Delphes Simulation

Michele Selvaggi

CERN

School on Future Colliders — 19/09/2018 — Pisa

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

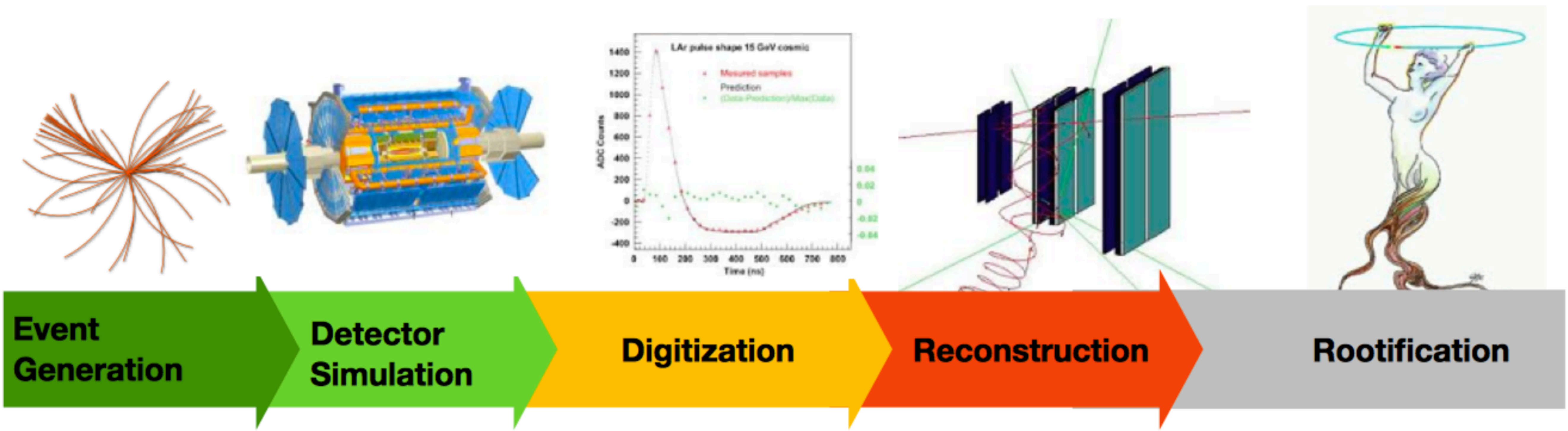
10-10^2 s/ev

- 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

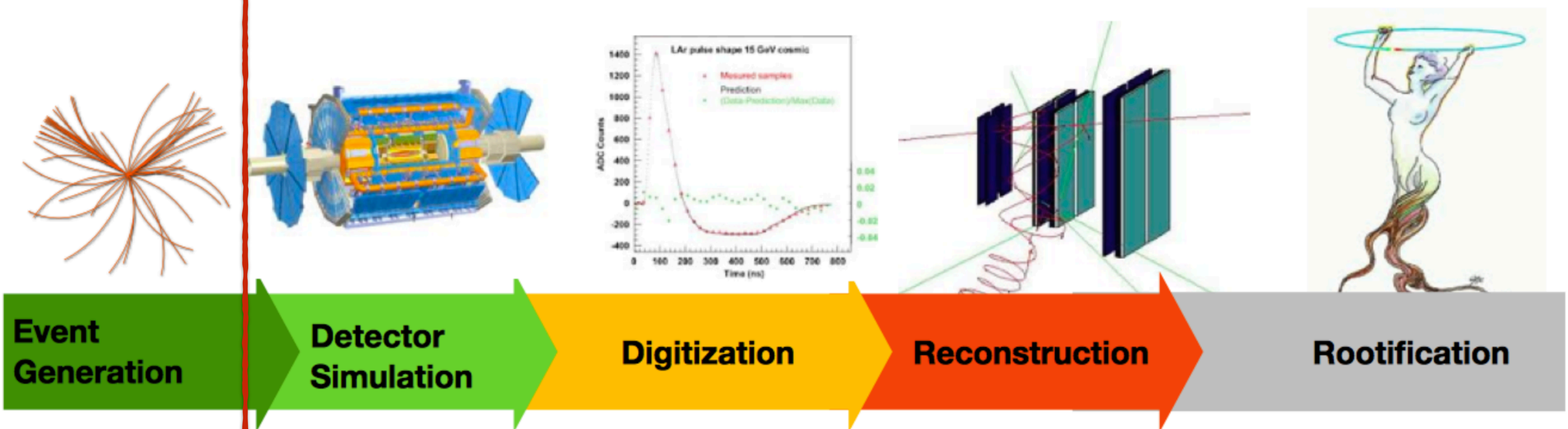
- Ultra Fast (ATOM, TurboSim):
  - from parton to detector object (smearing/lookup tables)

