Delphes

Framework for fast simulation of a generic collider experiment

Michele Selvaggi, for the Delphes Team

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

January 14, 2013

Detector simulation

- Full simulation (GEANT)

  - **simulates** particle-matter interaction (including e.m. showering, nuclear int., brehmstrahlung, photon conversions, etc ...) $\rightarrow$ 10 s /ev

- Fast simulation (ATLAS, CMS)

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

- Parametric simulation (PGS, Delphes)

  - **parameterize** detector response $\rightarrow$ 10 ms /ev

- Other (TurboSim)

  - **no detector**, build giant lookup table parton $\leftrightarrow$ reco

Parametric simulation

- What do we **expect** from **parametric detector simulation** ?

  - fast
  - realistic enough
  - flexible detector geometry
  - user-friendly

- Who needs it ?

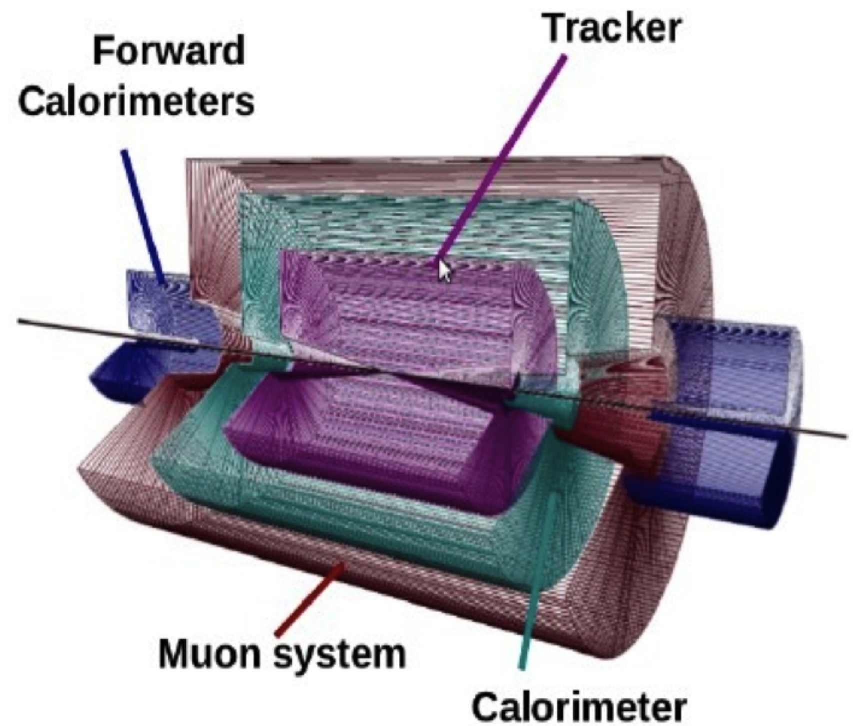
  → more advanced than parton-level studies
  → scan big parameter space (SUSY-like ...)
  → preliminary tests of new geometries/resolutions (upgrades ..)
  → testing analysis methods (multivariate/Matrix Element)
  → educational purpose (master thesis)

What is Delphes?

