By E. Wigner

(Received December 22, 1937)

1. Origin and Characterization of the Problem

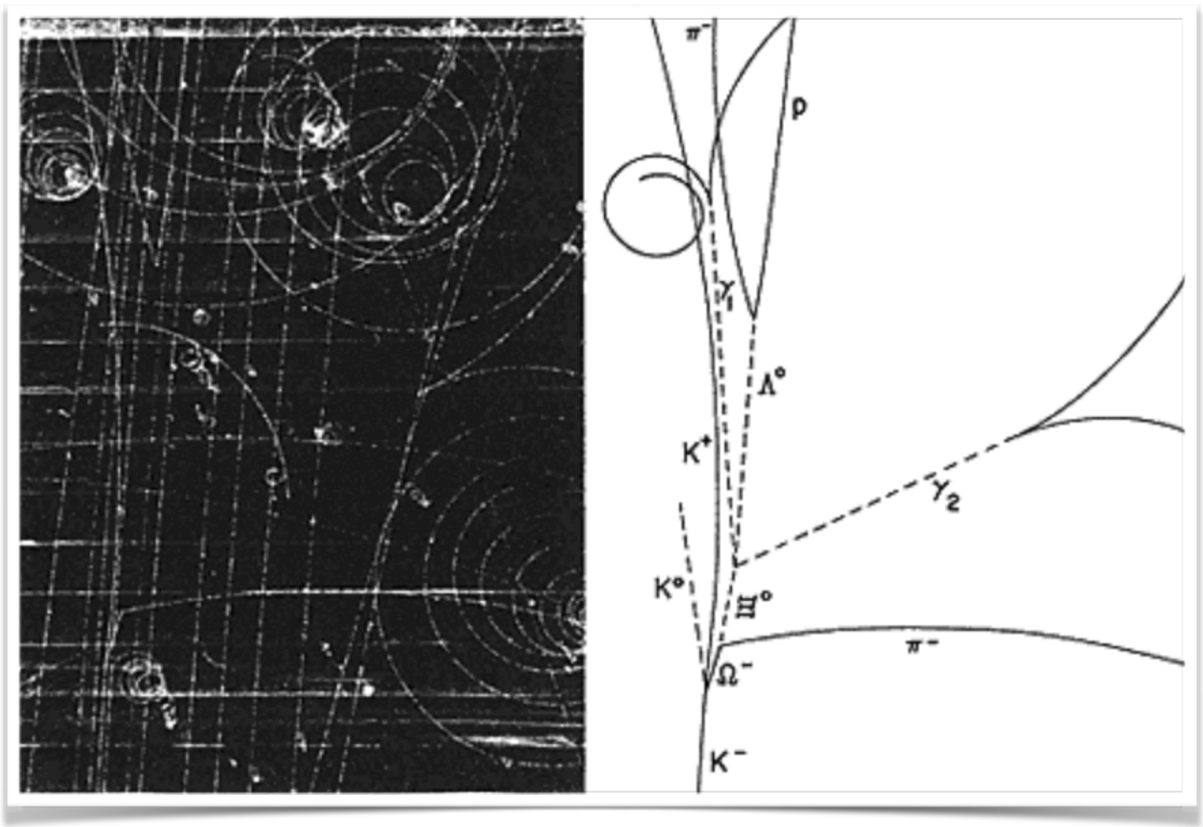
It is perhaps the most fundamental principle of Quantum Mechanics that the system of states forms a *linear manifold*,[1] in which a unitary *scalar product* is defined.[2] The states are generally represented by wave functions[3] in such a way that φ and constant multiples of φ represent the same physical state. It is possible, therefore, to normalize the wave function, i.e., to multiply it by a constant factor such that its scalar product with itself becomes 1. Then, only a constant factor of modulus 1, the so-called phase, will be left undetermined in the wave function. The linear character of the wave function is called the superposition principle. The square of the modulus of the unitary scalar product (ψ, φ) of two normalized wave functions ψ and φ is called the transition probability from the state ψ into φ, or conversely. This is supposed to give the probability that an experiment performed on a system in the state φ, to see whether or not the state is ψ, gives the result that it is ψ. If there are two or more different experiments to decide this (e.g., essentially the same experiment,

Experimentalist reaction:

What is a particle?

Experimentalist answer:

A particle is an object that interacts with your detector such that its track can be followed

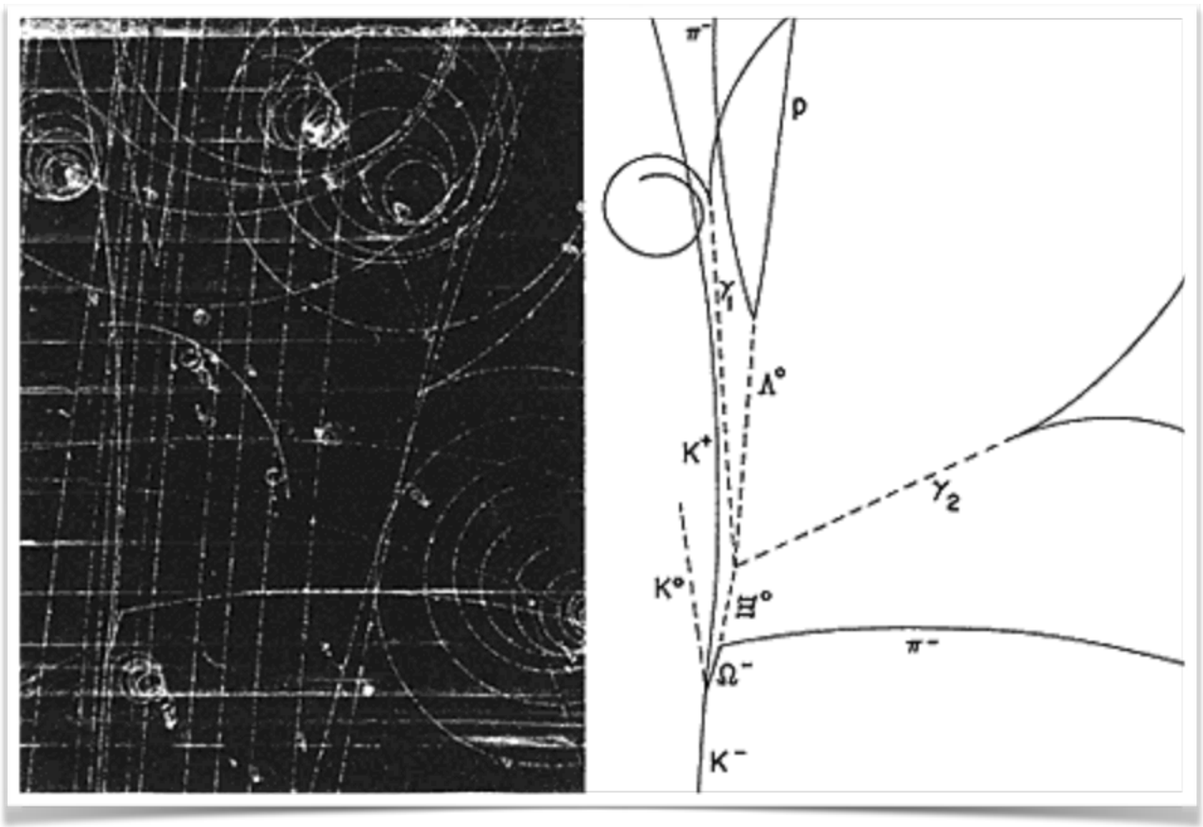


Ω- discovery (1964)

What is a particle?

Experimentalist answer:

A particle is an object that interacts with your detector such that its track can be followed



Ω- discovery (1964)

Stable particles:

electrons, photons, protons, neutrinos, (neutrons), K, pions

Un-(stable) particles in colliders ?

Collider Detector size: L = 10 m

- $\mu \rightarrow \tau = 10^{-6}$ s $\rightarrow L_{decay} = 1$ km
- $\pi^{+/-} \rightarrow \tau = 10^{-8}$ s $\rightarrow L_{decay} = 10$ m
- $K^{+/-} \rightarrow \tau = 5\ 10^{-9}$ s $\rightarrow L_{decay} = 5$ m
- $K^0_L \rightarrow \tau = 10^{-8}$ s $\rightarrow L_{decay} = 15$ m
- $K^0_S \rightarrow \tau = 10^{-11}$ s $\rightarrow L_{decay} = 2$ cm
- $\tau^{+/-} \rightarrow \tau = 10^{-13}$ s $\rightarrow L_{decay} = 100$ µm
- B $\rightarrow \tau = 5\ 10^{-13}$ s $\rightarrow L_{decay} = 500$ µm

Un-(stable) particles in colliders ?

Collider Detector size: L = 10 m

- $\mu \rightarrow \tau = 10^{-6}$ s $\rightarrow L_{decay} = 1$ km
- $\pi^{+/-} \rightarrow \tau = 10^{-8}$ s $\rightarrow L_{decay} = 10$ m
- $K^{+/-} \rightarrow \tau = 5\ 10^{-9}$ s $\rightarrow L_{decay} = 5$ m
- $K^0_L \rightarrow \tau = 10^{-8}$ s $\rightarrow L_{decay} = 15$ m

Stable particles:

e, p, γ, μ, π, $K^{+/-}$, K^0_L, n …

- $K^0_S \rightarrow \tau = 10^{-11}$ s $\rightarrow L_{decay} = 2$ cm

- $\tau^{+/-} \rightarrow \tau = 10^{-13}$ s $\rightarrow L_{decay} = 100\ \mu$m
- B $\rightarrow \tau = 5\ 10^{-13}$ s $\rightarrow L_{decay} = 500\ \mu$m

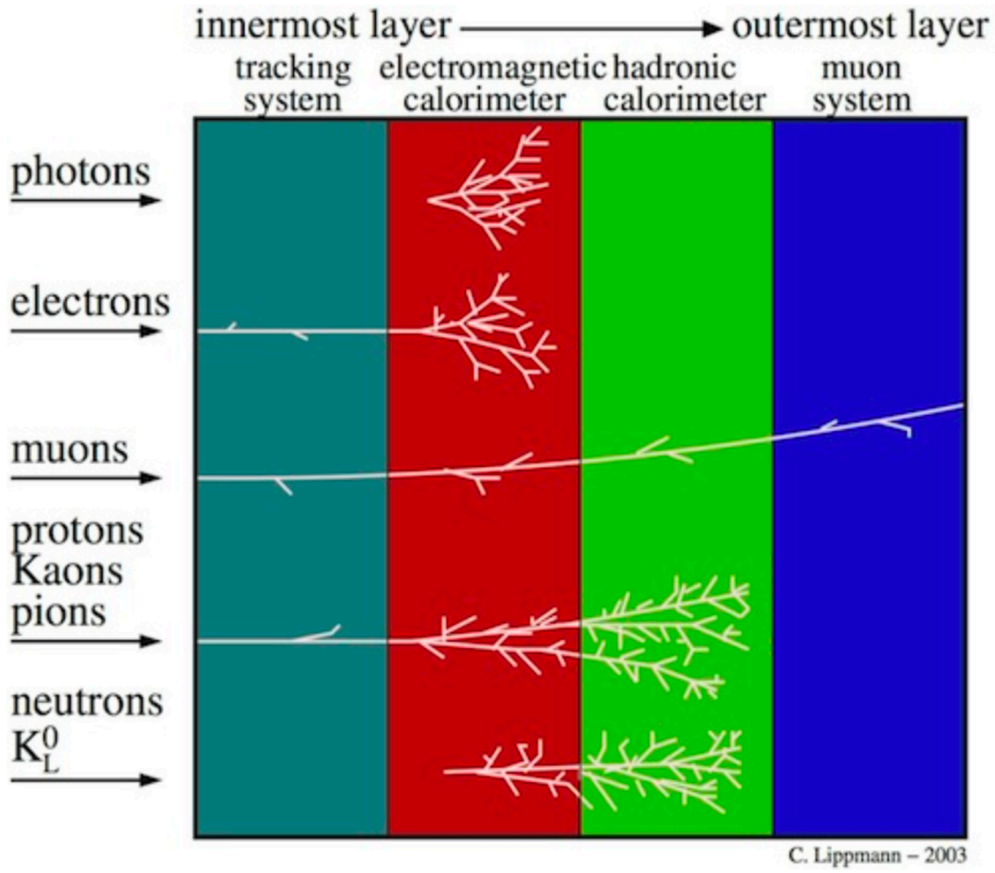
Vertex detectors:

give rise to displaced vertices

Particle Detectors

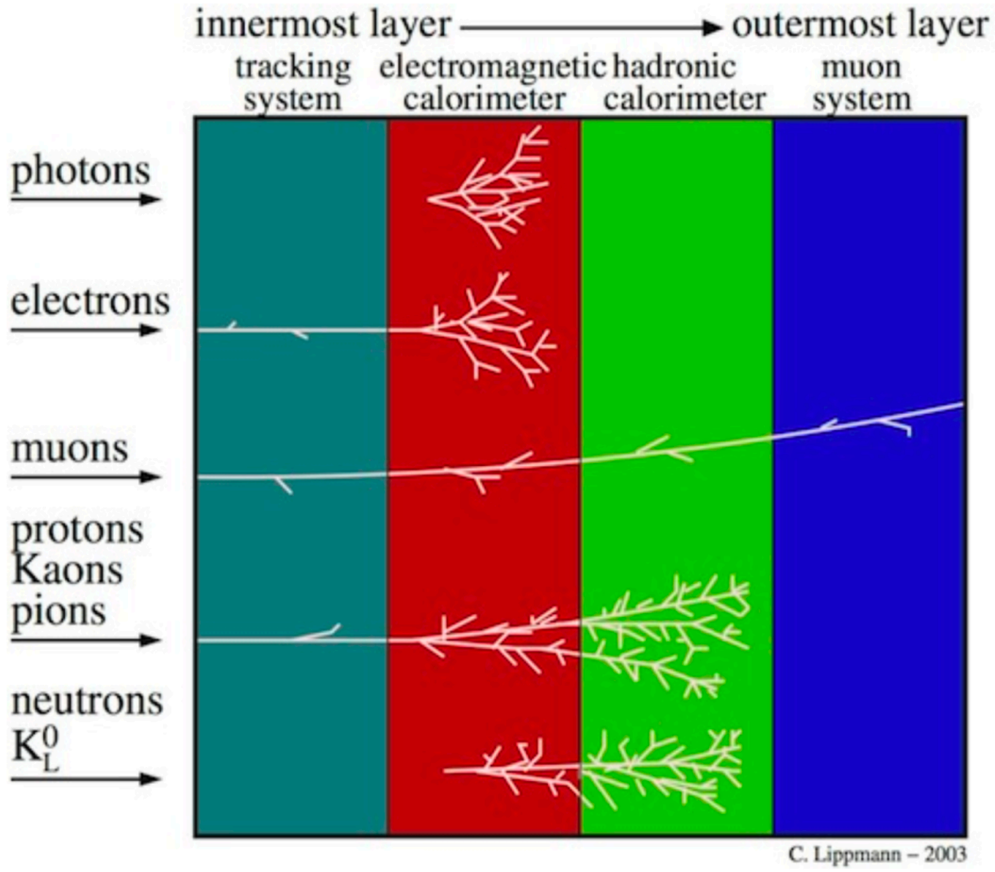
Particles are detected via their interaction with matter:

- electro-magnetic (e, μ, ɣ, p, $\pi^{+/-}$, $K^{+/-}$)
  - charged particle ionisation loss
  - EM cascades (ɣ pair production + brehmstrahlung)
- strong (p, n, K)
  - hadronic shower
- weak (ν) → negligible at colliders, gives missing energy



Particle Detectors

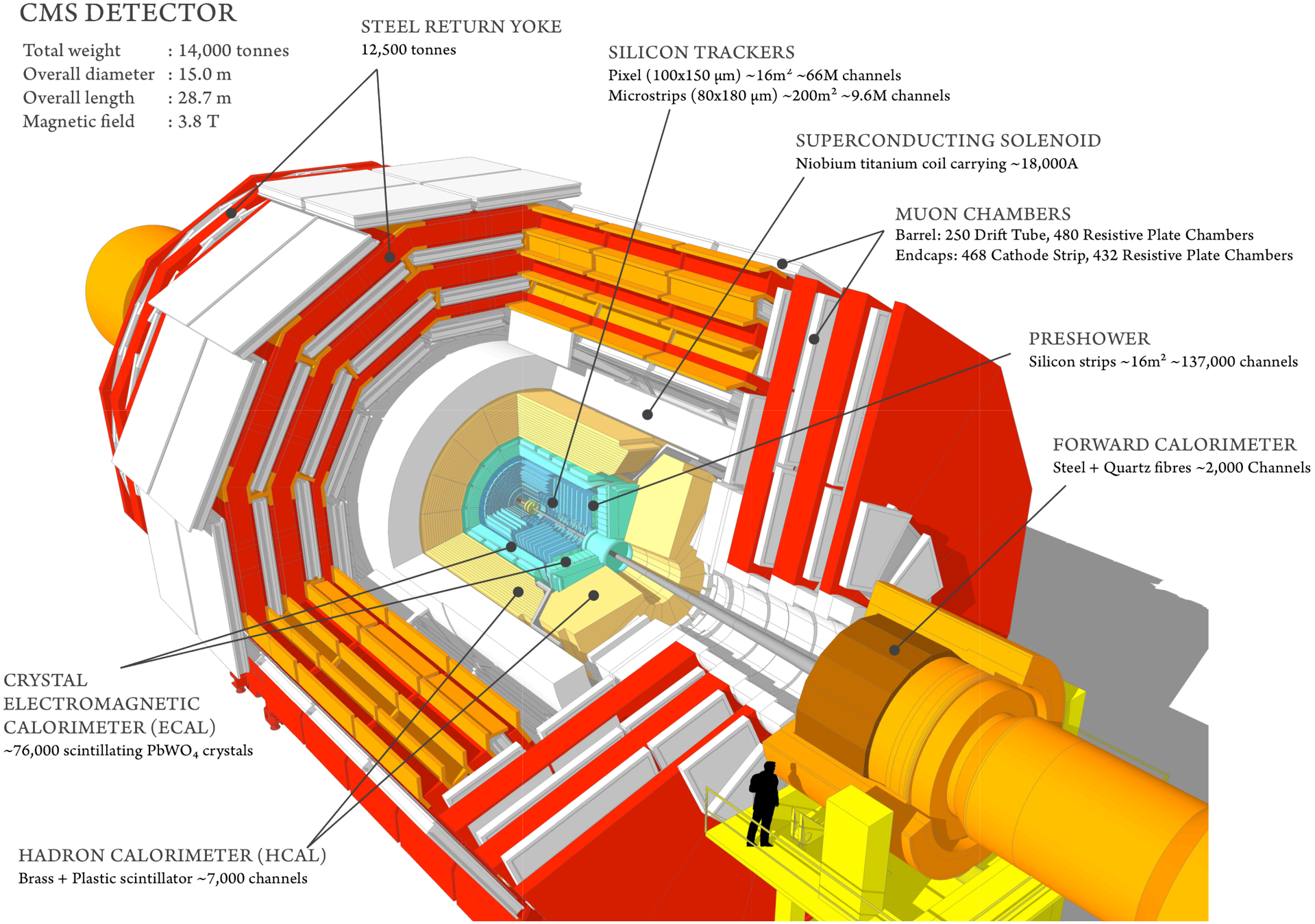
- Trackers:
  - Measures momenta of charged particles through curvature of trajectory inside magnetic field
- Calorimeters (EM, hadronic):
  - Measures total energy of particles by destroying it (scintillation light, ionisation loss)…



- These considerations tightly constrain the layout of a (collider) detector:
  - Tracker
  - EM calo
  - Hadronic calo
  - Muon chambers

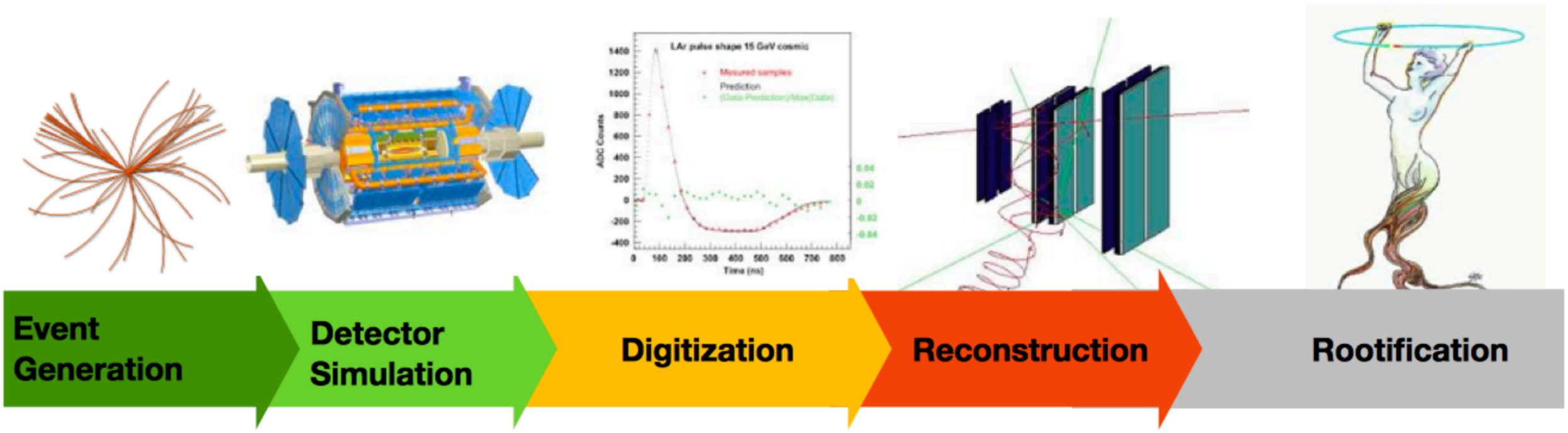
Particle Detector in real life

DELPHES
fast simulation



MonteCarlo EvGen

DELPHES
fast simulation



courtesy of Andreas Salzburger

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

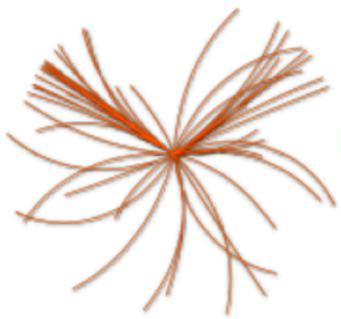
- Parametric simulation (**Delphes**, PGS):
  - **parameterise detector response** at the particle level(efficiency, resolution on tracks, calorimeter objects)
  - reconstruct **complex objects** and observables(use particle-flow, jets, missing ET, pile-up ..)

