Delphes
fast simulation

UCL
Université catholique de Louvain

Delphes

Fast Detector Simulation

Michele Selvaggi
(on behalf of the Delphes collaboration)

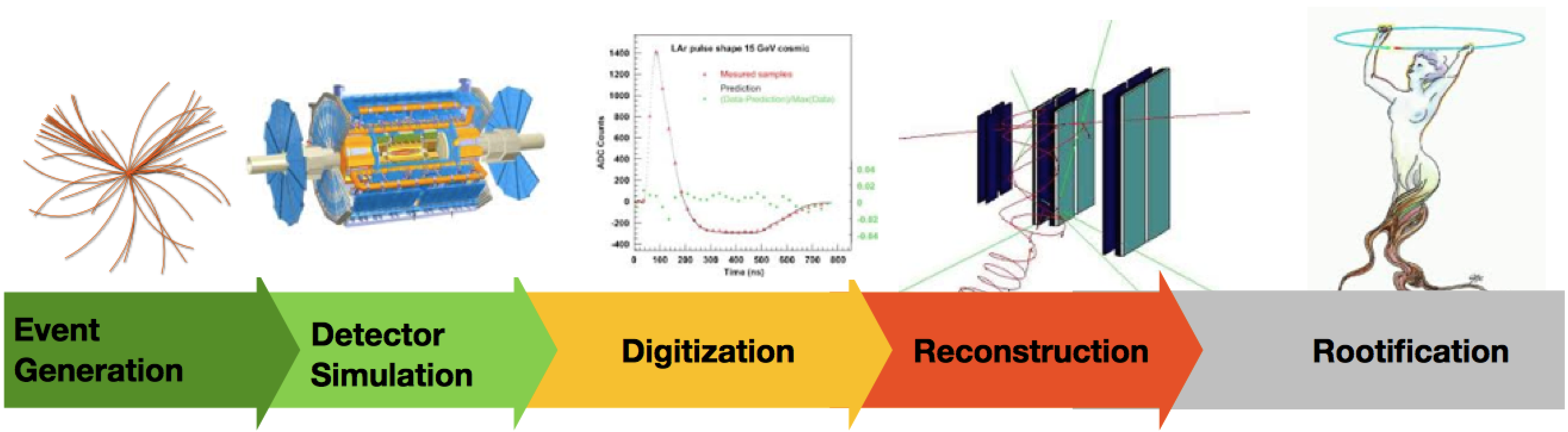
github.com/delphes
cp3.irmp.ucl.ac.be/projects/delphes

MC4BSM - Beijing
24/07/2016

DELPHES
fast simulation

MC chain

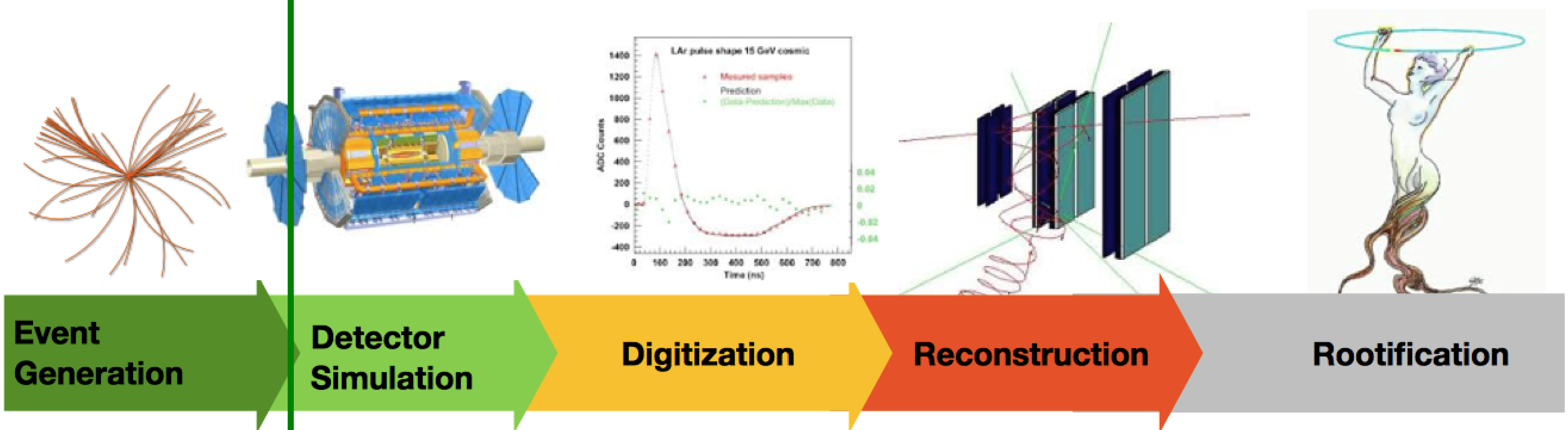
UCL
Université
catholique
de Louvain



courtesy of A. Salzburger

MC chain

FAST SIMULATION



courtesy of A. Salzburger

Detector simulation

- Full simulation (GEANT):
  - **simulates** particle-matter interaction (including e.m. showering, nuclear int., brehmstrahlung, photon conversions, etc ...) → 100 s /ev
- Experiment Fast simulation (ATLAS, CMS ...):
  - **simplifies** and makes faster simulation and reconstruction → 1 s /ev
- Parametric simulation (**Delphes**, PGS):
  - **parameterize** detector response, reconstruct complex objects
    B field propagation, Jets, Missing ET → 10 ms /ev
- Object smearing (Atom, Falcon, TurboSim):
  - from parton to detector object (lookup tables)

When FastSim?

- When to use FastSim?

  → test your model with detector simulation
  → **sensitive to acceptance and complex observable (Jets,MET)**
  → scan big parameter space (SUSY-like)
  → preliminary tests of new geometries/resolutions (future detectors)
  → educational purpose (bachelor/master thesis)

- When not to use FastSim?

  → very exotic topologies (HSCP, long-lived, ...) (NOT YET ...)

The Delphes Project

The Delphes project

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

- Since 2009, its development is **community-based**
  - **ticketing system** for improvement and bug-fixes
    → user proposed patches, can be forked from github and make pull-requests

- In 2013, **DELPHES 3** was released (DELPHES 2 NOT SUPPORTED ANYMORE !!):
  - C++ modular software
  - Dependencies: gcc, tcl, ROOT
  - is shipped with FastJet

- Delphes is itself distributed by various tools: MadGraph, MadAnalysis, CheckMate
- **Widely** tested and used by the community (pheno, Snowmass, Recasting, FCC, CMS upgrades ...)
- Repository: github.com/delphes
- Website and manual: https://cp3.irmp.ucl.ac.be/projects/delphes
- Original publication: JHEP 02 (2014) 057 [1307.6346]

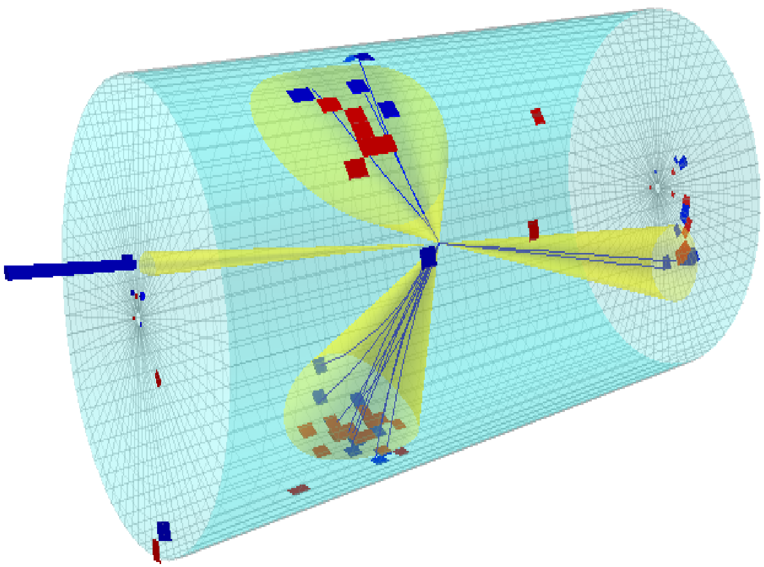
What is Delphes?

DELPHES fast simulation

UCL Université catholique de Louvain

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

- **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

Run Delphes

- Install ROOT from root.cern.ch
- Clone Delphes from github or download from website
- Type in shell:

```
./configure
make -j 4
```

- Run Delphes:

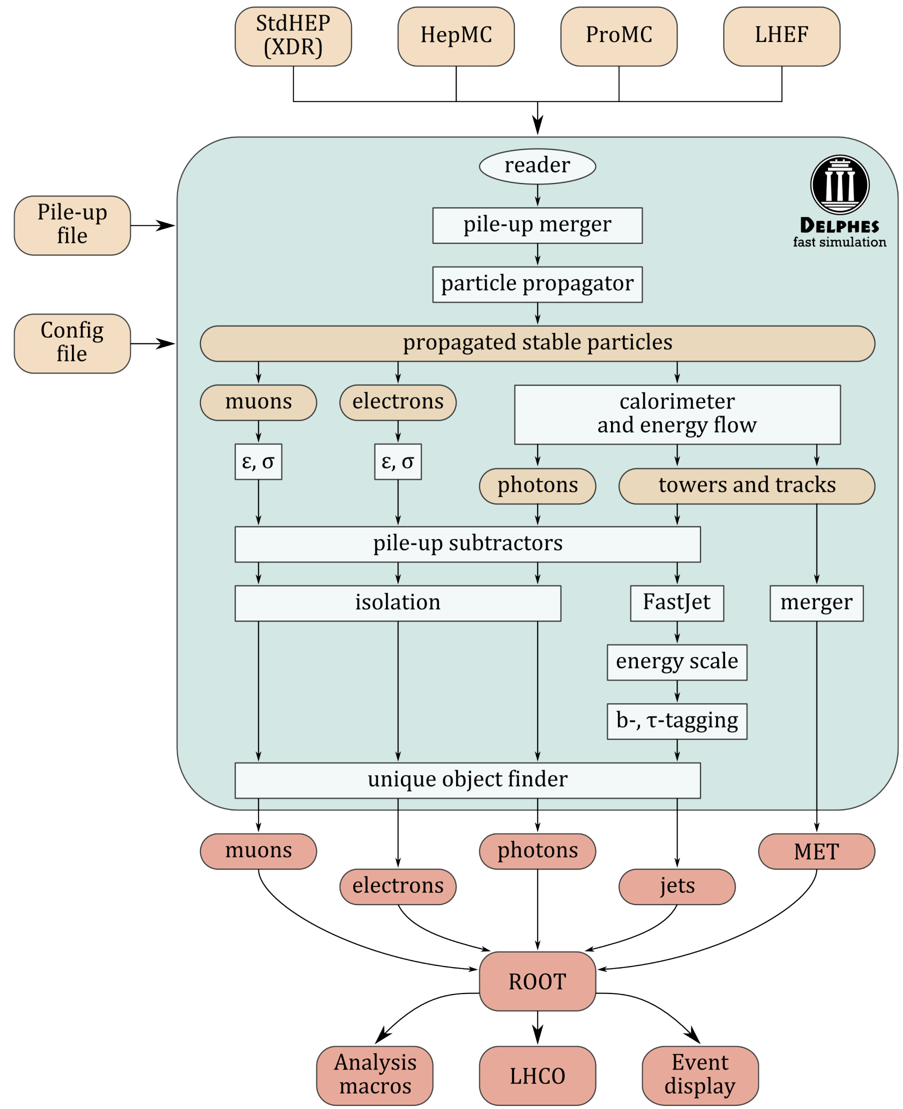
```
./DelphesSTDHEP [detector_card] [output] [input]
./DelphesHepMC [detector_card] [output] [input]
```