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

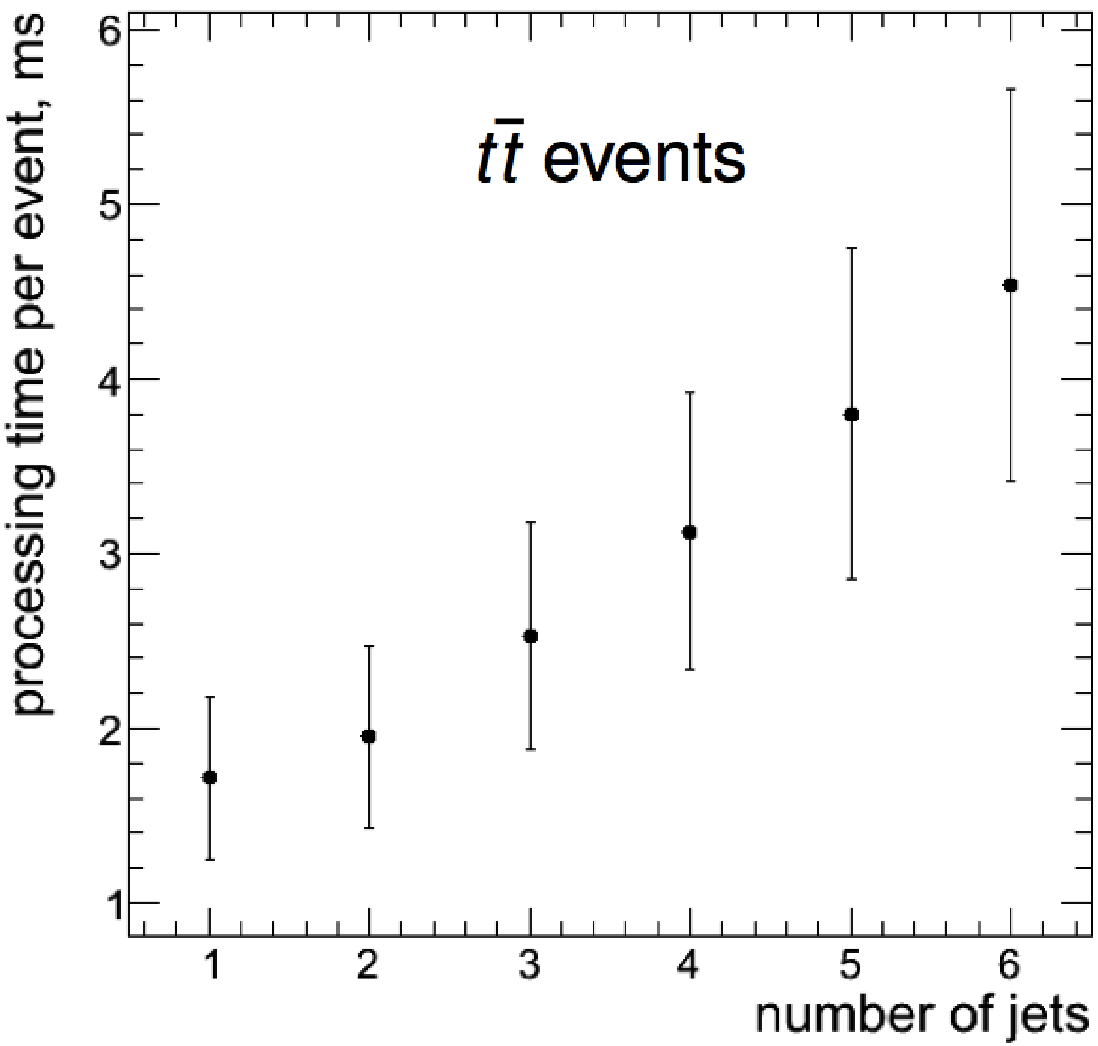
- **simulates**:
  - charged particle propagation in magnetic field: **tracking**
  - electromagnetic and hadronic **calorimeters**
  - **muon** system

