Delphes 3

(lastest developments)

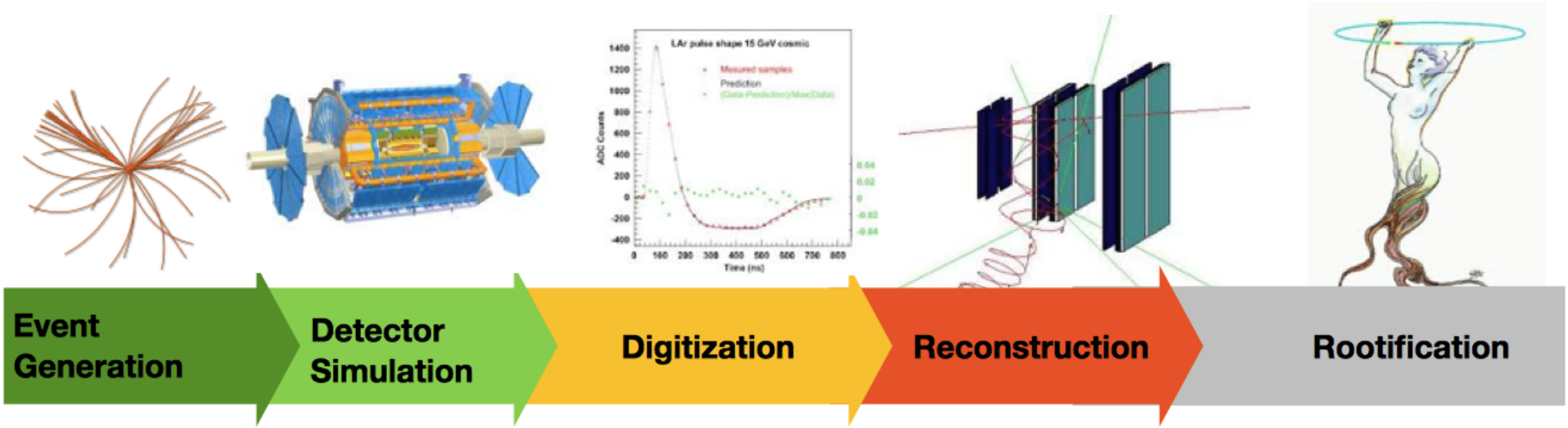
Michele Selvaggi

Université catholique de Louvain (UCL)

Center for Particle Physics and Phenomenology (CP3)

ACAT 2016 - Valparaiso
January 18th 2016

MC chain



courtesy of A. Salzburger

Detector simulation/reconstruction

- Full simulation (GEANT):

  - **simulates** particle-matter interaction (including e.m. showering, nuclear int., brehmstrahlung, photon conversions, etc ...) $\longrightarrow$ 100 s /ev

- Experiment Fast simulation (ATLAS, CMS ...):

  - **simplifies** and makes faster simulation and reconstruction $\longrightarrow$ 1 s /ev

- Parametric simulation:

  Delphes, PGS:

  - **parameterize** detector response, reconstruct complex objects $\longrightarrow$ 10 ms /ev

When and when not FastSim?

- When to use FastSim?

  → quick phenomenological studies
  → **sensitive to acceptance and complex observable (Jets,MET)**
  → scan big parameter space (SUSY-like)
  → preliminary tests of new geometries/resolutions (jet substructure)
  → educational purpose (bachelor/master thesis)

- When not to use FastSim?