MonteCarlo EvGen



courtesy of Andreas Salzburger

MonteCarlo EvGen



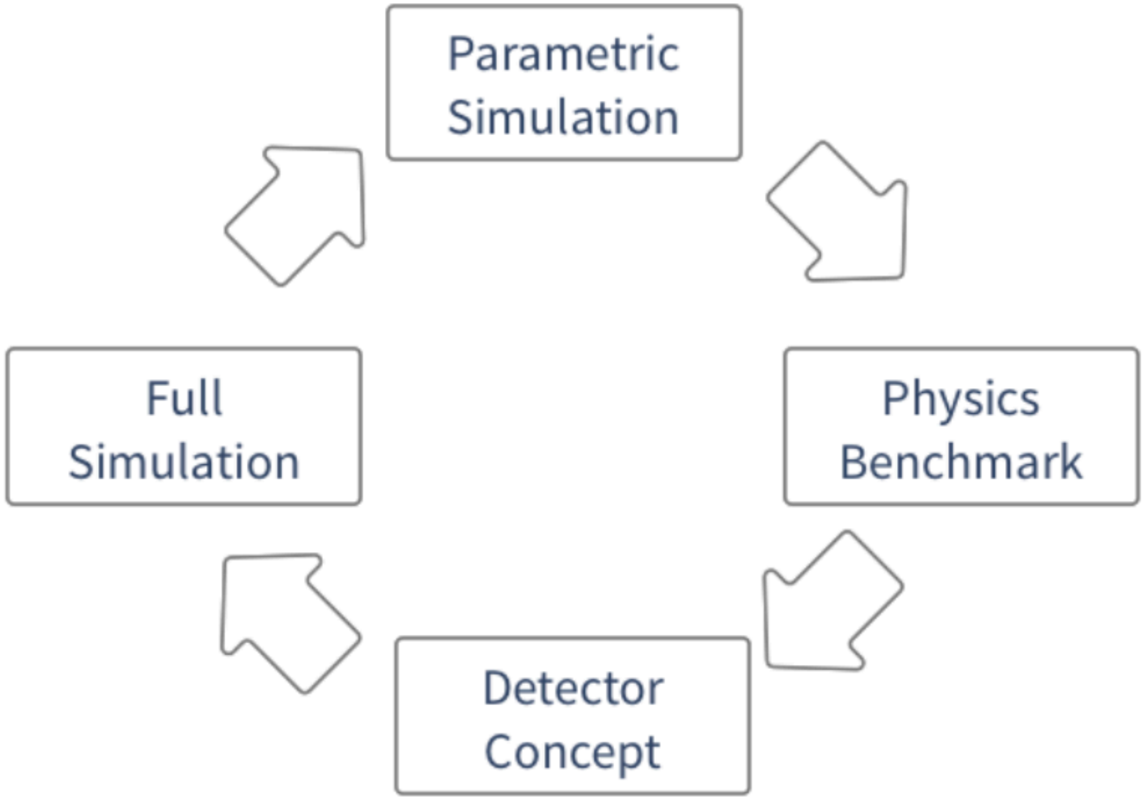
Delphes FastSim

courtesy of Andreas Salzburger

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

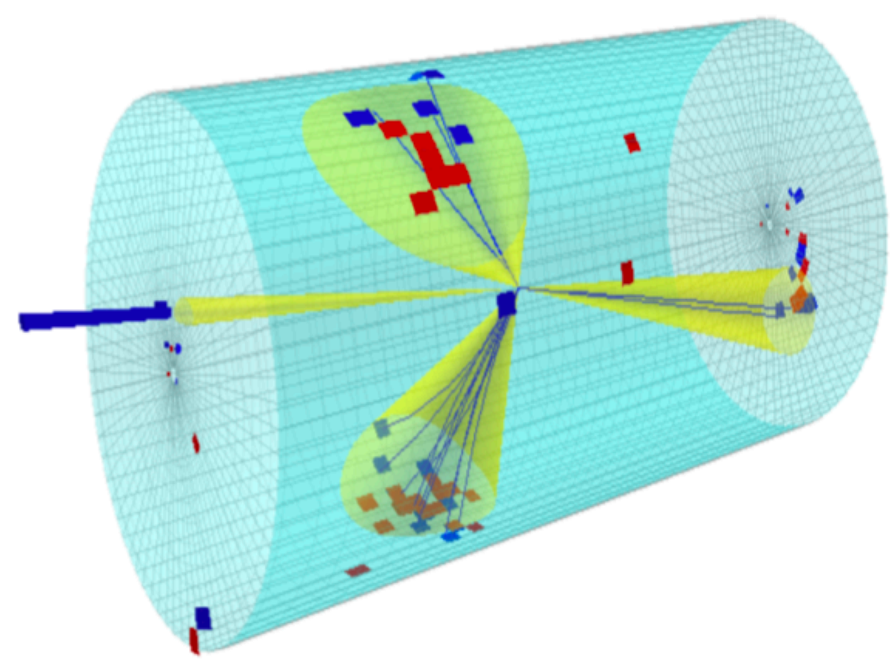
github: github.com/delphes
website: cp3.irmp.ucl.ac.be/projects/delphes

Delphes in a nutshell

- **Delphes** is modular framework that simulates the response of a **multipurpose 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



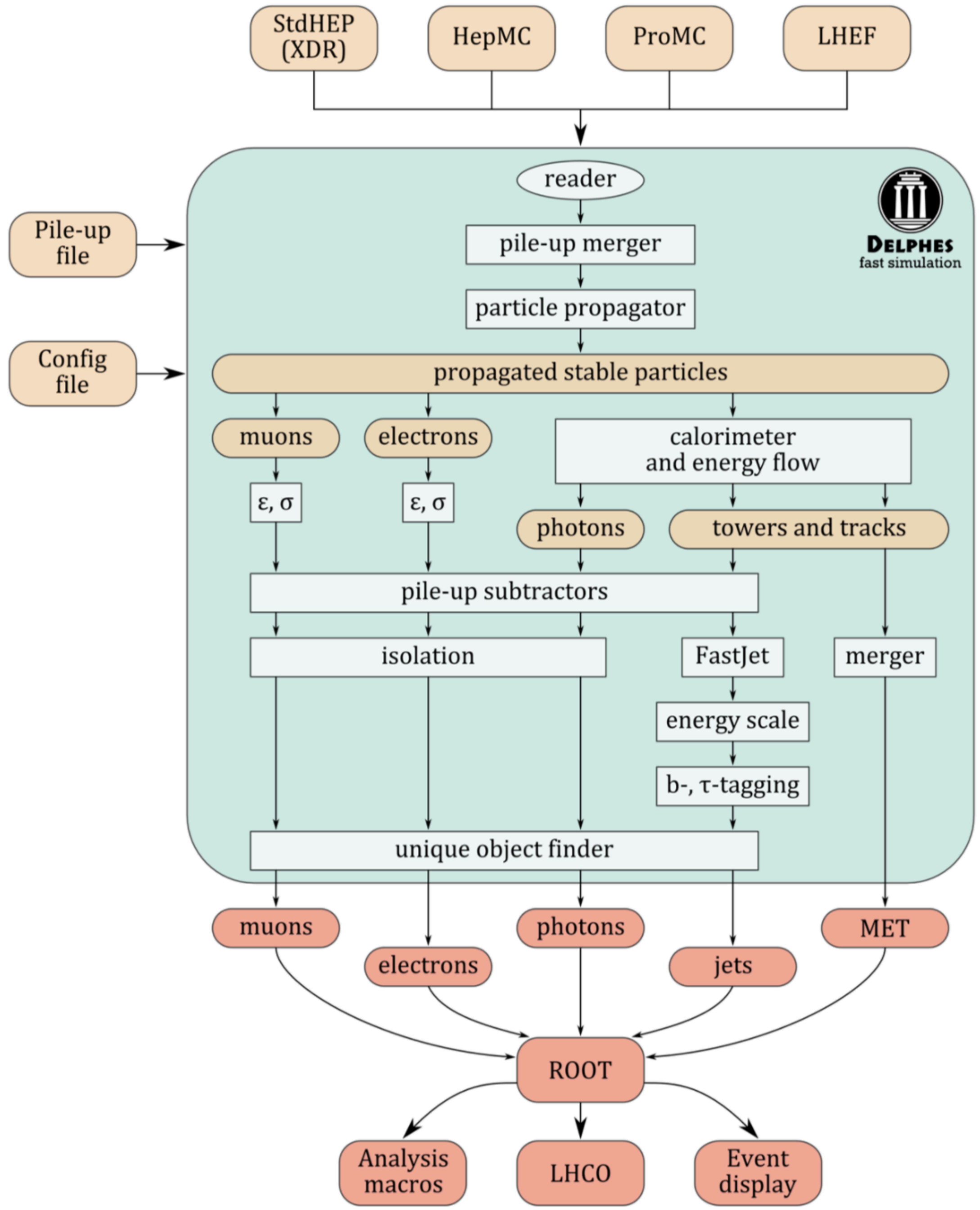
- designed to deal with hadronic environment
- well-suited also for e+e- studies
- detector cards for: CMS (current/PhaseII) - ATLAS - LHCb - FCC-hh - ILD - CEPC

Modularity

- The modular system allows the user to configure a detector a 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