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



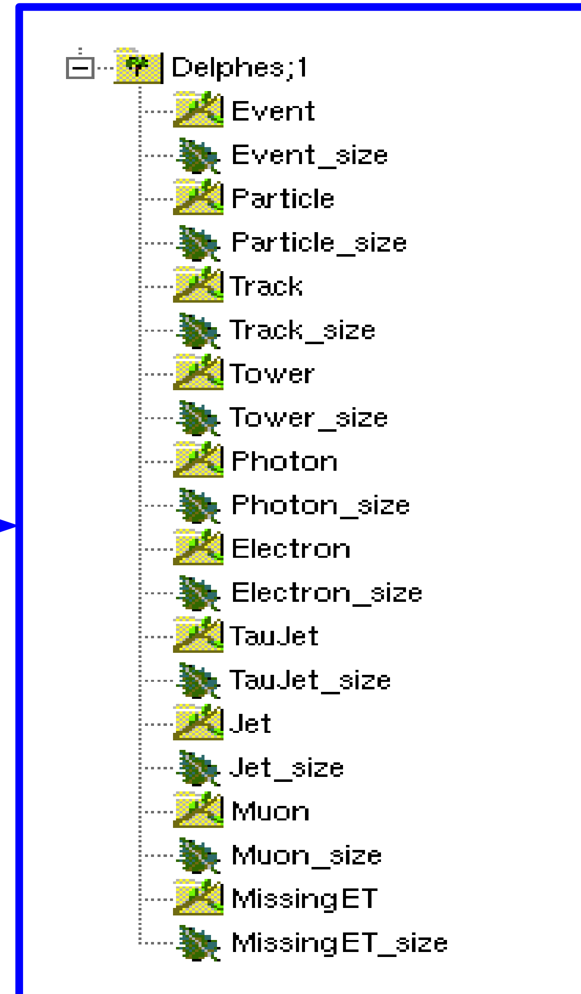
Speed

Processing time with a standard laptop



Technical features

- C++ code, uses ROOT classes
- Input
  - Pythia/Herwig output (HepMC,STDHEP)
  - LHE (MadGraph/MadEvent)
- Output
  - ROOT trees
- Configuration file
  - define geometry
  - reconstruction/selection criteria
  - output object collections

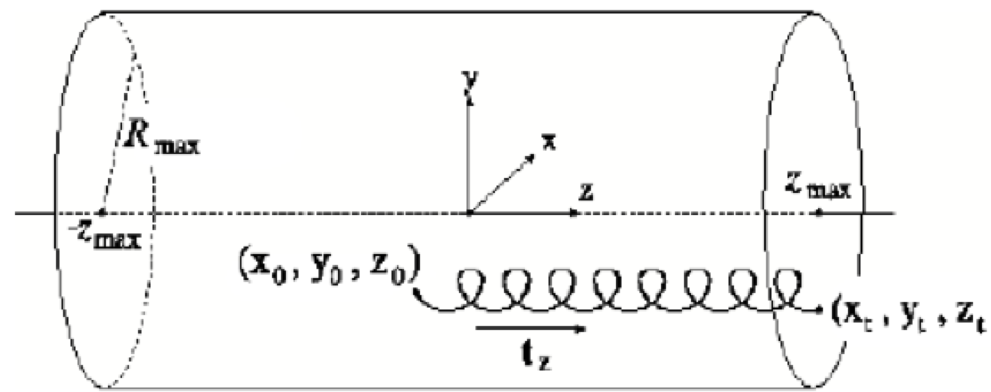


see details in tutorial ...

