DELPHES
fast simulation

Alexandre Mertens, for the Delphes Team

Université catholique de Louvain (UCL)
Center for Particle Physics and Phenomenology (CP3)

JHEP 02 (2014) 057

ACAT Workshop
1st September 2014

Detector simulation

- Full simulation (GEANT):
  - **simulates** particle-matter interactions in very fine-grained steps → 10-100 s /ev
- Experiment Fast simulation (ATLAS, CMS ...):
  - **simplifies** by simulating most interactions, but in a point-like way (i.e. once per detector layer or block)
  - mixes G4, parametric, libraries → 1 s /ev
- Parametric simulation:

Delphes, PGS**:**

  - **parameterize** detector response, reconstruct complex objects → 10 ms /ev

Overview

- The Delphes Project
- The Particle Flow Emulation
- PU Implementation
- Other new features
- Conclusion

The Delphes Project

The Delphes project: A bit of history

- Delphes project started back in 2007 at UCL as a side project to allow quick feasibility studies

- Since 2009, its development is **community-based**
  - **ticketing system** for improvement and bug-fixes
    → user proposed patches
  - **Quality control** and **core development** is done at the UCL

- In 2013, **DELPHES 3** was released:
  - modular software
  - new features
  - also included in MadGraph suite

- **Widely** tested and used by the community (pheno, Snowmass, CMS ECFA efforts, etc...)

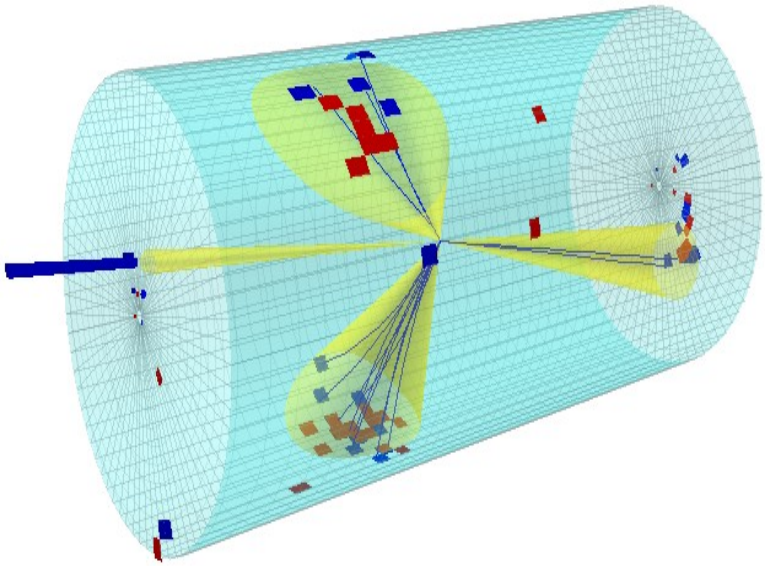
- Website and manual: https://cp3.irmp.ucl.ac.be/projects/delphes

- Paper: JHEP 02 (2014) 057

The Delphes project: Delphes in a nutshell

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

- **Includes**:
  - pile-up
  - charged particle **propagation** in magnetic field
  - electromagnetic and hadronic **calorimeters**
  - **muon** system



- **Provides**:
  - leptons (electrons and muons)
  - photons
  - jets and missing transverse energy (particle-flow)
  - taus and b's

The Delphes project: I/O and configurations

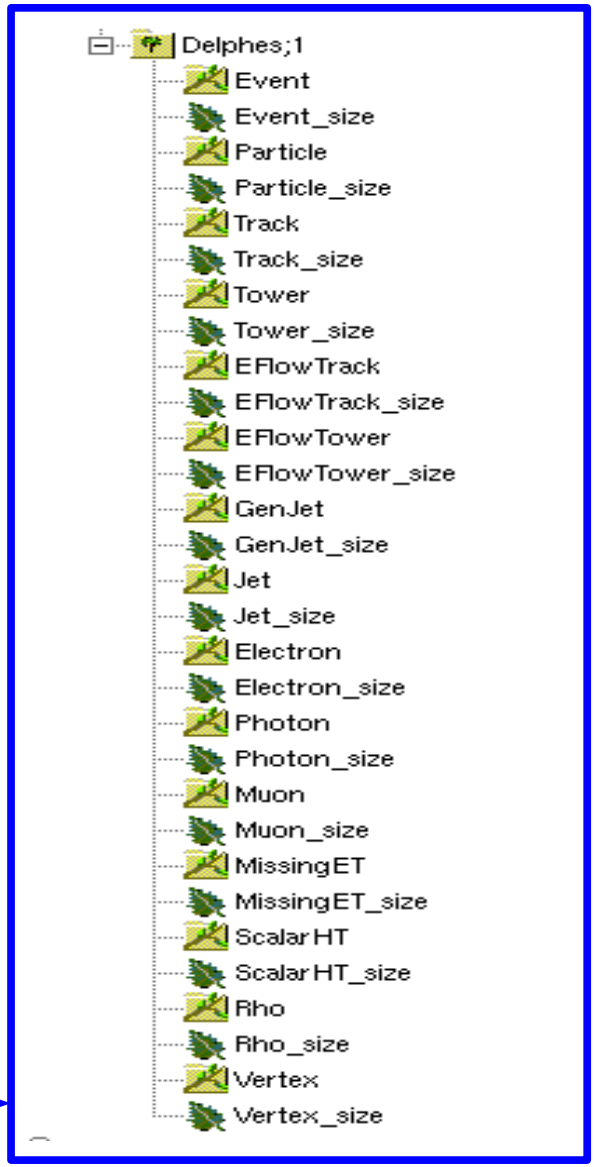
- **modular** C++ code

- Uses
  - ROOT classes [Comp. Phys. C. 180 (2009) 2499]
  - FastJet package [Eur. Phys. J. C 72 (2012) 1896]

- Input
  - Pythia/Herwig output (HepMC,STDHEP)
  - LHE (MadGraph/MadEvent)
  - ProMC

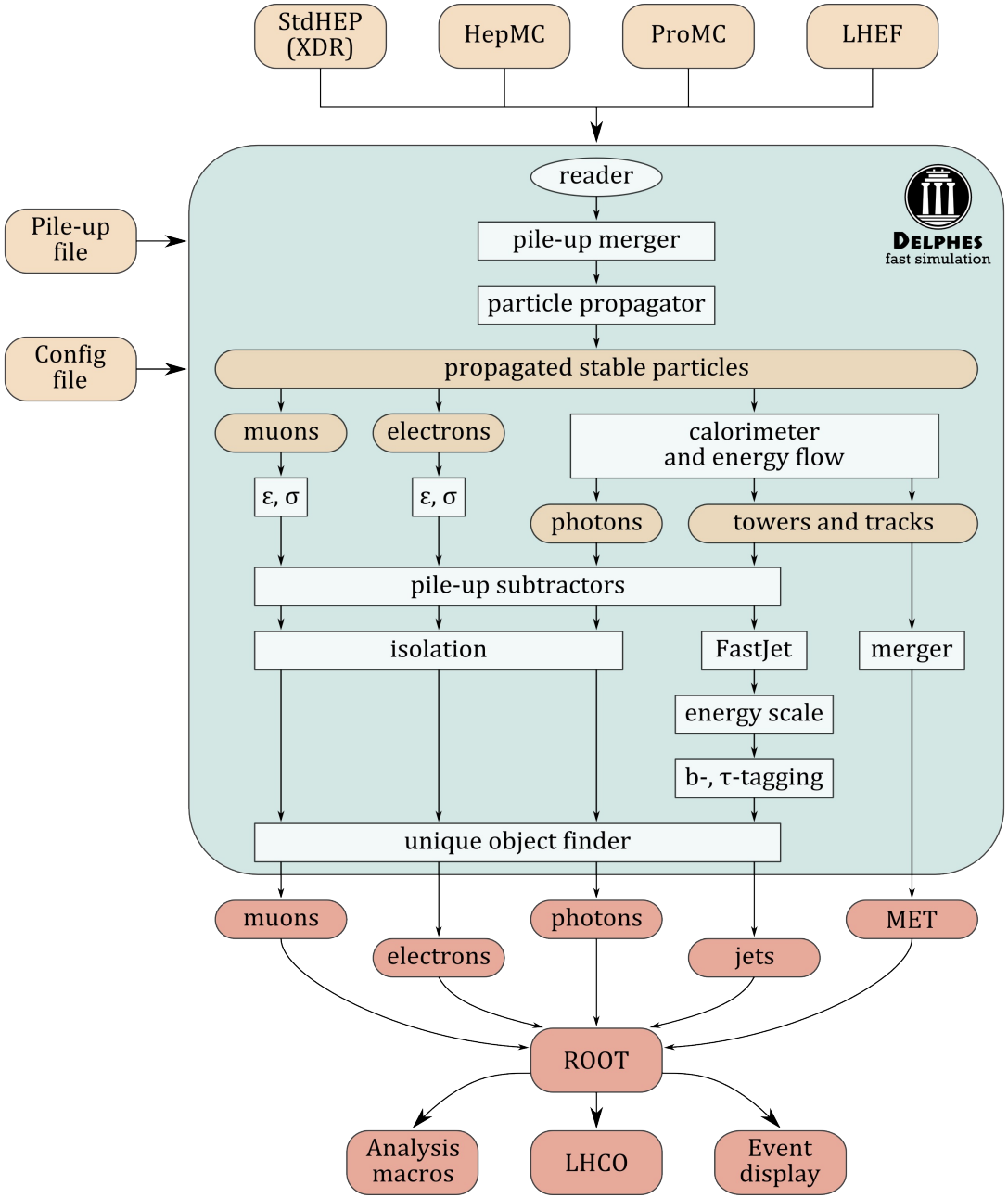
- Configuration file
  - Define geometry
  - Resolution/reconstruction/selection criteria
  - Output object collections