DELPHES fast simulation

```
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

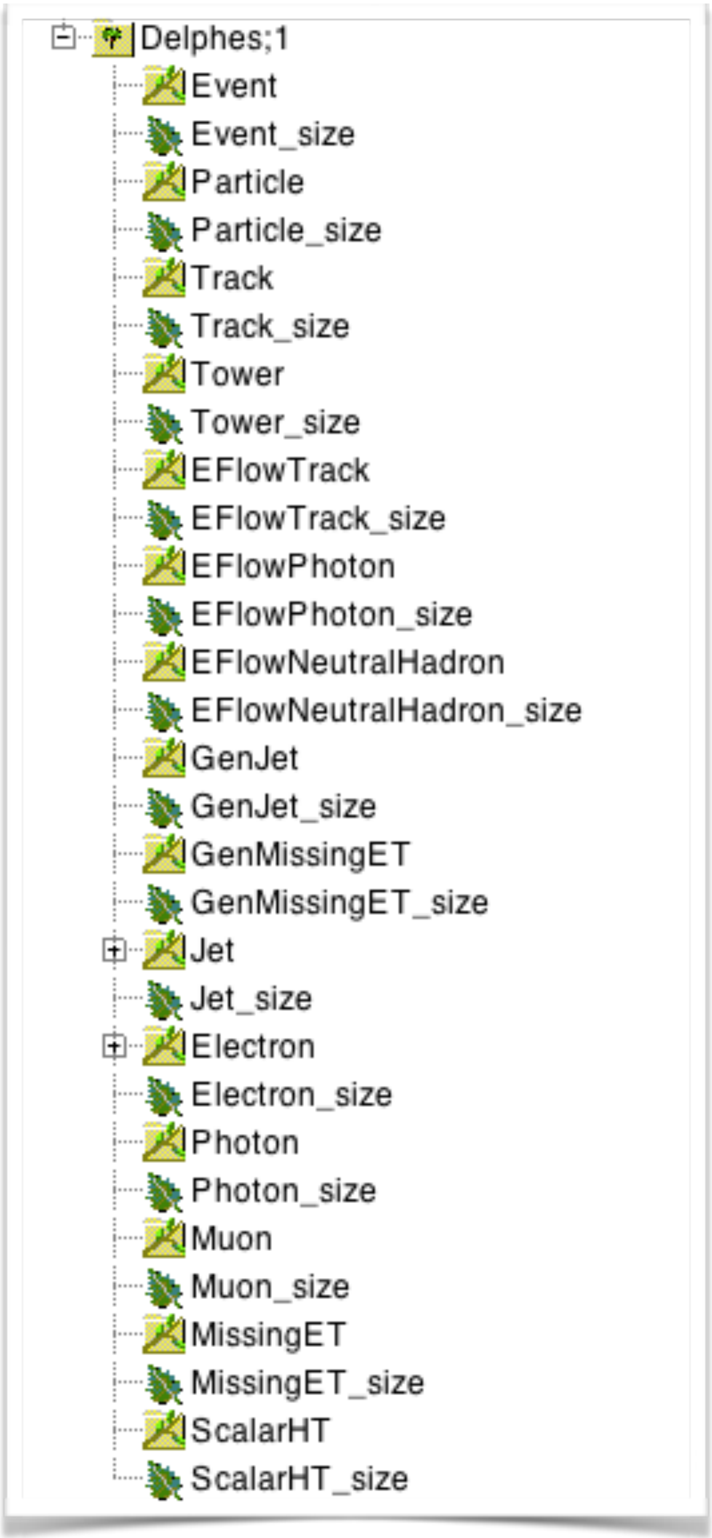
output(s) candidates

13

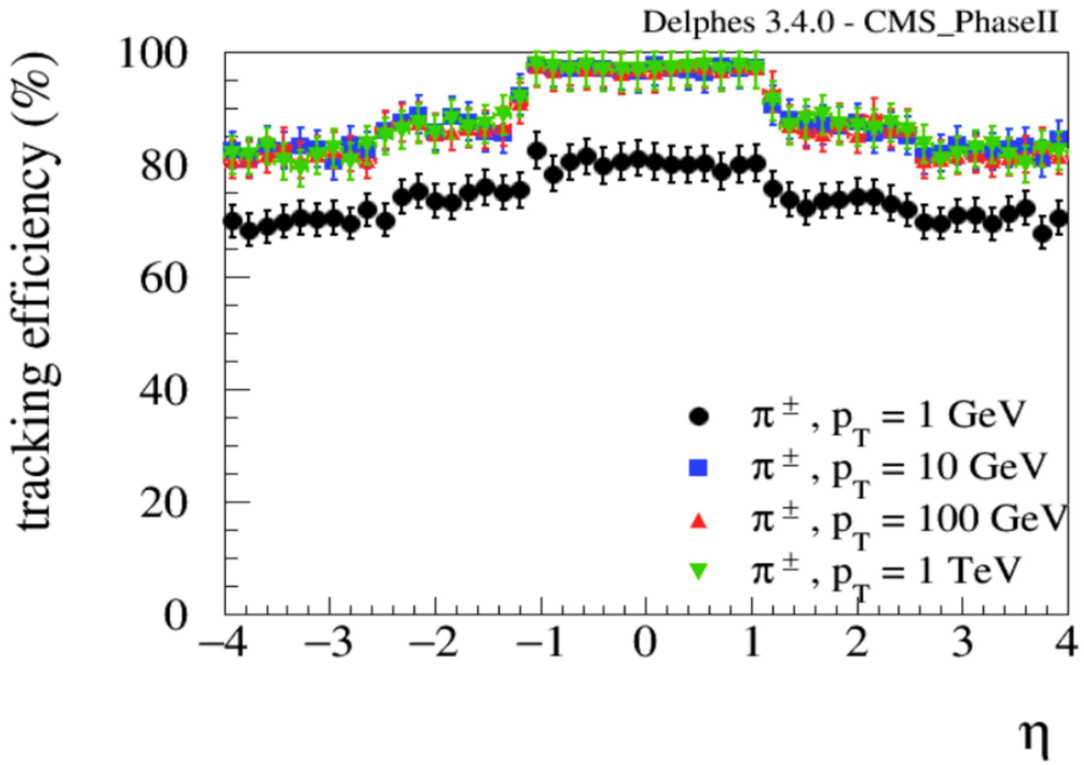
Configuration file

Output collections are configured in the TreeWriter module

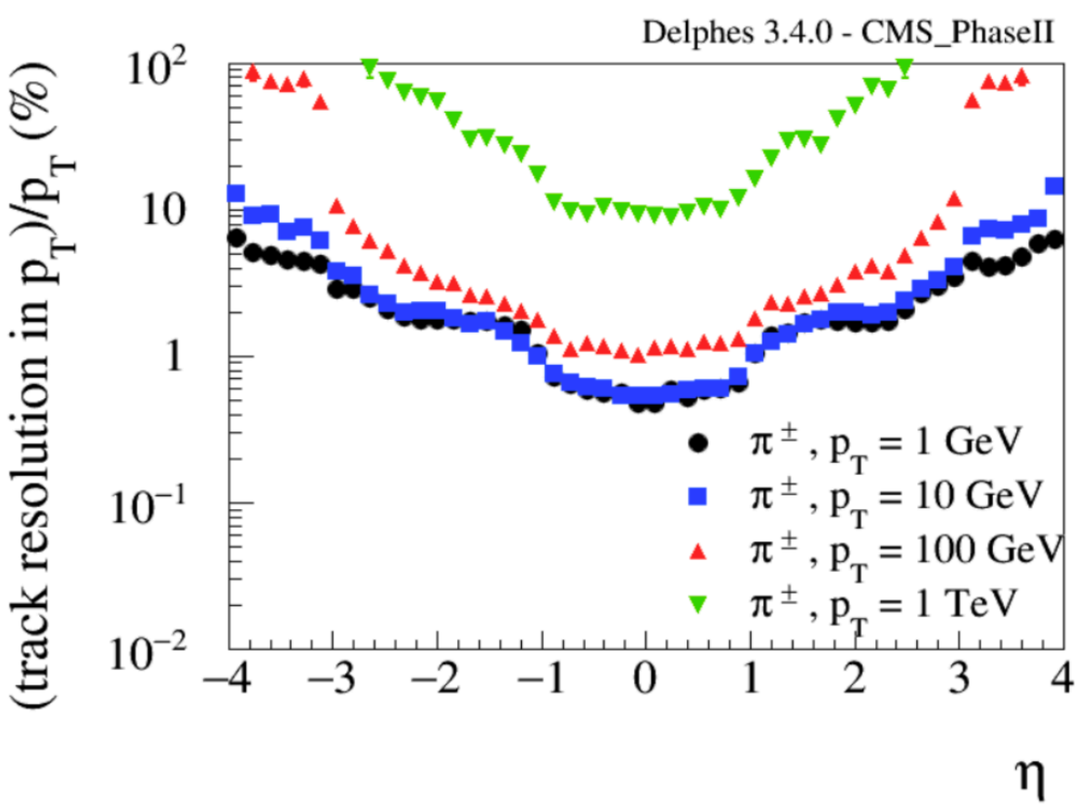
```
module TreeWriter TreeWriter {
# add Branch InputArray BranchName BranchClass
  add Branch Delphes/allParticles Particle GenParticle

  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
}
```



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) + \
```

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

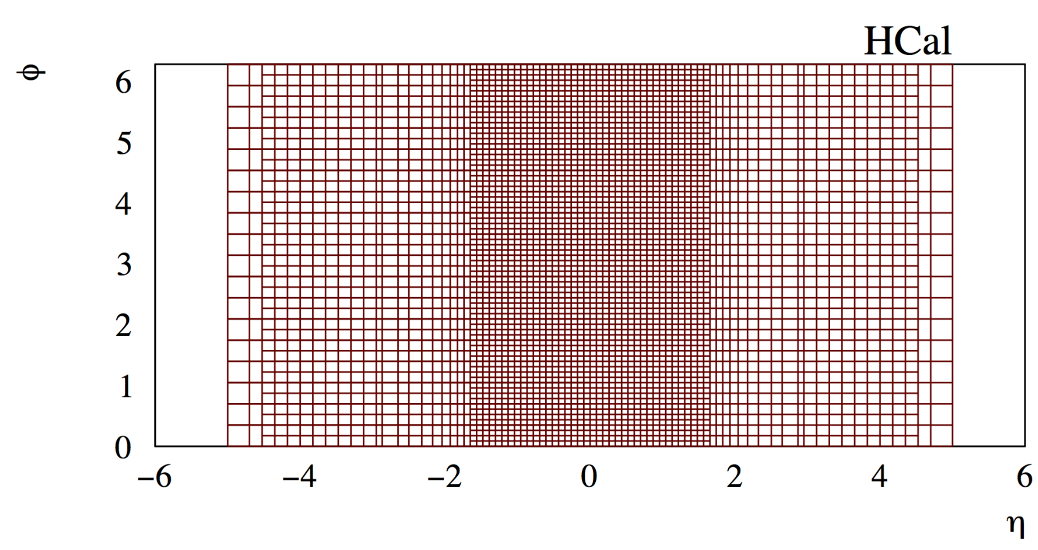
Calorimetry

- 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 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