  → high precision studies
  → very exotic topologies (heavy stable charged particles)

The Delphes Project

The Delphes project: A bit of history and status

- 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 (and interfaced with Pythia8)

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

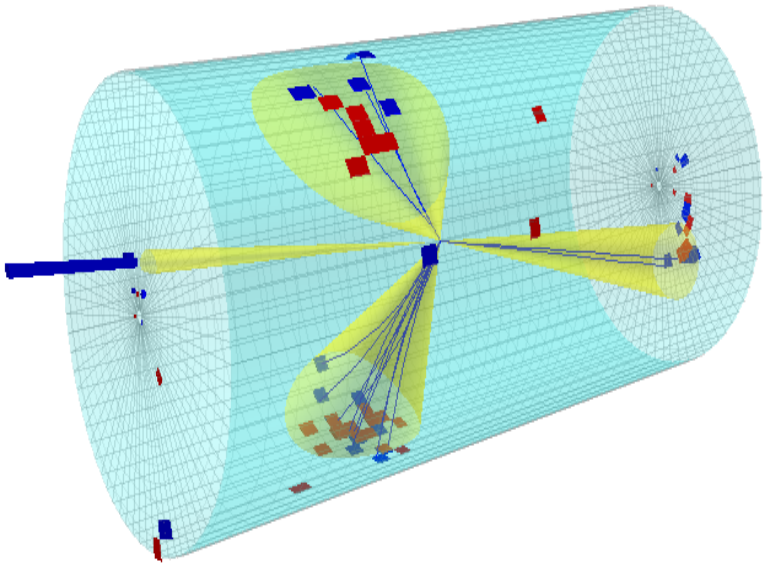
- Website and manual: https://cp3.irmp.ucl.ac.be/projects/delphes

- Paper: JHEP 02 (2014) 057

The Delphes project: Delphes in a nutshell

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

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



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

Latest Developments

- As of 2014, the **modular structure** was widely tested and stable

- Main developments go in the direction of **extending the reach** of Delphes

- Be able to describe **non trival geometries** (such as LHCb, under development) ...

  ... and detectors for **future colliders** (ILC, FCC ..)

- More subtle simulation and reconstruction features:

  → particle (mis-)identification map
  → separate granularity for ECAL and HCAL
  → **photon conversions**
  → **high energy particle-flow**

- Easy **interface** to MC event generators, or other hep tools

  → can run Pythia8 within Delphes
  → call Delphes as an external library (e.g FastJet)

Particle Flow

Particle-Flow Emulation

- In a **multipurpose** detector:
  - **tracking** provides **good** measurement of momenta at **low energy**
  - **calorimeter** provides **good** measurement of momenta at **high** energy

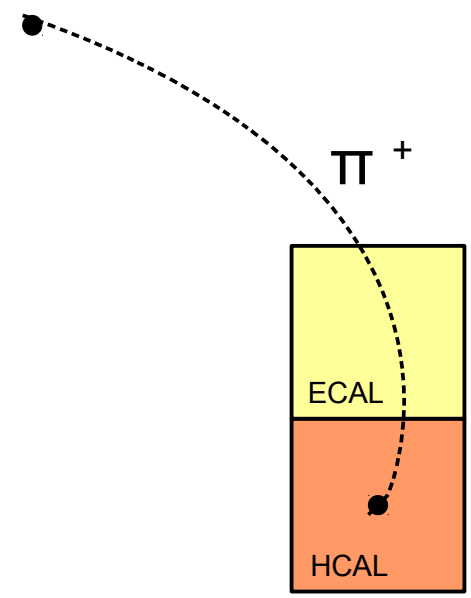
- In practice, PF means using optimally **tracking and calo** info to reconstruct high reso. input objects for later use (jets, E_T^{miss}, H_T)

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

Example: A charged pion of 10 GeV in ATLAS/CMS

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

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



Particle-Flow algorithm creates:

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

Particle-Flow Emulation

- In a **multipurpose** detector:
  - **tracking** provides **good** measurement of momenta at **low energy**
  - **calorimeter** provides **good** measurement of momenta at **high** energy