- Output
  - ROOT trees



default **CMS** and **ATLAS** configurations are included in any Delphes release

The Delphes project: A modular structure



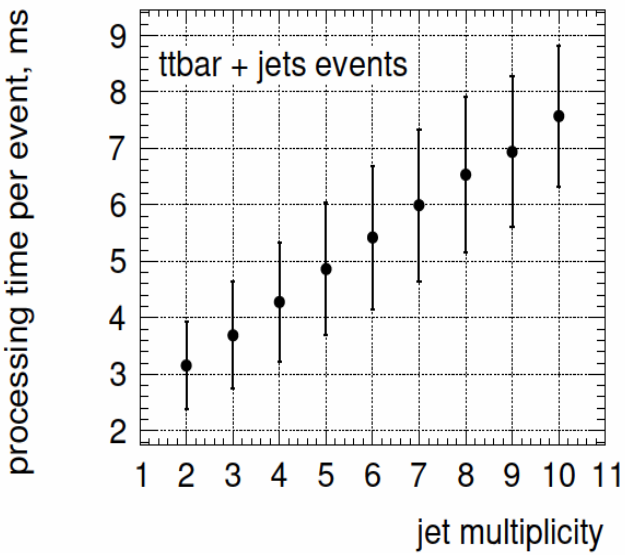
- Every Object in Delphes is a **Candidate**.
- All **modules** consume and produce **Arrays** of **Candidates**.
- Any module can **access** Arrays produced by other modules using **ImportArray** method: *ImportArray("ModuleName/arrayName")*
- The Delphes team provides a set of modules.
- A user can create **new modules** and define its **own sequence**.

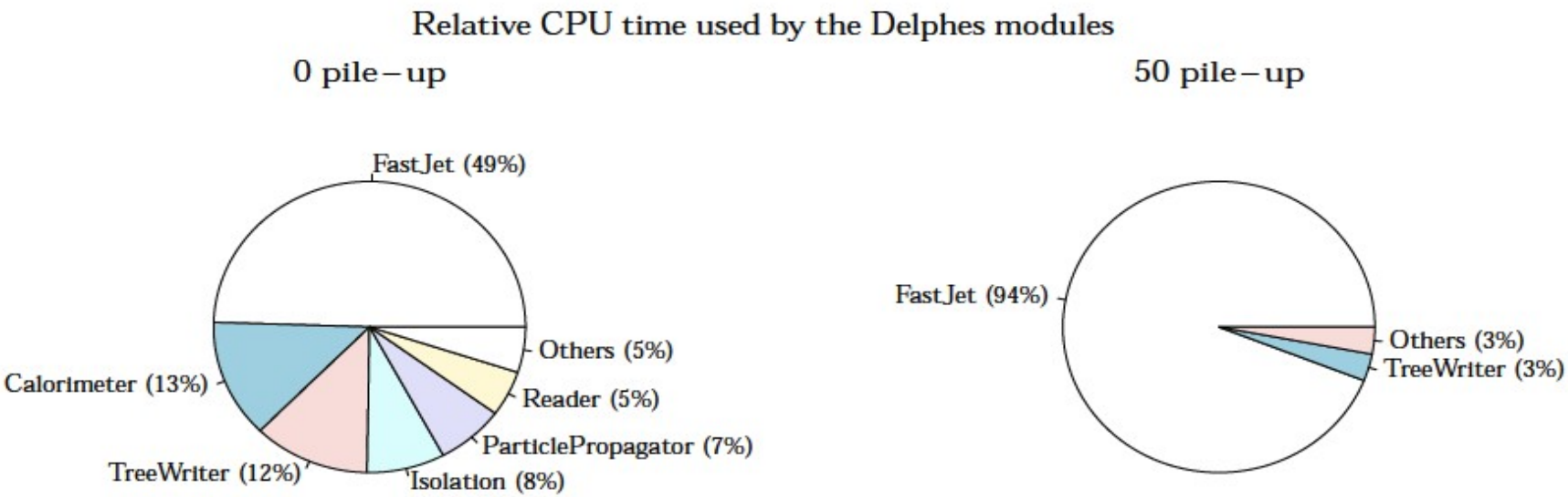
The Delphes Project: CPU time

Delphes reconstruction:

1ms (0 PU) – 1s (150 PU)

Mainly spent in the FastJet algorithm:





Particle Flow

Particle-Flow Emulation

- Idea: Reproduce realistically the performances of the Particle-Flow algorithm.

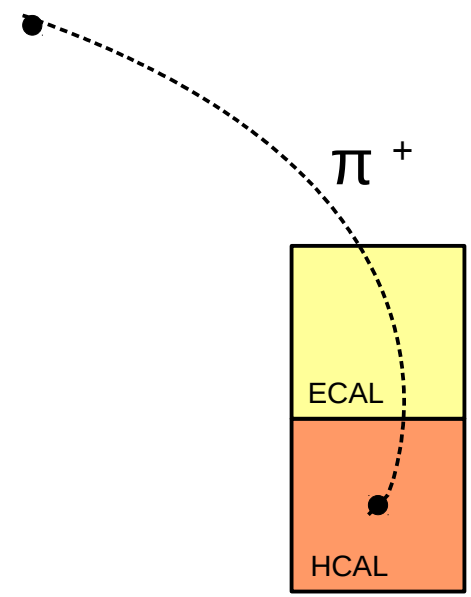
- In practice, in DELPHES use **tracking and calo** info to reconstruct high reso. input objects for later use (jets, E_T^{miss}, H_T)

  → assume **σ(trk) < σ(calo)**

Example 1: A pion of 10 GeV

→ $E^{HCAL}(\pi^+) = 15$ GeV

$E^{TRK}(\pi^+) = 11$ GeV



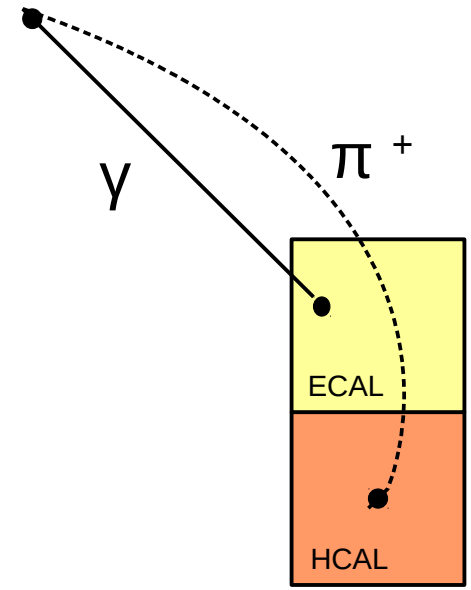
Particle-Flow algorithm creates:

→ PF-track, with energy $E^{PF\text{-}trk} = 11$ GeV

→ PF-tower, with energy $E^{PF\text{-}tower} = 4$ GeV

Particle-Flow Emulation

Example 2: A pion (10 GeV) and a photon (20 GeV)



→ $E^{ECAL}(\gamma) = 18$ GeV

→ $E^{HCAL}(\pi^+) = 15$ GeV

→ $E^{TRK}(\pi^+) = 11$ GeV

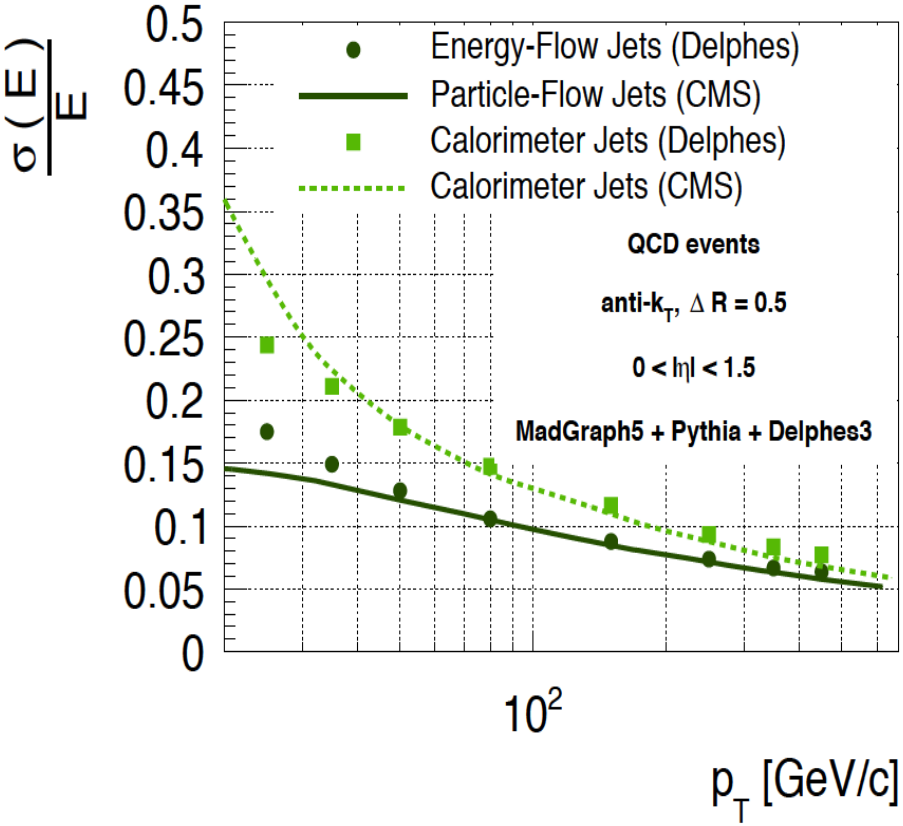
Particle-Flow algorithm creates:

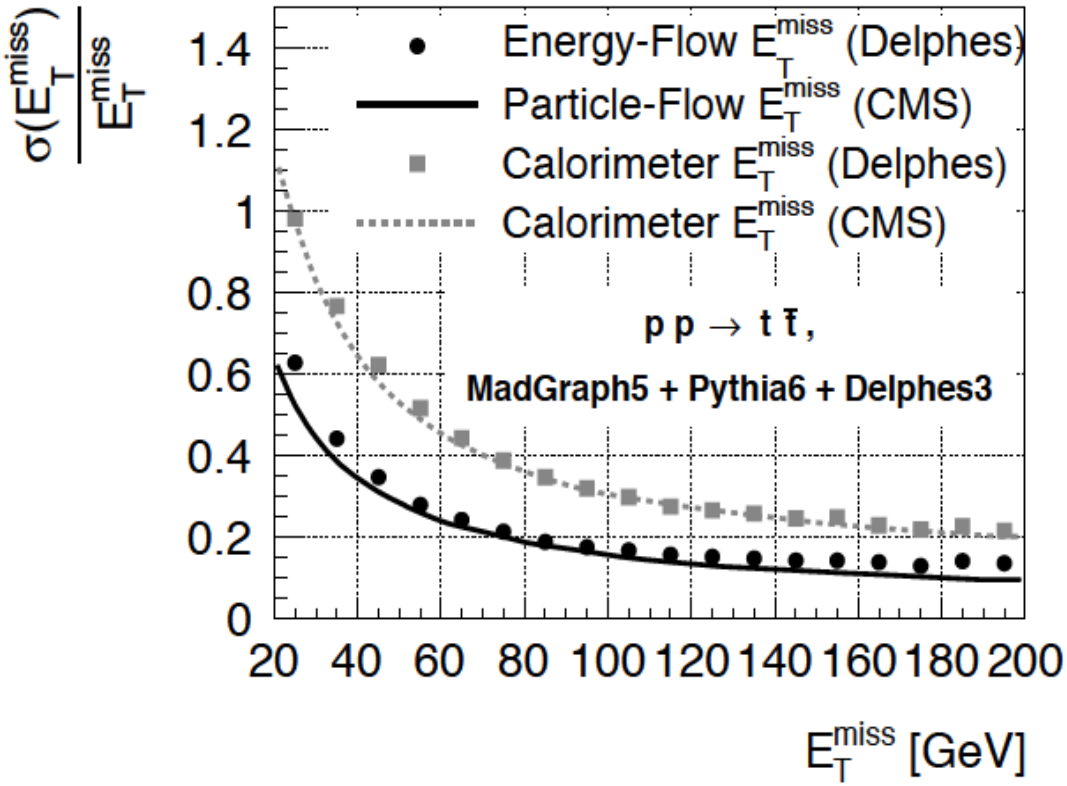
→ PF-track, with energy $E^{PF\text{-}trk} = 11$ GeV

→ PF-tower, with energy $E^{PF\text{-}tower} = 4 + 18$ GeV

Separate neutral and charged calo deposits has crucial implications for pile-up subtraction

Particle Flow: Validation





→ **good agreement**

Particle Flow: Physics example

- Reproduce part of **top mass measurement** in semi-leptonic decay (arXiv:1209:2319)

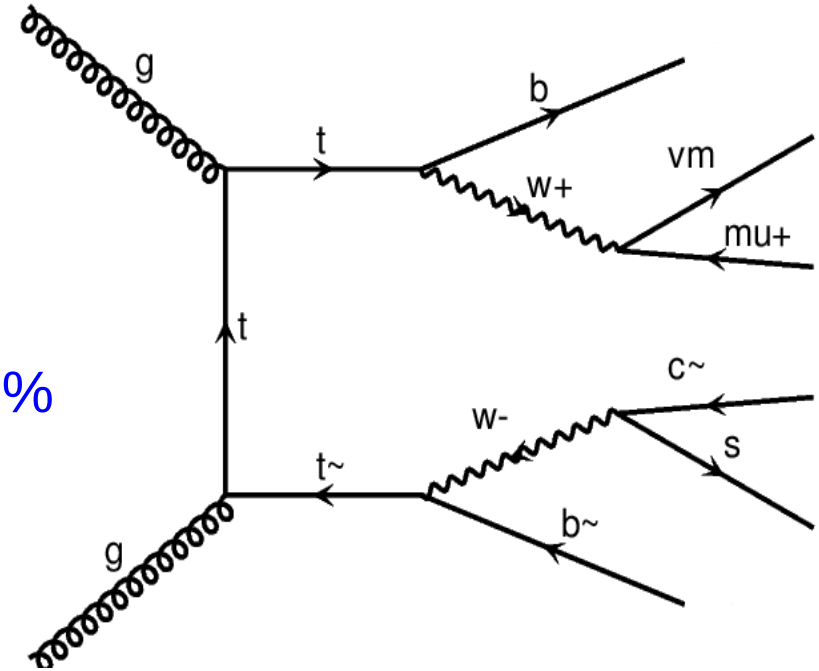
- **Signal** produced with MG5+Pythia+Delphes3

- Selection criteria:

  → = 1 lepton p_T > 30 GeV, |η| < 2.1
  → ≥ 4 jets p_T > 30 GeV, |η| < 2.4
  → ≥ 2 b-tagged jets, ≥ 2 light jets

- eff(Delphes) = 2.8% vs. eff(CMS) = 2.3%

→ **good agreement**

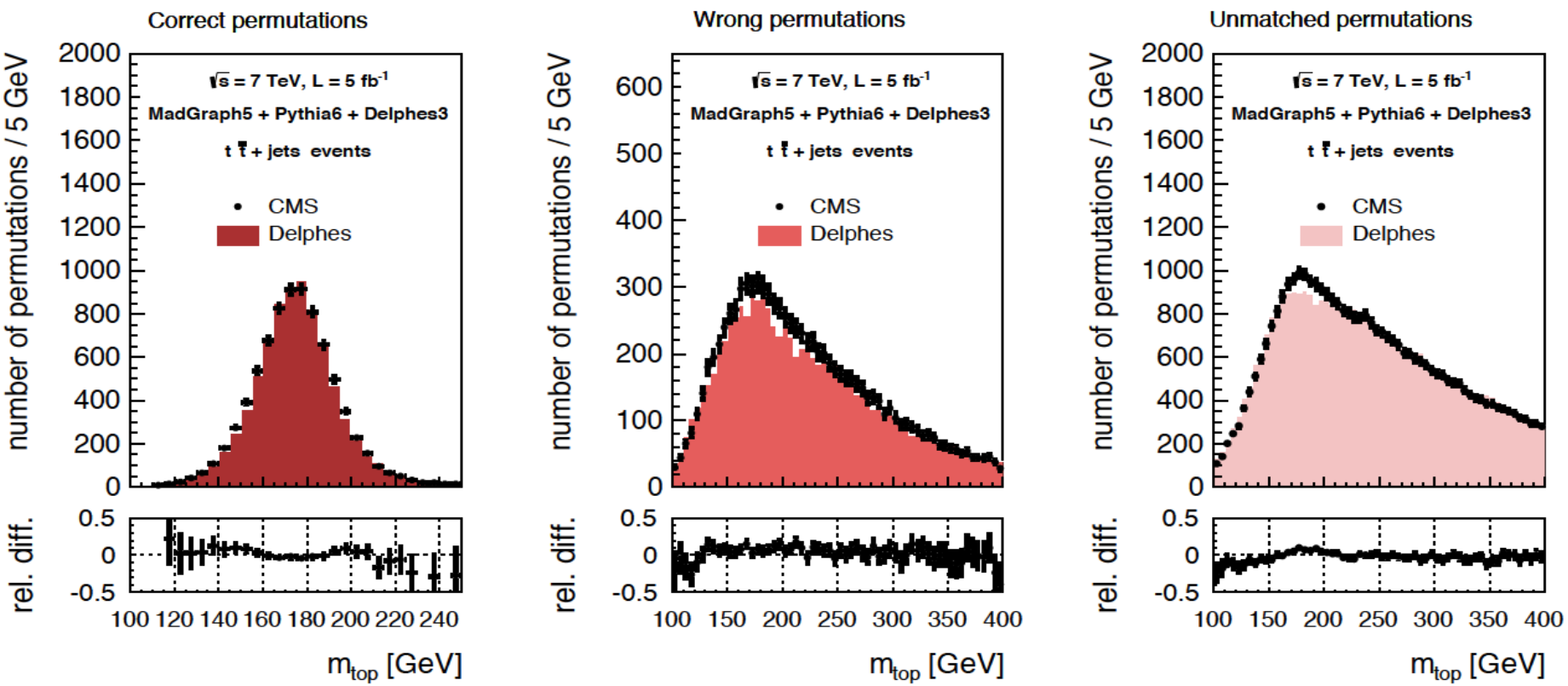


Particle Flow: Physics example

Look at **hardest** 2 b-tagged and 2 light jets (à la CMS):

- correct : 4 jets are good, associate right b with lights
- wrong : 4 jets are good, associate wrong b with lights
- unmatched : at least one of the jets doesn't match