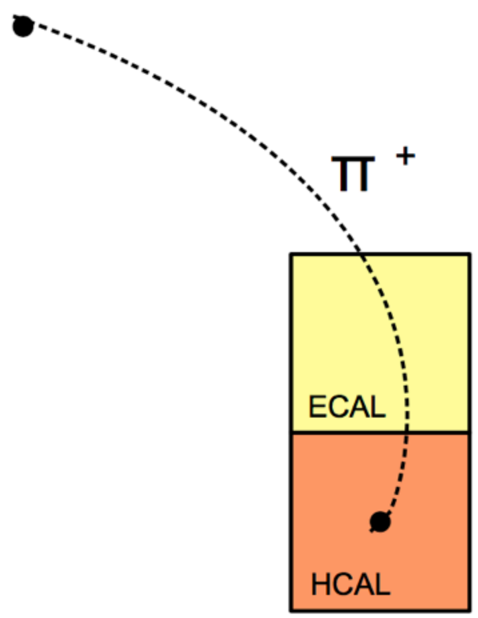
$$\left(\frac{\sigma}{E}\right)^2 = \left(\frac{S(\eta)}{\sqrt{E}}\right)^2 + \left(\frac{N(\eta)}{E}\right)^2 + C(\eta)^2$$



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



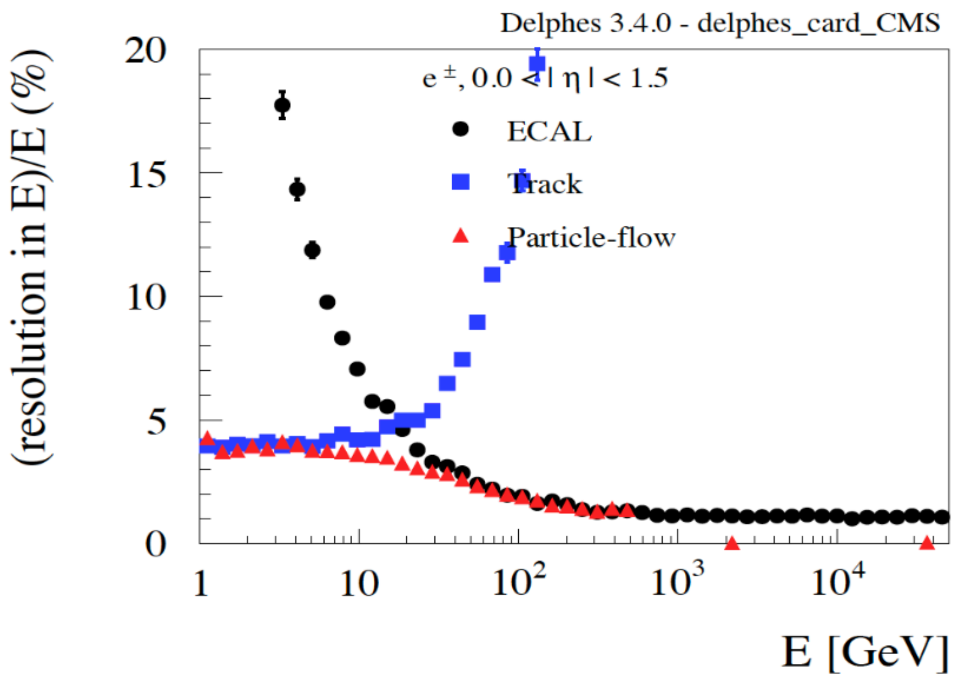
Algorithm:

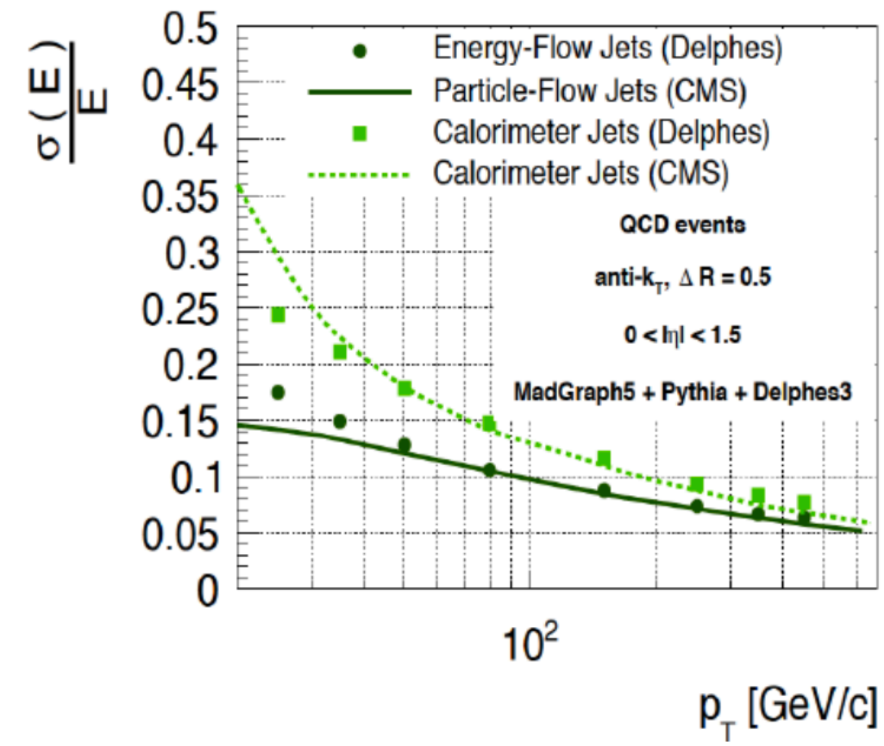
- If $E_{neutral} / \sqrt{(\sigma_{calo}^2 + \sigma_{trk}^2)} > S$:
  → create PF-neutral particle + PF-track

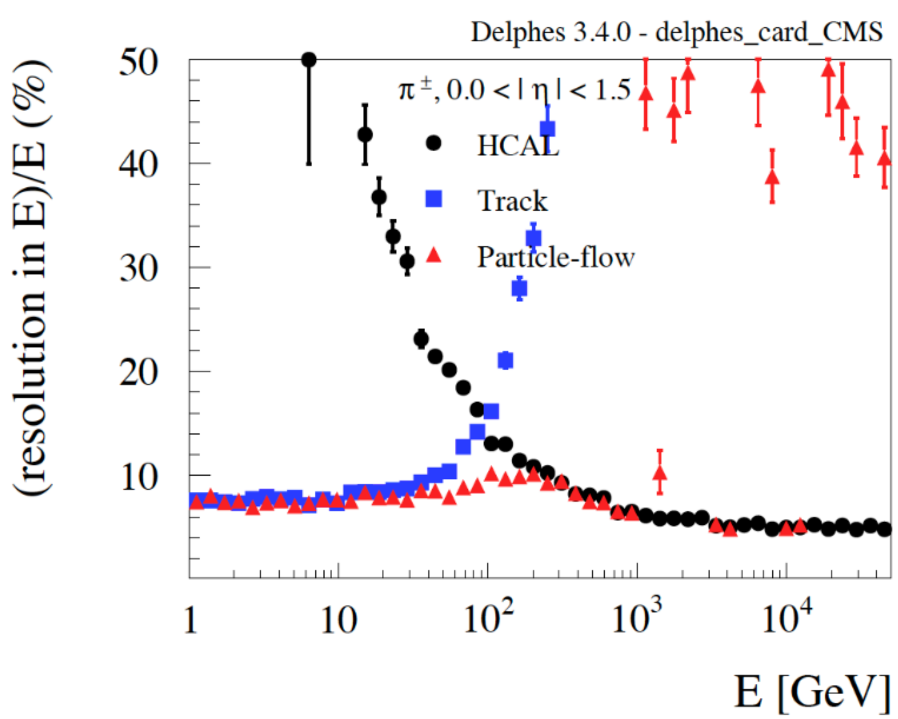
- Else:
  create PF-track and **rescale momentum** by combined calo+trk estimate