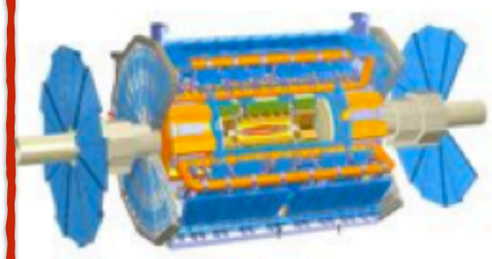
10^{-2} - 10^{-1} s/ev

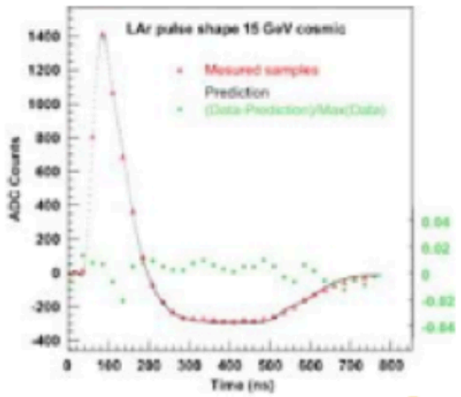
- Ultra Fast (ATOM, TurboSim):
  - from parton to detector object (smearing/lookup tables)

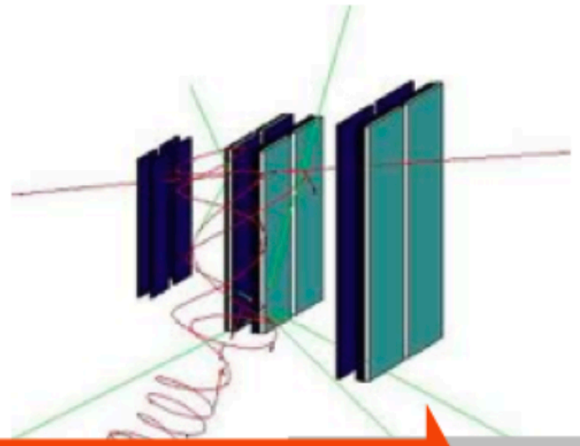
MonteCarlo EvGen

DELPHES
fast simulation









Event Generation → Detector Simulation → Digitization → Reconstruction → Rootification

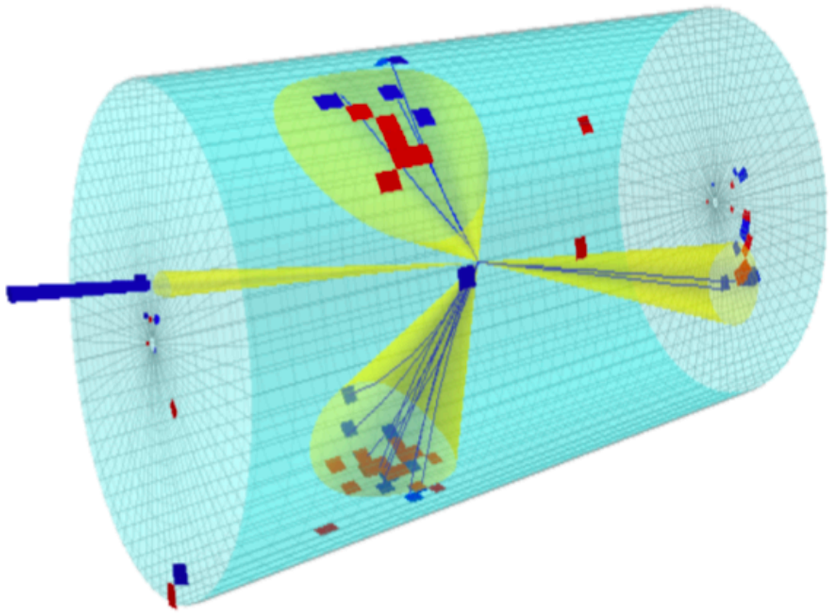
Delphes FastSim

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



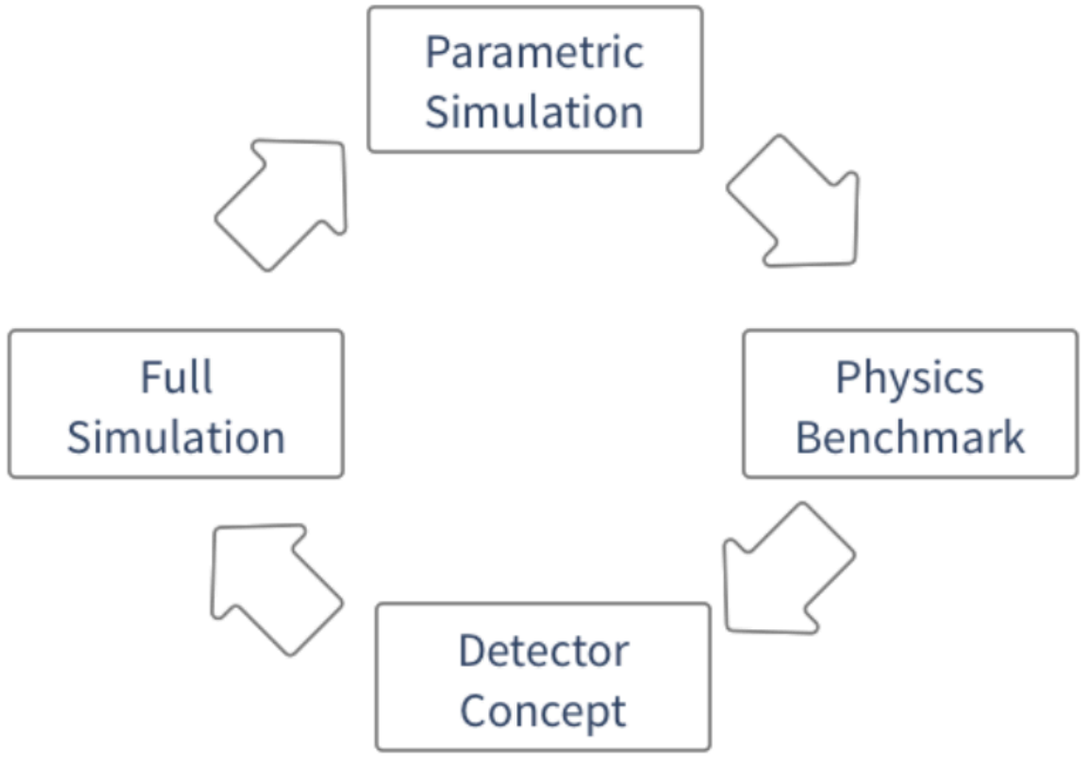
- Provides:
  - leptons, photons, neutral hadrons
  - jets, missing energy
  - heavy flavour tagging

- 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

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

What is Delphes ?

- Delphes started in 2007 as a project for Fast Detector Simulation

- Delphes 3, released in 2013 is community based:
  - on gitHub, ticket system
  - several user proposed patches
  - docker, singularity image in hepsim

- Reference FastSim tool for pheno community, SnowMass, ECFA, FCC, CMS

- Dependencies:
  - gcc, tcl, ROOT
  - is shipped with FastJet

github: github.com/delphes
website: cp3.irmp.ucl.ac.be/projects/delphes

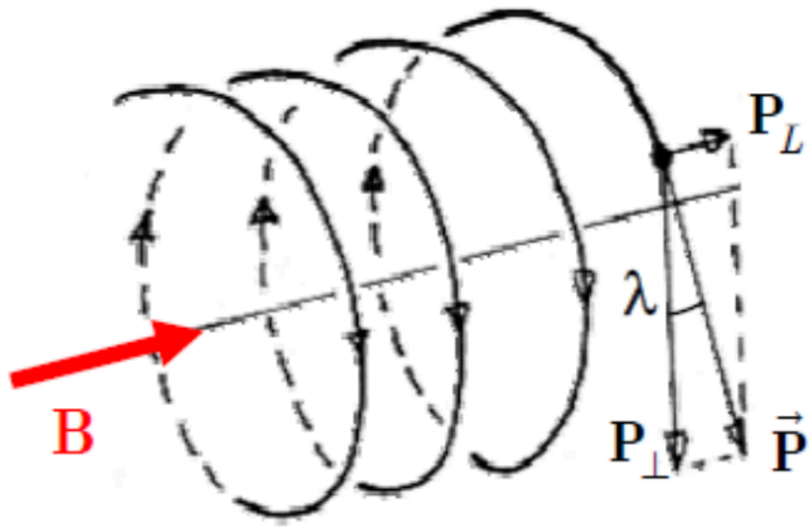
Overview

Charged Particle parameterisation

DELPHES fast simulation

- 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})

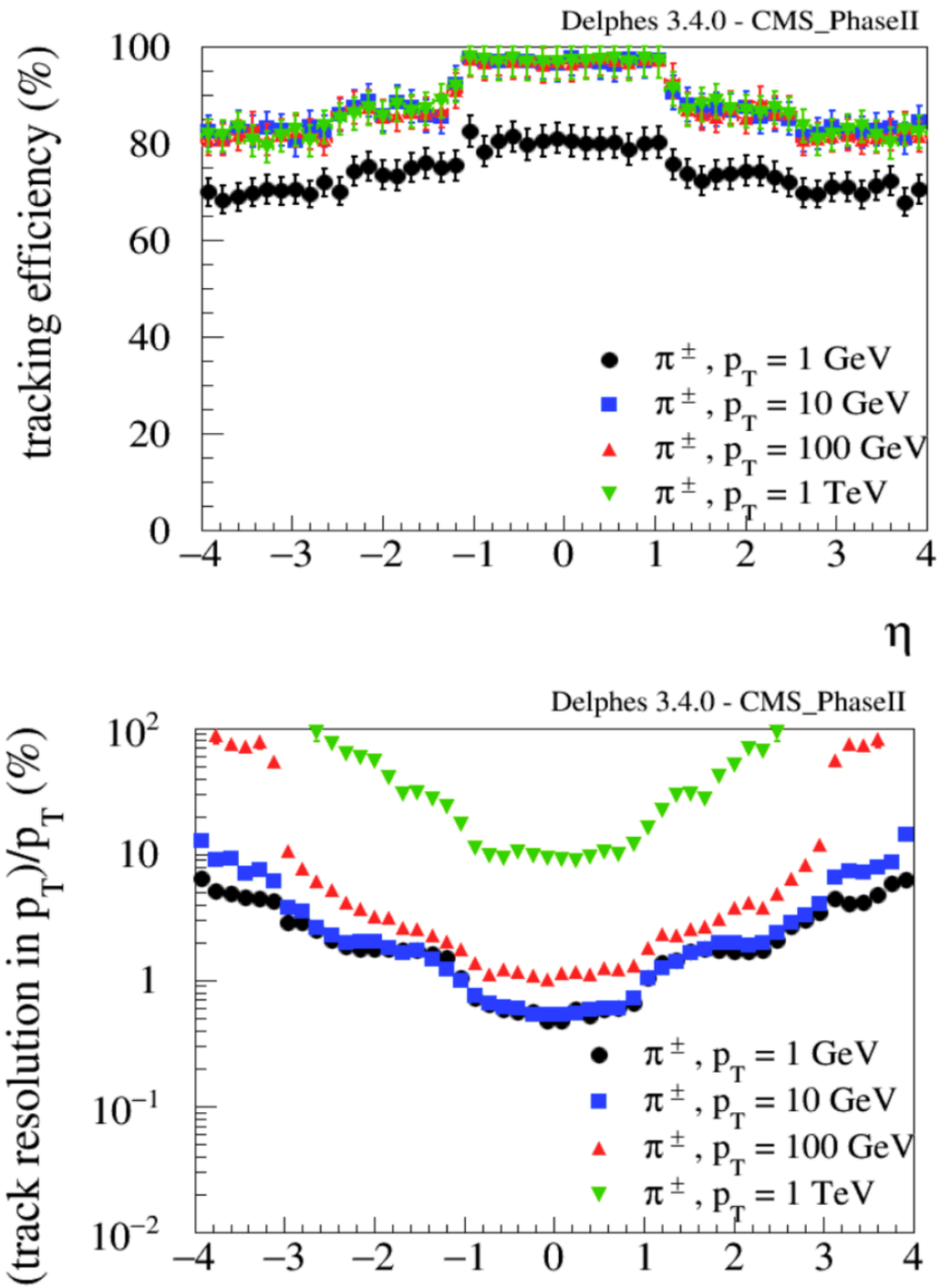


- **Efficiency** and **resolution** depends on:
  - particle ID (electron, muon or charged hadron)
  - particle 4-momentum