	CMS	DELPHES
correct	15.5 %	15.8 %
wrong	17.4 %	16.5 %
unmatched	67.1 %	67.7 %

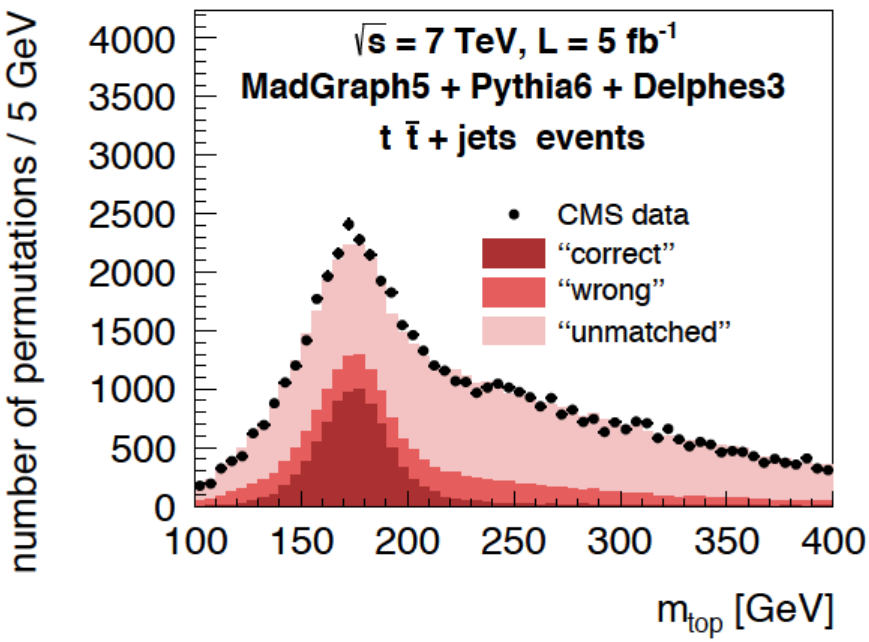


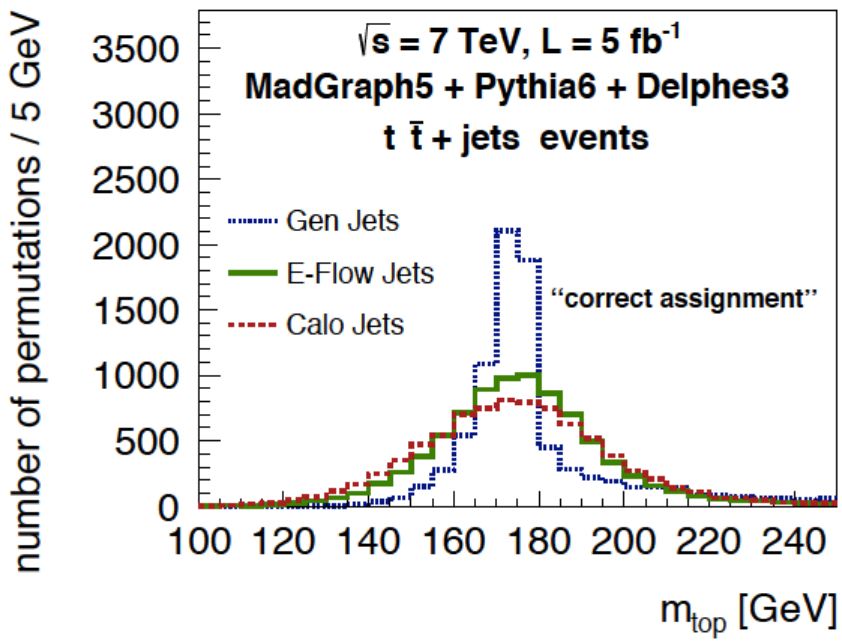
Particle Flow: Physics example

Look at **hardest** 2 b-tagged and 2 light jets (à la CMS):

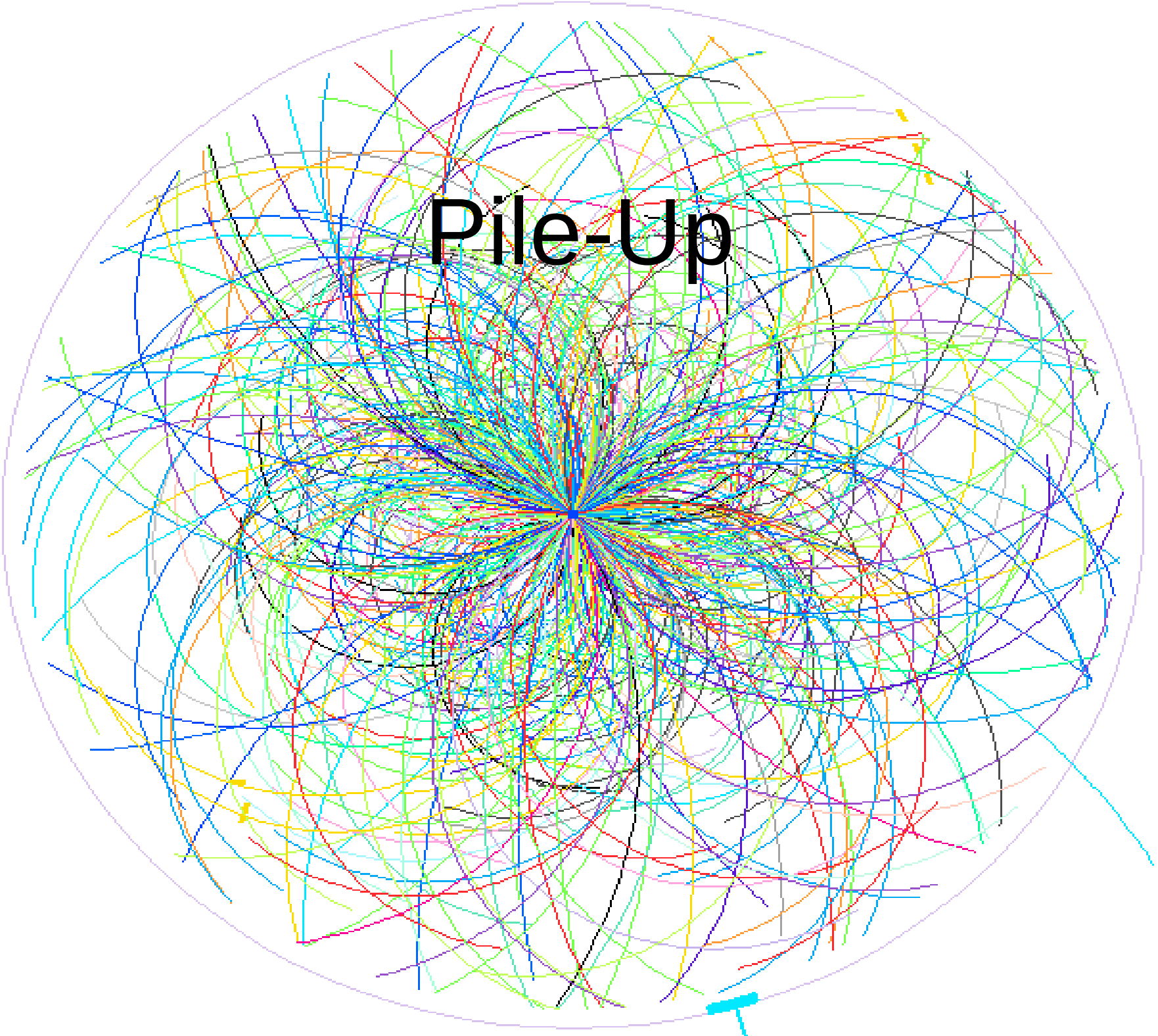
- correct : 4 jets are good, associate right b with lights
- wrong : 4 jets are good, associate wrong b with lights
- unmatched : at least one of the jets doesn't match

	CMS	DELPHES
correct	15.5 %	15.8 %
wrong	17.4 %	16.5 %
unmatched	67.1 %	67.7 %





Pile-Up



Pile-up motivations

- **Pile-up** becomes an issue at **high luminosity** LHC
  - reduced **efficiency**
  - worsened **resolution** (jets, E_T^{miss})
  - degraded **isolation**
  - **fake** tracks, jets

- **Efficiencies** and **resolutions** can be **tuned by hand** to mimic pile-up
- Fake objects (jets) need to be simulated. Also, we want to have some predictive power:

  → We therefore introduced: tunable **simulation** of pile-up

  pile-up **substraction** procedure.