- Input formats: HepMC, StdHep, ProMC, LHE
- Output: browsable ROOT tree

Modular

- The modular system allows the user to configure and schedule modules via a configuration file (.tcl), add modules, change data flow, alter output information
- Modules communicate entirely via collections (vectors) of universal objects (TObjArray of Candidate four-vector like objects).
- Any module can access TObjArrays produced by other modules using ImportArray method:

```
ImportArray("ModuleName/arrayName")
```



Configuration file

- Delphes configuration file is based on tcl scripting language
- This is where the detector, data-flow, and output tree is configured.
- Delphes provides tuned detector cards for most detectors:
  - ATLAS, CMS, LHCb, ILD, FCC.
  - can find other tunes in CheckMate, MadAnalysis.
- Order of execution of various modules is configured in the Execution Path:

```
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
```

input(s) candidates

```
  set TowerOutputArray towers
  set PhotonOutputArray photons

  set EFlowTrackOutputArray eflowTracks
  set EFlowPhotonOutputArray eflowPhotons
  set EFlowNeutralHadronOutputArray eflowNeutralHadrons
```

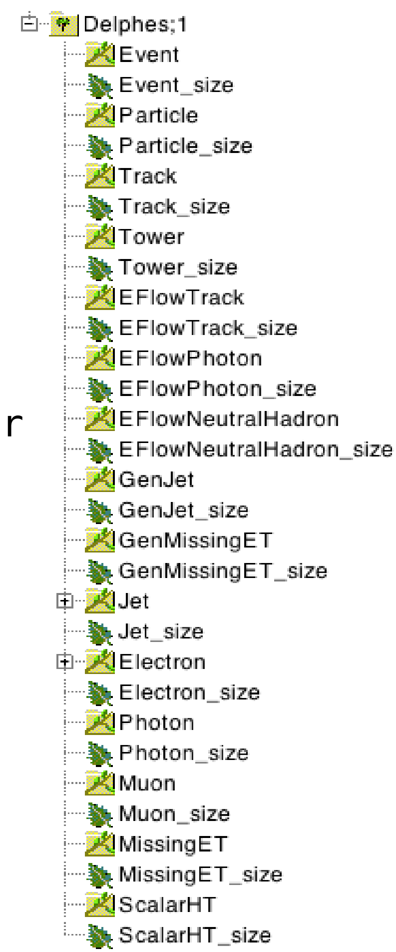
output(s) candidates

```
  ...

  # 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)}
```

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
}
```



Recent Features

Run Delphes with Pythia 8

- You can now run the full MC/reconstruction chain with one simple command by linking Delphes with Pythia8 (more info here).
- Set PYTHIA8 path variable and recompile Delphes:

```
export PYTHIA8=[path_to_pythia8_installation]
make HAS_PYTHIA8=true DelphesPythia8
```

- You can then directly either directly use Pythia8 matrix element, or use external LHE (also with matching available).
- In both case the input to Delphes will be a Pythia8 “cmnd” file:

```
./DelphesPythia8 [detector_card] [pythia8_cmnd] [output]
```

- Avoids storing huge intermediary event files (hepmc), all the parton/hadron-level information can accessed via the Particle branch in the output.
- If multiple weights were stored in LHE input, Delphes stores them in the Weights branch in a vector.

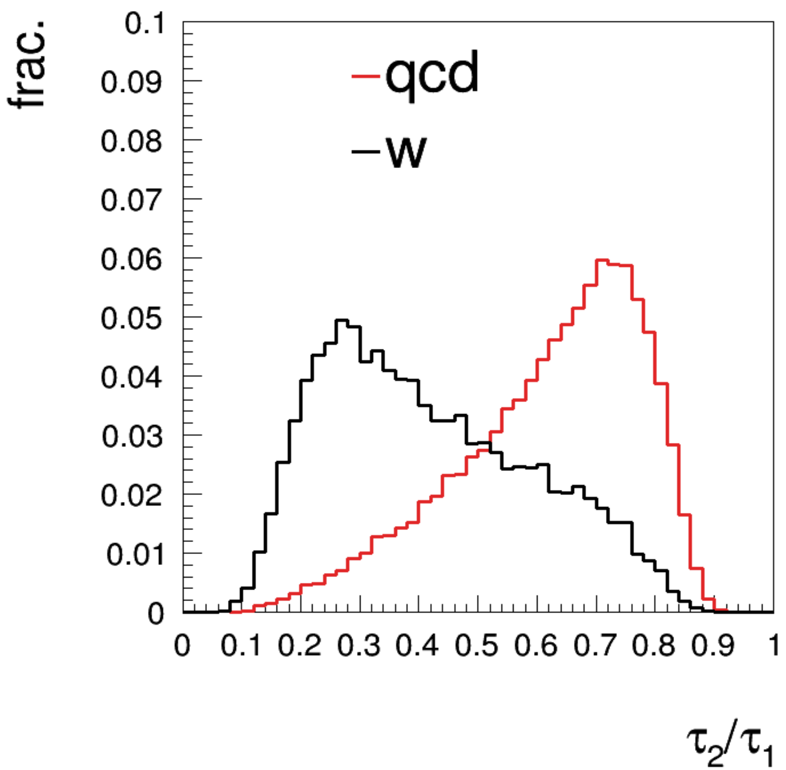
Run Delphes with external FastJet

- Delphes is distributed with full fastjet, with a subset of fastjet/contribs
- However if you want to use your own fastjet code, you have to write a new Delphes module, or alter existing FastJetFinder, which can be cumbersome..
- Instead you can simply use Delphes as low-level candidate producer (i.e particle-flow candidates, calorimeter towers, tracks, …) and feed those objects to fastjet
- We provide a shared object libDelphesNoFastjet.so that serves this purpose
- Complete instructions with examples can be found here.

Jet Substructure

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

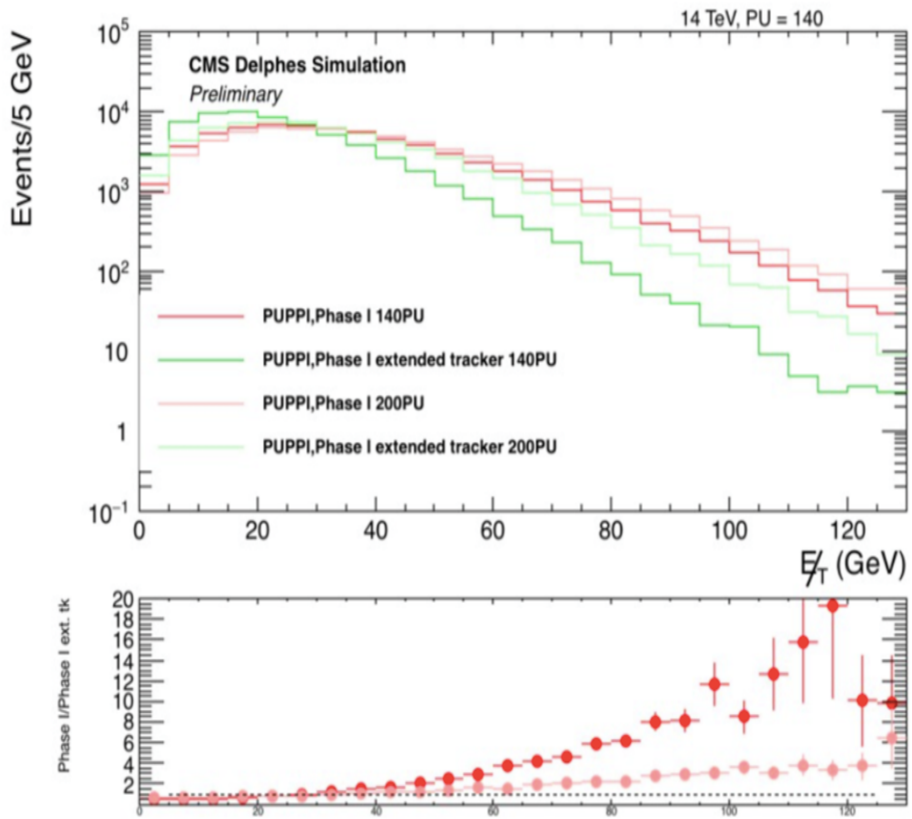
- Embedded in FastJetFinder module
- τ_1, τ_2, .. , τ_5 saved as jet members (N-subjettiness)
- Trimming, Pruning, SoftDrop ...




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

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 5
  set ParameterR 1.0

  set JetPTMin 200.0

  set ComputeTrimming true
  set ComputePruning true
  set ComputeSoftDrop true
  set ComputeNsubjettiness true

}
```

Pile-Up Subtraction

PUPPI has been included [arXiv:1407.6013]



Delphes and PUPPI in combination have been used to argue for a tracker extension up |eta| < 4 for CMS Phase II upgrades!!