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

Tracking parameterisation

DELPHES
fast simulation




```
####################################
# 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) + \
```

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

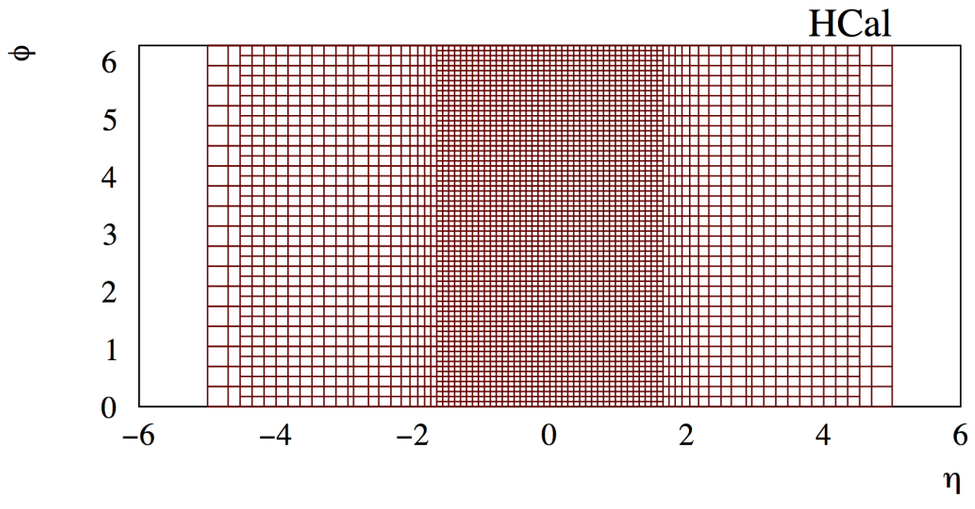
DELPHES fast simulation

- 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 γ π^0	1	0
Long-lived neutral hadrons (K^0_S , Λ^0)	0.3	0.7
ν μ	0	0
others	0	1

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

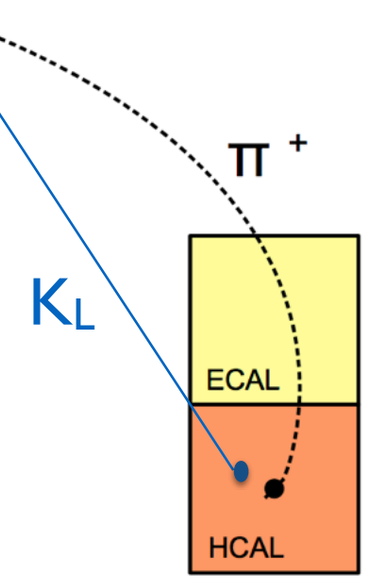
$$\left(\frac{\sigma}{E}\right)^2 = \left(\frac{S(\eta)}{\sqrt{E}}\right)^2 + \left(\frac{N(\eta)}{E}\right)^2 + C(\eta)^2$$



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 $E_{Neutral} = E_{calo} - E_{trk}$



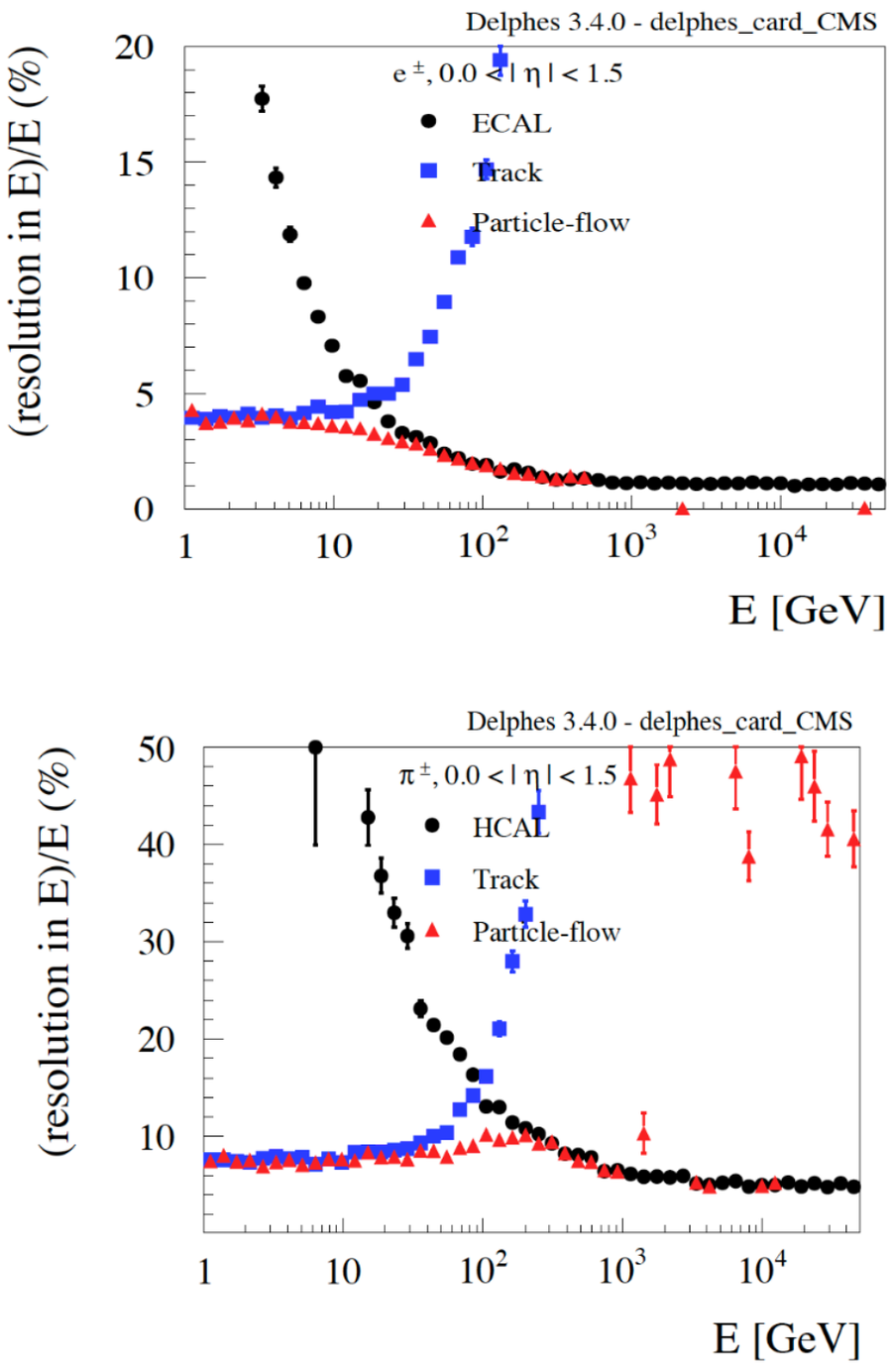
Algorithm:

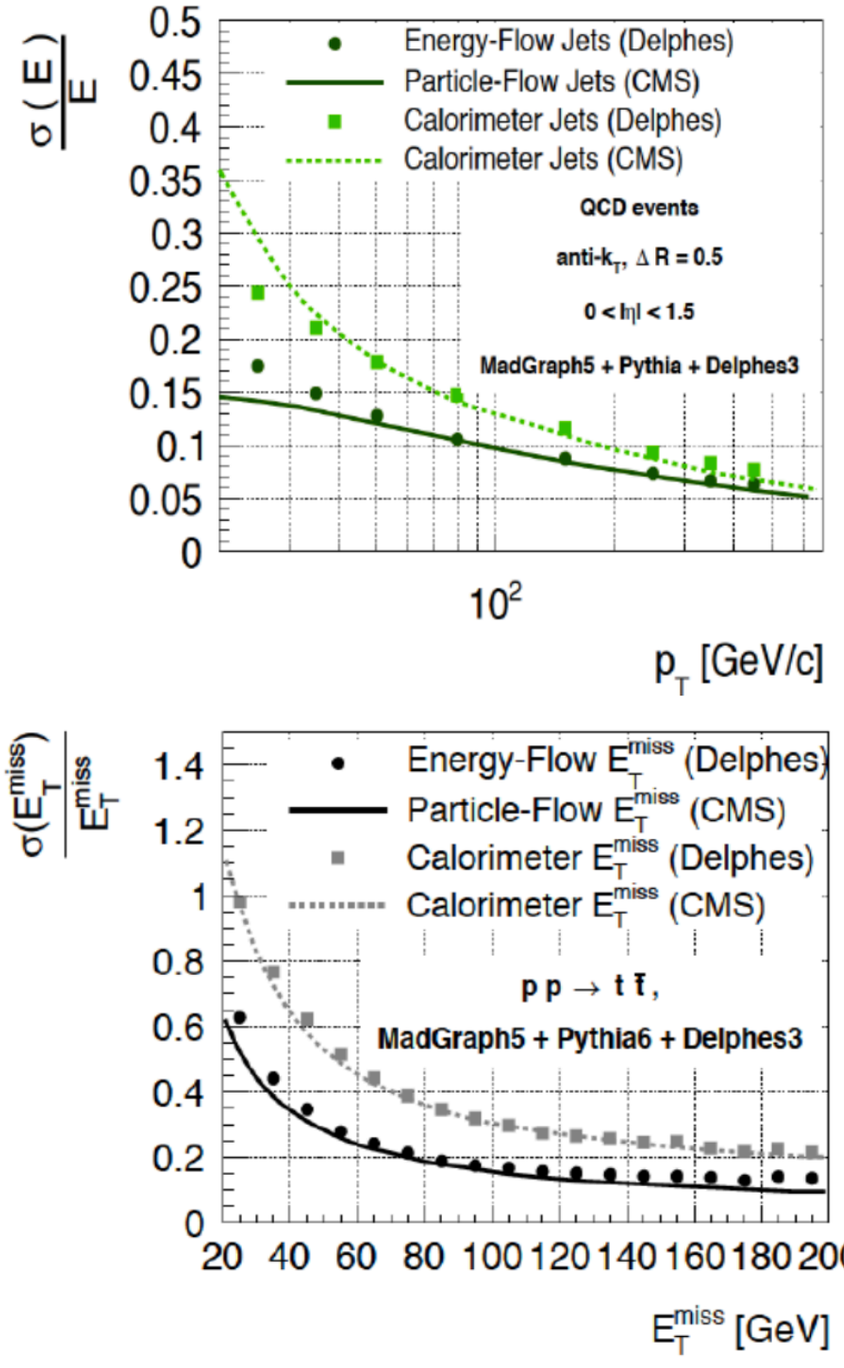
- If $E_{neutral}/\sqrt{(\sigma_{calo}^2 + \sigma_{trk}^2)} > S$:
  → create PF-neutral particle + PF-track