Pile-up: implementation

- **Pile-up** is implemented in Delphes **since version 3.0.4**
  - **mixes** N minimum bias events with hard event sample
  - spreads **poisson(N)** events along z-axis with configurable spread
  - rotate event by random angle φ wrt z-axis

- **Charged** Pile-up subtraction (most effective if used with PF algo)

  - if **z < |Zres|** keep all **charged and neutrals** (→ ch. particles too close to hard scattering to be rejected)

  - if **z > |Zres|** keep only **neutrals** (perfect charged subtraction)

  - allows user to tune amount of charged particle subtraction by **adjusting Z spread/resolution**

- **Residual** pile-up subtraction is needed for jets and isolation.
  - Use the FastJet Area approach (Cacciari, Salam, Soyez)
    - compute ρ = event pile-up density
    - jet correction : $pT \rightarrow pT - \rho A$ (JetPileUpSubtractor)
    - isolation : $\sum pT \rightarrow \sum pT - \rho\pi R^2$ (Isolation module itself)
  - Subtraction can be |η| dependent

Pile-Up : Event Display

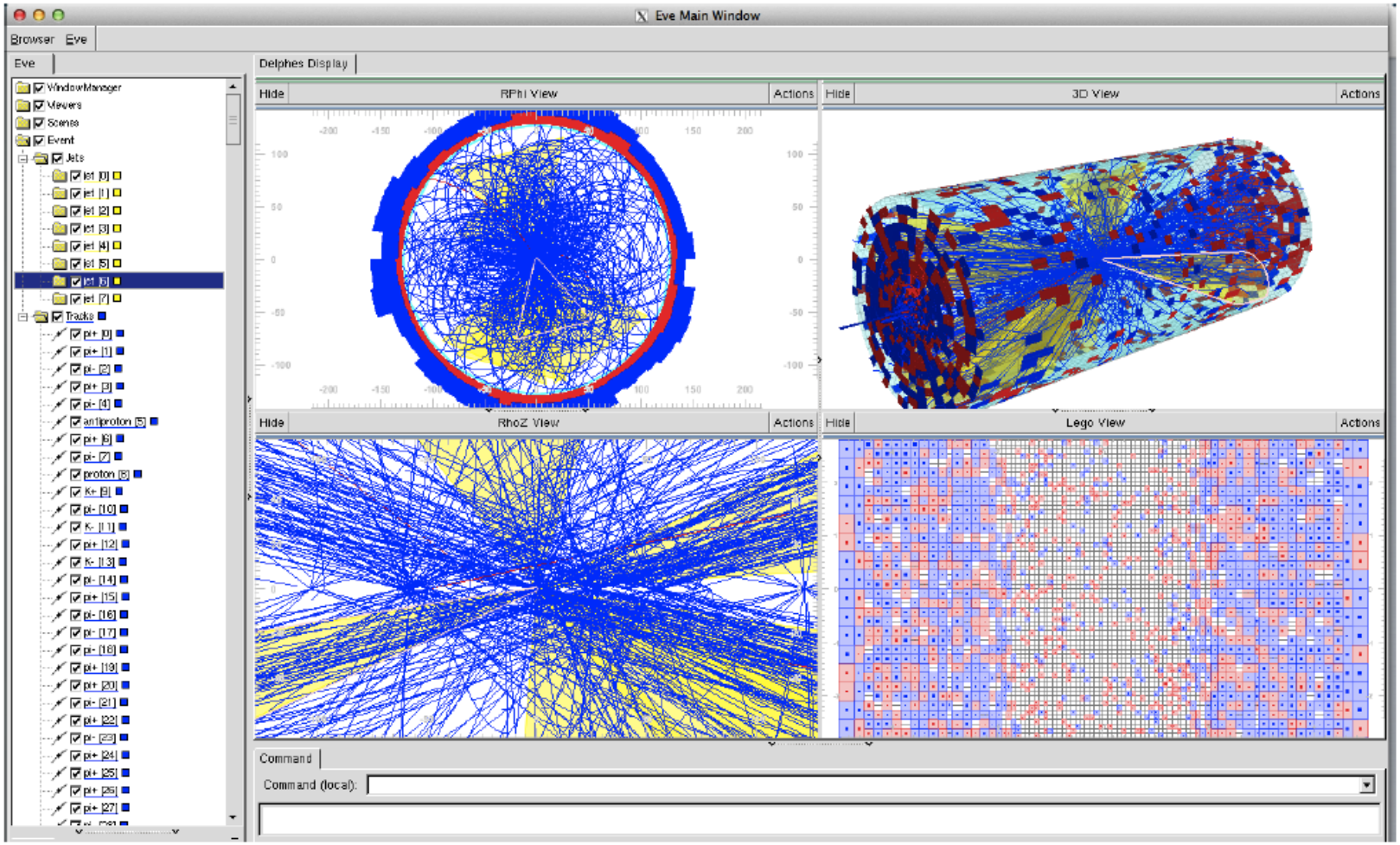
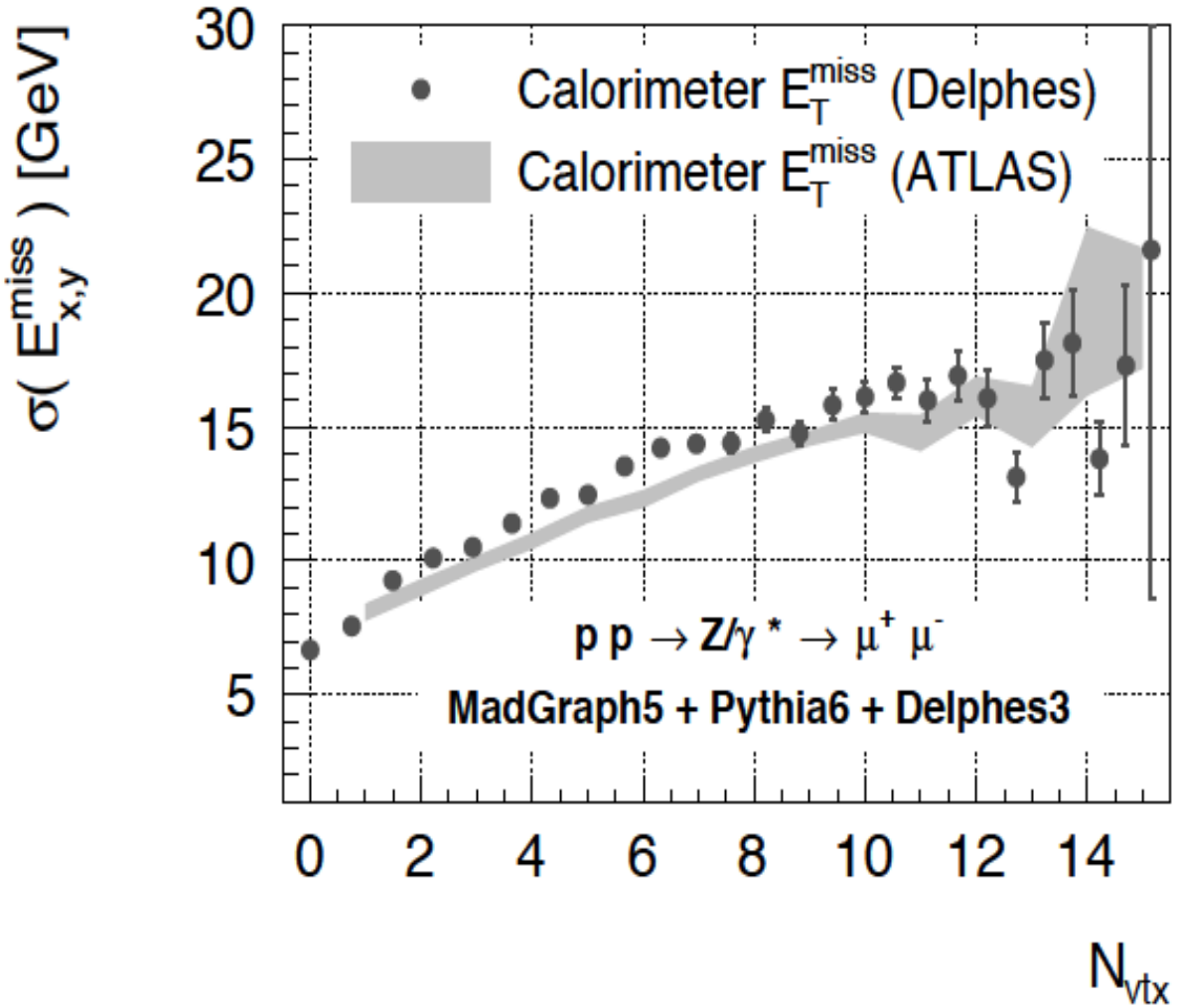


Figure 3. QCD event with 50 pile-up interactions shown with the DELPHES event display based on the ROOTEVE libraries [12]. Transverse view (top left), longitudinal view (bottom left), 3D view (top right), (η,ϕ) view (bottom right).

Pile-Up: Validation



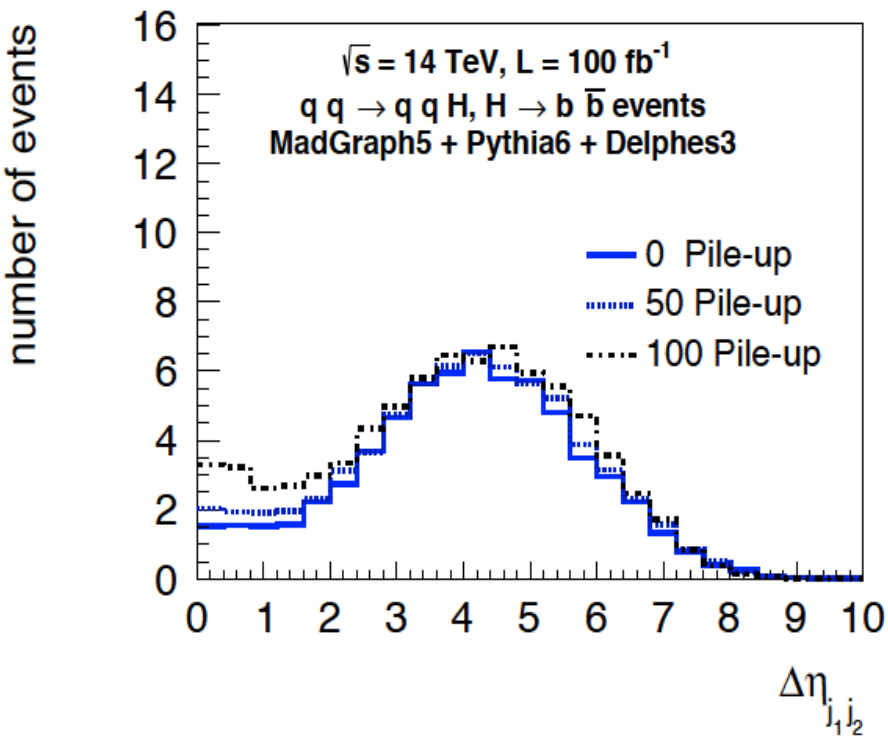
→ **good agreement**

Pile-Up: Physics example

- H → bb in **VBF channel** expected to be highly affected by pile-up
- Irreducible background **bb+jets**
- Select >4 jets with pT > 80, 60, 40, 40 (at least 2 b-tagged, at least 2 light)