DELPHES
fast simulation

Future Studies

UCL
Université catholique de Louvain

- Delphes has been designed to deal with **high number of hadrons** environment:
    - Jets, MET and object isolation are modeled realistically
    - pile-up simulation subtraction (FastJet Area method, PUPPI, SoftKiller)
- Recent improvements:
    - **different segmentation** for ECAL and HCAL
    - **jet substructure** for boosted objects
    - Included configuration card for future collider studies (ILD, FCC)
- Allows for:
    - **reverse engineering**:
      → you have some target for jet invariant mass resolution
      what granularity and resolution are needed to achieve it?
    - impact of pile-up on isolation, jet substructure, multiplicities ...
    - how much does timing information help for pile-up mitigation

Conclusions

- **Delphes 3** has been out for two years now, with **major improvements**:
    - modularity
    - default cards giving results on par with published performance from LHC experiments
    - updated configurations for future e+e- and hh colliders
    - interfaced within MadGraph5/Py8, CheckMate/MadAnalysis

- Delphes 3 can be used right away for fast and realistic simulation for present and future collider studies
- **Delphes is used both by experimentalist and theorists**
- Continuous development (vertexing, conversions, fakes, timing ...)
- Feel free to contribute!

Tutorial:

https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook/Tutorials/Mc4Bsm

Contributors

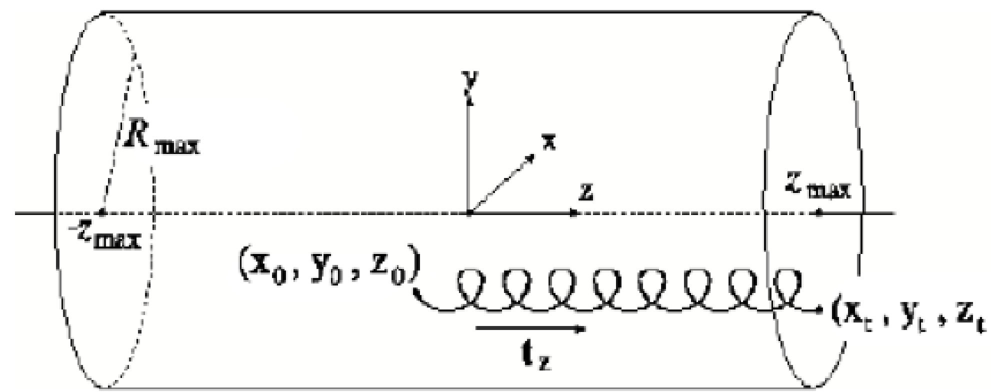
Jerome de Favereau
Christophe Delaere
Pavel Demin
Andrea Giammanco
Vincent Lemaitre
Alexandre Mertens
Michele Selvaggi

the community ...

Back-up

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 (but can be implemented)
→ no dE/dx measurements

Calorimetry

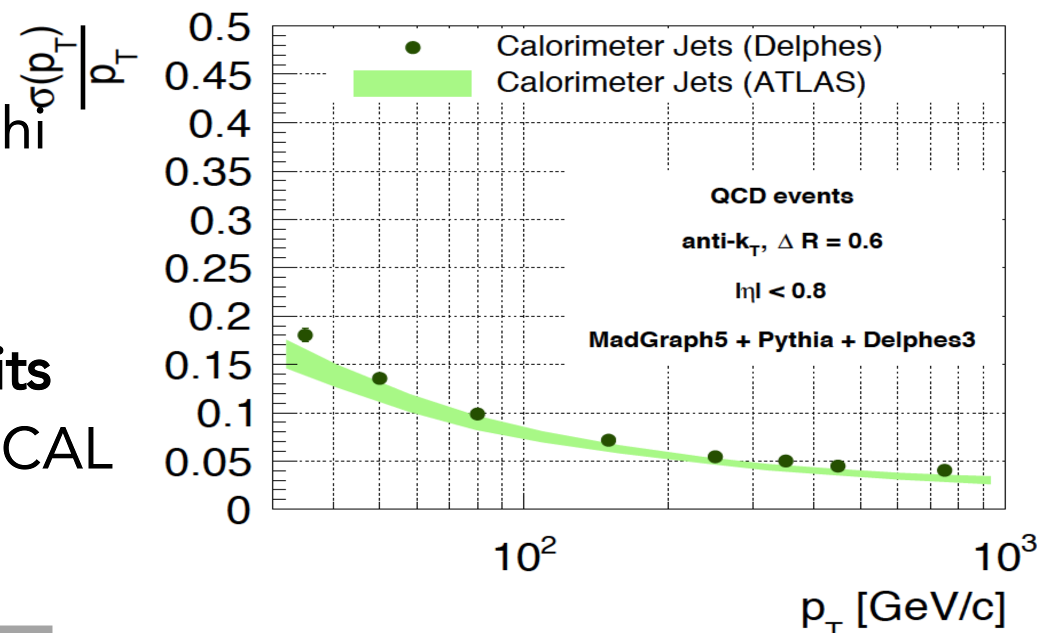
DELPHES fast simulation

UCL Université catholique de Louvain

- Can specify separate ECAL/HCAL **segmentation** in eta/phi

- Each particle that reaches the calorimeters **deposits a fraction of its energy** in one ECAL cell (f_{EM}) and HCAL cell (f_{HAD}), depending on its type:

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 is **smeared** according to the calorimeter cell it reaches

No Energy sharing between the neighboring cells
No longitudinal segmentation in the different calorimeters

Particle-Flow

- 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)

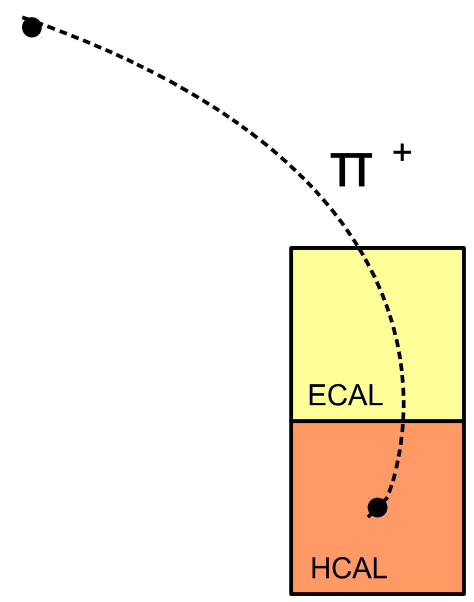
→ If **σ(trk) < σ(calo) (low energy)**

Example: A pion of 10 GeV

$$E^{HCAL}(\pi^+) = 9 \text{ GeV}$$
$$E^{TRK}(\pi^+) = 11 \text{ GeV}$$

Particle-Flow algorithm creates:

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



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

Particle-Flow

- 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)

→ If **σ(trk) < σ(calo) (low energy)**

Example: A pion of 10 GeV

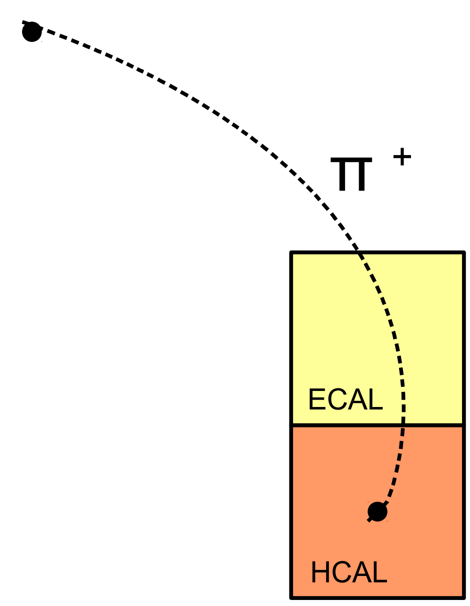
$E^{HCAL}(\pi^+) = 15$ GeV

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



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

Particle-Flow

- 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)