- 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

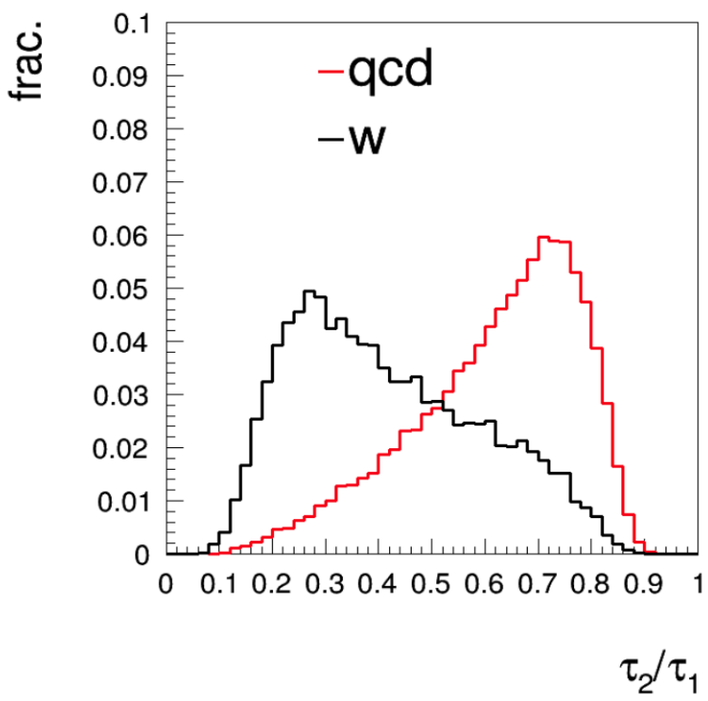




Jets and Substructure

- FastJet performs **jet clustering** via the FastJetFinder module
- Most used Jet substructure algorithms are included (N-subjettiness, SoftDrop, Trimming, Pruning …)
- Delphes can also be used as a library for producing detector 4-vector objects: tracks, calo-towers or particle-flow candidates (see info here)

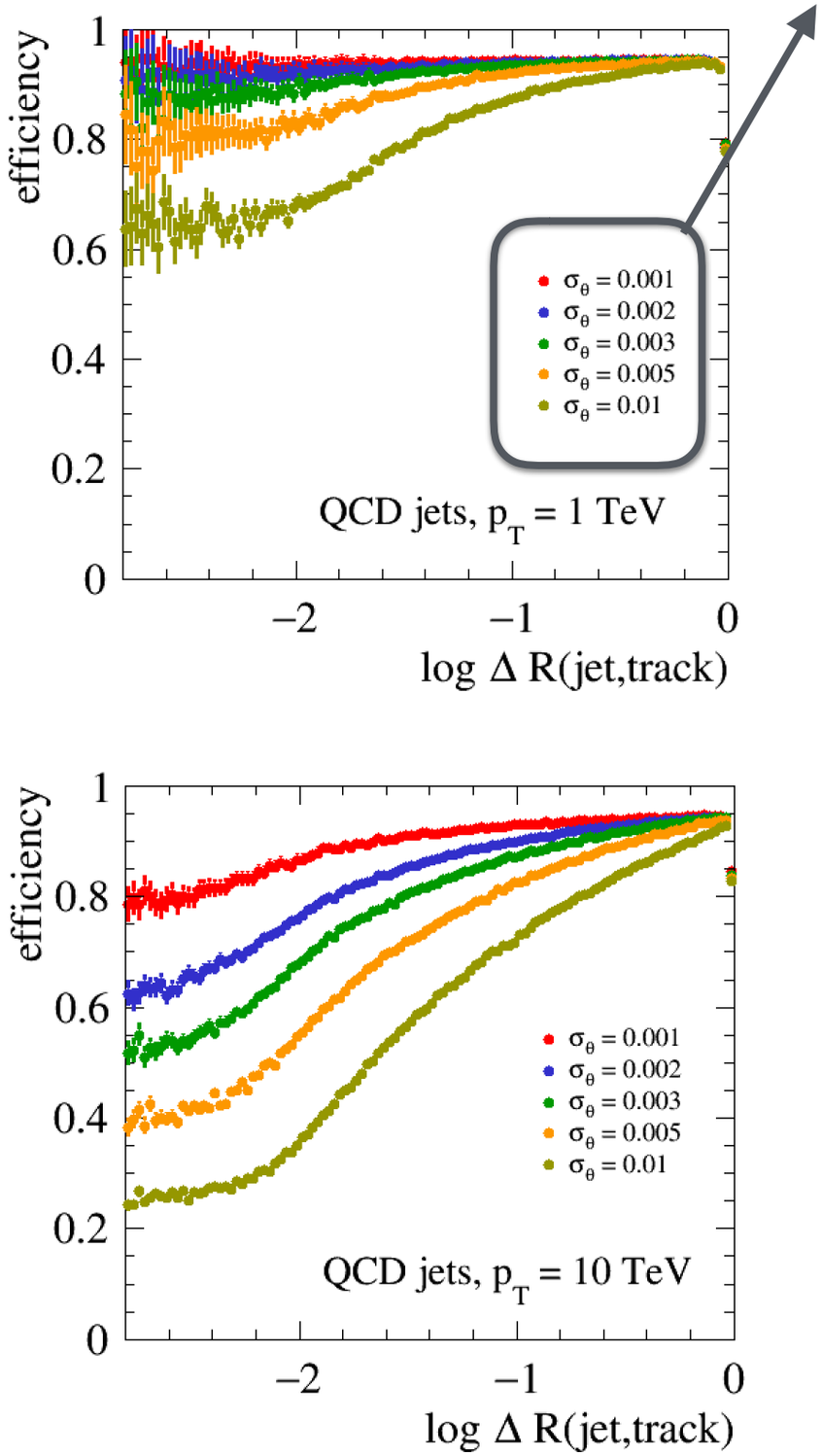
```
############
# Jet finder
############

module FastJetFinder FatJetFinder {
#  set InputArray TowerMerger/towers
  set InputArray EFlowMerger/eflow

  set OutputArray jets

  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 200.0
}
```

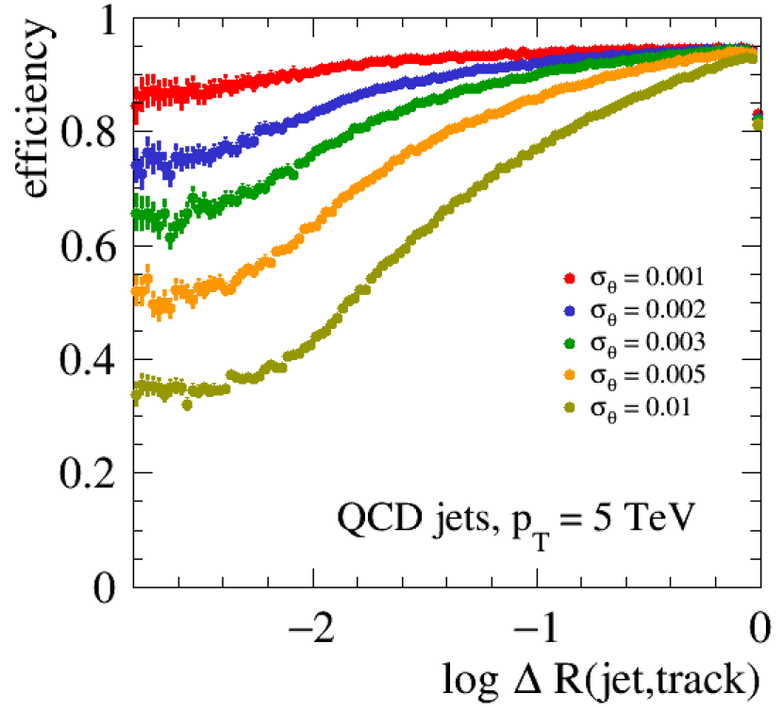


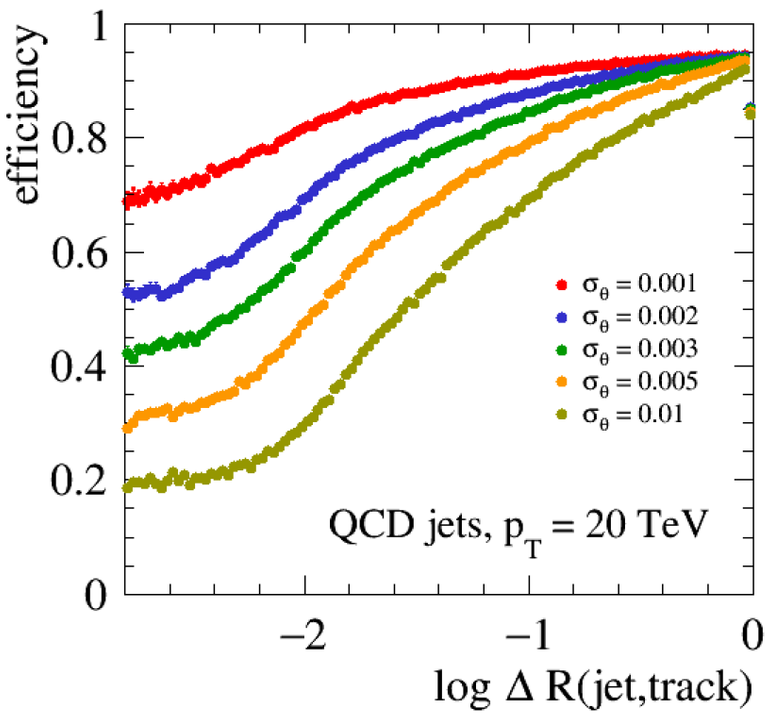
Jet substructure in Dense environment

DELPHES fast simulation

Instrinsic tracking angular resolution

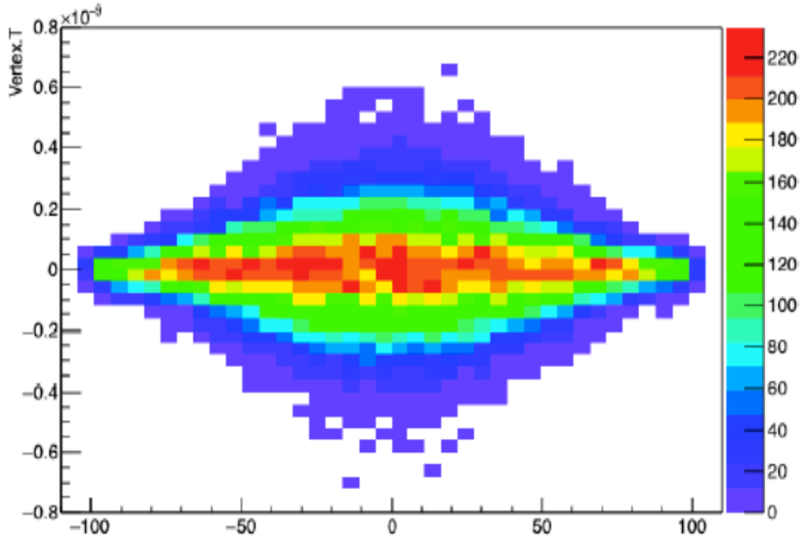






Pile-up Simulation and Timing

- Pile-up can be mixed with hard event, with f(z,t) profile
- Time of flight automatically computed in Delphes and propagated in the tracking volume, timing measurement simulated with TimeSmearing module