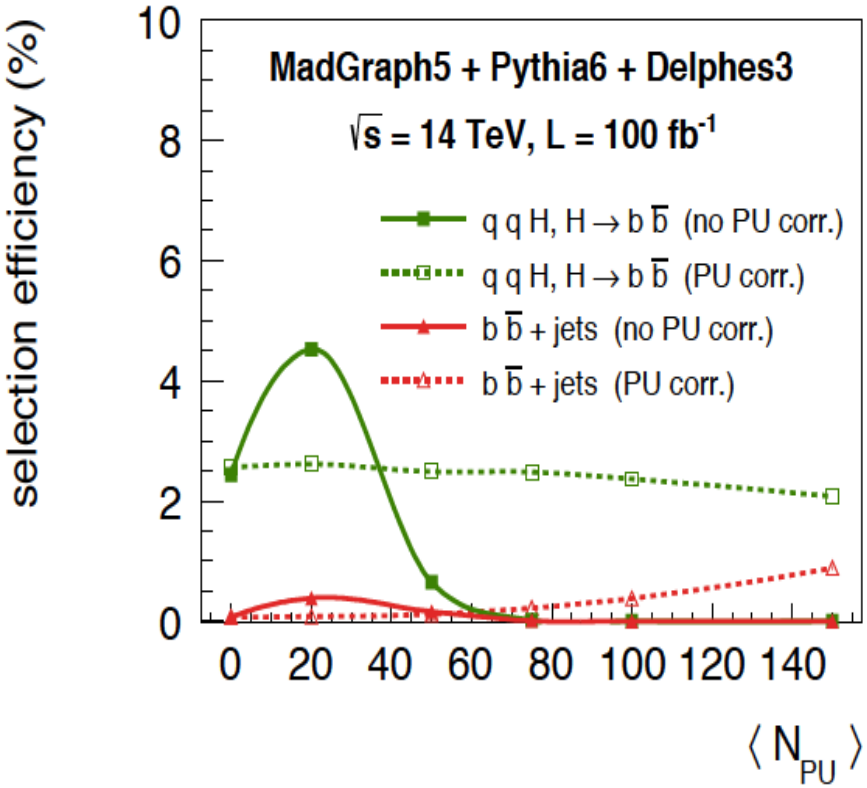
Emergence of pile-up jets in the central region:

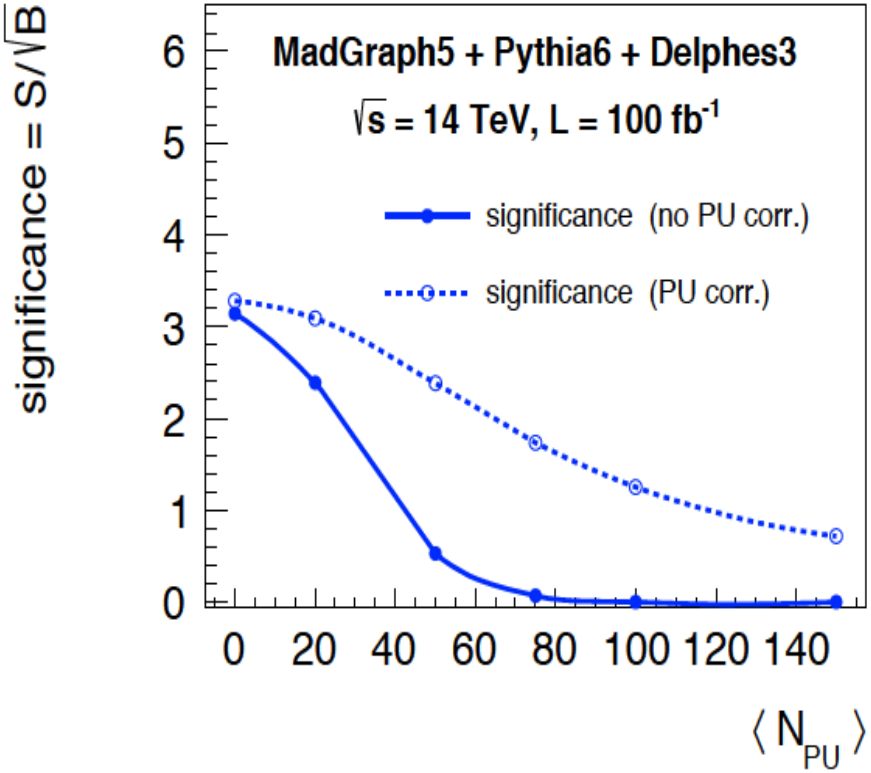
→ **depletion of rapidity gap**



Pile-Up : Physics example

- Require **large rapidity gap** between light jets, **no hadronic activity** in between
- 100 < m(bb) < 200 GeV



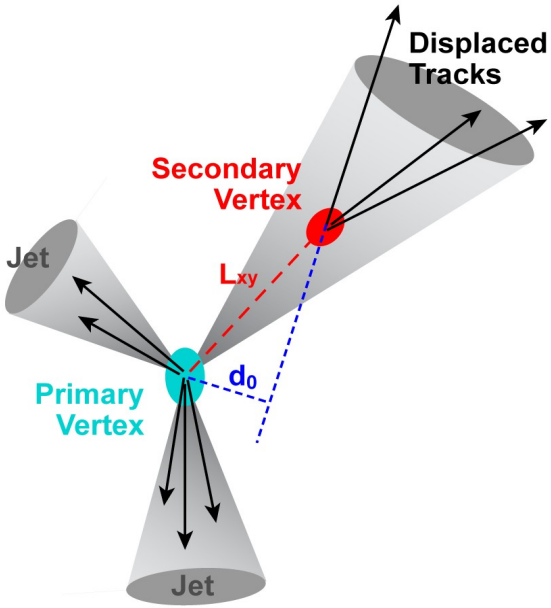


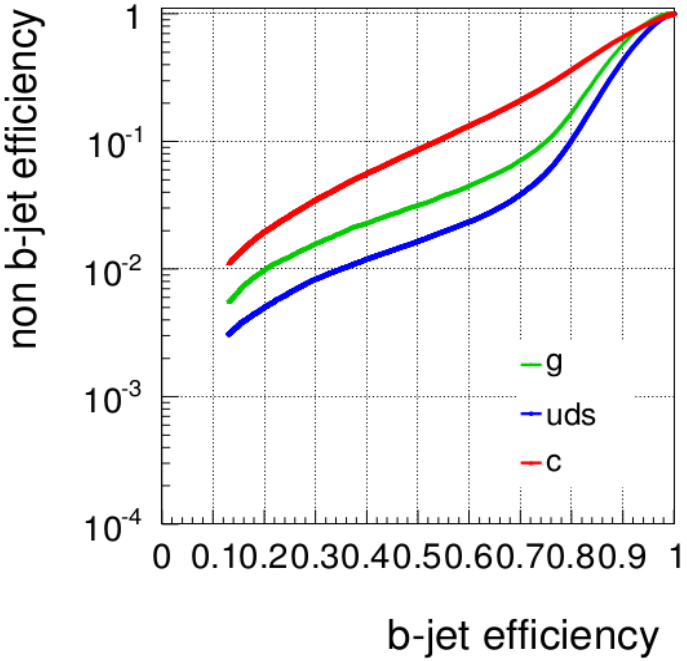
Other new features

Track Counting B-tagging

Parametric **b-tagging**:
- Check if there is a b,c-quark in the cone of size DeltaR
- Apply a **parametrized Efficiency** (PT, eta)

Track Counting :
- Smearing the impact parameter (d_0) of each track inside a jet.
- Application of the **Track Counting** algorithm.
- Provide comparable performances to CMS results
- **Predictive**

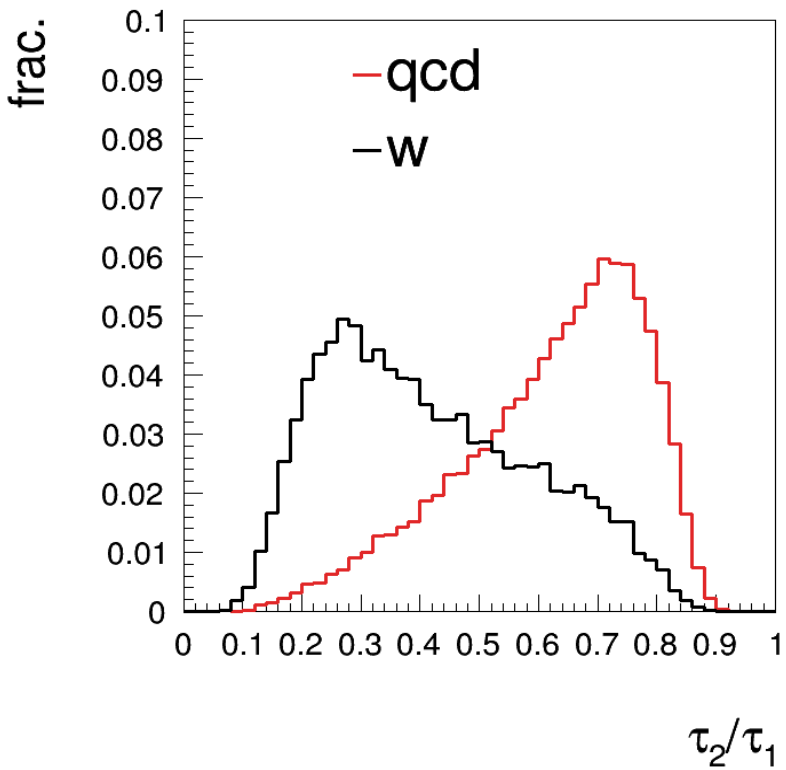




Jet Sub-structure (N-subjettiness)

JHEP 1103:015 (2011), JHEP 1202:093 (2012) and JHEP 1404:017 (2014)

- very useful for identifying **sub-structure** of highly-boosted jets.
- build ratios τ_N / τ_M to **discriminate** between **N or M-prong**



- Embedded in FastJetFinder module
- Variables τ_1, τ_2, .. , τ_5 saved as jet members (N-subjettiness)

Used in the context of hyperboosted jets

- Impact of calorimeter cell size [M. Selvaggi*]

* https://indico.cern.ch/event/302395/session/20/contribution/43/material/slides/0.pdf

Conclusions

- **Delphes** is an advanced parametric simulation program

  - Allow a ~10.000 time gain in CPU time.
  - Widely tested by the community.