Charged Particle Propagation

- Charged 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

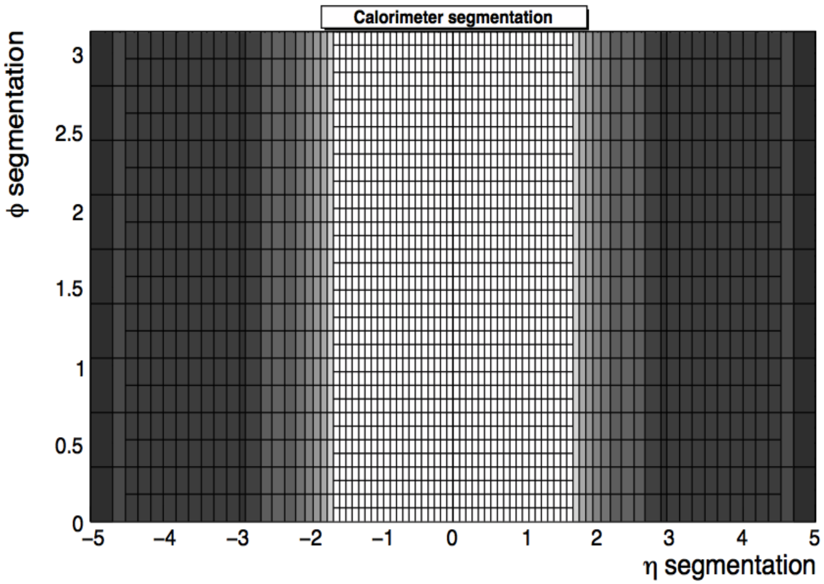
```
# efficiency formula for muons
add EfficiencyFormula {13} {                                    (pt <= 0.1)   * (0.000) + \
                                    (abs(eta) <= 1.5) * (pt > 0.1   && pt <= 1.0)   * (0.750) + \
                                    (abs(eta) <= 1.5) * (pt > 1.0)         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

Not real tracking/vertexing !!
→ no fake tracks/ conversions (but can be easily implemented)
→ no dE/dx measurements

Calorimetry

- em/had calorimeters have same **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}
$e\ \gamma\ \pi^0$	1	0
Long-lived neutral hadrons (K^0_S , Λ^0)	0.3	0.7
$\nu\ \mu$	0	0
others	0	1

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

$$E_{smeared} = gauss(f_{EM}E, \sigma_{EM}(\eta)) + gauss(f_{HAD}E, \sigma_{HAD}(\eta))$$

$$\sigma^2(\eta) = N^2(\eta) + S^2(\eta)E + C^2(\eta)E^2$$

Leptons, photons

- Muons/photons/electrons
  - **identified** via their PDG id
  - inside the **tracker coverage** for electrons and muons
  - muons do not deposit energy in calo (independent smearing parameterized in p_T and η)
  - electrons and photons smeared according to electromagnetic calorimeter resolution