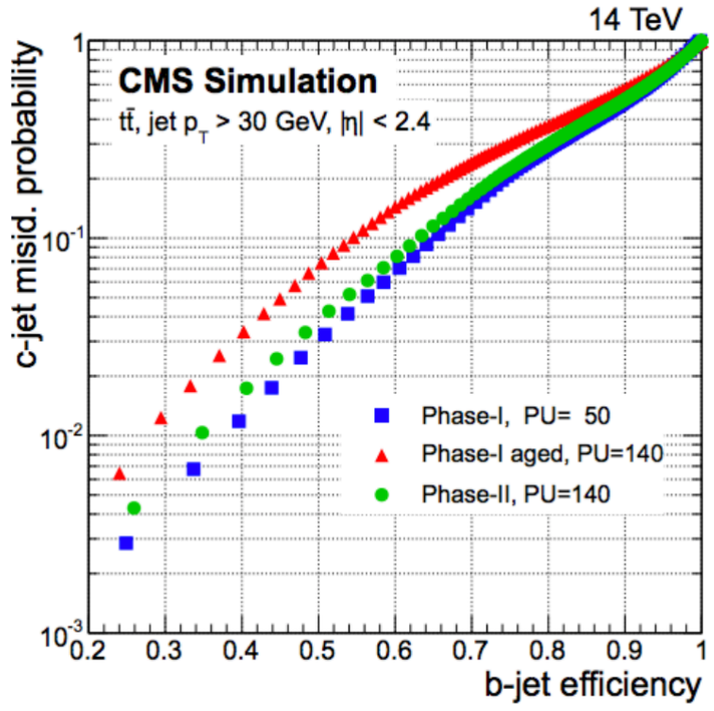


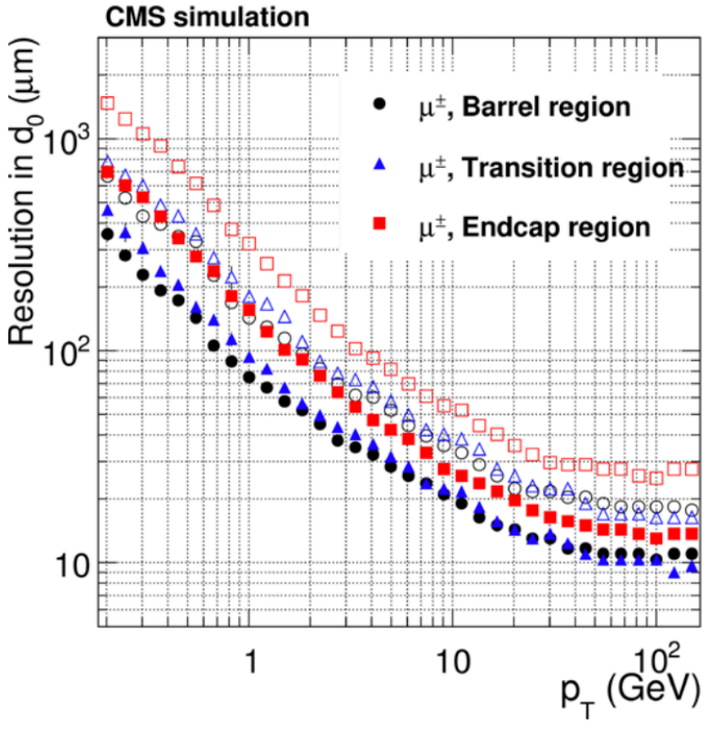

```
########################################
#   Time Smearing Neutral Photons
########################################

module TimeSmearing TimeSmearingPhotons {
  set InputArray ECal/eflowPhotons
  set OutputArray photons
  set TimeResolution {
                        (abs(eta) > 0.0 && abs(eta) <= 6.0) * sqrt(20e-12^2 + 150e-12^2)/energy^2
                     }
}
```

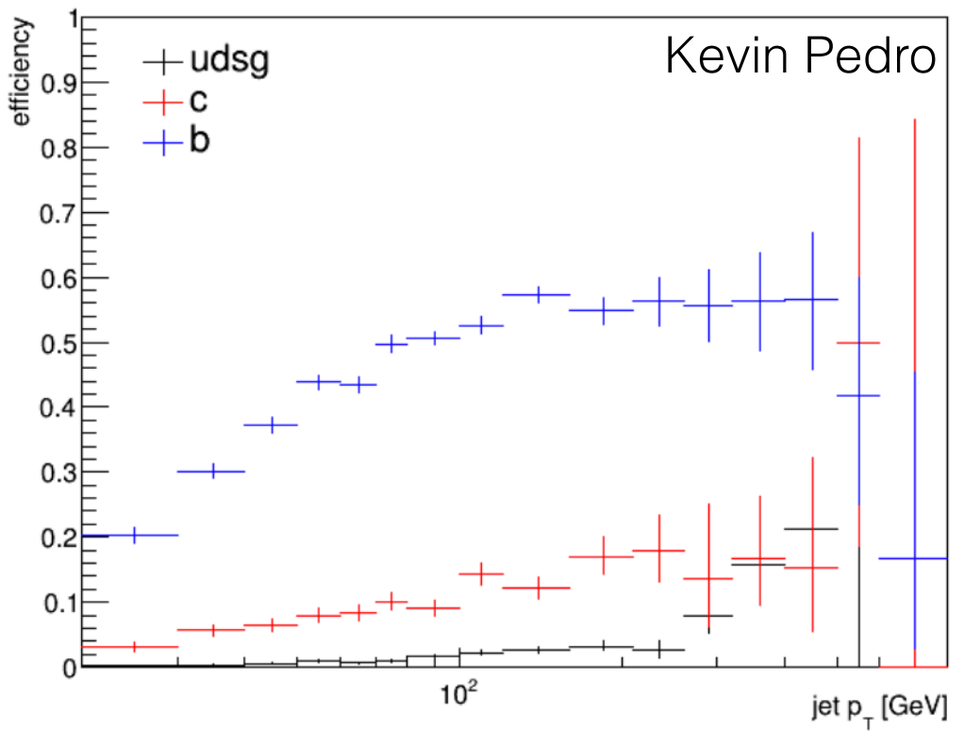
Heavy flavour Flavor Tagging

- Parametric efficiencies and mis-identification rates (both for b and τ tagging)

- Track Counting B-Tagging:
  - parameterise **longitudinal** and **transverse impact parameter resolution (via TrackSmearing module)**
  - count number of tracks with significant displacement







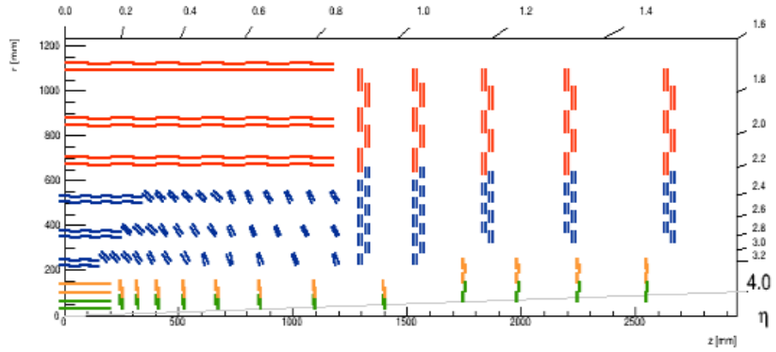
Going beyond

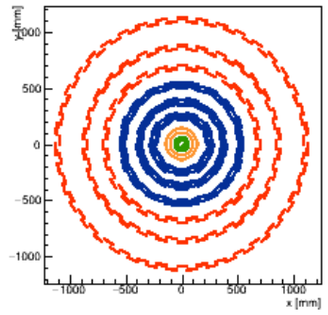
tkLayout

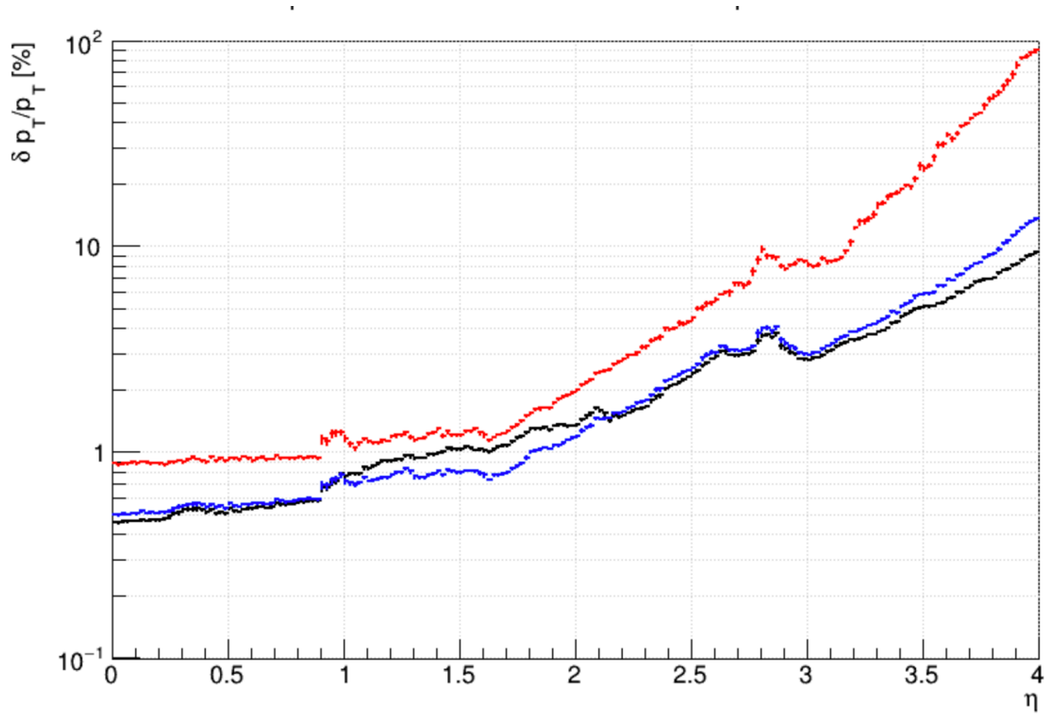
- **TrackSmearing** module allows for **diagonal smearing of (d_0, d_z, p, ctg θ, φ)**

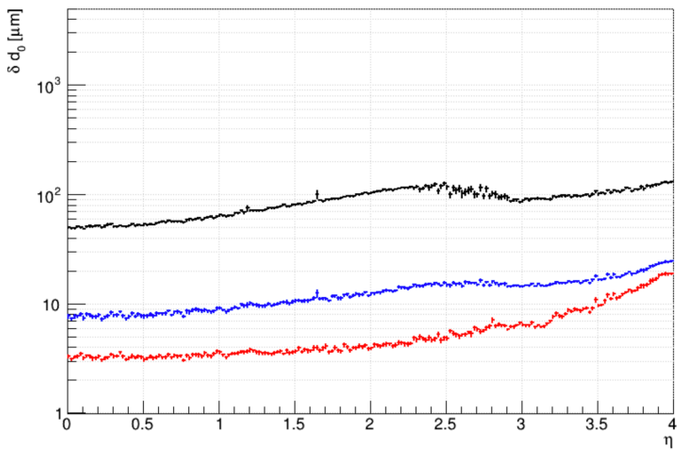
- Tracking performance can be predicted with **tkLayout:**
  - used for CMS PhaseII , FCC-hh
  - public tool