→ If **σ(trk) > σ(calo) (high energy)**

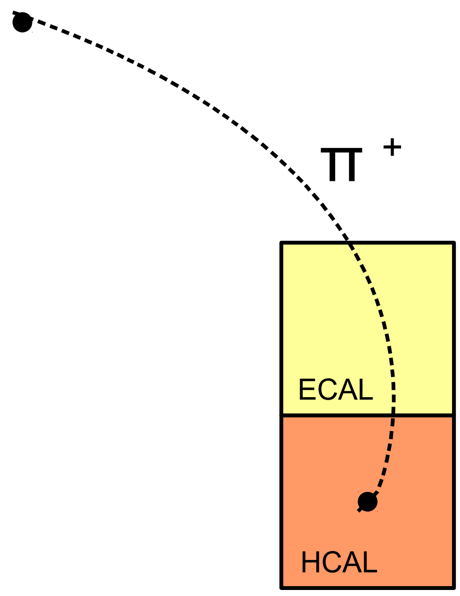
Example: A pion of 500 GeV

$E^{HCAL}(\pi^+) = 550$ GeV

$E^{TRK}(\pi^+) = 400$ GeV

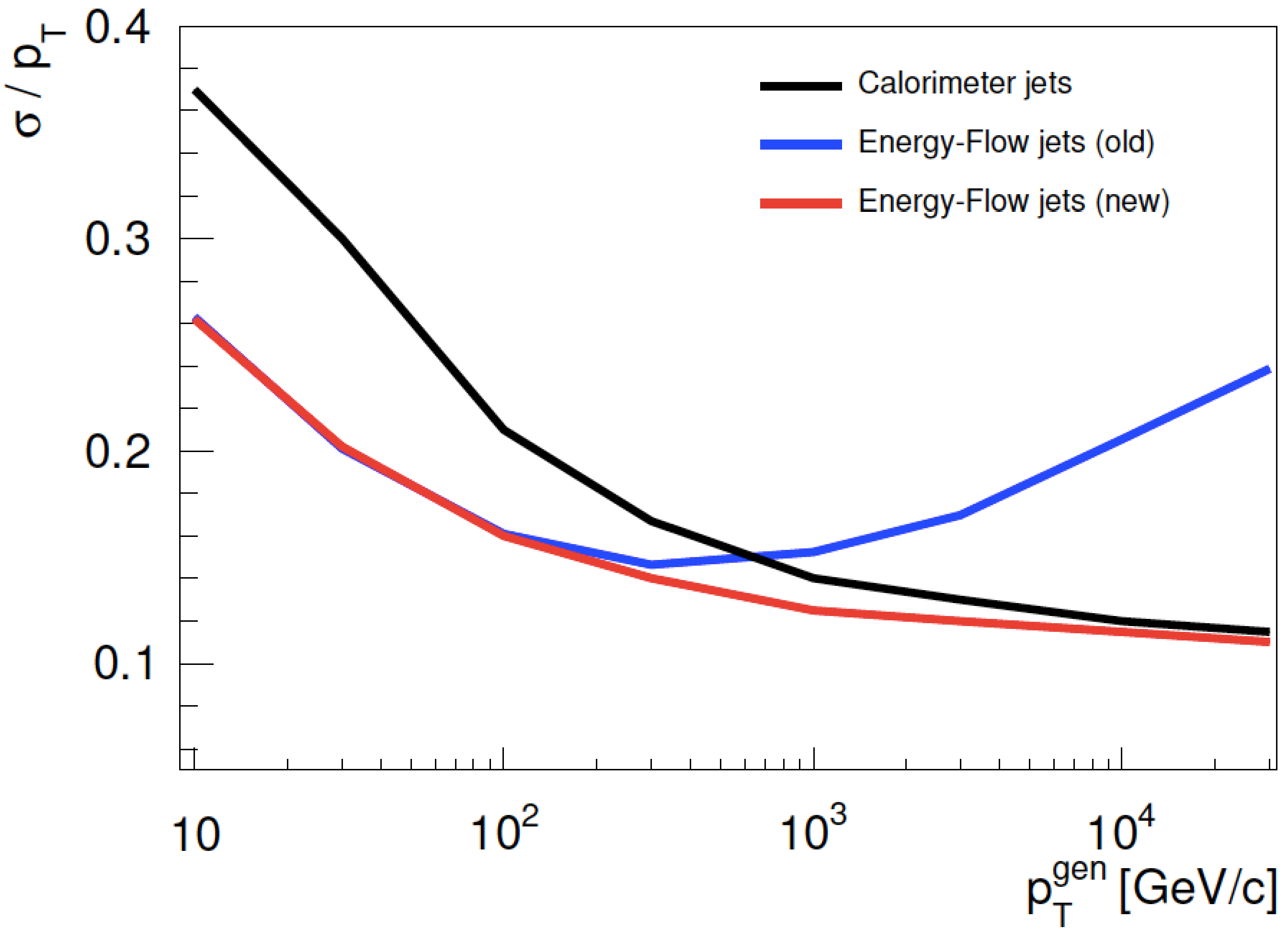
Particle-Flow algorithm creates:

PF-track, with energy $E^{PF\text{-}trk} = 550$ GeV
and no PF-tower



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

Validation



Calorimeter jets
Energy-Flow jets (old)
Energy-Flow jets (new)
σ / p_T
0.4
0.3
0.2
0.1
10
10^2
10^3
10^4
p_T^gen [GeV/c]

Leptons, photons

DELPHES fast simulation

UCL Université catholique de Louvain

- Muons/photons/electrons
  - muons **identified** via their PDG id, do not deposit energy in calo (independent smearing parameterized in p_T and η)
  - electrons and photons reconstructed according to particle-flow

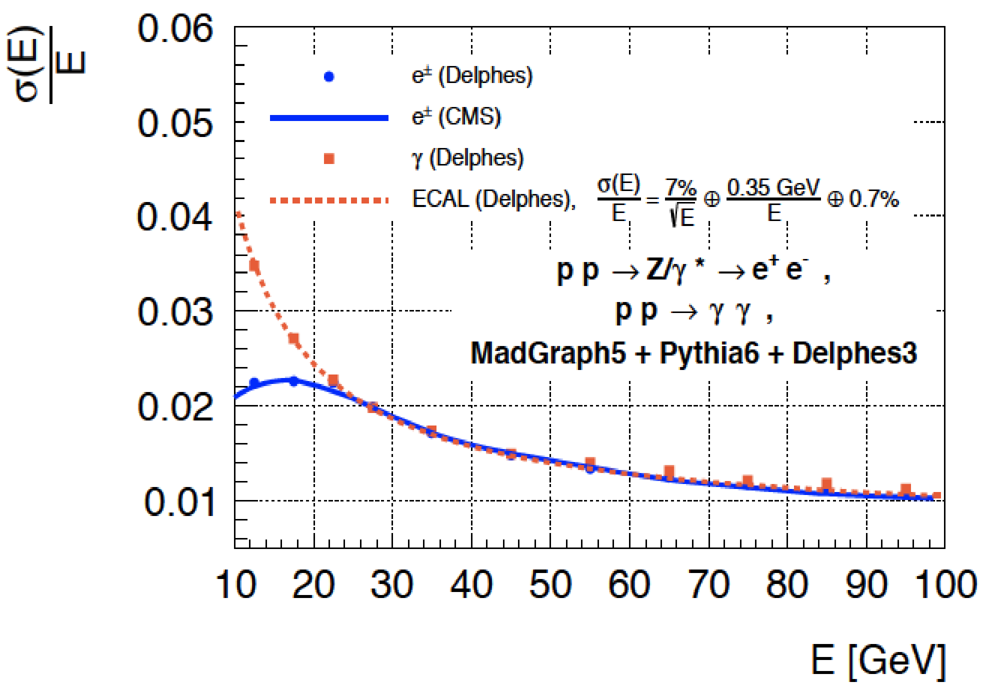
- Isolation:

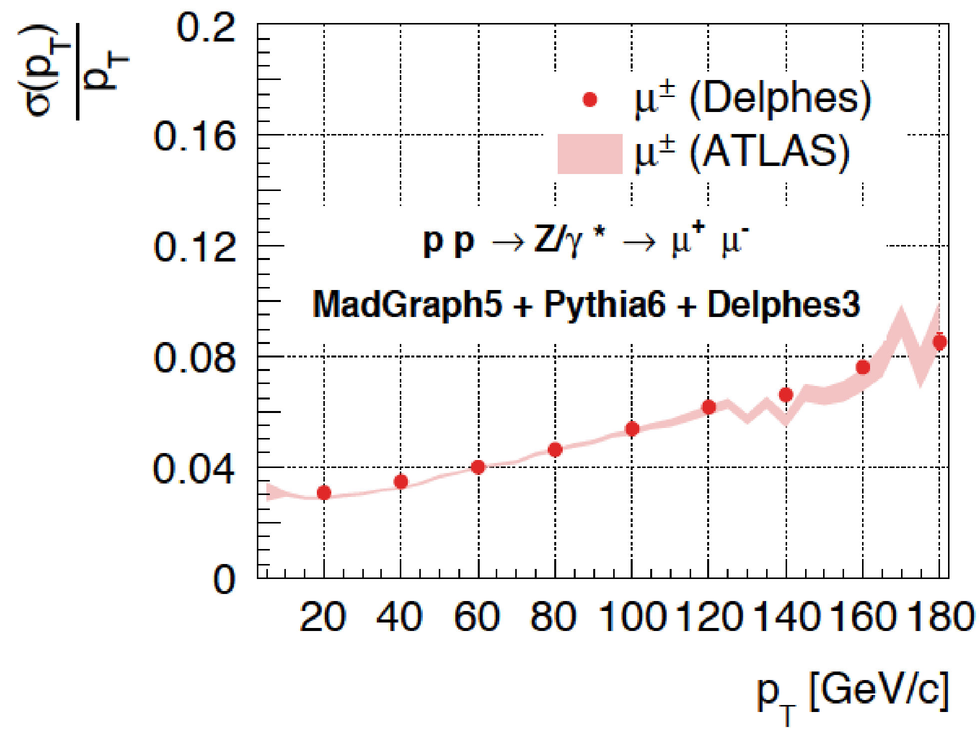
If **I(P) < Imin**, the lepton is **isolated**

User can specify parameters I_{min}, ΔR, p_T^{min}

$$I(P) = \frac{\sum_{i \neq P}^{\Delta R < R,\ p_T(i) > p_T^{min}} p_T(i)}{p_T(P)}$$

Validation





→ excellent agreement

b and τ jets

- if **b** parton is found in a cone ΔR w.r.t jet direction
  → apply **efficiency**
- if **c** parton is found in a cone ΔR w.r.t jet direction
  → apply **c-mistag rate**
- if **u,d,s,g** parton is found in a cone ΔR w.r.t jet direction
  → apply **light-mistag rate**

b-tag **flag** is then stored in the jet collection

- tau-jets

- if tau lepton is found in a cone ΔR w.r.t jet direction
  → apply **efficiency**
- else
  → apply **tau-mistag rate**