- Isolation: $$\mathrm{rel.Iso} = \frac{\sum_{\Delta R<0.5} p_T^{track}}{p_T}$$

  If rel.Iso ~ 1, the lepton is isolated

- Not taken into account:
  - fakes, punch-through, brehmstrahlung, conversions

Jets

- **FastJet** library used for jet clustering
  - all clustering algos supported: **anti-kT**, SisCone, ...

- Jets are formed with “**particle-flow**” like input:
  - inside tracker volume

    → tracks
    → calorimeter towers for neutral particles

  - outside tracker volume

    → calorimeter towers

b and tau jets

- <u>b-jets</u>

  - if **b** parton is found in a cone $\Delta R = 0.5$ w.r.t jet direction
    → apply **efficiency** (40% default)
  - if **c** parton is found in a cone $\Delta R = 0.5$ w.r.t jet direction
    → apply **c-mistag rate** (10% default)
  - if **u,d,s,g** parton is found in a cone $\Delta R = 0.5$ w.r.t jet direction
    → apply **light-mistag rate** (0.1% default)

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

- <u>tau-jets</u>

  - if tau lepton is found in a cone $\Delta R = 0.5$ w.r.t jet direction
    → apply **efficiency** (40% default)
  - else
    → apply **tau-mistag rate** (1% default)

tau jets have their own collection (no leptonic tau decays)

see tutorial for **p_T and η dependent** efficiency and mistag rate

Missing Energy

missing transverse energy is computed by default as:

$$\mathbf{E_T^{miss}} = -\sum_i \mathbf{p_T^{muons}}(i) - \sum_i \mathbf{E_T^{towers}}(i)$$

or as “particle flow”:

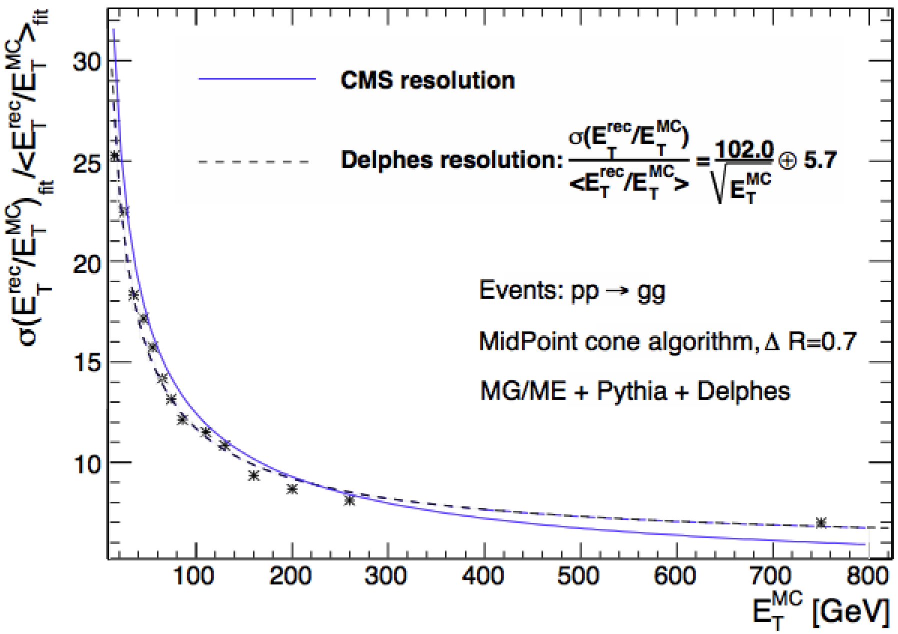
$$\mathbf{E_T^{miss}} = -\sum_{i \in barrel} \mathbf{p_T^{track}}(i) - \sum_{i \in barrel} \mathbf{E_T^{neutral\ towers}}(i) - \sum_{i \in endcaps} \mathbf{E_T^{towers}}(i)$$

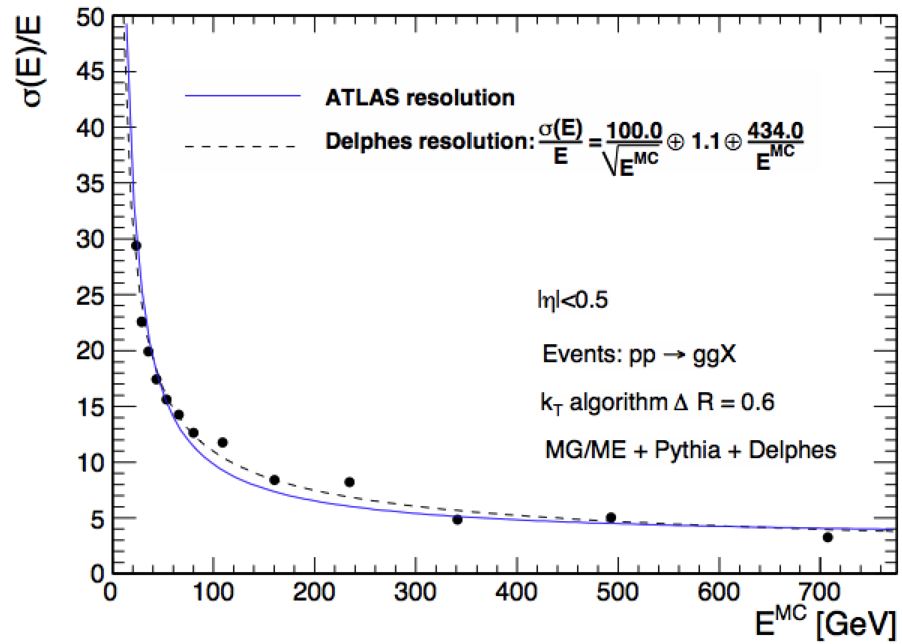
Effects not simulated:

- cracks (can be simulated via efficiency formula)
- dead channels
- noise ...

Validation: jets

- Electrons, muons and photons are auto-validated by construction
- **Jets** and missing energy need to be tested:

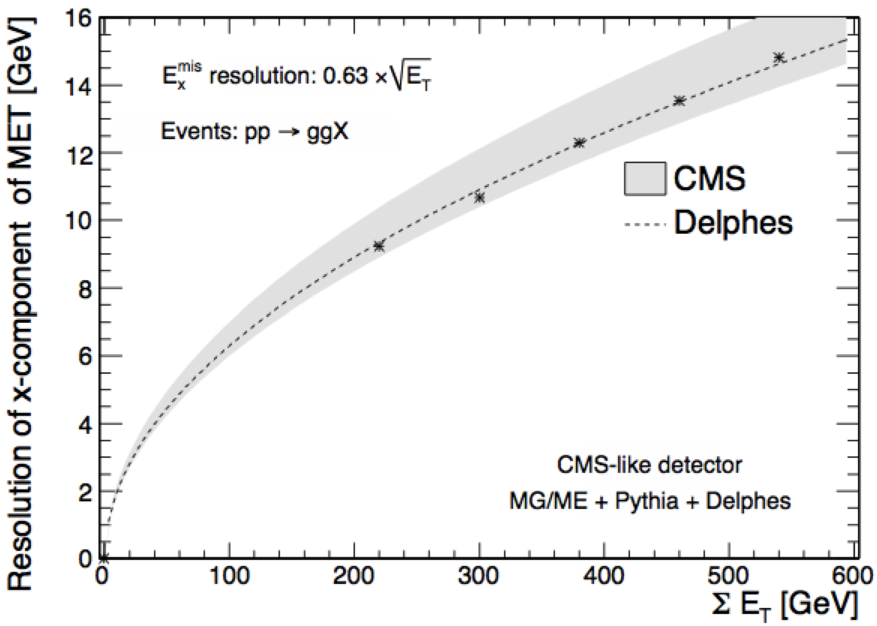


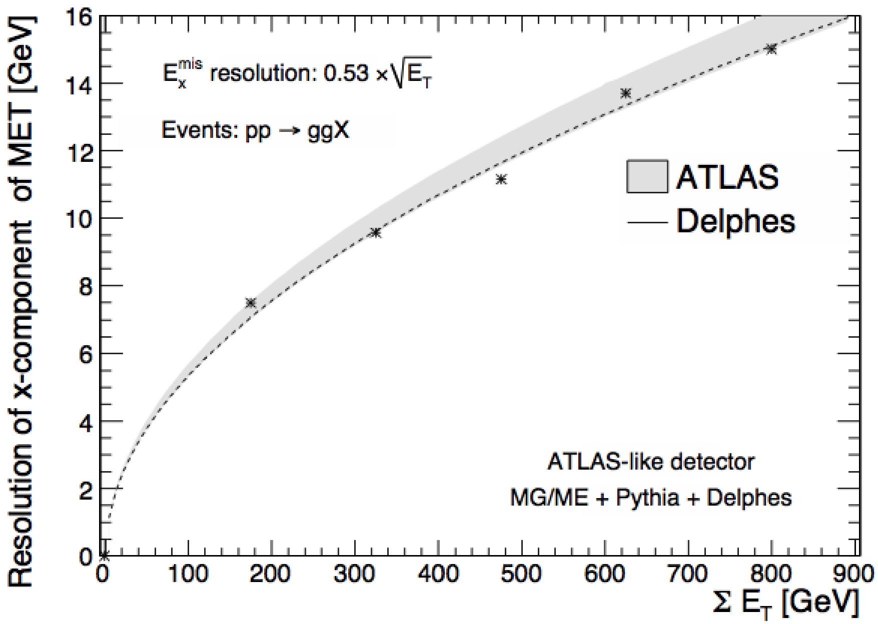


→ **good agreement**

Validation: E_T^{miss}

- Electrons, muons and photons are auto-validated by construction
- Jets and **missing energy** need to be tested:





→ **good agreement**

Development

- Delphes project started back in 2007

- 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

Widely tested and used by the community > 100 citations !!

- **Major change ongoing:**
  - **modular** version of the software has been written
    → more flexible
    → more user-friendly
    → faster
  - we are in the **beta-testing** phase right now

Please test the beta !!!

and give us some feedback ...

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