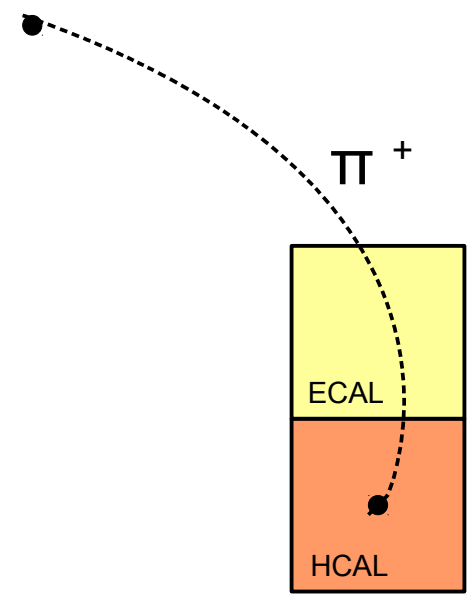
- In practice, PF means using optimally **tracking and calo** info to reconstruct high reso. input objects for later use (jets, E_T^{miss}, H_T)

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

Example: A charged pion of 10 GeV in ATLAS/CMS

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

- In a multipurpose detector:
  - **tracking** provides **good** measurement of momenta at **low energy**
  - **calorimeter** provides **good** measurement of momenta at **high** energy

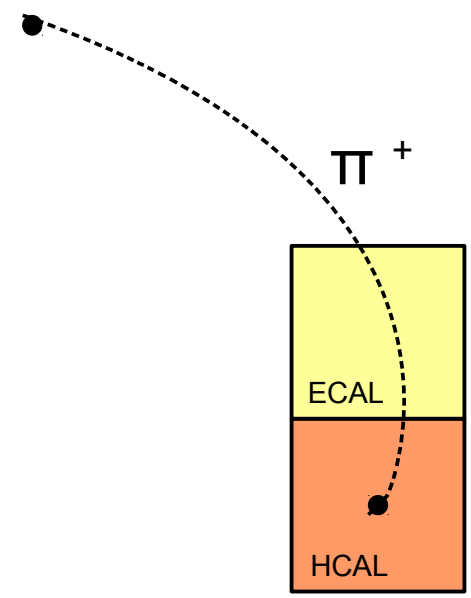
- In practice, PF means using optimally **tracking and calo** info to reconstruct high reso. input objects for later use (jets, E_T^{miss}, H_T)

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

Example: A charged pion of 500 GeV in ATLAS/CMS

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

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



Particle-Flow algorithm creates:

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

Particle-Flow Emulation

The general algorithm:

- For a given tower we have a total deposit E^{CALO}

- We also have:

→ N_{good} tracks, $\sigma(trk) < \sigma(calo)$, low p_T, depositing total energy E^{TRK}_{GOOD} ,
→ N_{bad} tracks, $\sigma(trk) > \sigma(calo)$, high p_T, depositing total energy E^{TRK}_{BAD} ,

- If <u>only</u> "good" tracks, $N_{bad} = 0$ **(ex: high-pile-up environment)**:

→ compute **excess energy neutral energy** $\Delta E = \max(E^{CALO} - E^{TRK}_{GOOD}, 0)$
if $\Delta E > E_{min}$ (E_{min} = **minimal energy** set by user, typically ~ 1 GeV)
and if $\Delta E / \sigma_{CALO}(\Delta E) > S_{min}$ (S_{min} = **minimal significance** set by user ~ 1.0)

then create the following objects:

→ 1 *Eflow Tower* of energy $E^{TOWER} = E^{CALO} - E^{TRK}_{GOOD} = \Delta E$ (neutral deposit)
→ N_{good} *Eflow Tracks* of original track energy

Particle-Flow Emulation

- If at least one "bad" track, $N_{bad} > 0$, (ex: highly boosted jet):

  → can't say if excess $\Delta E = \max(E^{CALO} - E^{TRK}_{GOOD}, 0)$ is really due to neutral excess or simply to poor tracking resolution.