- **Delphes 3** has been out for one year now, with **major improvements**:

  - Modularity.
  - Pile-Up, Particle Flow, Predictive B-tagging, jet sub-structure.
  - Development ongoing (new communities of users, new features, etc..)

- Able to mimic CMS and ATLAS **performances** and reproduce analysis.

  - FCC, LHCb and Alice simulation under preparation.

- Only software in the market able to tackle the 200 PU scenario for future LHC runs.

  - Used for next run of the LHC and future accelerator proposals

- Test it, and give us feedback and **contribute**!

People

Jerome de Favereau
Christophe Delaere
Pavel Demin
Andrea Giammanco
Vincent Lemaitre
Alexandre Mertens
Michele Selvaggi

the community ...

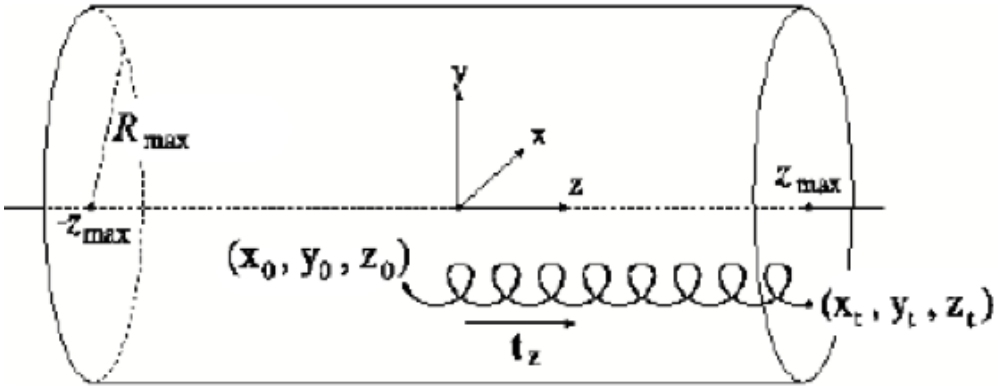
Back-up

The modules: Particle Propagation

- **Charged** and **neutral** particles are propagated in the magnetic field until they reach the calorimeters

- Propagation parameters:

  - magnetic field **B**
  - **radius** and **half-length** (R_{max}, z_{max})



- Efficiency/resolution depends on:

  - particle ID
  - transverse momentum
  - pseudorapidity

```
# efficiency formula for muons
add EfficiencyFormula {13} {                                                    (pt <= 0.1)   * (0.000) + \
                                          (abs(eta) <= 1.5) * (pt > 0.1   && pt <= 1.0)   * (0.750) + \
                                          (abs(eta) <= 1.5) * (pt > 1.0)                  * (1.000) + \
                        (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1   && pt <= 1.0)   * (0.700) + \
                        (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0)                  * (0.975) + \
                        (abs(eta) > 2.5)                                                  * (0.000)}
```

No real tracking/vertexing !!

→ no fake tracks/ conversions (but can be implemented)

→ no dE/dx measurements

The modules: Particle-Flow Emulation

- Idea: Reproduce realistically the performances of the Particle-Flow algorithm.

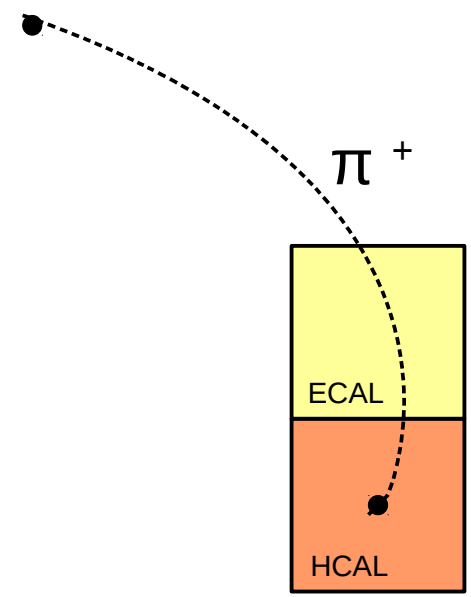
- In practice, in DELPHES use **tracking and calo** info to reconstruct high reso. input objects for later use (jets, E_T^{miss}, H_T)

  → assume **σ(trk) < σ(calo)**

Example 1: A pion of 10 GeV

→ $E^{HCAL}(\pi^+) = 15$ GeV

$E^{TRK}(\pi^+) = 11$ GeV



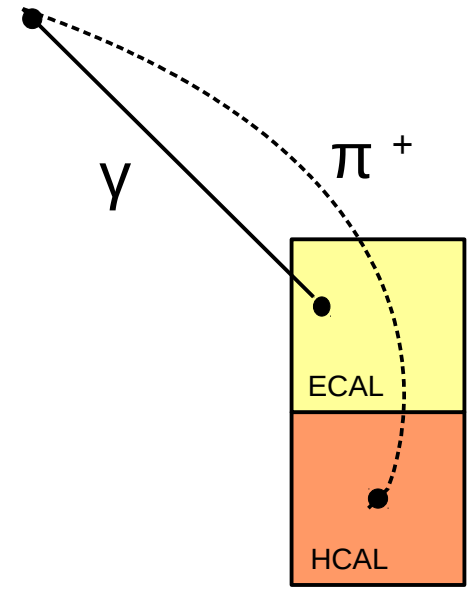
Particle-Flow algorithm creates:

→ PF-track, with energy $E^{PF\text{-}trk} = 11$ GeV

→ PF-tower, with energy $E^{PF\text{-}tower} = 4$ GeV

The modules: Particle-Flow Emulation

Example 2: A pion (10 GeV) and a photon (20 GeV)



→ $E^{ECAL}(\gamma) = 18$ GeV

→ $E^{HCAL}(\pi^+) = 15$ GeV

→ $E^{TRK}(\pi^+) = 11$ GeV

Particle-Flow algorithm creates:

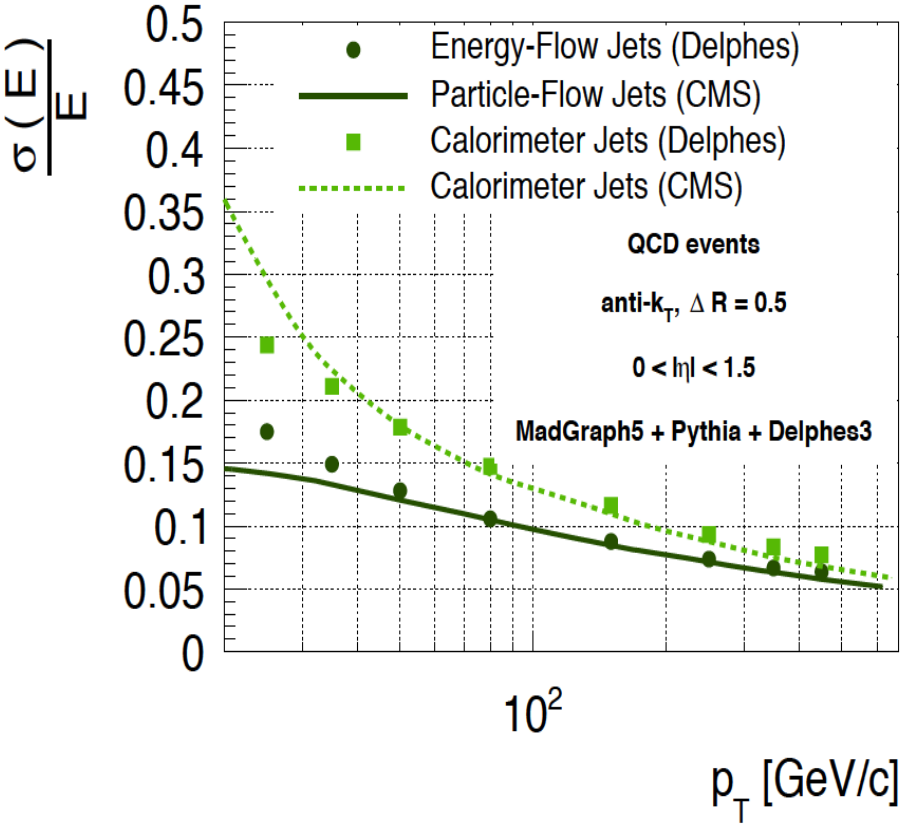
→ PF-track, with energy $E^{PF\text{-}trk} = 11$ GeV