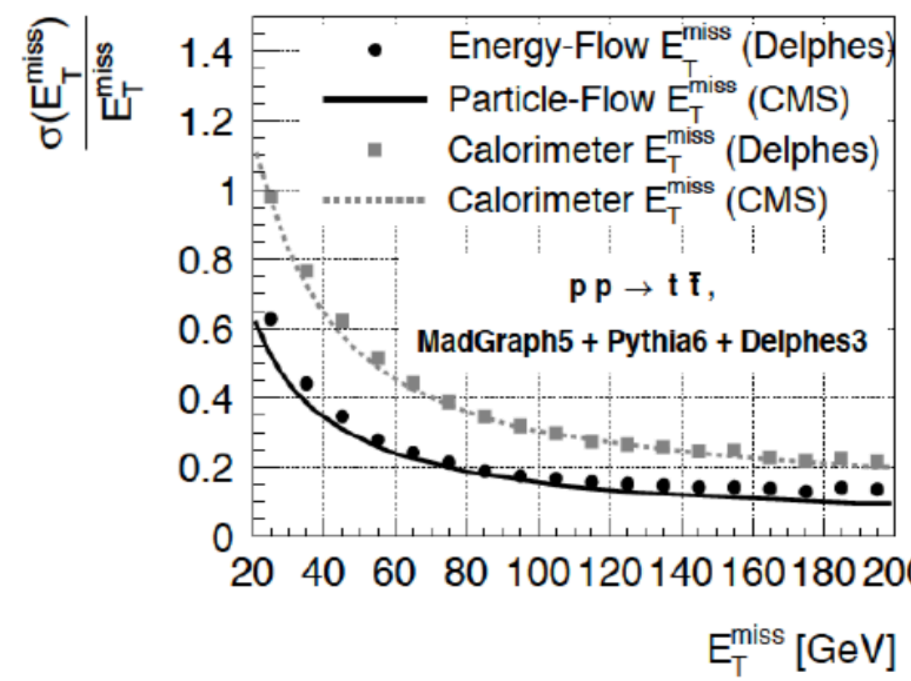
  - EM (had) deposit 100% in ECAL (HCAL)
  - No propagation in calorimeters
  - No clustering (topological) clustering, exploiting pre-defined grid

Particle-Flow





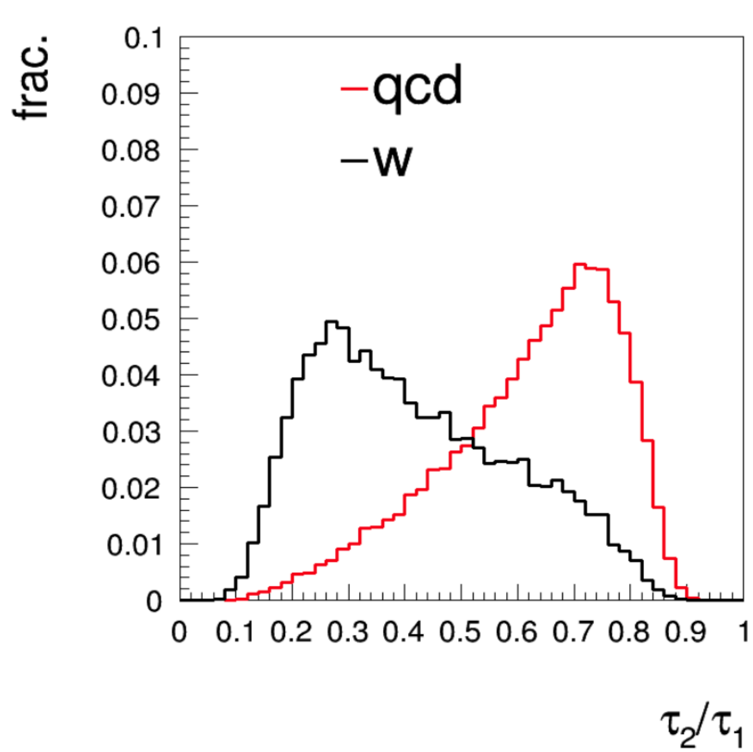




Jets and Substructure

- 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)

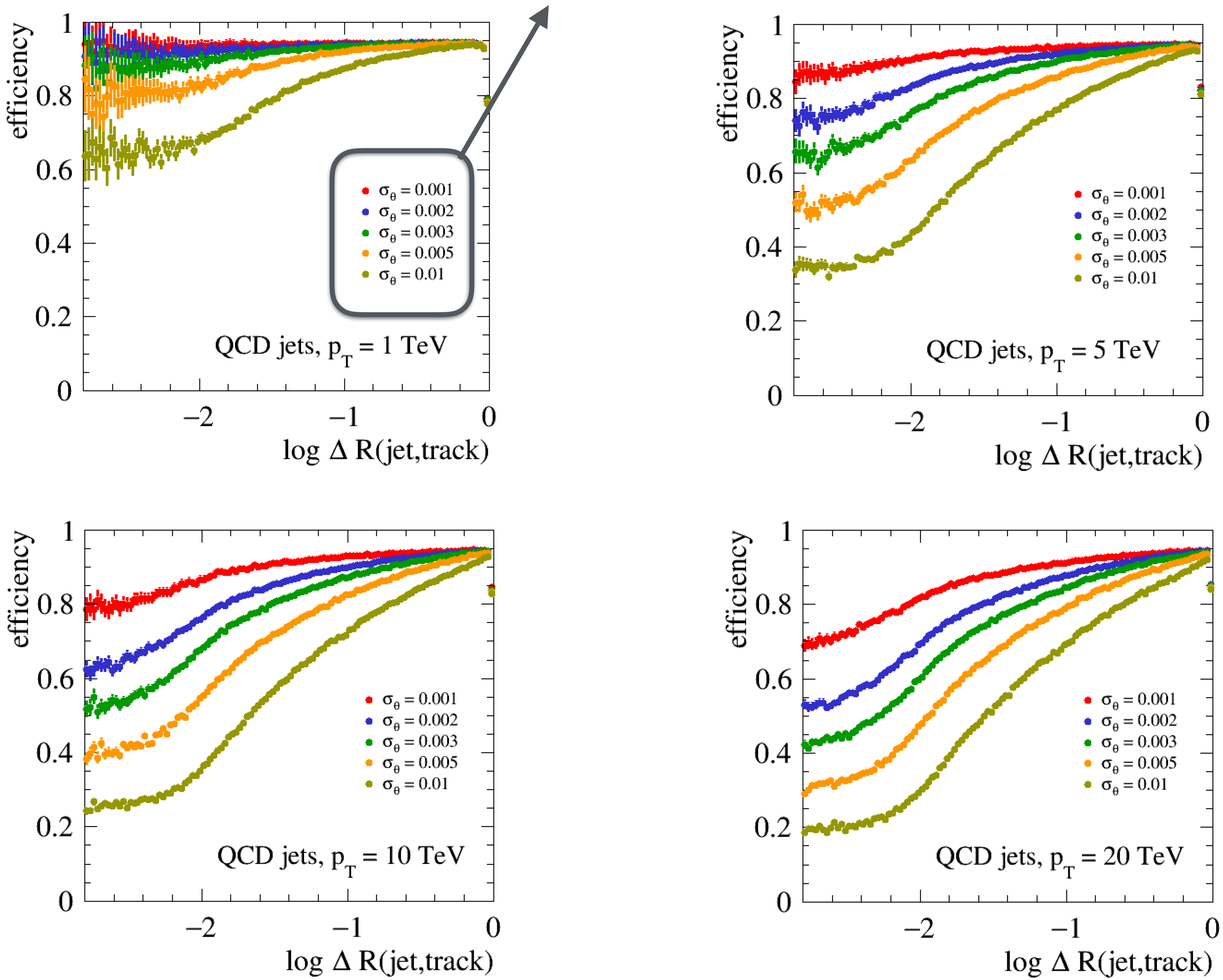
```
###########
# Jet finder
###########

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
}
```



Tracking in Dense environment (NEW!)