  → use superior calorimeter resolution to assess momenta of "bad" tracks:

Rescale 4-momenta of "bad" tracks by $f = (E^{CALO} - E^{TRK}_{GOOD}) / E^{TRK}_{BAD}$

Then create the following objects:

→ no *Eflow Towers*

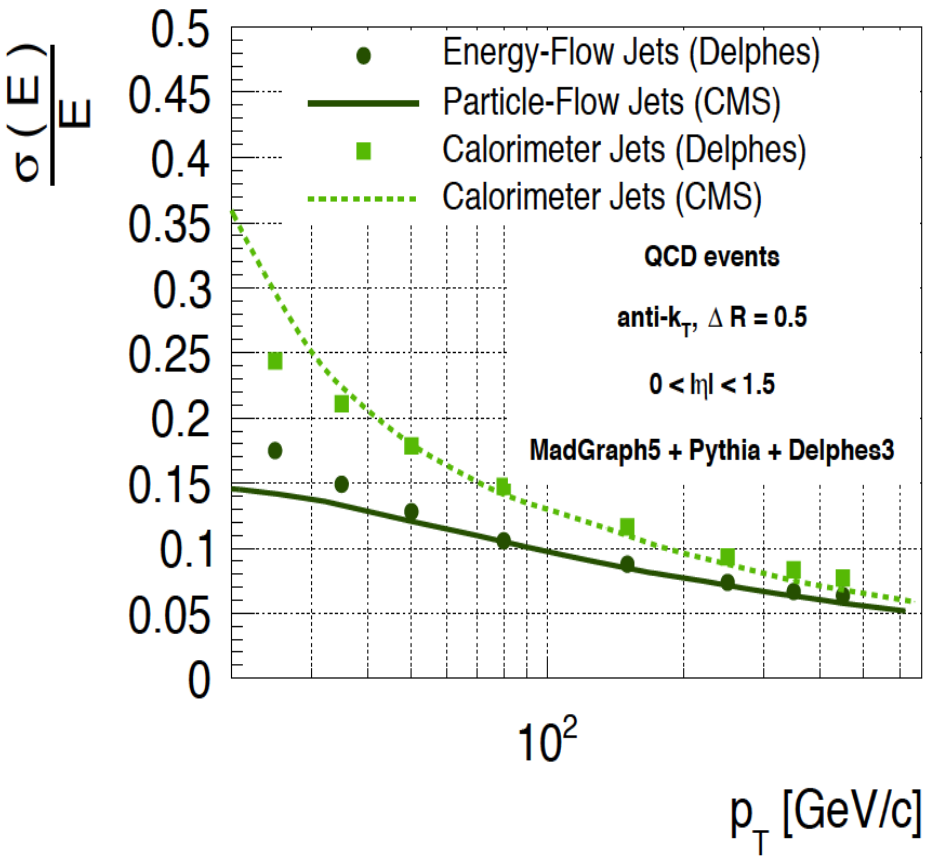
→ N_{good} *Eflow Tracks* of original track energy