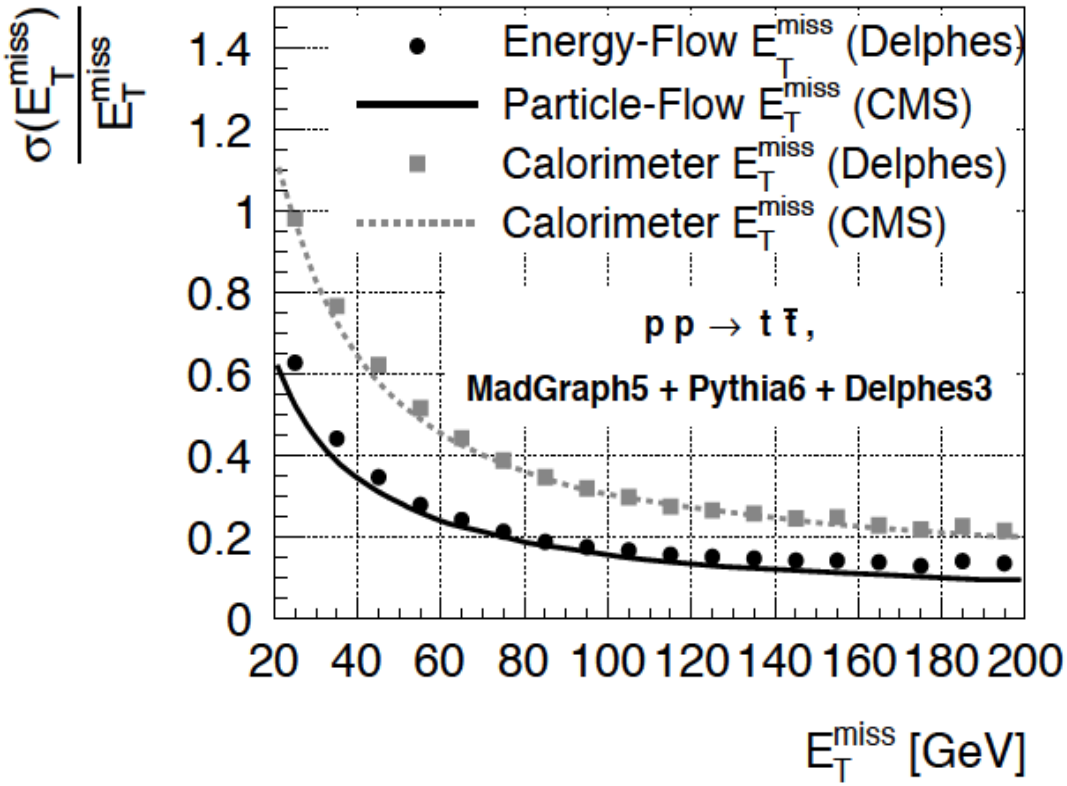
→ PF-tower, with energy $E^{PF\text{-}tower} = 4 + 18$ GeV

Separate neutral and charged calo deposits has crucial implications for pile-up subtraction

No separation between “Photons” and “Neutral Hadrons” in the output.

The modules: Validation of the PF emulation





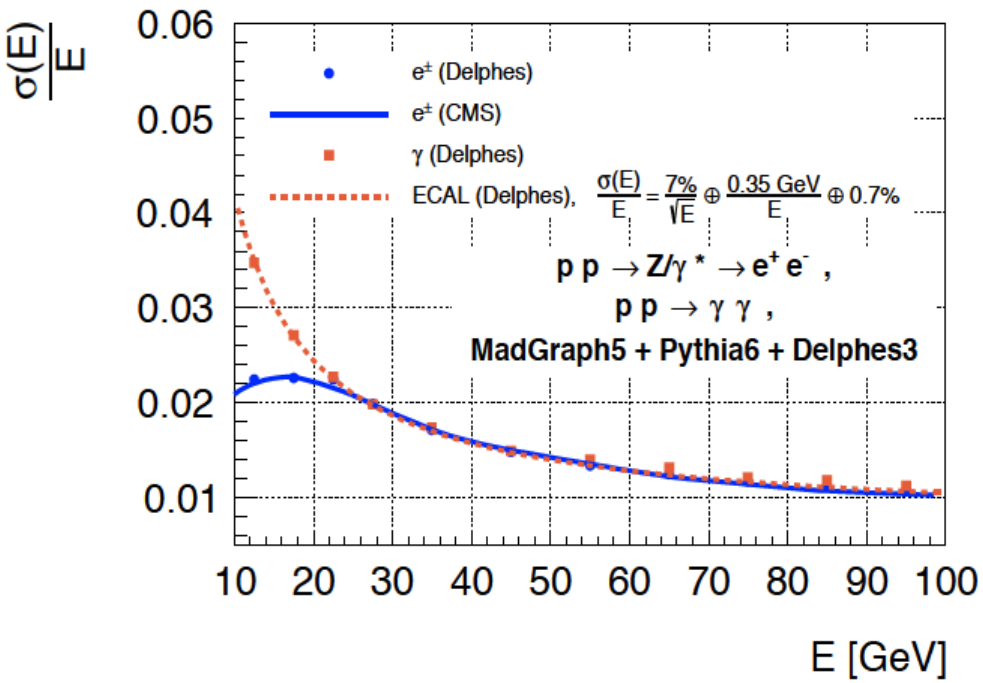
→ **good agreement**

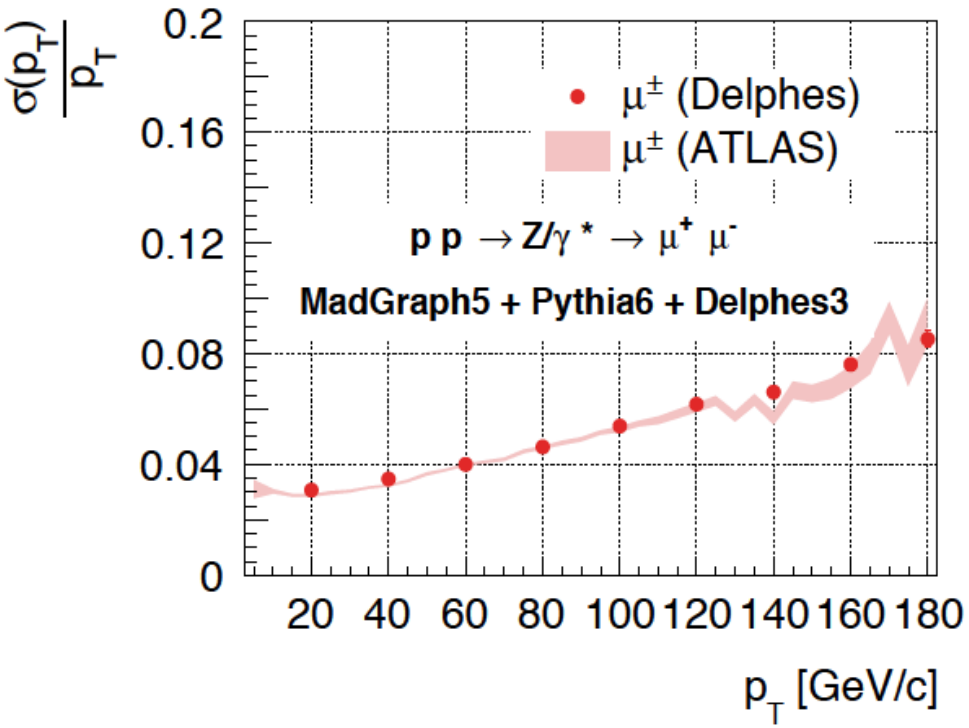
The modules: Jets / E_T^{miss} / H_T

- Delphes uses **FastJet** libraries for jet clustering
- Inputs **calorimeter towers** or "**particle-flow**" objects

```
module FastJetFinder FastJetFinder {
#  set InputArray Calorimeter/towers
  set InputArray EFlowMerger/eflow

  set OutputArray jets

  # algorithm: 1 CDFJetClu, 2 MidPoint, 3 SIScone, 4 kt, 5 Cambridge/Aachen, 6 antikt
  set JetAlgorithm 6
  set ParameterR 0.7

  set ConeRadius 0.5
  set SeedThreshold 1.0
  set ConeAreaFraction 1.0
  set AdjacencyCut 2.0
  set OverlapThreshold 0.75

  set MaxIterations 100
  set MaxPairSize 2
  set Iratch 1

  set JetPTMin 20.0
}
```

The modules: Validation for leptons and photons





→ **excellent agreement (by construction)**

The modules: b-jets (tau-jets)

Parametric **b-tagging**:

- Check if there is a b,c-quark in the cone of size DeltaR
- Apply a **parametrized Efficiency** (PT, eta)

```
module BTagging BTagging {
  set PartonInputArray Delphes/partons
  set JetInputArray JetEnergyScale/jets

  set BitNumber 0
  set DeltaR 0.5
  set PartonPTMin 1.0
  set PartonEtaMax 2.5

  # default efficiency formula (misidentification rate)
  add EfficiencyFormula {0} {0.001}

  # efficiency formula for c-jets (misidentification rate)
  add EfficiencyFormula {4} {                                    (pt <= 15.0) * (0.000) + \
                                              (abs(eta) <= 1.2) * (pt > 15.0) * (0.2*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 15.0) * (0.1*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 2.5)                                  * (0.000)}

  # efficiency formula for b-jets
  add EfficiencyFormula {5} {                                    (pt <= 15.0) * (0.000) + \
                                              (abs(eta) <= 1.2) * (pt > 15.0) * (0.5*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 15.0) * (0.4*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 2.5)                                  * (0.000)}
}
```

Not Predictive