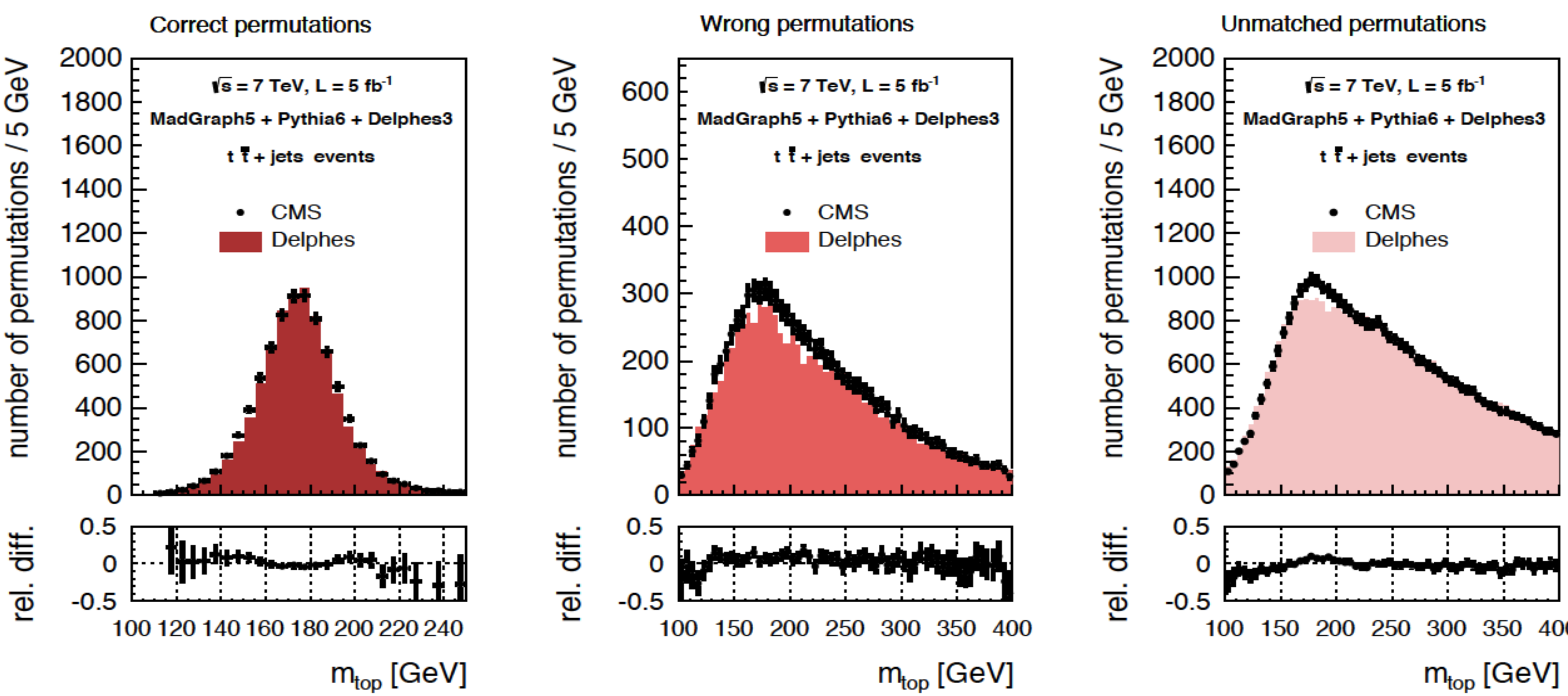
p_T and η , and n_{prong} dependent efficiency and mistag rate

Physics example

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

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

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

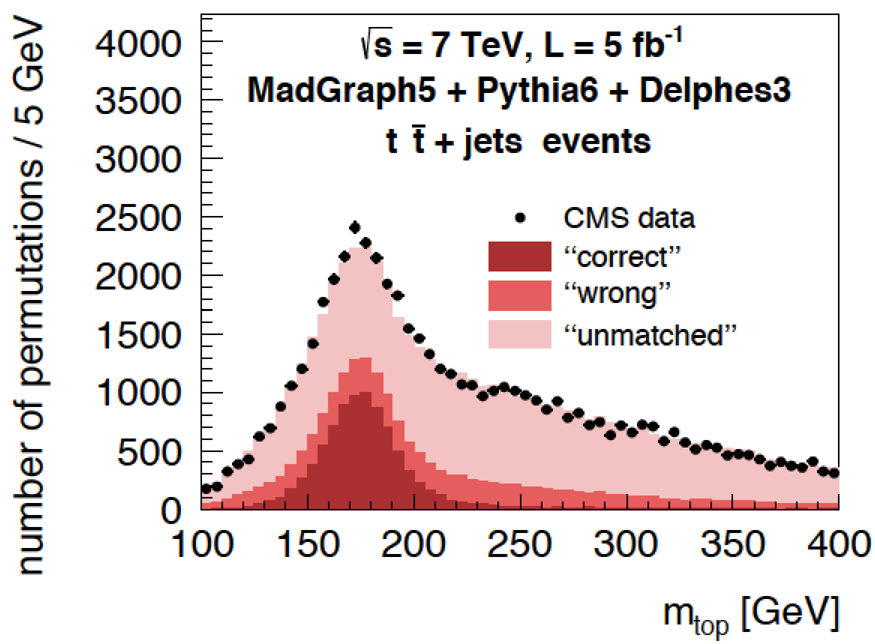


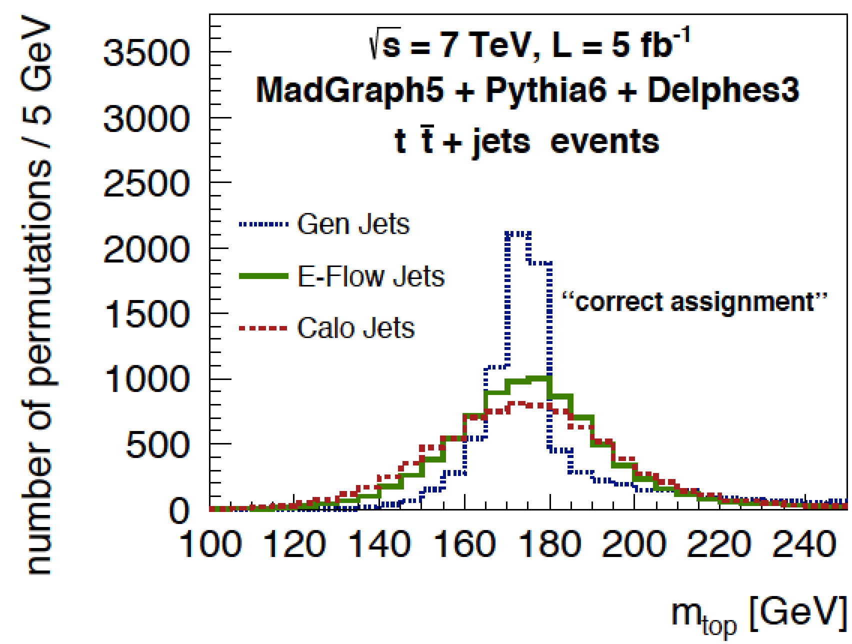
Physics example

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

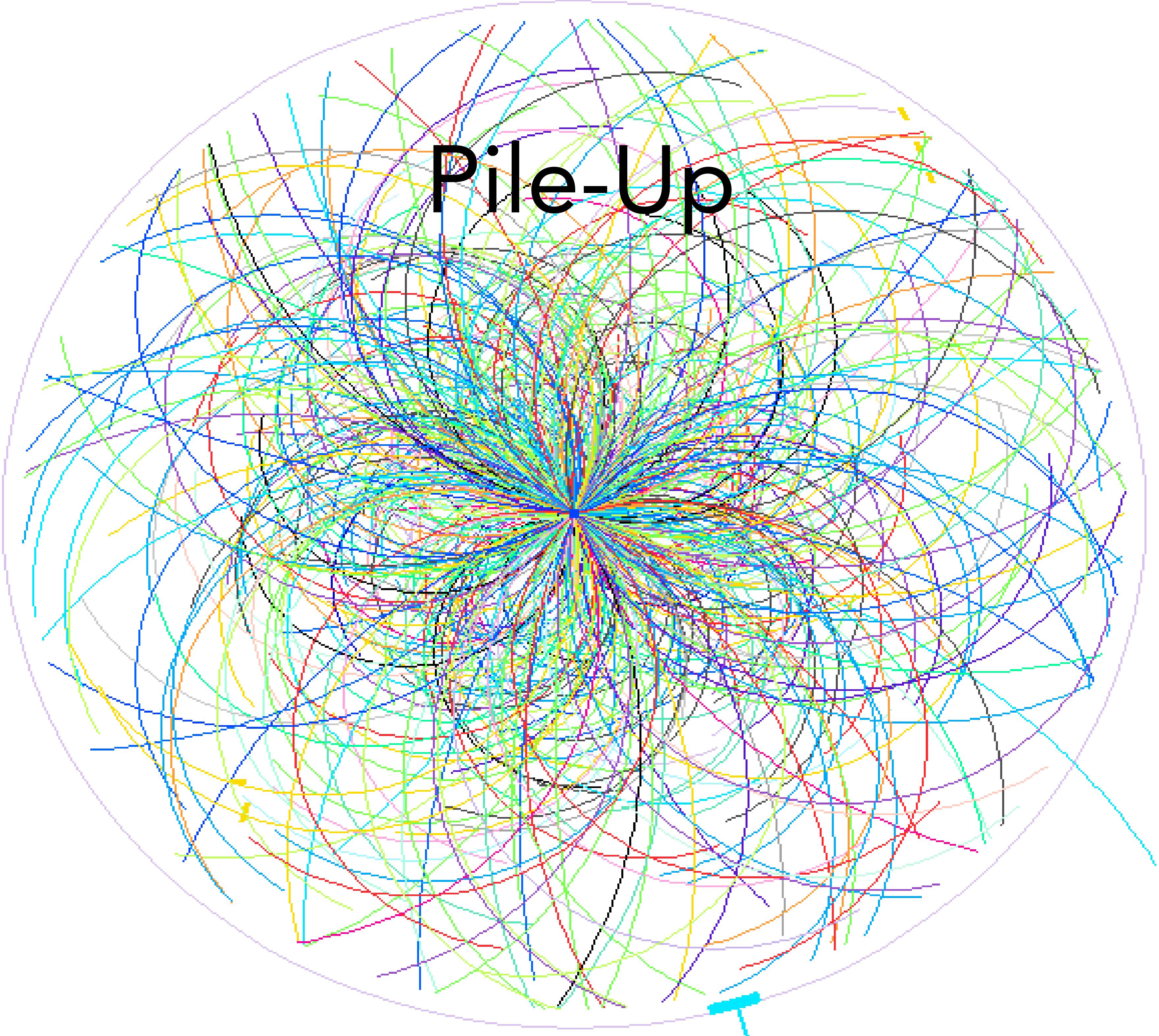
- correct : 4 jets are good, match right b with lights
- wrong : 4 jets are good, match wrong b with lights
- unmatched : at least one of the jets don't match