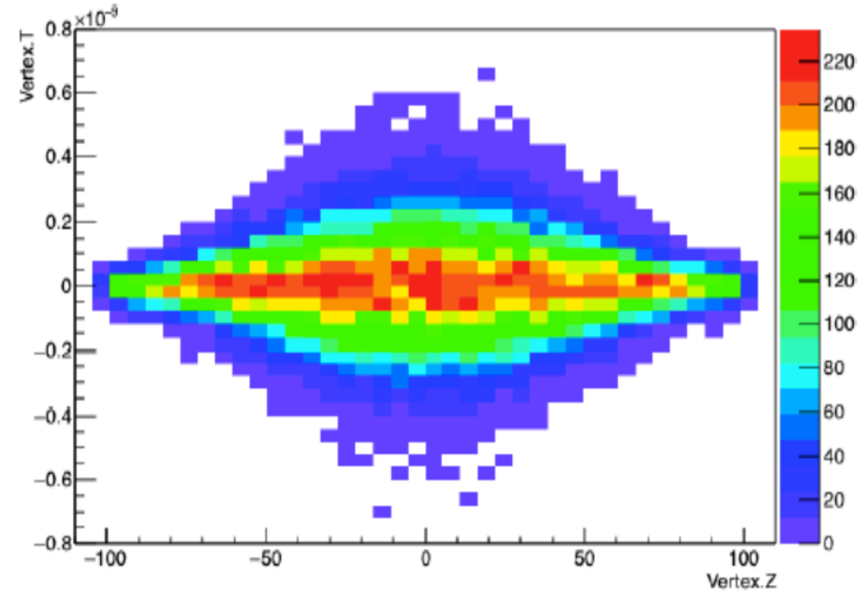
DELPHES fast simulation

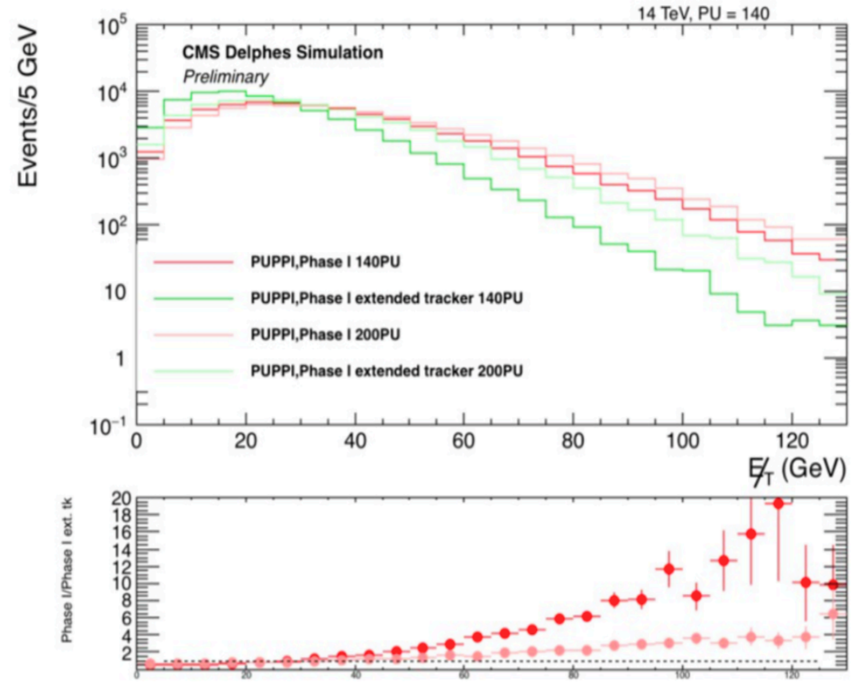
Instrinsic tracking angular resolution



Pile-up Simulation and Subtraction

- Pile-up can be mixed with hard event, with f(z,t) profile
- Charged Hadron Subtraction performed according to smearing longitudinal impact parameter
- Neutral Subtraction performed either with GridMedianEstimator, SoftKiller (FastJet) or PUPPI