- Use "delphize" scripts to produce **Delphes-friendly** parameterisations





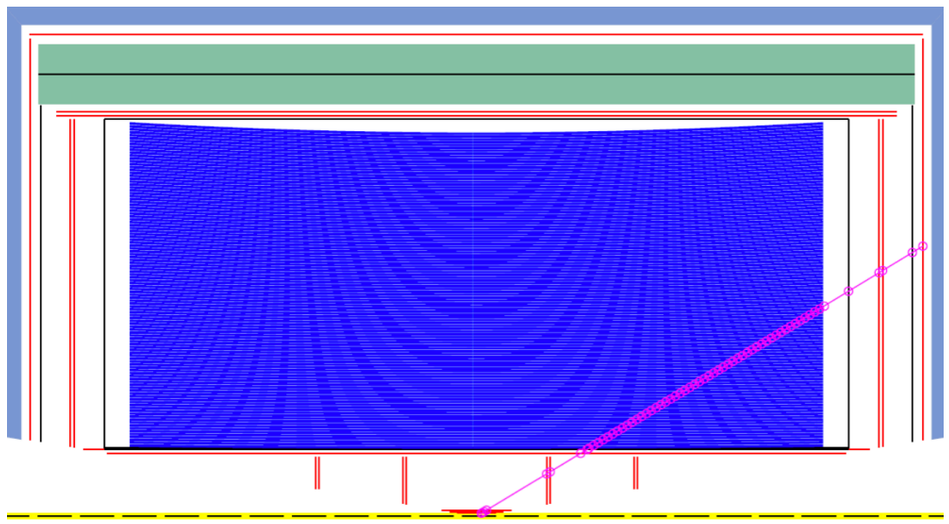


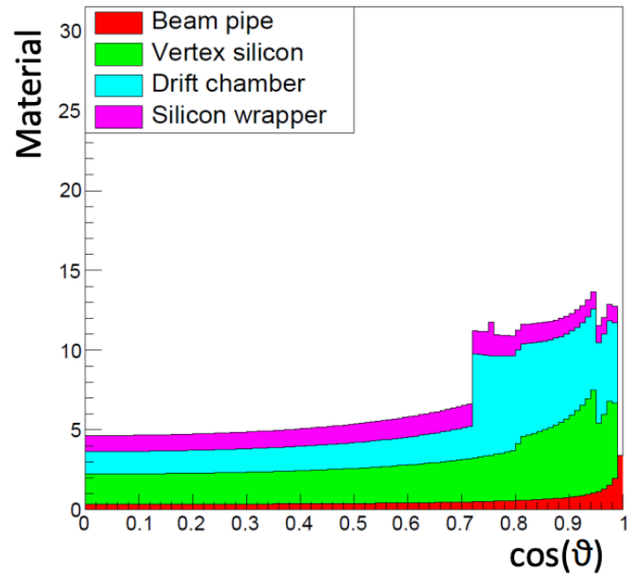


http://tklayout.web.cern.ch/

FastTrackCovariance

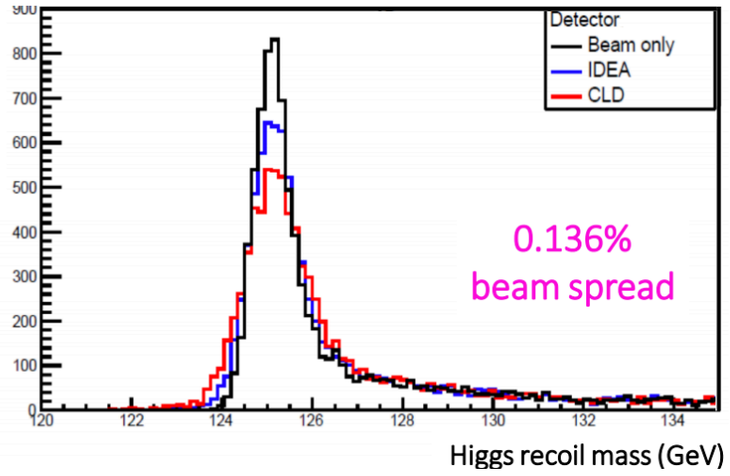
DELPHES
fast simulation





Track Smearing

- **Standalone C++ code**
- Simple tracker geometry implementation, **including material**
- computes **full covariance matrix** (depend on pT, θ)
- includes material multiple scattering
- computes smeared track (including off-diagonal terms)
- being included in Delphes



F. Bedeschi

more details here

Tracking

DELPHES
fast simulation

pattern recognition
track fitting

Full Simulation

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

TkLayout

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

parameterisation

(d_0, d_z, p, ctg θ, φ)

Delphes

FastTrackCovariance

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

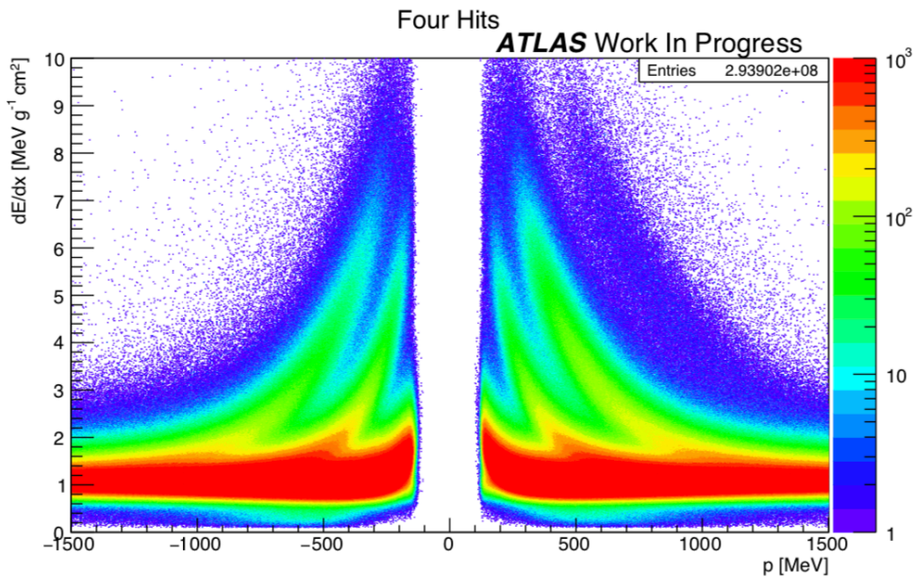
dEdx method (beta)

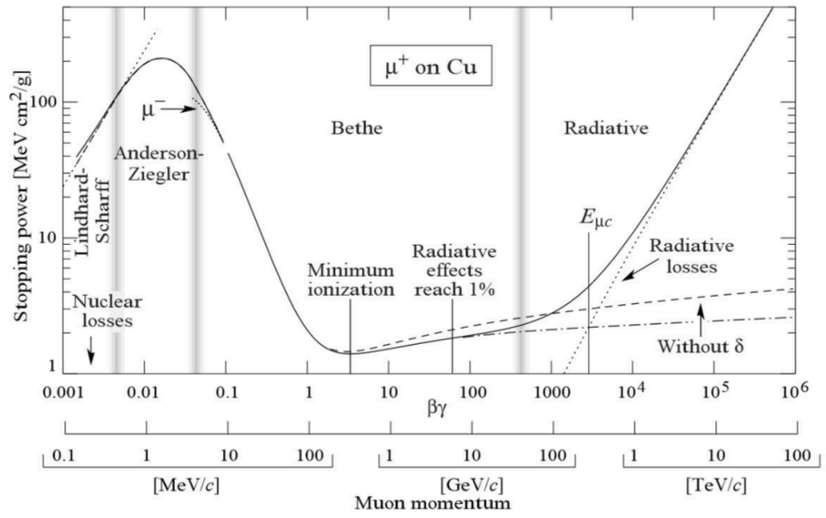
DELPHES
fast simulation

Algorithm:

For each track:

- Compute **path length L** (already done in Delphes):
  - **material properties**: e.g Silicon
  - **sensor thickness**: 100 um of Silicon
  - **fraction of active material** e.g. f = 1mm/1.5m
  - **resolution of charge measurement** in sensor: r in MeV/cm
  - **number of measurements** computed by the module as Nhits = f*L/thickness
  - specify fraction of measurements to be thrown away (truncated mean)
  - assume all charge collected
- Given material composition, particle velocity, etc… compute **ΔE_i from Landau distribution**
- Produce additional smearing with Gaussian resolution R to simulate finite resolution in charge collection
- Compute truncated mean





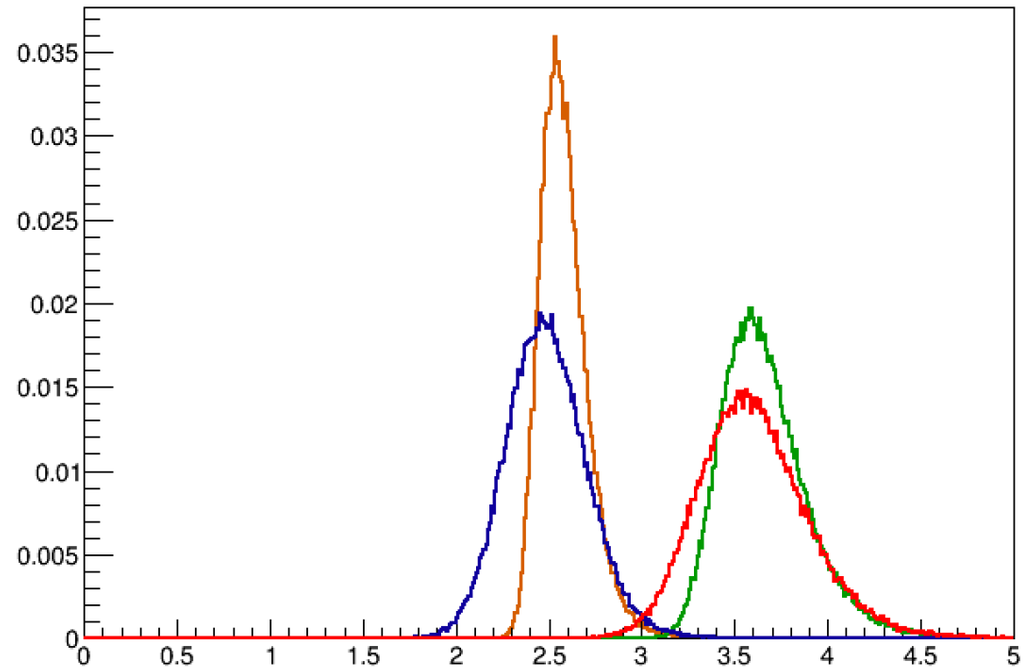
Degrees of freedom:

- **TruncatedMeanFraction**
- **Resolution**

dEdX in Silicon Tracker

DELPHES
fast simulation

```
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)

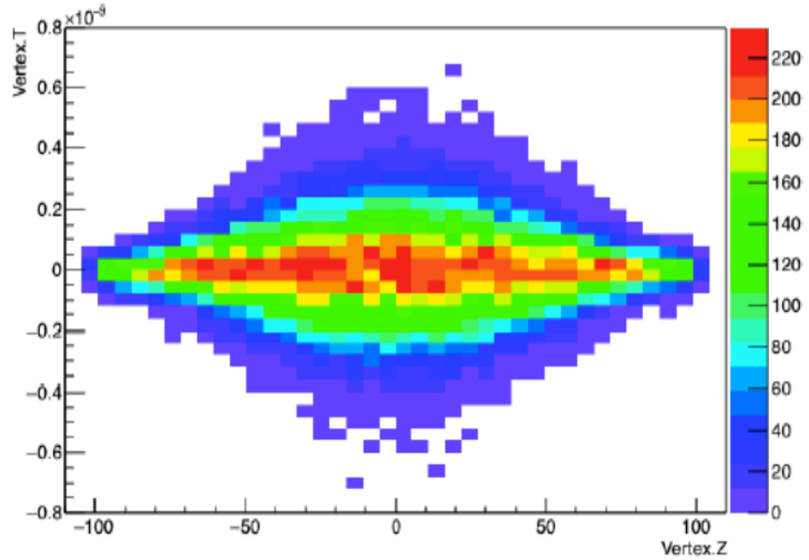
beta =0.75 (LanGauss)

Timing detectors

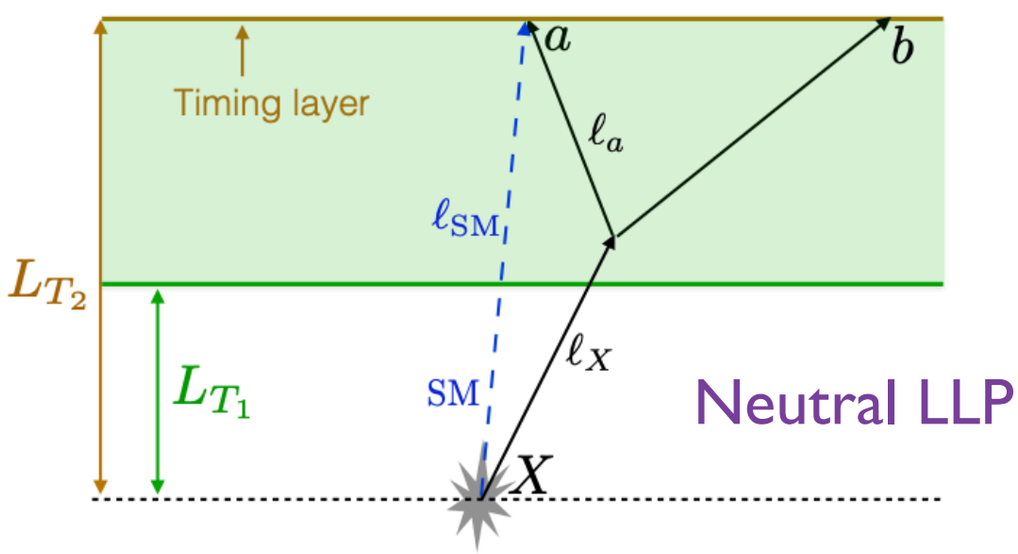
DELPHES
fast simulation

- 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 ALICE?

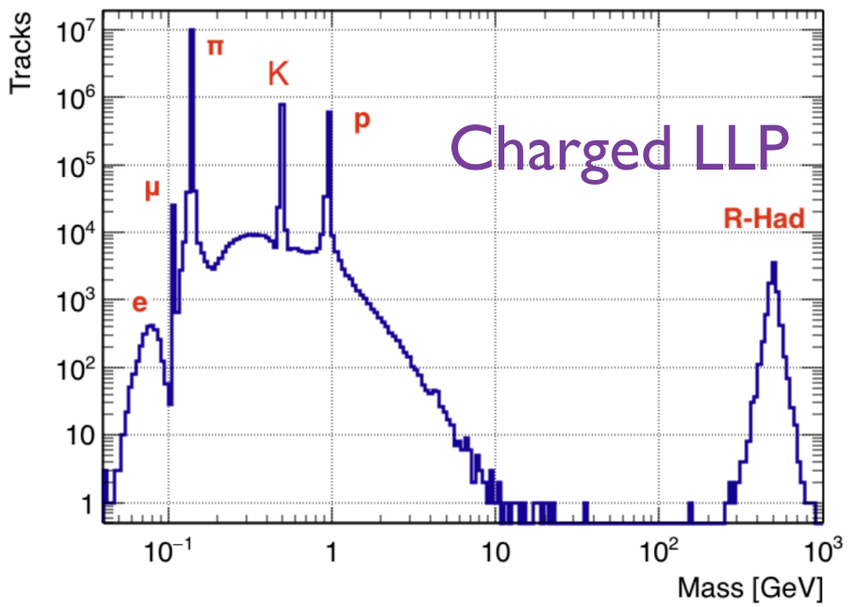
- Timing can be used to measure TOF, and hence for particle ID (either SM or BSM long lived particles)



$$t_{\mathrm{OF}} = \frac{L(p_T, \eta)}{cp}\sqrt{p^2 + m^2}.$$



Neutral LLP



Charged LLP

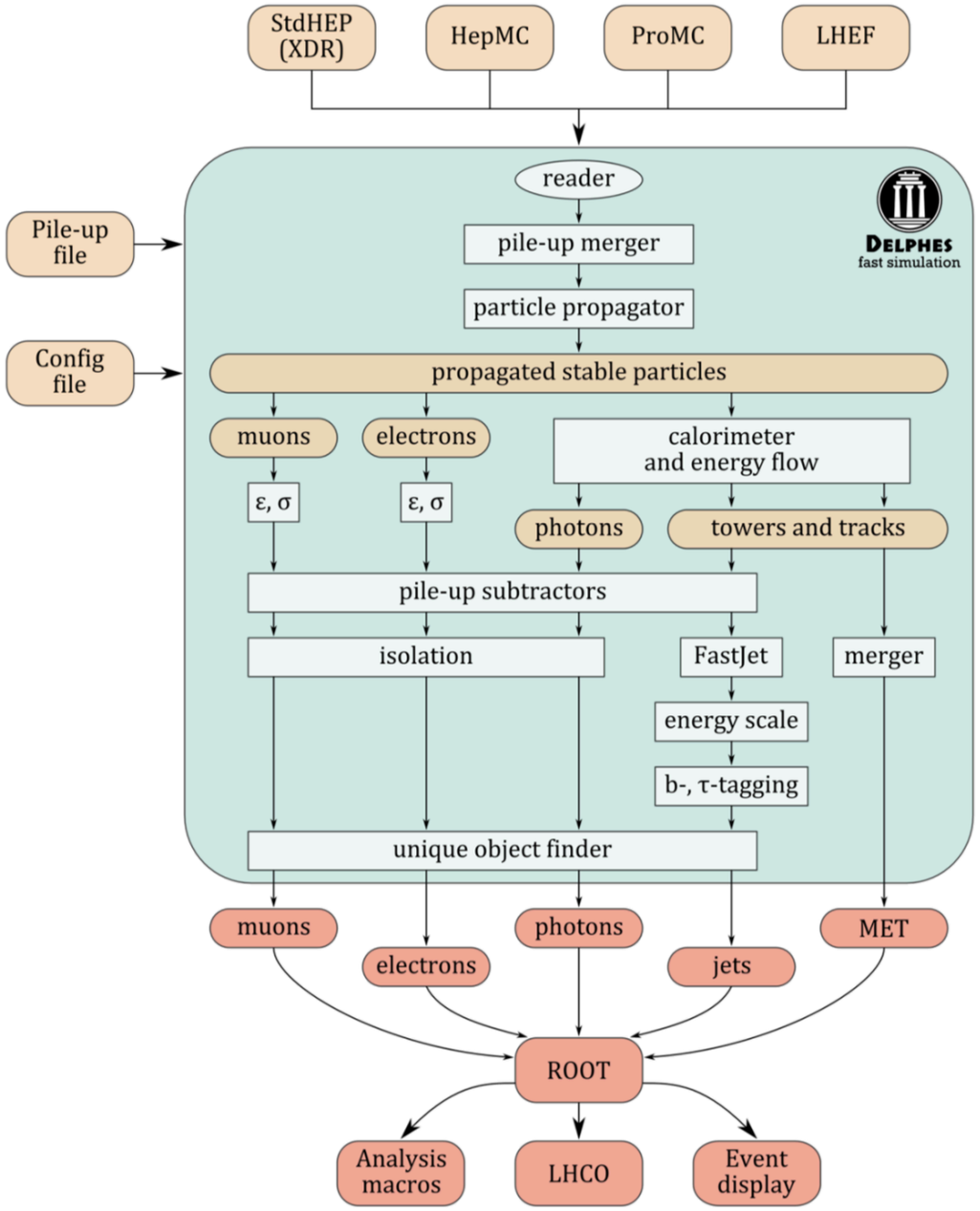
Practicalities

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)}
```

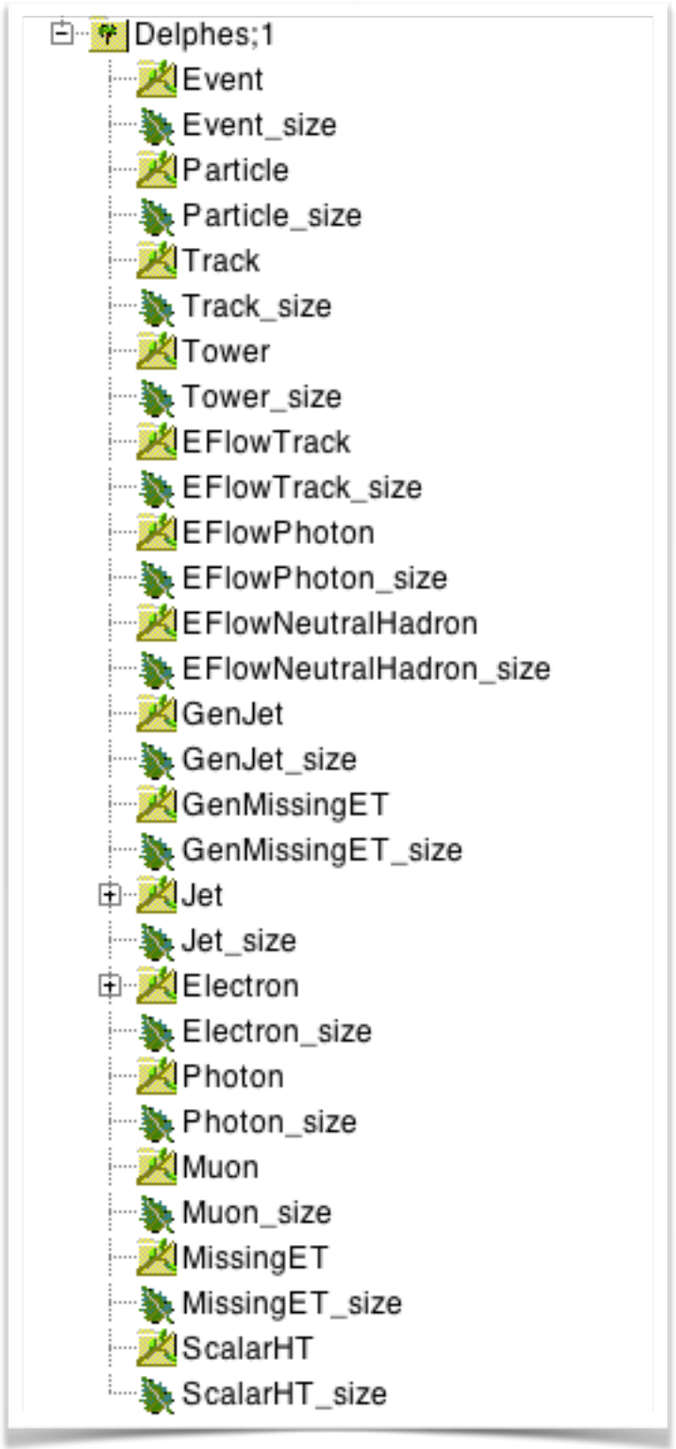
input(s) candidates

output(s) candidates

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
}
```



Conclusion

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

- Can be used and configured for:
  - quick **phenomenological** studies
  - as an **alternative for full-sim** if accurately tuned

- **Future plans** include:

  - More accurate description of tracking and material budget to allow for more detailed studies and prediction of HF tagging performance (**FastTracking Module**)
  - Secondary **(displaced) vertexing**
  - Improve **PID modelling capabilities**, would allow to extend the scope of BSM sensitivity studies:
    - **Timing/TOF**
    - **dEdx**

TUTORIAL

In this tutorial you will learn:

- how to produce a Delphes ROOT sample starting from an HepMC file
- how to navigate through the Delphes output
- how to analyse the Delphes output interactively and with a simple python script
- to understand the configuration file (detector card)
- how to change simple parameters such as reconstruction efficiency and detector resolution

For simplicity we will study simple Drell Yan and Z' (TeV) samples:

- $pp \rightarrow Z \rightarrow ee/\mu\mu$
- $pp \rightarrow Z' \rightarrow ee/\mu\mu$

TUTORIAL

Make sure your have properly installed ROOT, Pythia8 and Delphes
or have installed the Virtual Machine:

https://twiki.cern.ch/twiki/bin/view/VBSCan/PREFIT20#How_to_setup_the_tools_code_AN2

Tutorial:

https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook/Tutorials/Prefit

I will be stay connected and Alessia Saggio will be helping in live (thanks!)