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



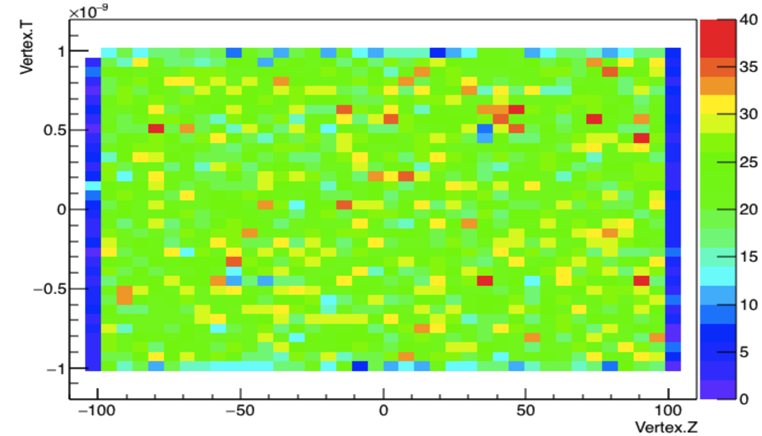


Pile-Up



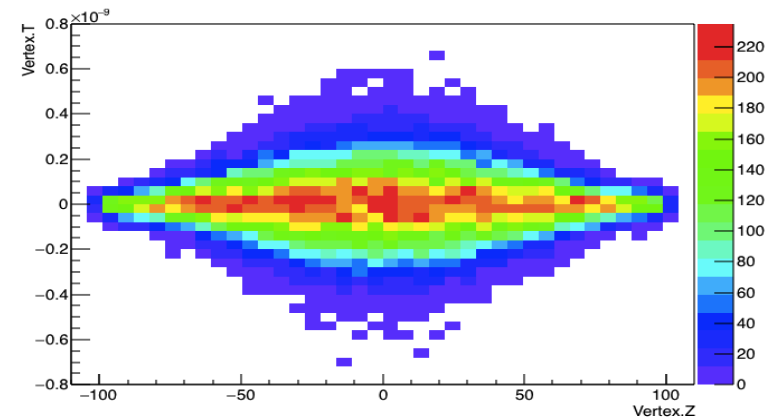
Pile - Up

Pile-up is implemented in Delphes **since version 3.0.4**



PileUpMerger module:

- **mixes** N minimum bias events with hard event sample
- spreads **poisson(N), Gauss(N)** events along z-axis with configurable (z,t) beamspot profle
- rotate event by random angle ϕ wrt z-axis



Pile – Up

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

  - if **$z < |Z_{res}|$** keep all **charged and neutrals** ($\rightarrow$ ch. particles too close to hard scattering to be rejected)

  - if **$z > |Z_{res}|$** keep only **neutrals** (perfect charged subtraction)

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

- **Residual** eta dependent pile-up substraction 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)

DELPHES
fast simulation

Pile - Up

UCL
Université catholique de Louvain

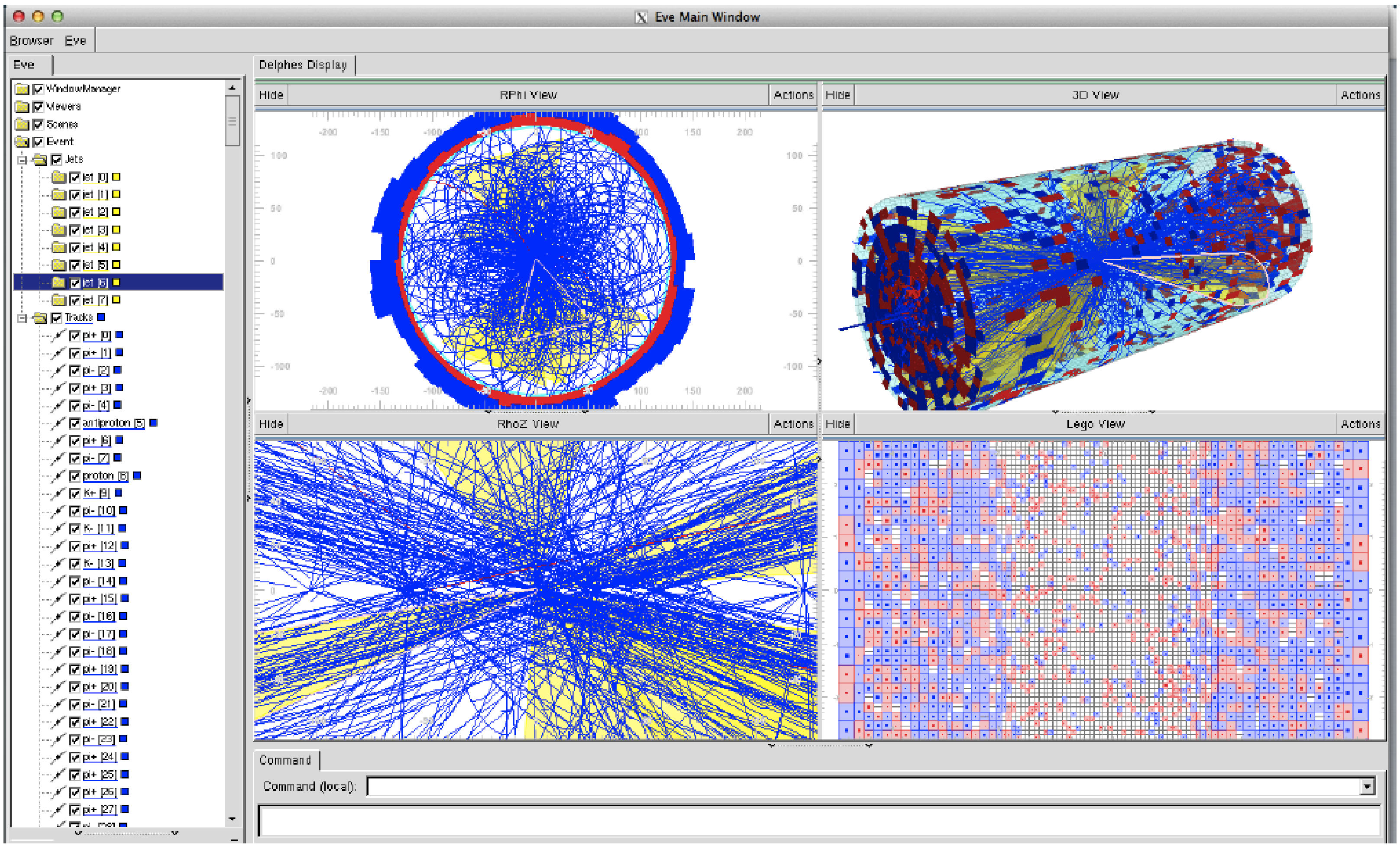
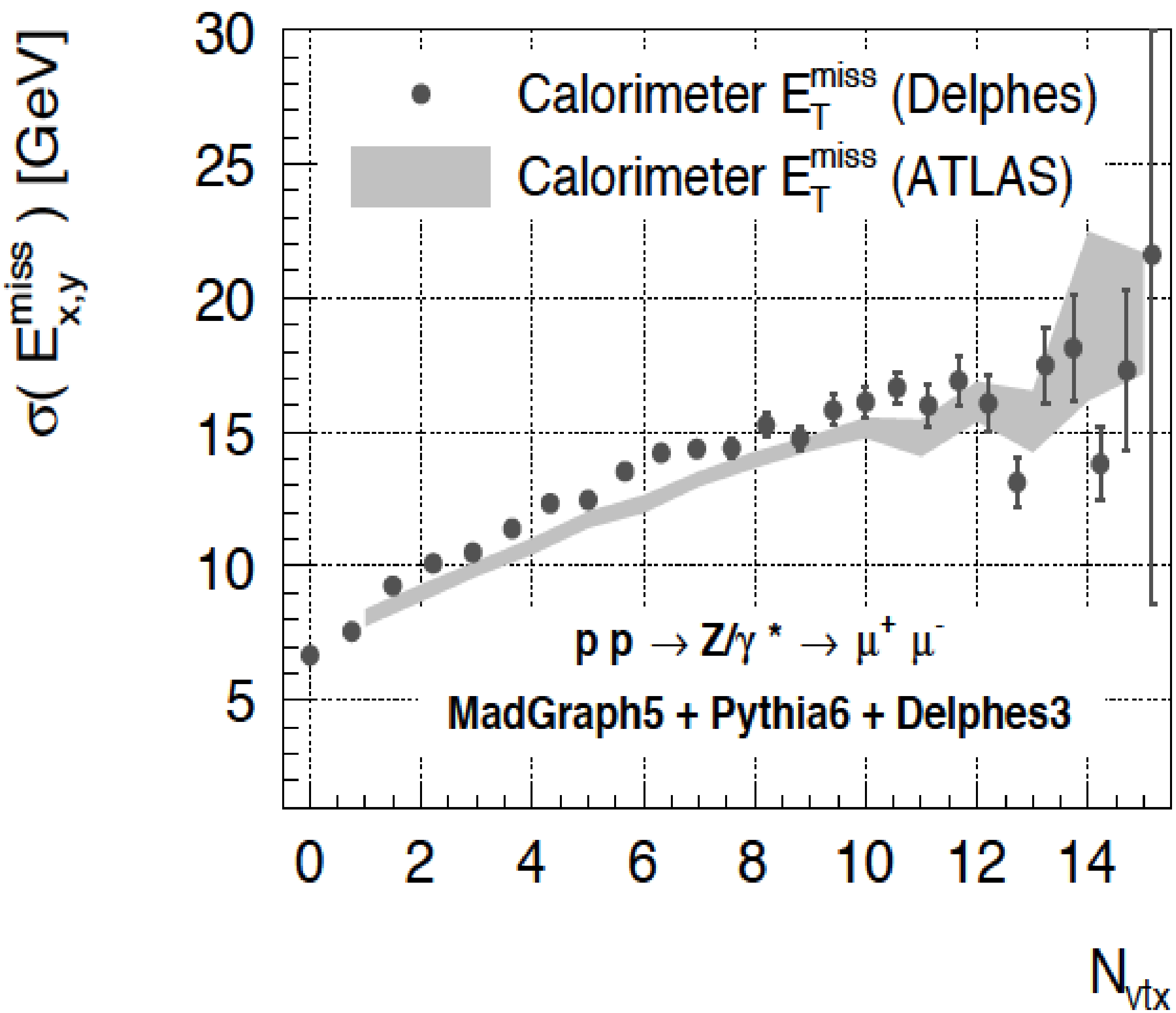


Figure 3. QCD event with 50 pile-up interactions shown with the DELPHES event display based on the ROOTEVE libraries [12]. Transverse view (top left), longitudinal view (bottom left), 3D view (top right), (η,ϕ) view (bottom right).