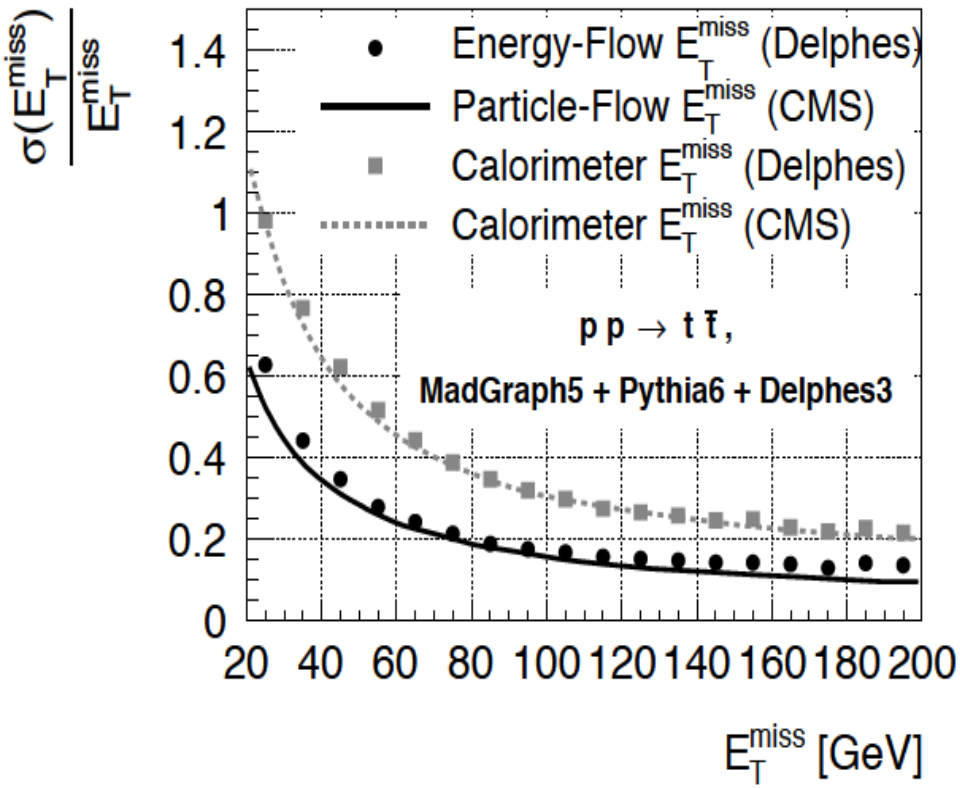
→ N_{bad} *Eflow Tracks* with energy $E_{trk}' = f\, E_{trk}$

- By construction we then have **total (calorimeter) energy conservation:** $\sum E^{TRK}_{BAD}(i) = E^{CALO} - E^{TRK}_{GOOD}$
- We still use the **optimal angular resolution** of the **tracking** system

Present approach assumes we can reconstruct flat tracking reconstruction as a function of "hit density".

Validation: Particle-Flow





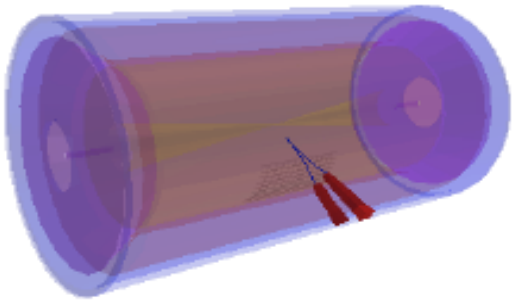
→ good agreement

Photon Conversions

Photon Conversions

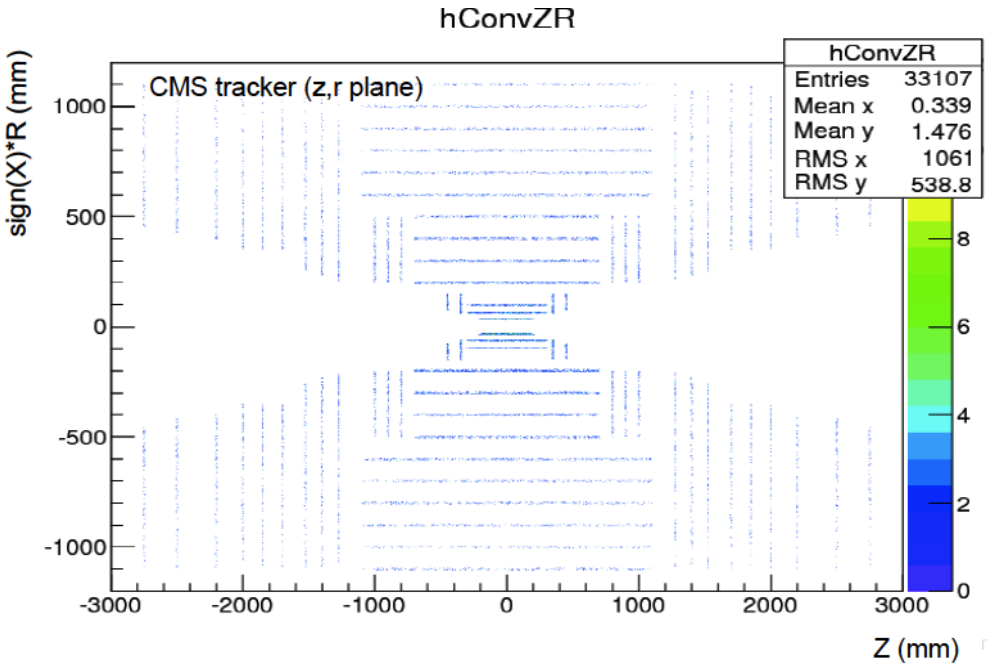
- probability of converting after distance "Δx"