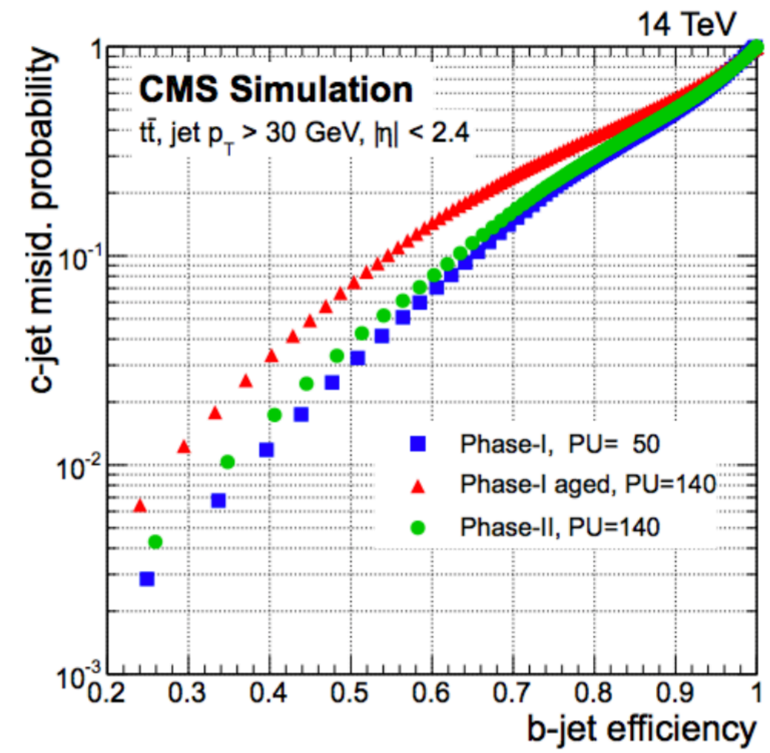


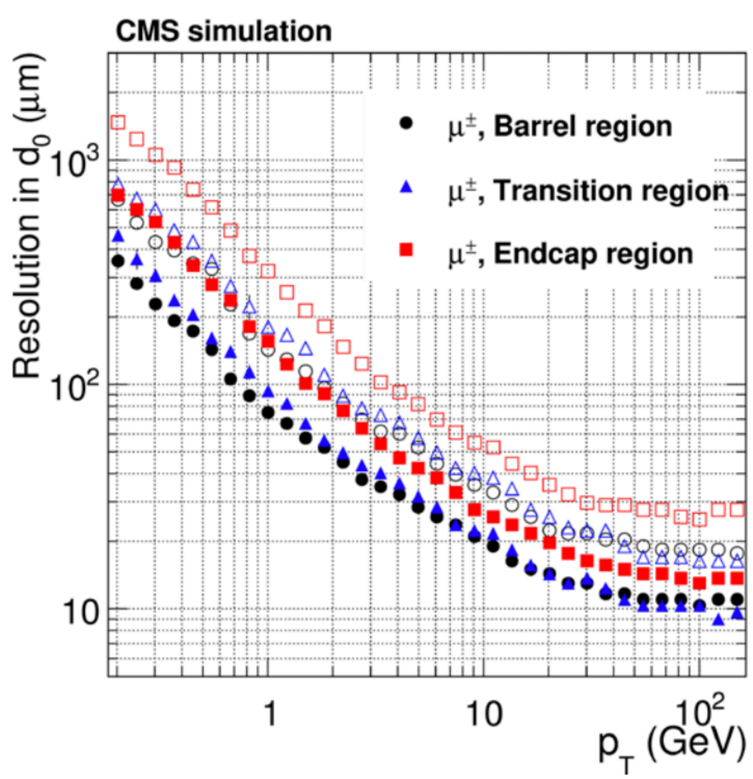
Heavy flavour Flavor Tagging

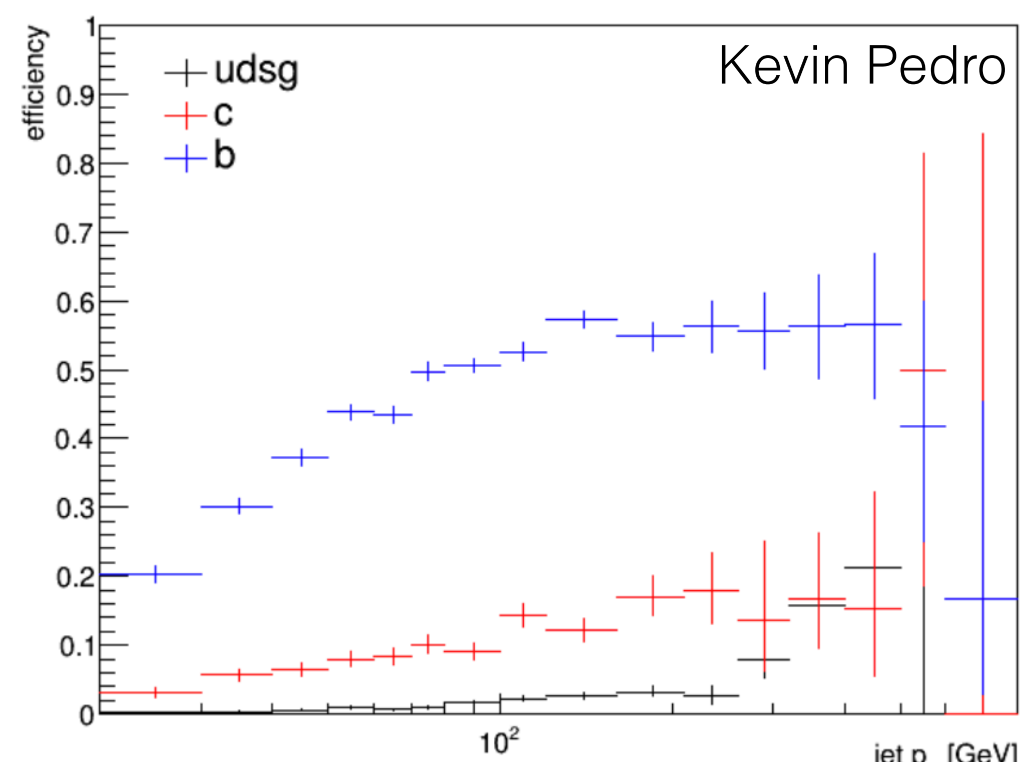
DELPHES fast simulation

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

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







Conclusion

- Delphes provides a **simple, highly modular framework** for performing fast detector simulation
- **Integrated** in MG5 suite and in the FCCSW framework
- Includes:
  - efficiency/ identification/ fake-rate maps
  - Tracking/Calorimeter **smearing** and Particle-Flow
  - Jet clustering (with FastJet) and jet substructure
  - **pile-up** simulation and modern PU subtraction techniques
- Can be used and configured for:
  - quick **phenomenological** studies
  - as an **alternative for full-sim** if accurately tuned

TUTORIAL

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

Pre-requisites

Make sure your have properly installed the VM and followed all instructions:

https://indico.cern.ch/event/669093/attachments/1615913/2770286/TutorialSchoolPisav2.pdf

Tutorial:

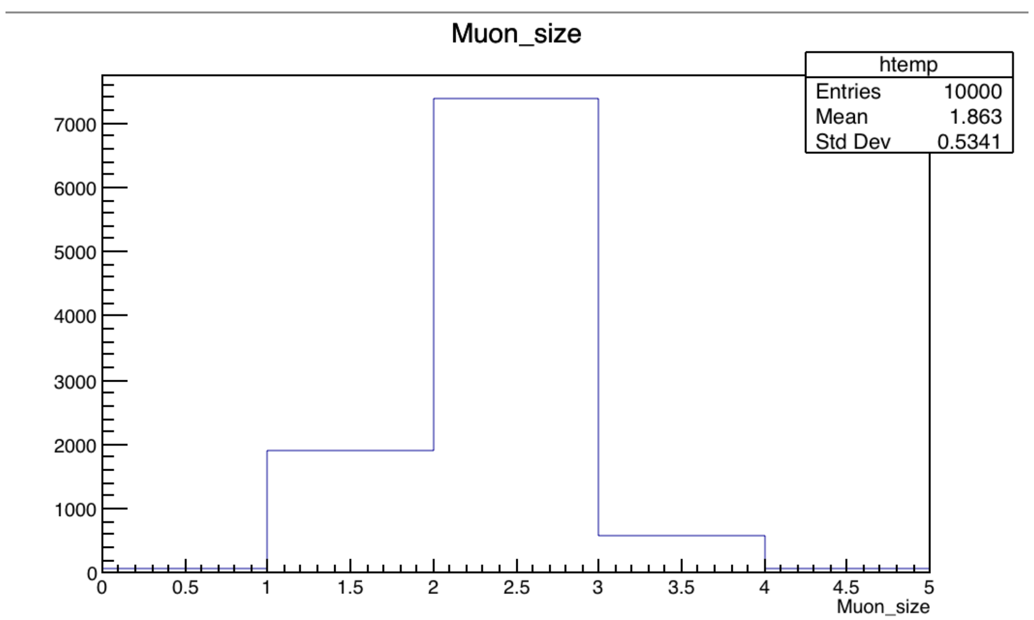
https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook/Tutorials/Pisa

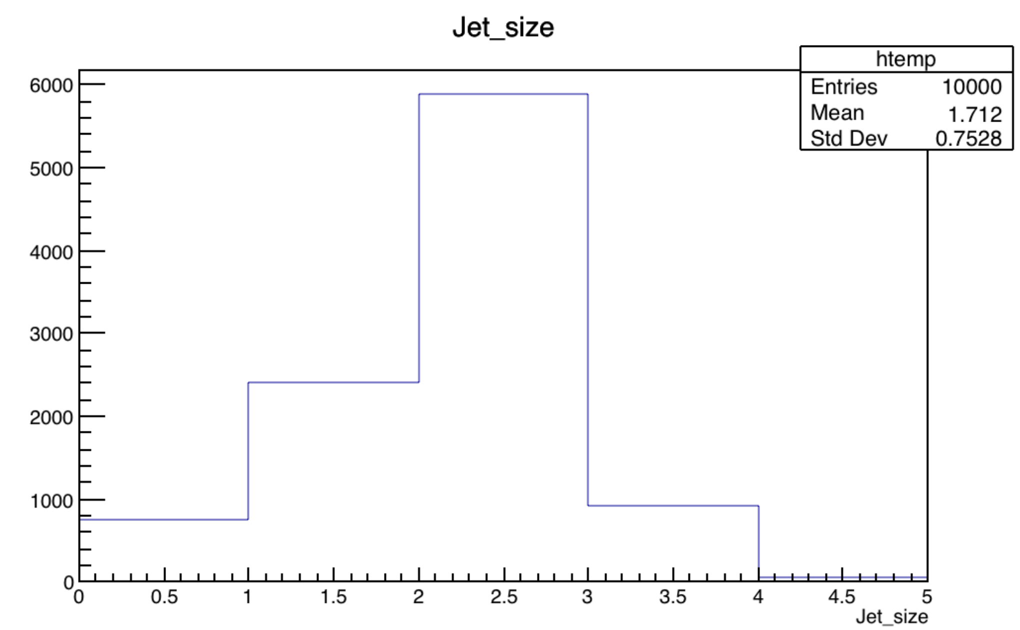
Outline

In this tutorial you will learn how to:

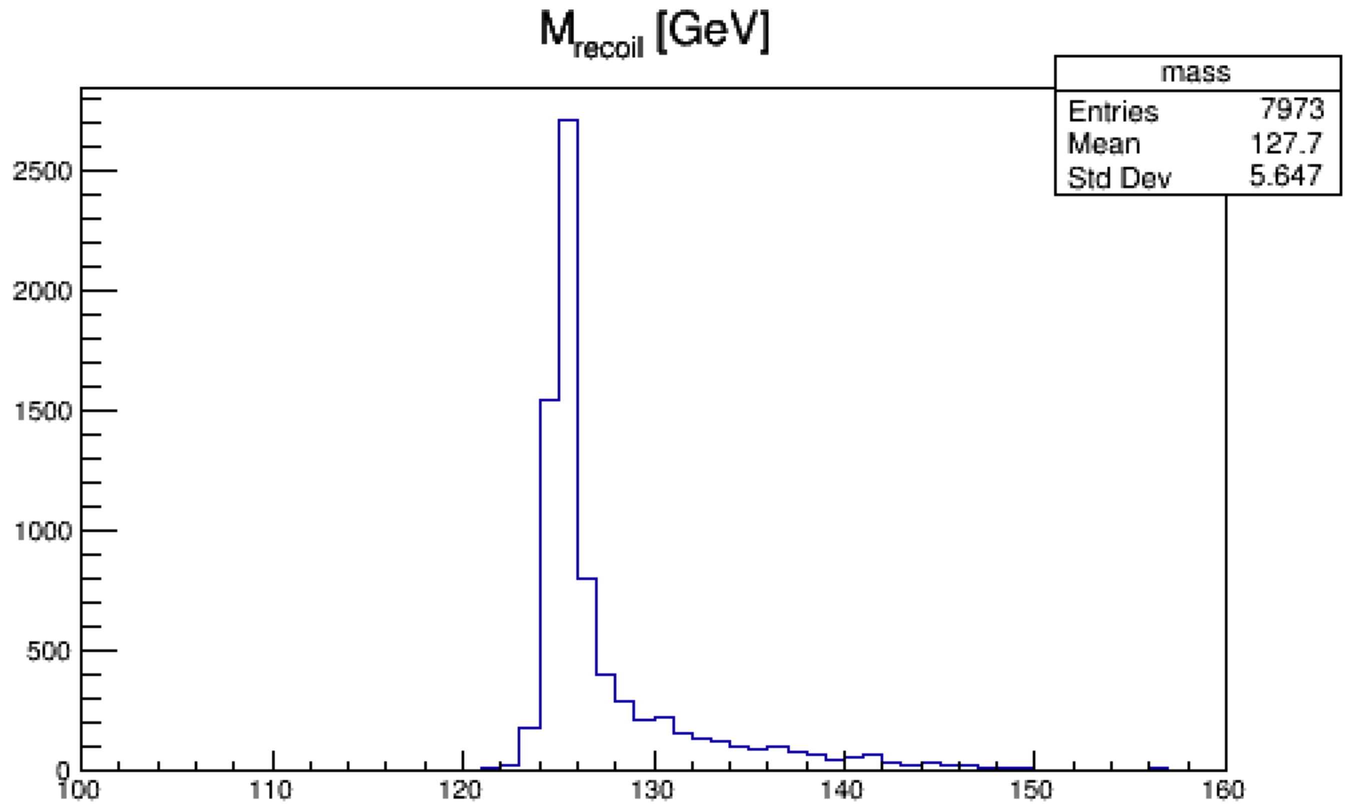
- interface Delphes with the Pythia8 event generator
- configure and generate events with Pythia8 + Delphes
- navigate through the Delphes output
- analyse the Delphes events with an analysis macro
- configure a detector card

Plots (I)


Muon_size
htemp
Entries 10000
Mean 1.863
Std Dev 0.5341
7000
6000
5000
4000
3000
2000
1000
0
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
Muon_size

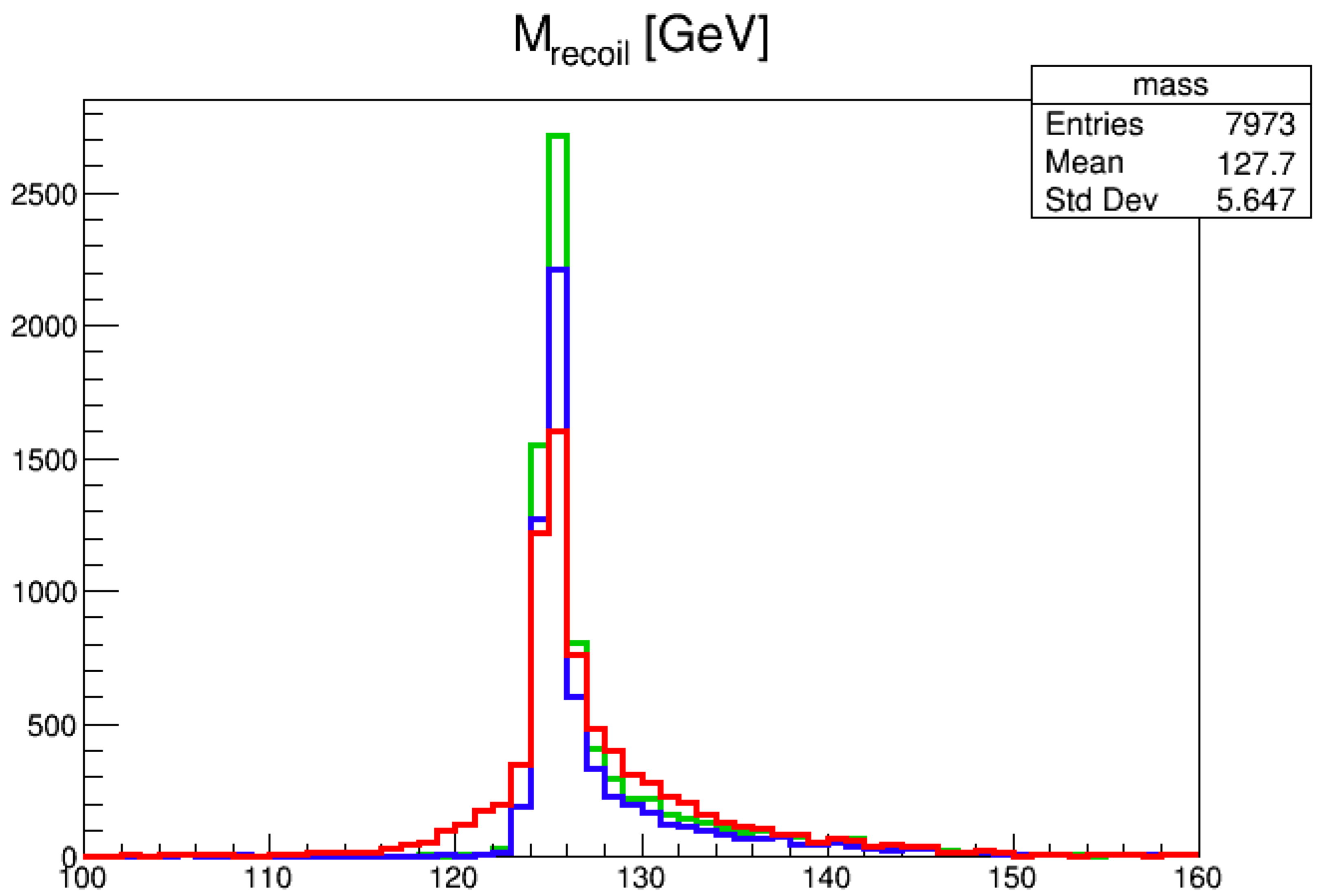

Jet_size
htemp
Entries 10000
Mean 1.712
Std Dev 0.7528
6000
5000
4000
3000
2000
1000
0
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
Jet_size

Plots (II)



M_{recoil} [GeV]
mass
Entries 7973
Mean 127.7
Std Dev 5.647
2500
2000
1500
1000
500
0
100
110
120
130
140
150
160

Plots (III)



M$_{recoil}$ [GeV]
mass
Entries 7973
Mean 127.7
Std Dev 5.647
2500
2000
1500
1000
500
0
100
110
120
130
140
150
160