Pile - Up



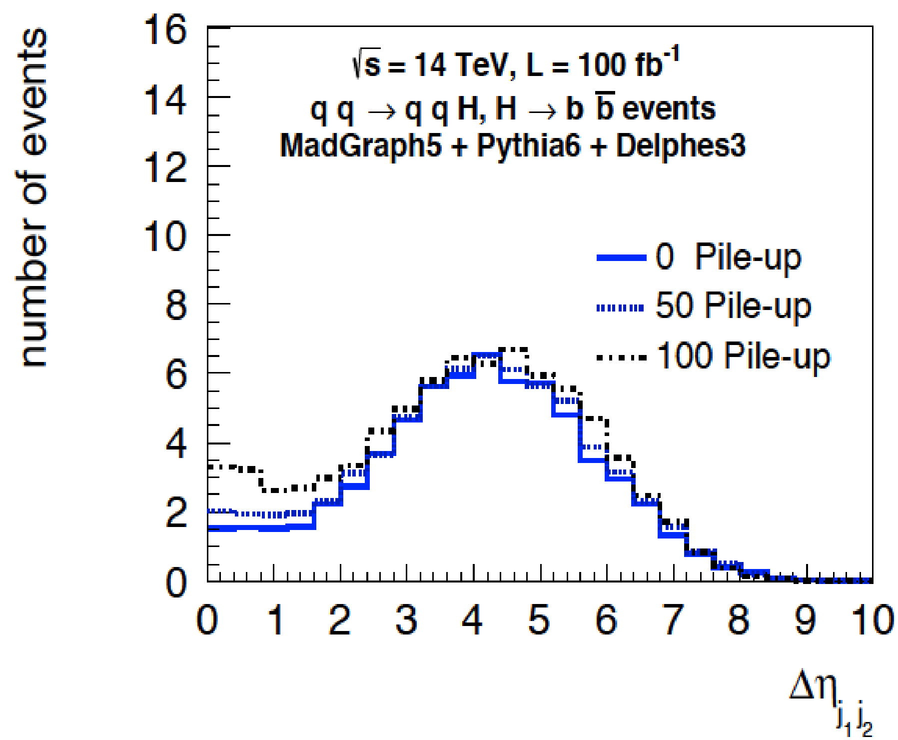
→ **good agreement**

Pile Up validation

- 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)

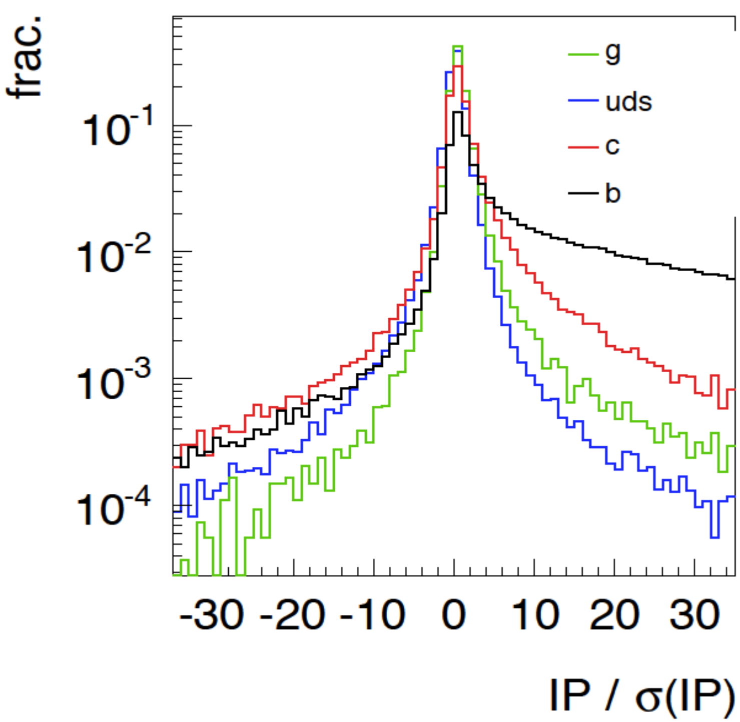
Emergence of pile-up jets in the central region:

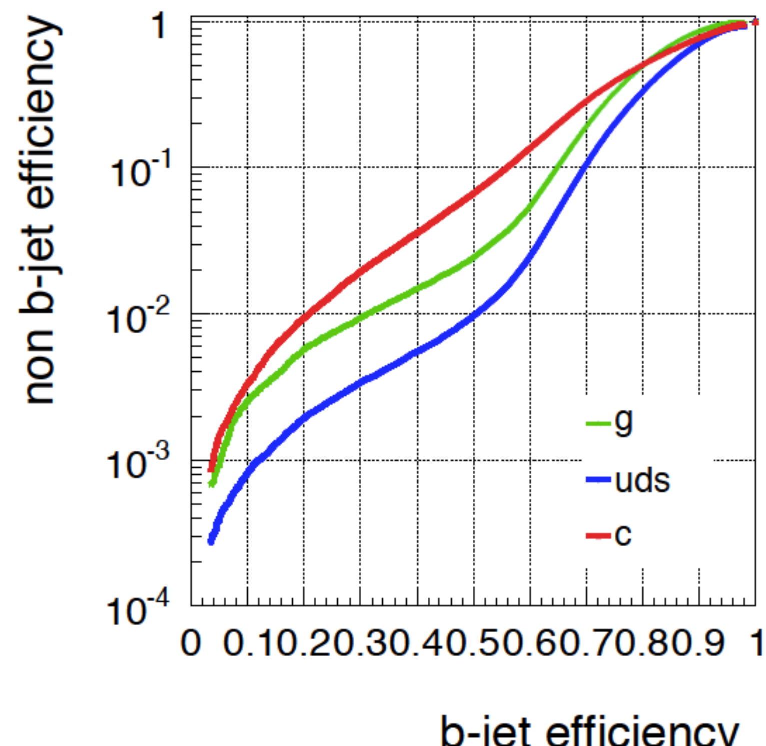
→ **depletion of rapidity gap**



TC – btagging

- Track parameters (p_T, d_{XY}, d_Z) derived from **track fitting** in real experiments
- In Delphes we can **smear** directly $\mathbf{d_{XY}}$, $\mathbf{d_Z}$ according to **($\mathbf{p_T}$, $\boldsymbol{\eta}$)** of the track
- **Count tracks** within jet with **large impact parameter** significance.





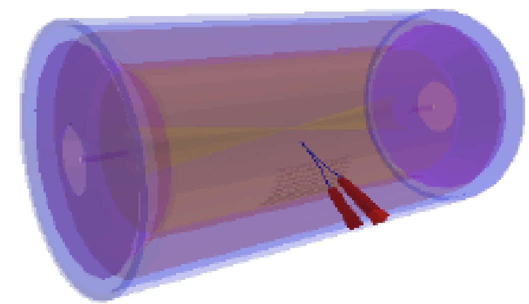
→ although very simple is predictive

→ ignore correlations among track parameters

Photon Conversions

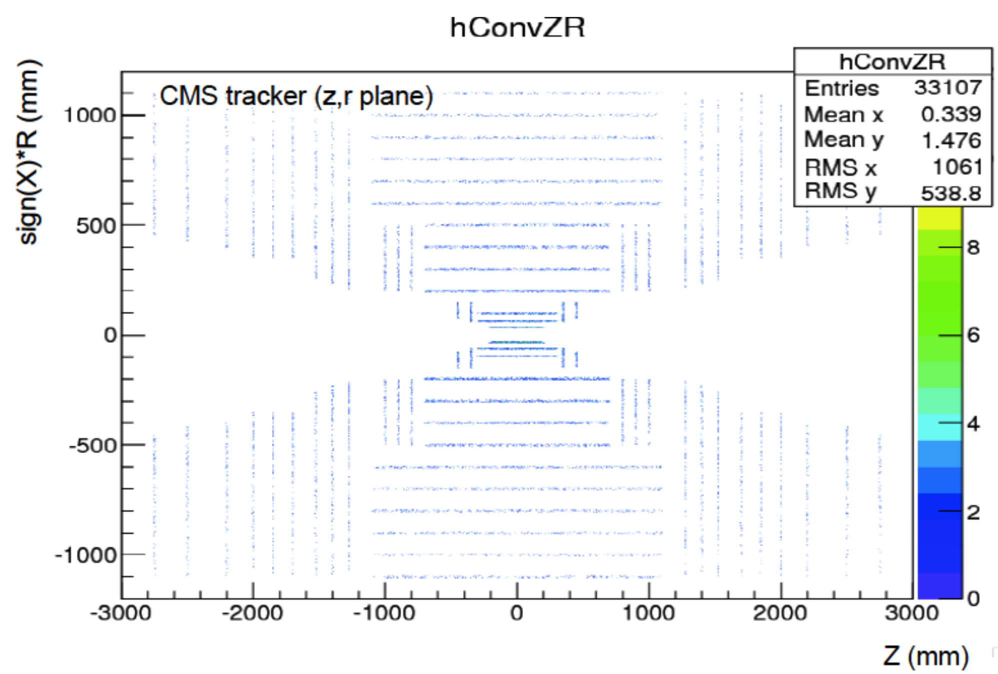
- probability of converting after distance "Δx"

$$P(\text{conv. after } \Delta x) = 1 - \exp(-\Delta x / \lambda)$$



1) material budget map can be provided via

$\lambda^{-1}(r, z, phi)$ = average conversion rate per unit length (m^{-1})
$= 7/9 * \rho / X_0$



2) step length "Δx"