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



1) material budget map can be provided via

λ^{-1} (r , z , phi) = average conversion rate per unit length (m^{-1})



2) step length "Δx"

3) the photon annihilation cross-section

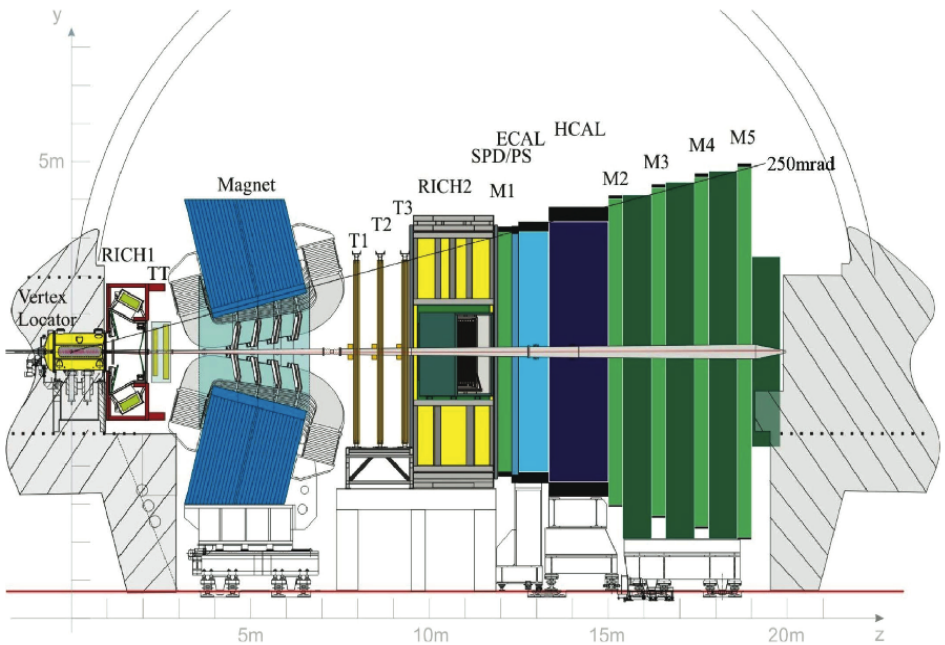
$$d\sigma / dx \sim 1 - 4/3\ x\,(1 - x)$$

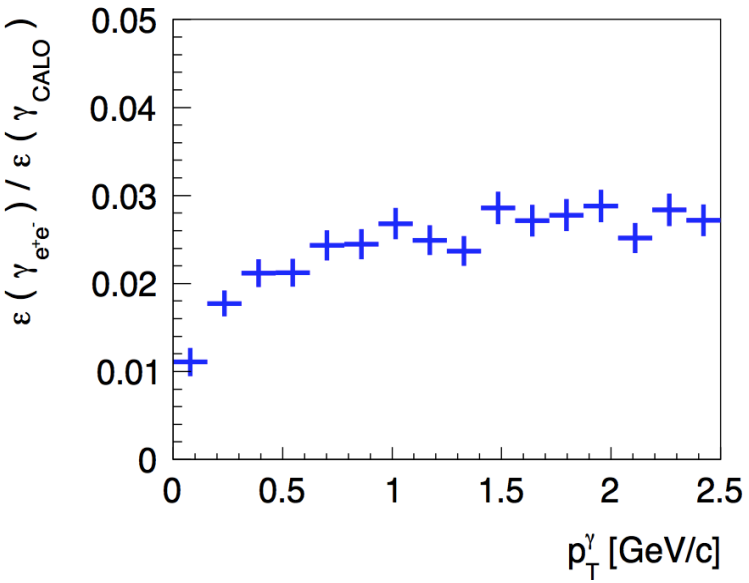
More info:
https://cp3.irmp.ucl.ac.be/projects/delphes/raw-attachment/wiki/WorkBook/Modules/delphes_conversions.pdf

Photon Conversions

- LHCb is good playground for conversions, because of **non ermetic acceptance**:

→ expect low momentum e^+e^- pairs from conversions in early stages of the tracker, to not make it to the calorimeters.

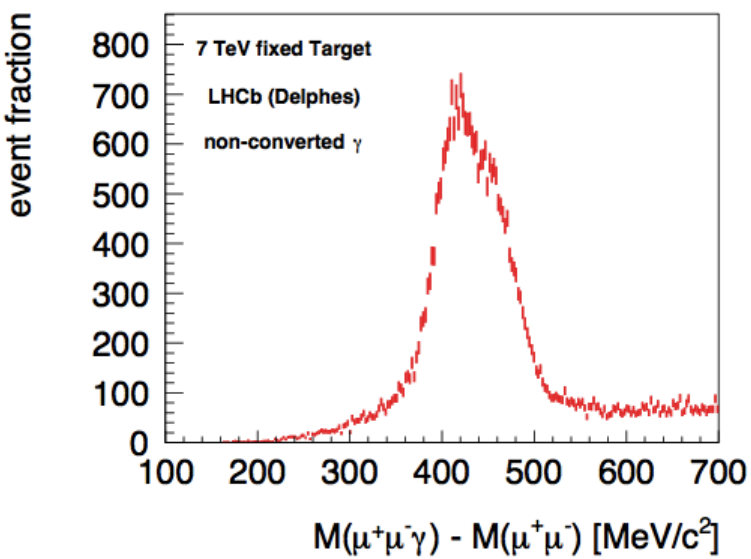


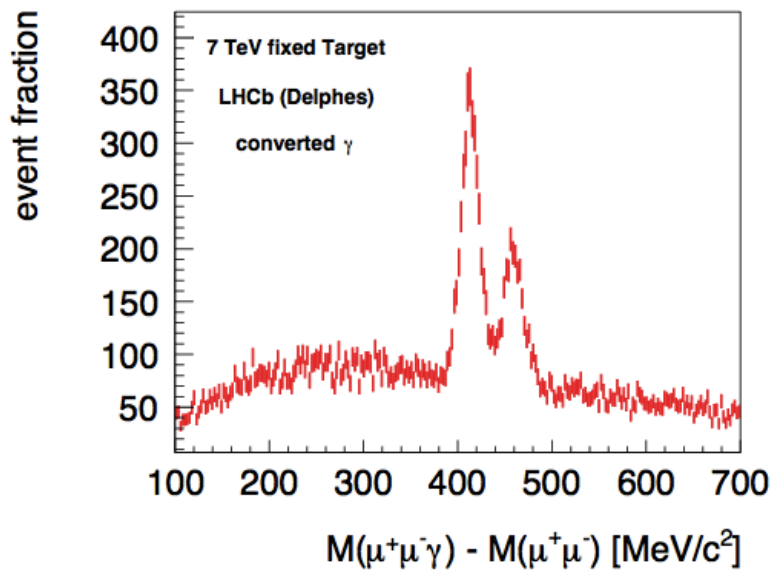


More info:
https://cp3.irmp.ucl.ac.be/projects/delphes/raw-attachment/wiki/WorkBook/Modules/delphes_conversions.pdf

Photon Conversions

- converted photons can be used to improve photon momentum resolution
- useful for resolving overlapping resonances (χ_{c1}, χ_{c2})





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

Delphes and Future colliders

Delphes and future colliders

- Delphes has been designed to deal with **high number of hadrons** environment:

  - Jets, MET and object isolation are modeled realistically
  - pile-up subtraction (FastJet Area method, Charged Hadron Subtraction)
  - pile-up JetId

- Recent improvements

  - **different segmentation** for ECAL and HCAL
  - Impact parameter smearing: allow for **predictive b-tagging** (now parametrized)
  - **jet substructure** for boosted objects (N-(sub)jettiness)
  - Included configuration card for future collider studies
  - Improved Particle-Flow description at high energy
  - Embed Pythia8 parton shower inside Delphes simulation

Delphes and future colliders

Delphes can be used **right-away** for **future colliders** studies ...

<u>What can you do with Delphes?</u>

- 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 structure, multiplicities ...

<u>In which context?</u>

- **preliminary physics studies** can be performed in **short time**

- can be used **in parallel** with full detector simulation

- flexible software structure allows **integration** in other frameworks
  (can be called from others programs, see manual)

Conclusions

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

  - modularity
  - pile-up
  - visualization tool based on ROOT EVE
  - default cards giving results on par with published performance from LHC experiments
  - fully integrated within MadGraph5/Py8
  - updated configurations for future e+e- and hh colliders

- Delphes 3 can be used right away for fast and realistic simulation

- Continuous development (new detector cards, improved Particle Flow reconstruction, photon conversions ...)

Website and manual:

https://cp3.irmp.ucl.ac.be/projects/delphes

People

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

the community ...

Back-up

Run delphes ...

Running Delphes

- Install ROOT (and load environment):

```
source [path-to-root-installation]/bin/thisroot.sh
```

- Download, unpack and install latest Delphes version

```
wget http://cp3.irmp.ucl.ac.be/downloads/Delphes-3.3.1.tar.gz
tar xzvf Delphes-3.3.1.tar.gz
cd Delphes-3.3.1
make -j 4
```

- To run you need an hadron-level input file (produced by MG+Py/Herwig). Delphes accepts both *.hep or *.hepmc format.

  You can download a small example sample from here (or generate one):

```
wget http://cp3.irmp.ucl.ac.be/downloads/z_ee.hep.gz
gunzip z_ee.hep.gz
```

- And run with the default CMS detector card:

```
./DelphesSTDHEP cards/delphes_card_CMS.tcl delphes_output.root z_ee.hep
```

- Follow README file for a quick start tutorial, starting from section “Simple analysis [...]”

Running Delphes

- Install ROOT (and load environment):

```
source [path-to-root-installation]/bin/thisroot.sh
```

- Download, unpack and install latest Delphes version

```
wget http://cp3.irmp.ucl.ac.be/downloads/Delphes-3.3.1.tar.gz
tar xzvf Delphes-3.3.1.tar.gz
cd Delphes-3.3.1
make -j 4
```

- To run you need an hadron-level input file (produced by MG+Py/Herwig). Delphes accepts both *.hep or *.hepmc format.

  You can download a small example sample from here (or generate one):

```
wget http://cp3.irmp.ucl.ac.be/downloads/z_ee.hep.gz
gunzip z_ee.hep.gz
```

- And run with the default CMS detector card:

```
./DelphesSTDHEP cards/delphes_card_CMS.tcl output.root z_ee.hep
```

detector card — output file — input event file