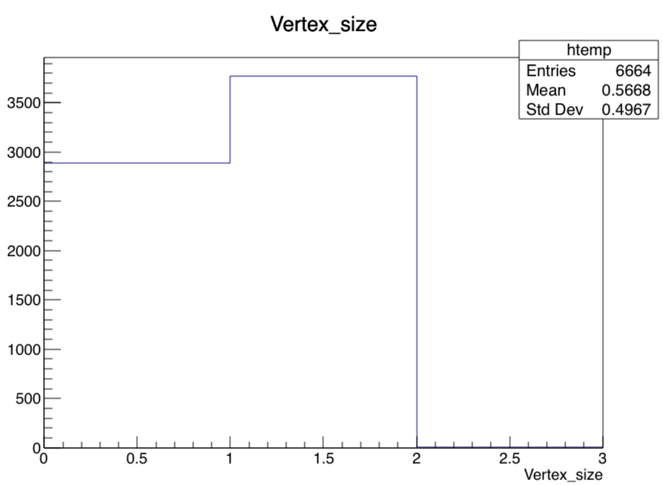
3) the photon annihilation cross-section

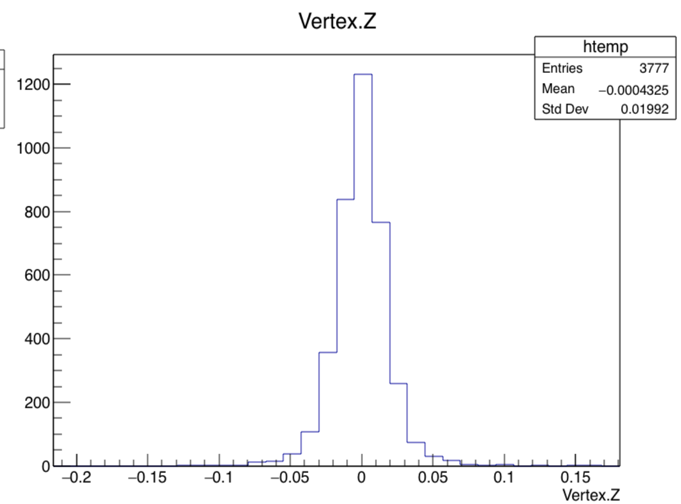
$$d\sigma / dx \sim 1 - 4/3\, x\,(1 - x)$$

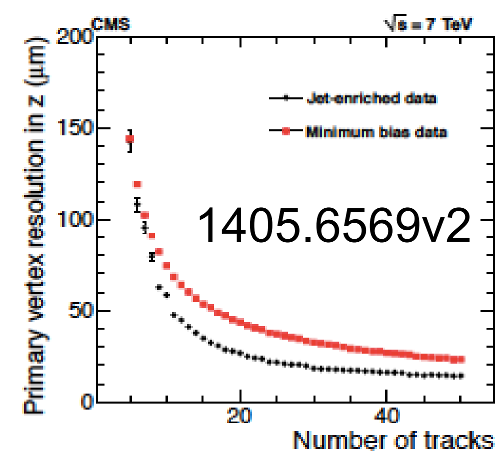
More info:
https://cp3.irmp.ucl.ac.be/projects/delphes/raw-attachment/wiki/WorkBook/Modules/delphes_conversions.pdf

VertexFinder

- The algorithm:
  - Find every track with $p_\mathrm{T} > \texttt{SeedMinPT}$ (we use $\texttt{SeedMinPT} = 1\,\mathrm{GeV}$); these are used as seeds to grow clusters.
  - Starting from the highest p_T seed, add all tracks with $|z - z_\mathrm{cluster}| < 2\sigma_z$; tracks are added starting from the nearest and the cluster position is updated after each track is added.
  - Reject any clusters with <2 tracks.
- The cluster with the highest $\sum p_\mathrm{T}^2$ is chosen as the PV of interest.







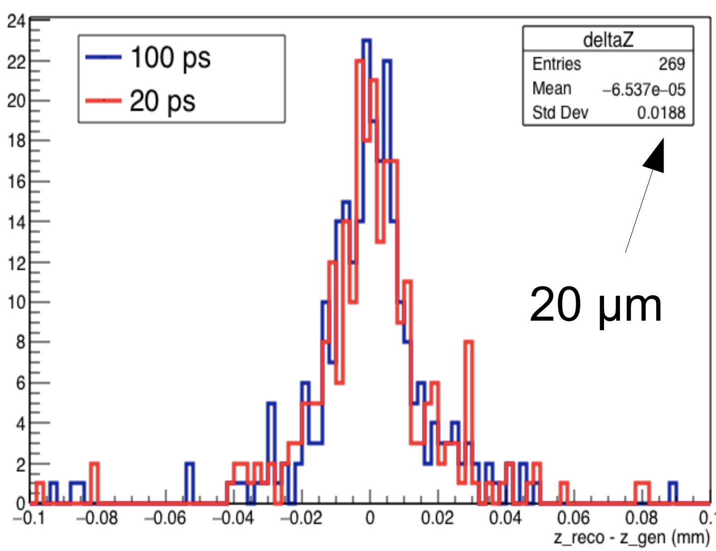
- Running "out of the box", seems low efficiency (parameters need to be tuned probably)
- Vertex resolution seems ok (CMS resolution obtained with Deterministic Annealing

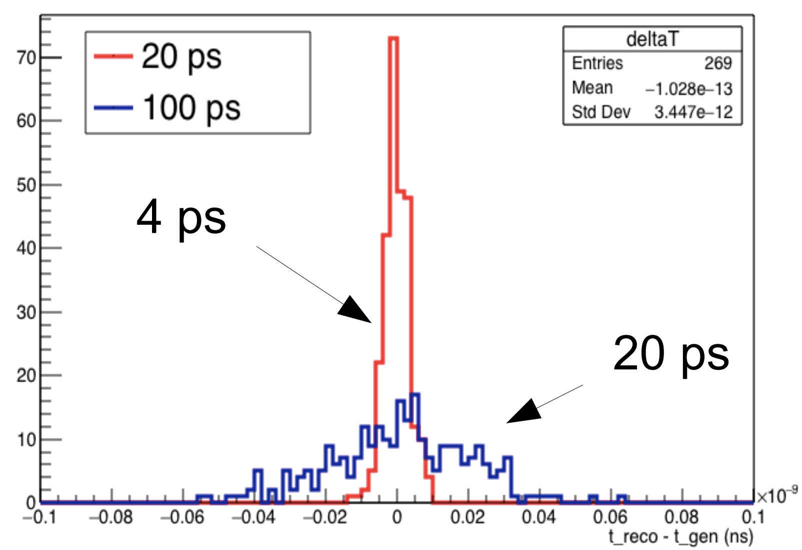
Thanks to A. Hart

VertexFinder4D

- Vertexing algorithm including time information of tracks
- Original implementation can be found in CMS software
- The DA-clusterizer in 4D is now implemented in Delphes
- Example with 160 ps x 5.3 cm beamspot and 20 ps time resolution on tracks measurement

Thanks to L. Gray





Vertex with highest $\sum p_T^2$ is taken for comparison.

Contributors

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

the community ...

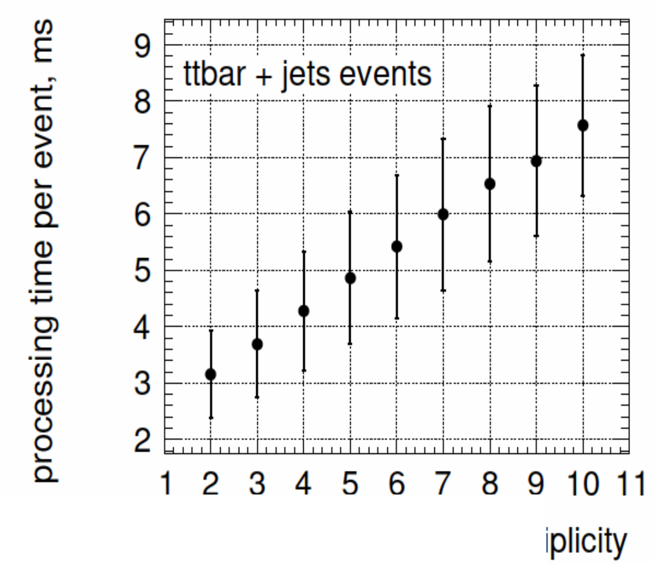
CPU time

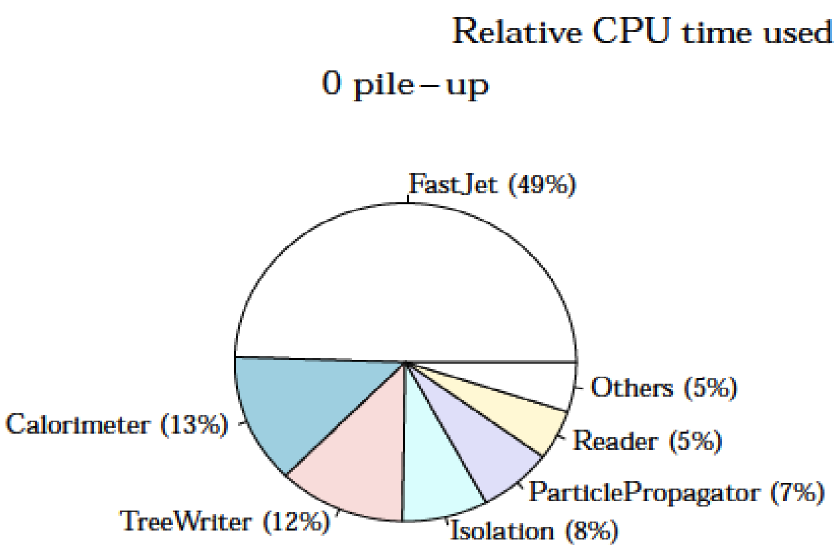
Delphes **reconstruction time** per event:

0 Pile-Up = 1 ms

150 Pile-Up = 100 ms - 1s

Mainly spent in the FastJet algorithm:





Disk usage

Disk **space** for 10k ttbar events (upper limit, store all constituents):

0 Pile-Up = 300 Mb

100 Pile-Up = 3 Gb

Mainly taken by list of MC particles and Calo towers: