Delphes, a framework for fast simulation of a generic collider experiment

S. Ovyn*, X. Rouby, V. Lemaître

Center for Particle Physics and Phenomenology (CP3),
Université catholique de Louvain,
B-1348 Louvain-la-Neuve, Belgium

Abstract

This paper presents a new C++ framework, *Delphes*, performing a fast multipurpose detector response simulation. The simulation includes a tracking system, embedded into a magnetic field, calorimeters and a muon system, and possible very forward detectors arranged along the beamline. The framework is interfaced to standard file formats (e.g. Les Houches Event File or HepMC) and outputs observables such as isolated leptons, missing transverse energy and collection of jets which can be used for dedicated analyses. The simulation of the detector response takes into account the effect of magnetic field, the granularity of the calorimeters and subdetector resolutions. A simplified preselection can also be applied on processed events for trigger emulation. Detection of very forward scattered particles relies on the transport in beamlines with the *Hector* software. Finally, the FROG 2D/3D event display is used for visualisation of the collision final states.

Preprint: CP3-09-01, arXiv:0903.2225 [hep-ph]

PROGRAM SUMMARY

Program Title: DELPHES
Current version: 1.8
Journal Reference:
Catalogue identifier:
Distribution format: tar.gz
Programming language: C++
External routines/libraries: ROOT environment
Subprograms used: HepMC, StdHEP, FASTJET, *Hector*, FROG. All provided within *Delphes* distribution.
*URL:*http://www.fynu.ucl.ac.be/delphes.html

Key words: *Delphes*, detector simulation, event reconstruction, trigger, LHC
PACS: 29.85.-c, 07.05.Tp, 29.90.+r, 29.50.+v

1. Introduction

Multipurpose detectors at high energy colliders are very complex systems. Precise data analyses require a full detector simulation, including transport of the primary and secondary particles through the detector material accounting for the various detector inefficiencies, the dead material, the imperfections and the geometrical details. Their simulation is in general performed by means of the GEANT [1] package and final observables used for analyses usually require sophisticated reconstruction algorithms.

*Corresponding author: +32.10.47.32.29.
Email address: severine.ovyn@uclouvain.be (S. Ovyn)

This complexity can only be handled by large collaborations. Such simulation is very complicated, technical and requires a large CPU power. Phenomenological studies, looking for the observability of given signals, require in general only fast but realistic estimates of the expected signal signatures and their associated backgrounds.

In this context, a new framework, called *Delphes* [2], has been developped, for a fast simulation of a general-purpose collider experiment. Using this framework, observables such as cross-sections and efficiencies after event selection can be estimated for specific reactions. Starting from the output of event generators, the simulation of the detector response takes into account the subdetector resolutions, by smearing the kinematics of final-state particles (i.e. those considered as stable by the event generator [1]).

Delphes includes the most crucial experimental features, such as (Fig. 1):

1. the geometry of both central and forward detectors,
2. the effect of magnetic field on tracks,
3. the reconstruction of photons, leptons, jets, b-jets, τ-jets and missing transverse energy,
4. a lepton isolation,
5. a trigger emulation,
6. an event display.

Although *Delphes* yields much realistic results than a simple "parton-level" analysis, it has some limitations. Detector geometry is idealised, being uniform, symmetric around the beam axis, and having no cracks nor dead material. Secondary interactions, multiple scatterings, photon conversion and bremsstrahlung are also neglected.

Several common datafile formats can be used as input in *Delphes* [a], in order to process events from many different generators. *Delphes* creates output data in a ROOT ntuple [3]. This output contains a copy of the generator-level data, the analysis data objects after reconstruction, and possibly the results of the trigger emulation [b]. In option *Delphes* can produce a reduced output file in `*.lhco` text format, which is limited to the list of the reconstructed high-level objects in the final states [c].

2. Simulation of the detector response

The overall layout of the multipurpose detector simulated by *Delphes* is shown in Fig. 2. It consists in a central tracking system (TRACKER) surrounded by an electromagnetic and a hadron calorimeters (ECAL and HCAL, each with a central region and two endcaps) and two forward calorimeters (FCAL). Finally, a muon system (MUON) encloses the central detector volume.

[1] In the current *Delphes* version, particles other than electrons ($e^{\pm}$), photons (γ), muons ($\mu^{\pm}$), neutrinos (ν_e, ν_μ and ν_τ) and neutralinos are simulated as hadrons for their interactions with the calorimeters. The simulation of stable particles beyond the Standard Model should therefore be handled with care [d].

Table 1: Default extension in pseudorapidity η of the different subdetectors. Full azimuthal (ϕ) acceptance is assumed.

	η	ϕ
TRACKER	$[-2.5; 2.5]$	$[-\pi; \pi]$
ECAL, HCAL	$[-1.7; 1.7]$	$[-\pi; \pi]$
ECAL, HCAL endcaps	$[-3; -1.7]$ & $[1.7; 3]$	$[-\pi; \pi]$
FCAL	$[-5; -3]$ & $[3; 5]$	$[-\pi; \pi]$
MUON	$[-2.4; 2.4]$	$[-\pi; \pi]$

A detector card [e] allows a large spectrum of running conditions by modifying basic detector parameters, including calorimeter and tracking coverage and resolution, thresholds or jet algorithm parameters. Even if *Delphes* has been developped for the simulation of general-purpose detectors at the LHC (namely, CMS and ATLAS), this input parameter file interfaces a flexible parametrisation for other cases, e.g. at future linear colliders [f]. If no detector card is provided, predefined values based on "typical" CMS acceptances and resolutions are used. The geometrical coverage of the various subsystems used in the default configuration are summarised in Tab. 1. The detector is assumed to be strictly symmetric around the beam axis.

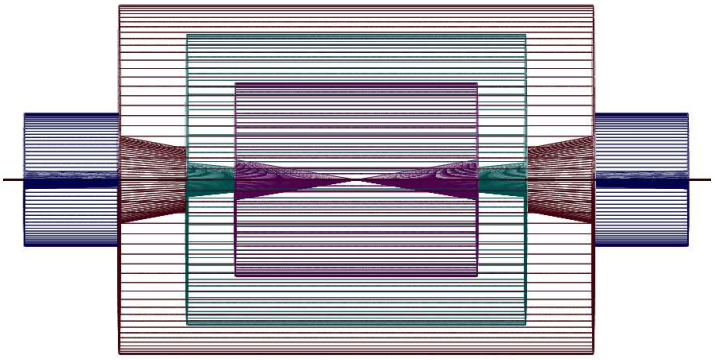

Figure 2: Profile of layout of the generic detector geometry assumed in *Delphes*. The innermost layer, close to the interaction point, is a central tracking system (pink). It is surrounded by a central calorimeter volume (green) with both electromagnetic and hadronic sections. The outer layer of the central system (red) is muon system. In addition, two end-cap calorimeters (blue) extend the pseudorapidity coverage of the central detector. Additional forward detectors are not depicted.

2.1. Magnetic field

In addition to the subdetectors, the effects of a solenoidal magnetic field are simulated for the charged particles [h]. This affects the position at which charged particles enter the calorimeters and their corresponding tracks. The field extension is limited to the tracker volume and is in particular not applied for muon chambers. However, this is not a limiting factor as the resolution applied for muon reconstruction is the one expected by the experiment, which consequently includes the effects of the magnetic field within the muon system.

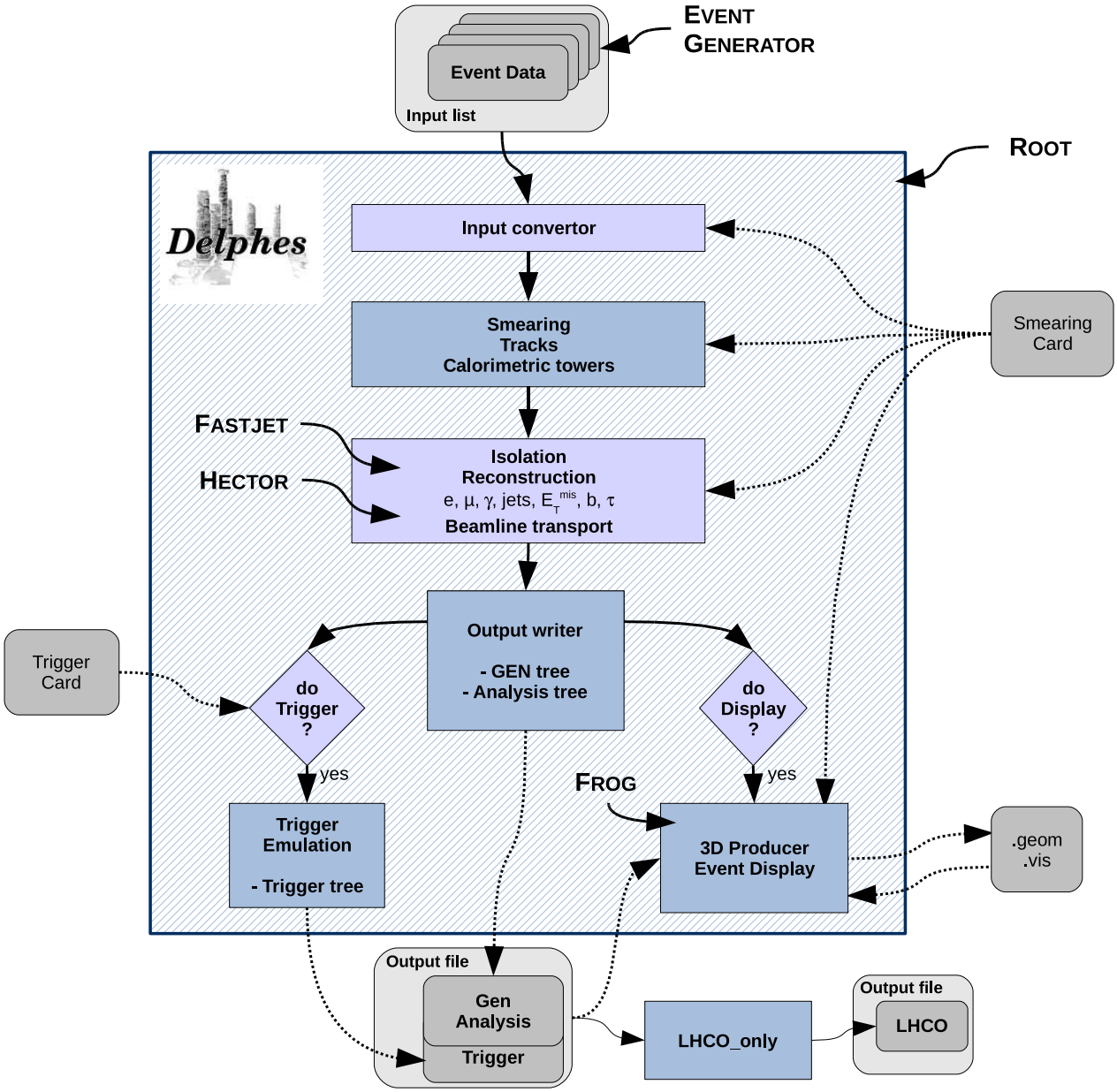

Figure 1: Flow chart describing the principles behind *Delphes*. Event files coming from external Monte Carlo generators are read by a converter stage (top). The kinematics variables of the final-state particles are then smeared according to the tunable subdetector resolutions. Tracks are reconstructed in a simulated solenoidal magnetic field and calorimetric cells sample the energy deposits. Based on these low-level objects, dedicated algorithms are applied for particle identification, isolation and reconstruction. The transport of very forward particles to the near-beam detectors is also simulated. Finally, an output file is written, including generator-level and analysis-object data. If requested, a fully parametrisable trigger can be emulated. Optionally, the geometry and visualisation files for the 3D event display can also be produced. All user parameters are set in the *Detector/Smearing Card* and the *Trigger Card*.

2.2. Tracks reconstruction

Every stable charged particle with a transverse momentum above some threshold and lying inside the detector volume covered by the tracker provides a track. By default, a track is assumed to be reconstructed with 90% probability if its transverse momentum p_T is higher than 0.9 GeV/c and if its pseudorapidity $|\eta| \leq 2.5$ [i]. No smearing is currently applied on tracks.

2.3. Calorimetric cells

The response of the calorimeters to energy deposits of incoming particles depends on their segmentation and resolution, as well as on the nature of the particles themselves. In CMS and ATLAS detectors, for instance, the calorimeter characteristics are not identical in every direction, with typically finer resolution and granularity in the central regions [5, 6]. It is thus very important to compute the exact coordinates of the entry point of the particles into the calorimeters, via the magnetic field calculations.

The smallest unit for geometrical sampling of the calorimeters is a *cell*; it segments the (η, ϕ) plane for the energy measurement. No longitudinal segmentation is available in the simulated calorimeters. *Delphes* assumes that ECAL and HCAL have the same segmentations and that the detector is symmetric in ϕ and with respect to the $\eta = 0$ plane [l]. Fig. 3 illustrates the default calorimeter segmentation.

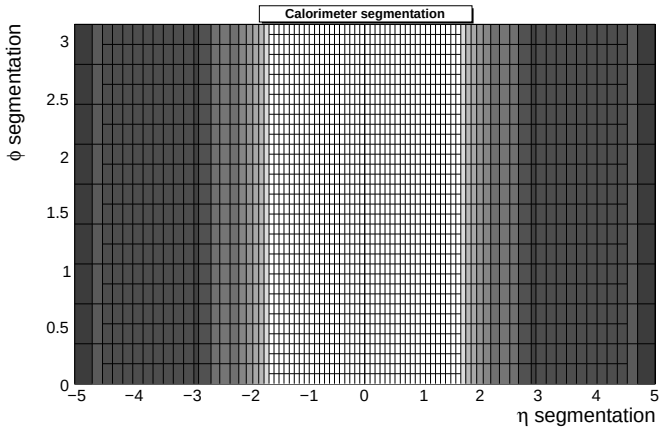

Figure 3: Default segmentation of the calorimeters in the (η, ϕ) plane. Only the central detectors (ECAL, HCAL) and FCAL are considered. ϕ angles are expressed in radians.

The calorimeter response is parametrised through a Gaussian smearing of the accumulated cell energy with a variance σ:

$$\frac{\sigma}{E} = \frac{S}{\sqrt{E}} \oplus \frac{N}{E} \oplus C, \tag{1}$$

where S, N and C are the *stochastic*, *noise* and *constant* terms, respectively, and $\oplus$ stands for quadratic additions [j].

In the default parametrisation, ECAL and HCAL are assumed to cover the pseudorapidity range $|\eta| < 3$, and FCAL between 3.0 and 5.0, with different response to electromagnetic objects ($e^{\pm}, \gamma$) or hadrons. Muons and neutrinos are assumed not to interact with the calorimeters [d]. The default values of the stochastic, noise and constant terms are given in Tab. 2.

Table 2: Default values for the resolution of the central and forward calorimeters (for both electromagnetic and hadronic parts). Resolution is parametrised by the *stochastic* (S), *noise* (N) and *constant* (C) terms (Eq. 1) [g].

	S (GeV$^{1/2}$)	N (GeV)	C
ECAL	0.05	0.25	0.0055
ECAL, end caps	0.05	0.25	0.0055
FCAL, e.m. part	2.084	0	0.107
HCAL	1.5	0	0.05
HCAL, end caps	1.5	0	0.05
FCAL, had. part	2.7	0	0.13

Electrons and photons leave their energy in the electromagnetic parts of the calorimeters (ECAL and FCAL, e.m.), while charged and neutral final-state hadrons interact with the hadronic parts (HCAL and FCAL, had.). Some long-living particles, such as the K_s^0 and Λ's, with lifetime $c\tau$ smaller than 10 mm are considered as stable particles by the generators although they may decay before the calorimeters. The energy smearing of such particles is therefore performed using the expected fraction of the energy, determined according to their decay products, that would be deposited into the ECAL (E_{ECAL}) and into the HCAL (E_{HCAL}). Defining F as the fraction of the energy leading to a HCAL deposit, the two energy values are given by

$$\begin{cases} E_{\text{HCAL}} = E \times F \\ E_{\text{ECAL}} = E \times (1 - F) \end{cases} \tag{2}$$

where $0 \leq F \leq 1$. The electromagnetic part is handled similarly as for electrons and photons. The resulting calorimetry energy measurement given after the application of the smearing is then $E = E_{\text{HCAL}} + E_{\text{ECAL}}$. For K_S^0 and Λ hadrons, the energy fraction is F is assumed to be 0.7 [k].

No sharing between neighbouring cells is implemented when particles enter a cell very close to its geometrical edge. Due to the finite segmentation, the smearing, as defined in Eq. 1, is applied directly on the accumulated electromagnetic and hadronic energies of each calorimetric cell. The calorimetric cells enter in the calculation of the missing transverse energy (MET), and are used as input for the jet reconstruction algorithms.

3. High-level object reconstruction

The output file created by *Delphes* [m] stores the final collections of particles ($e^{\pm}$, $\mu^{\pm}$, γ) and objects (light jets, b-jets, τ-jets, $\not{E}_T^{\text{miss}}$). In addition, some detector data are added, such as tracks, calorimetric cells and hits in the very forward detectors (ZDC, RP220 and FP420, see Sec. 5). While electrons, muons and photons are easily identified, other quantities are more difficult to evaluate as they rely on sophisticated algorithms (e.g. jets or missing energy).

For most of these objects, their four-momentum and related quantities are directly accessible in *Delphes* output (E, $\vec{p}$, p_T, η and ϕ). Additional properties are available for specific objects (like the charge and the isolation status for $e^{\pm}$ and $\mu^{\pm}$, the result of application of b-tag for jets and time-of-flight for some detector hits).

3.1. Photon and charged lepton

From here onwards, *electrons* refer to both positrons (e^+) and electrons (e^-), and *charged leptons* refer to electrons and muons ($\mu^{\pm}$), leaving out the $\tau^{\pm}$ leptons as they decay before being detected.

The electron, muon and photon collections contains only the true final-state particles identified via the generator-data. In addition, these particles must pass fiducial cuts taking into account the magnetic field effects and some additional reconstruction cuts.

Consequently, no fake candidates enter these collections. However, when needed, fake candidates can be added into the collections at the analysis level, when processing *Delphes* output data. As effects like bremsstrahlung are not taken into account along the lepton propagation in the tracker, no clustering is needed for the electron reconstruction in *Delphes*.

Electrons and photons

Real electron ($e^{\pm}$) and photon candidates are associated to the final-state collections if they fall into the acceptance of the tracking system and have a transverse momentum above some threshold (default: $p_T > 10$ GeV/c). Assuming a good measurement of the track parameters in the real experiment, the electron energy can be reasonably recovered. *Delphes* assumes a perfect algorithm for clustering and Brehmstrahlung recovery. Electron energy is smeared according to the resolution of the calorimetric cell where it points to, but independently from any other deposited energy is this cell. Electrons and photons may create a candidate in the jet collection.

Muons

Generator-level muons entering the muon detector acceptance (default: $-2.4 \leq \eta \leq 2.4$) and overpassing some threshold (default : $p_T > 10$ GeV/c) are considered as good candidates for analyses. The application of the detector resolution on the muon momentum depends on a Gaussian smearing of the p_T [n]. Neither η nor ϕ variables are modified beyond the calorimeters: no additional magnetic field is applied. Multiple scattering is neglected. This implies that low energy muons have in *Delphes* a better resolution than in a real detector. At last, the particles which might leak out of the calorimeters into the muon systems (*punch-through*) are not considered as muon candidates in *Delphes*.

Charged lepton isolation

To improve the quality of the contents of the charged lepton collections, isolation criteria can be applied. This requires that electron or muon candidates are isolated in the detector from any other particle, within a small cone. In *Delphes*, charged lepton isolation demands that there is no other charged particle with $p_T > 2$ GeV/c within a cone of $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2} < 0.5$ centered on the cell associated to the charged lepton ℓ, obviously taking the magnetic field into account.

The result (i.e. *isolated* or *not*) is added to the charged lepton measured properties. In addition, the sum P_T of the transverse momenta of all tracks but the lepton one within the isolation cone is provided [o]:

$$P_T = \sum_{i \neq \ell}^{\text{tracks}} p_T(i)$$

No calorimetric isolation is applied, but the charged lepton collections contain also the ratio ρ_ℓ between (1) the sum of the transverse energies in all calorimetric cells in a $N \times N$ grid around the lepton, and (2) the lepton transverse momentum [p]:

$$\rho_\ell = \frac{\Sigma_i E_T(i)}{p_T(\ell)} \text{ , } i \text{ in } N \times N \text{ grid centred on } \ell.$$

3.2. Jet reconstruction

A realistic analysis requires a correct treatment of partons which have hadronised. Therefore, the most widely currently used jet algorithms have been integrated into the *Delphes* framework using the FastJet tools[2]. Six different jet reconstruction schemes are available [8, r]. For all of them, the calorimetric cells are used as inputs. Jet algorithms differ in their sensitivity to soft particles or collinear splittings, and in their computing speed performances.

Cone algorithms

1. *CDF Jet Clusters* [9]: Cone algorithm forming jets by combining cells lying within a circle (default radius $\Delta R = 0.7$) in the (η, ϕ) space. Jets are seeded by all cells with transverse energy E_T above a given threshold (default: $E_T > 1$ GeV) [t].

2. *CDF MidPoint* [10]: Cone algorithm with additional "midpoints" (energy barycentres) in the list of seeds.

3. *Seedless Infrared Safe Cone* [11]: The SISCone algorithm is simultaneously insensitive to additional soft particles and collinear splittings.

Recombination algorithms

The next three jet algorithms rely on recombination schemes where calorimeter cell pairs are successively merged:

4. *Longitudinally invariant k_t jet* [12],

5. *Cambridge/Aachen jet* [13],

6. *Anti k_t jet* [14], where hard jets are exactly circular in the (y, ϕ) plane.

The recombination algorithms are safe with respect to soft radiations (*infrared*) and collinear splittings. Their implementations are similar except for the definition of the *distances* used during the merging procedure.

By default, reconstruction uses the CDF cone algorithm. Jets are stored if their transverse energy is higher than 20 GeV [s].

Energy flow

In jets, several particle can leave their energy into a given calorimetric cell, which broadens the jet energy resolution. However, the energy of charged particles associated to jets can be deduced from their reconstructed track, thus providing a way to identify some of the components of cells with multiple hits. When the *energy flow* is switched on in *Delphes*, the energy of tracks pointing to calorimetric cells is subtracted and smeared separately, before running the chosen jet reconstruction algorithm. This option allows a better jet E reconstruction [u].

3.3. b-tagging

A jet is tagged as b-jets if its direction lies in the acceptance of the tracker and if it is associated to a parent b-quark. By default, a b-tagging efficiency of 40% is assumed if the jet has a parent b quark. For c-jets and light jets (i.e. originating in u, d, s quarks or in gluons), a fake b-tagging efficiency of 10% and 1% respectively is assumed [v]. The (mis)tagging relies on the identity of the most energetic parton within a cone around the jet axis, with a radius equal to the one used to reconstruct the jet (default: ΔR of 0.7). In current version of *Delphes*, the displacement of secondary vertices is not simulated.

[2] A more detailed description of the jet algorithms is given in the User Manual, in appendix.

3.4. τ identification

Jets originating from τ-decays are identified using a procedure consistent with the one applied in a full detector simulation [5]. The tagging relies on two properties of the τ lepton. First, 77% of the τ hadronic decays contain only one charged hadron associated to a few neutrals (Tab. 3). Secondly, the particles arisen from the τ lepton produce narrow jets in the calorimeter (this is defined as the jet *collimation*).

Table 3: Branching ratios for τ^- lepton [15]. $h^\pm$ and h^0 refer to charged and neutral hadrons, respectively. $n \geq 0$ and $m \geq 0$ are integers.

Leptonic decays	
$\tau^- \to e^- \, \overline{\nu}_e \, \nu_\tau$	17.9%
$\tau^- \to \mu^- \, \overline{\nu}_\mu \, \nu_\tau$	17.4%
Hadronic decays	
$\tau^- \to h^- \, (n \times h^\pm) \, (m \times h^0) \, \nu_\tau$	64.7%
$\tau^- \to h^- \, (m \times h^0) \, \nu_\tau$	50.1%
$\tau^- \to h^- \, h^+ h^- (m \times h^0) \, \nu_\tau$	14.6%

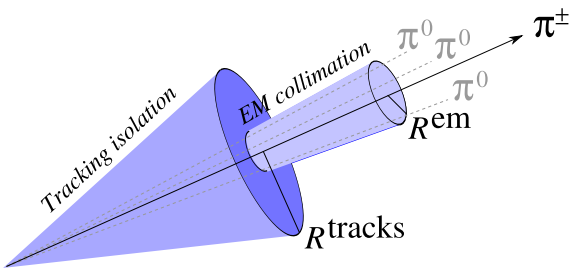

Figure 4: Illustration of the identification of τ-jets (1−prong). The jet cone is narrow and contains only one track. The small cone serves to apply the *electromagnetic collimation*, while the broader cone is used to reconstruct the jet originating from the τ-decay.

Table 4: Default values for parameters used in τ-jet reconstruction algorithm. Electromagnetic collimation requirements involve the inner *small* cone radius $R^{\rm em}$, the minimum transverse energy for calorimetric cells $E_T^{\rm cell}$ and the collimation factor C_τ. Tracking isolation constrains the number of tracks with a significant transverse momentum $p_T^{\rm tracks}$ in a cone of radius $R^{\rm tracks}$. Finally, the τ-jet collection is purified by the application of a cut on the p_T of τ-jet candidates [w].

Electromagnetic collimation	
$R^{\rm em}$	0.15
min $E_T^{\rm cell}$	1.0 GeV
C_τ	0.95
Tracking isolation	
$R^{\rm tracks}$	0.4
min $p_T^{\rm tracks}$	2 GeV/c
τ-jet candidate	
min p_T	10 GeV/c

Electromagnetic collimation

To use the narrowness of the τ-jet, the *electromagnetic collimation* C_τ is defined as the sum of the energy of cells in a small cone of radius $R^{\rm em}$ around the jet axis, divided by the energy of the reconstructed jet. To be taken into account, a calorimeter cell should have a transverse energy $E_T^{\rm cell}$ above a given threshold. A large fraction of the jet energy is expected in this small cone. This fraction, or *collimation factor*, is represented in Fig. 5 for the default values (see Tab. 4).

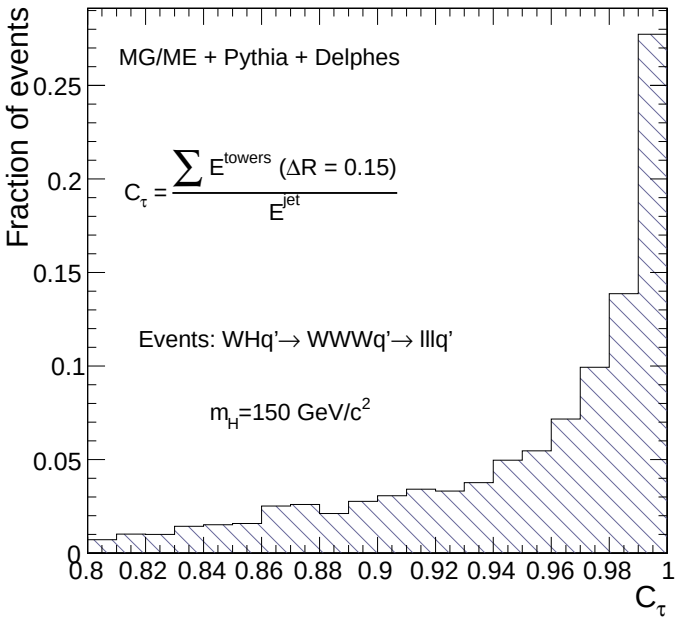

Figure 5: Distribution of the electromagnetic collimation C_τ variable for true τ-jets, normalised to unity. This distribution is shown for associated WH photoproduction [16], where the Higgs boson decays into a W^+W^- pair. Each W boson decays into a $\ell\nu_\ell$ pair, where $\ell = e, \mu, \tau$. Events generated with MadGraph/MadEvent [17]. Final state hadronisation is performed by *Pythia* [18]. Histogram entries correspond to true τ-jets, matched with generator-level data.

Tracking isolation

The tracking isolation for the τ identification requires that the number of tracks associated to particles with significant transverse momenta is one and only one in a cone of radius $R^{\rm tracks}$ (3−prong τ-jets are dropped). This cone should be entirely incorporated into the tracker to be taken into account. Default values of these parameters are given in Tab. 4.

Purity

Once both electromagnetic collimation and tracking isolation are applied, a threshold on the p_T of the τ-jet candidate is requested to purify the collection. This procedure selects τ leptons decaying hadronically with a typical efficiency of 66%.

3.5. Missing transverse energy

In an ideal detector, momentum conservation imposes the transverse momentum of the observed final state $\overrightarrow{p_T}^{\rm obs}$ to be equal and

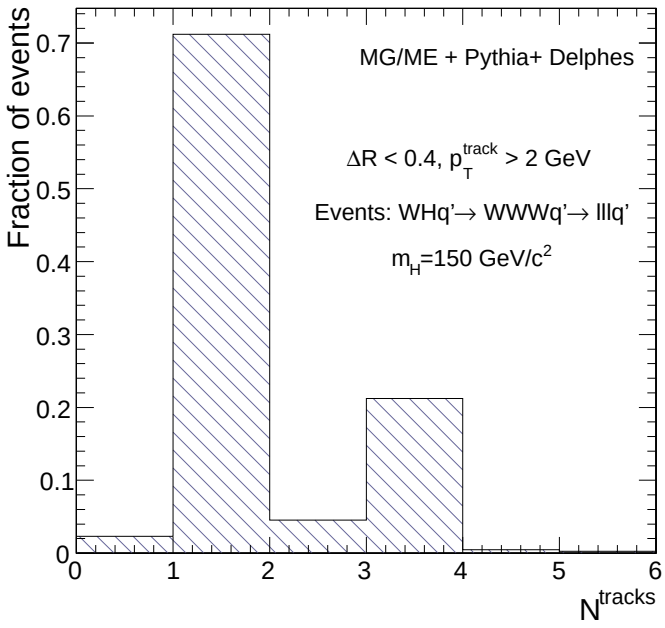

Figure 6: Distribution of the number of tracks N^{tracks} within a small jet cone for true τ-jets, normalised to unity. Photoproduced WH events, where W bosons decay leptonically (e, μ, τ), as in Fig. 5. Histogram entries correspond to true τ-jets, matched with generator-level data.

in opposite direction to the $\overrightarrow{p_T}$ vector sum of the invisible particles, written $\overrightarrow{p_T}^{\text{miss}}$. The *true* missing transverse energy, i.e. at generator-level, is calculated as the opposite of the vector sum of the transverse momenta of all visible particles – or equivalently, to the vector sum of invisible particle transverse momenta. In a real experiment, calorimeters measure energy and not momentum. Any problem affecting the detector (dead channels, misalignment, noisy cells, cracks) worsens directly the measured missing transverse energy $\overrightarrow{E_T}^{\text{miss}}$. In *Delphes*, MET is based on the calorimetric cells only. Muons and neutrinos are therefore not taken into account for its evaluation:

$$\overrightarrow{E_T}^{\text{miss}} = -\sum_{i}^{\text{cells}} \overrightarrow{E_T}(i) \tag{3}$$

However, as muon candidates, tracks and calorimetric cells are available in the output file, the missing transverse energy can always be reprocessed a posteriori with more specialised algorithms.

4. Trigger emulation

Most of the usual trigger algorithms select events containing leptons, jets, and MET with an energy scale above some threshold. This is often expressed in terms of a cut on the transverse momentum of one or several objects of the measured event. Logical combinations of several conditions are also possible. For instance, a trigger path could select events containing at least one jet and one electron such as $p_T^{\text{jet}} > 100$ GeV/c and $p_T^e > 50$ GeV/c.

A trigger emulation is included in *Delphes*, using a fully parametrisable *trigger table* [x]. When enabled, this trigger is applied on analysis-object data. In a real experiment, the online selection is often divided into several steps (or *levels*). corresponding to the different trigger levels. First-level triggers are fast and simple but based only on partial data as not all detector front-ends are readable within the decision latency. Higher level triggers are more complex, of finer-but-not-final quality and based on full detector data.

Real triggers are thus intrinsically based on reconstructed data with a worse resolution than final analysis information. On the contrary, the same information is used in *Delphes* for the trigger emulation and for final analyses.

Table 5: Default parameters for the forward detectors: distance from the interaction point and detector acceptance. The LHC beamline is assumed around the fifth LHC interaction point (IP). For the ZDC, the acceptance depends only on the pseudorapidity η of the particle, which should be neutral and stable. It is expressed in terms of the particle energy (E). All detectors are located on both sides of the interaction point.

Detector	Distance	Acceptance	
ZDC	±140 m	$\lvert\eta\rvert > 8.3$	for n and γ
RP220	±220 m	$E \in [6100; 6880]$ (GeV)	at 2 mm
FP420	±420 m	$E \in [6880; 6980]$ (GeV)	at 4 mm

5. Very forward detector simulation

Most of the recent experiments in beam colliders have additional instrumentation along the beamline. These extend the η coverage to higher values, for the detection of very forward final-state particles. In *Delphes*, Zero Degree Calorimeters, roman pots and forward taggers have been implemented (Fig. 7), similarly as for CMS and ATLAS collaborations [5, 6].

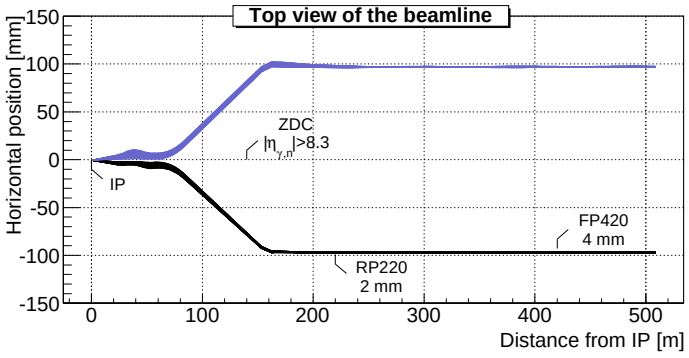

Figure 7: Default location of the very forward detectors, including ZDC, RP220 and FP420 in the LHC beamline. Incoming (beam 1, red) and outgoing (beam 2, black) beams on one side of the fifth interaction point (IP5, $s = 0$ m on the plot). The Zero Degree Calorimeter is located in perfect alignment with the beamline axis at the interaction point, at 140 m, where the beam paths are well separated. The forward taggers are near-beam detectors located at 220 m and 420 m. Beamline simulation with *Hector* [7]. All very forward detectors are located symmetrically around the interaction point.

5.1. Zero Degree Calorimeters

In direct sight of the interaction point, on both sides of the central detector, the Zero Degree Calorimeters (ZDCs) are located at zero angle, i.e. are aligned with the beamline axis at the interaction point. They are placed beyond the point where the paths of incoming and outgoing beams separate. These allow the measurement of stable neutral particles (γ and n) coming from the interaction point, with large pseudorapidities (e.g. $|\eta_{n,\gamma}| > 8.3$ in ATLAS and CMS).

The trajectory of the neutrals observed in the ZDCs is a straight line, while charged particles are deflected away from their acceptance window by the powerful magnets located in front of them. The fact that additional charged particles may enter the ZDC acceptance is neglected in the current versions of *Delphes*.

The ZDCs have the ability to measure the time-of-flight of the particle. This corresponds to the delay t after which the particle is observed in the detector, with respect to the bunch crossing reference time at the interaction point (t_0):

$$t = t_0 + \frac{1}{v} \times \left(\frac{s - z}{\cos\theta}\right) \approx \frac{1}{c} \times (s - z), \tag{4}$$

where t_0 is thus the true time coordinate of the vertex from which the particle originates, v the particle velocity, s is the ZDC distance to the interaction point, z is the longitudinal coordinate of the vertex, θ is the particle emission angle. It is assumed that the neutral particle observed in the ZDC is highly relativistic and very forward. For the time-of-flight measurement, a Gaussian smearing can be applied according to the detector resolution (Tab. 6) [g].

The ZDCs are composed of an electromagnetic and a hadronic sections, for the measurement of photons and neutrons, respectively. The energy of the observed neutral is smeared according to Eq. 1 and the corresponding section resolutions (Tab. 6). The ZDC hits do not enter in the calorimeter cell list used for reconstruction of jets and missing transverse energy.

Table 6: Default values for the resolution of the zero degree calorimeters. Resolution on energy measurement is parametrised by the *stochastic* (S), *noise* (N) and *constant* (C) terms (Eq. 1) [g]. The time-of-flight is smeared according to a Gaussian function.

ZDC, electromagnetic part		hadronic part
S (GeV$^{1/2}$)	0.7	1.38
N (GeV)	0	0
C	0.08	0.13
ZDC, timing resolution		
σ_t (s)	0	

The reconstructed ZDC hits correspond to neutral particles with a lifetime long enough to reach these detectors (default: $c\tau \geq$ 140 m) and very large pseudorapidities (default: $|\eta| > 8.3$). Photons and neutrons are identified if their energy overpasses a given threshold (def. $E_\gamma \leq 20$ GeV and $E_n \leq 50$ GeV) [q].

5.2. Forward taggers

Forward taggers (called here RP220, for "roman pots at 220 m" and FP420 for "forward proton taggers at 420 m", as at the LHC) are meant for the measurement of particles following very closely the beam path. Such devices, also used at HERA and Tevatron, are located very far away from the interaction point (further than 150 m in the LHC case).

To be able to reach these detectors, particles must have a charge identical to the beam particles, and a momentum very close to the nominal value of the beam. These taggers are near-beam detectors located a few millimetres from the true beam trajectory and this distance defines their acceptance (Tab. 5). For instance, roman pots at 220 m from the IP and 2 mm from the beam will detect all forward protons with an energy between 120 and 900 GeV [7]. In practice, in the LHC, only positively charged muons (μ^+) and protons can reach the forward taggers as other particles with a single positive charge coming from the interaction points will decay before their possible tagging. In *Delphes*, extra hits coming from the beam-gas events or secondary particles hitting the beampipe in front of the detectors are not taken into account.

While neutral particles propagate along a straight line to the ZDC, a dedicated simulation of the transport of charged particles is needed for RP220 and FP420. This fast simulation uses the *Hector* software [7], which includes the chromaticity effects and the geometrical aperture of the beamline elements of any arbitrary collider.

Forward taggers are able to measure the hit positions (x, y) and angles (θ_x, θ_y) in the transverse plane at the location of the detector (s meters away from the IP), as well as the time-of-flight[3] (t). Out of these the particle energy (E) and the momentum transfer it underwent during the interaction (q^2) can be reconstructed at the analysis level (it is not implemented in the current versions of *Delphes*. The time-of-flight measurement can be smeared with a Gaussian distribution (default value $\sigma_t = 0$ s) [y].

6. Validation

Delphes performs a fast simulation of a collider experiment. Its performances in terms of computing time and data size are directly proportional to the number of simulated events and on the considered physics process. As an example, 10,000 $pp \to t\bar{t}X$ events are processed in 110 s on a regular laptop and use less than 250 MB of disk space. The quality and validity of the output are assessed by comparing the resolutions on the reconstructed data to the expectations of both CMS [5] and ATLAS [6] detectors.

Electrons and muons are by construction equal to the experiment designs, as the Gaussian smearing of their kinematics properties is defined according to the detector specifications. Similarly, the b-tagging efficiency (for real b-jets) and misidentification rates (for fake b-jets) are taken directly from the expected values of the experiment. Unlike these simple objects, jets and missing transverse energy should be carefully cross-checked.

6.1. Jet resolution

The majority of interesting processes at the LHC contain jets in the final state. The jet resolution obtained using *Delphes* is

[3] It is worth noting that for both CMS and ATLAS experiments, the taggers located at 220 m are not able to measure the time-of-flight, contrary to FP420 detectors.

therefore a crucial point for its validation, both for CMS- and ATLAS-like detectors. This validation is based on $pp \to gg$ events produced with MadGraph/MadEvent and hadronised using *Pythia* [17, 18].

For a CMS-like detector, a similar procedure as the one explained in published results is applied here. The events were arranged in 14 bins of gluon transverse momentum $\hat{p}_T$. In each $\hat{p}_T$ bin, every jet in *Delphes* is matched to the closest jet of generator-level particles, using the spatial separation between the two jet axes

$$\Delta R = \sqrt{(\eta^{\rm rec} - \eta^{\rm MC})^2 + (\phi^{\rm rec} - \phi^{\rm MC})^2} < 0.25. \quad (5)$$

The jets made of generator-level particles, here referred as *MC jets*, are obtained by applying the algorithm to all particles considered as stable after hadronisation (i.e. including muons). Jets produced by *Delphes* and satisfying the matching criterion are called hereafter *reconstructed jets*. All jets are computed with the clustering algorithm (JetCLU) with a cone radius R of 0.7.

The ratio of the transverse energies of every reconstructed jet $E_T^{\rm rec}$ to its corresponding MC jet $E_T^{\rm MC}$ is calculated in each $\hat{p}_T$ bin. The $E_T^{\rm rec}/E_T^{\rm MC}$ histogram is fitted with a Gaussian distribution in the interval ±2 RMS centred around the mean value. The resolution in each $\hat{p}_T$ bin is obtained by the fit mean $\langle x \rangle$ and variance $\sigma^2(x)$:

$$\frac{\sigma\left(\frac{E_T^{\rm rec}}{E_T^{\rm MC}}\right)_{\rm fit}}{\left\langle \frac{E_T^{\rm rec}}{E_T^{\rm MC}} \right\rangle_{\rm fit}} \left(\hat{p}_T(i)\right), \text{ for all } i. \quad (6)$$

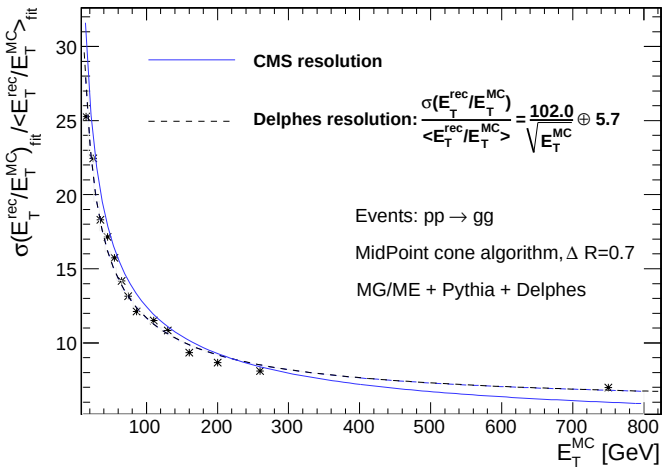

Figure 8: Resolution of the transverse energy of reconstructed jets $E_T^{\rm rec}$ as a function of the transverse energy of the closest jet of generator-level particles $E_T^{\rm MC}$, in a CMS-like detector. The jets events are reconstructed with the JetCLU clustering algorithm with a cone radius of 0.7. The maximum separation between the reconstructed and MC-jets is $\Delta R = 0.25$. Dotted line is the fit result for comparison to the CMS resolution [5], in blue. The $pp \to gg$ dijet events have been generated with MadGraph/MadEvent and hadronised with *Pythia*.

The resulting jet resolution as a function of $E_T^{\rm MC}$ is shown in Fig. 8. This distribution is fitted with a function of the following form:

$$\frac{a}{E_T^{\rm MC}} \oplus \frac{b}{\sqrt{E_T^{\rm MC}}} \oplus c, \quad (7)$$

where a, b and c are the fit parameters. It is then compared to the resolution published by the CMS collaboration [5]. The resolution curves from *Delphes* and CMS are in good agreement.

Similarly, the jet resolution is evaluated for an ATLAS-like detector. The $pp \to gg$ events are here arranged in 8 adjacent bins in p_T. A k_T reconstruction algorithm with $R = 0.6$ is chosen and the maximal matching distance between the MC-jets and the reconstructed jets is set to $\Delta R = 0.2$. The relative energy resolution is evaluated in each bin by:

$$\frac{\sigma(E)}{E} = \sqrt{\left\langle \left(\frac{E^{\rm rec} - E^{\rm MC}}{E^{\rm rec}} \right)^2 \right\rangle - \left\langle \frac{E^{\rm rec} - E^{\rm MC}}{E^{\rm rec}} \right\rangle^2}. \quad (8)$$

Figure 9 shows a good agreement between the resolution obtained with *Delphes*, the result of the fit with Equation 7 and the corresponding curve provided by the ATLAS collaboration [6].

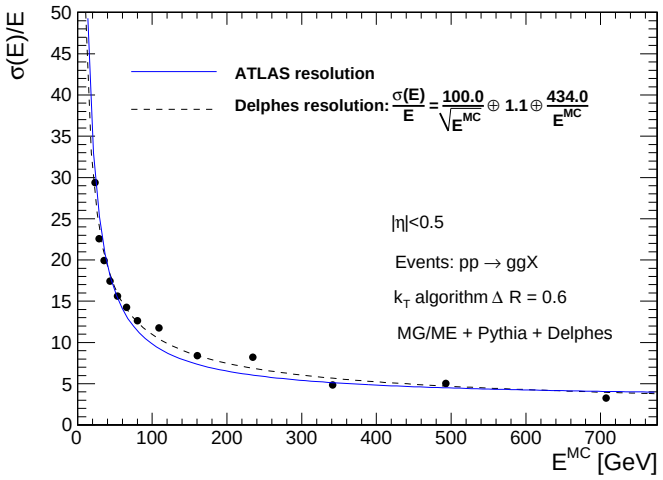

Figure 9: Relative energy resolution of reconstructed jets as a function of the energy of the closest jet of generator-level particles $E^{\rm MC}$, in an ATLAS-like detector. The jets are reconstructed with the k_T algorithm with a radius $R = 0.6$. The maximal matching distance between MC- and reconstructed jets is $\Delta R = 0.2$. Only central jets are considered ($|\eta| < 0.5$). Dotted line is the fit result for comparison to the ATLAS resolution [6], in blue. The $pp \to gg$ di-jet events have been generated with MadGraph/MadEvent and hadronised with *Pythia*.

6.2. MET resolution

All major detectors at hadron colliders have been designed to be as much hermetic as possible in order to detect the presence of one or more neutrinos and/or new weakly interacting particles through apparent missing transverse energy. The resolution of the $\overrightarrow{E_T}^{\rm miss}$ variable, as obtained with *Delphes*, is then crucial.

The samples used to study the MET performance are identical to those used for the jet validation. It is worth noting that the contribution to $E_T^{\rm miss}$ from muons is negligible in the studied sample. The input samples are divided in five bins of scalar E_T sums (ΣE_T). This sum, called *total visible transverse energy*, is defined as the scalar sum of transverse energy in all cells. The quality of the MET reconstruction is checked via the resolution on its horizontal component $E_x^{\rm miss}$.

The $E_x^{\rm miss}$ resolution is evaluated in the following way. The distribution of the difference between $E_x^{\rm miss}$ in *Delphes* and at generator-level is fitted with a Gaussian function in each (ΣE_T) bin. The fit RMS gives the MET resolution in each bin. The resulting value is plotted in Fig. 10 as a function of the total visible transverse energy, for CMS- and ATLAS-like detectors.

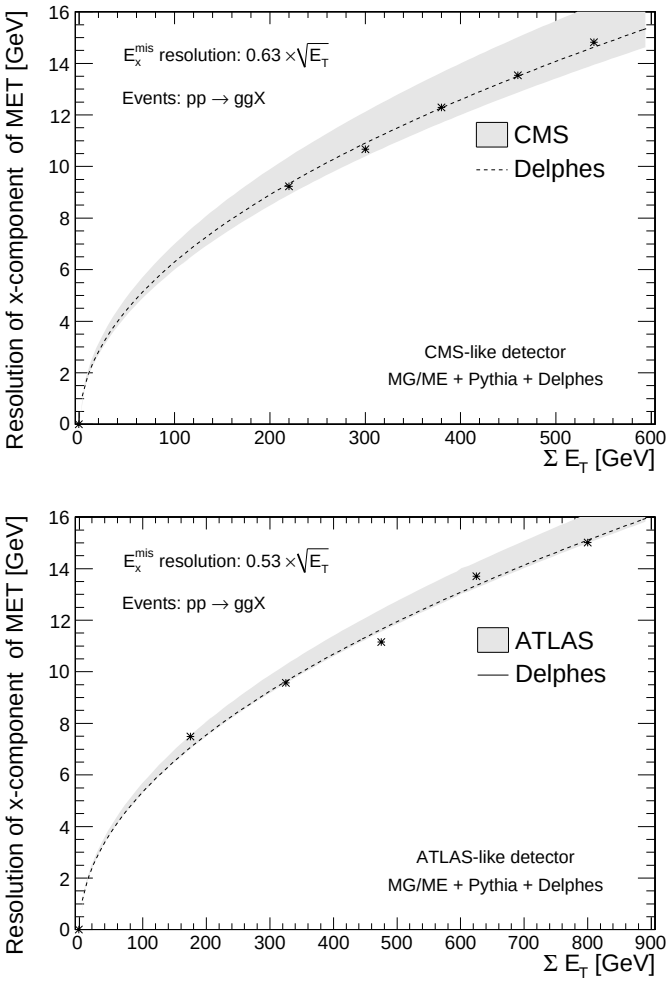

Figure 10: $\sigma(E_x^{\rm mis})$ as a function on the scalar sum of all cells (ΣE_T) for $pp \to gg$ events, for a CMS-like detector (top) and an ATLAS-like detector (bottom), for di-jet events produced with MadGraph/MadEvent and hadronised with *Pythia*.

The resolution σ_x of the horizontal component of MET is observed to behave like

$$\sigma_x = \alpha \sqrt{E_T} \quad (\mathrm{GeV}^{1/2}), \tag{9}$$

where the α parameter depends on the resolution of the calorimeters.

The MET resolution expected for the CMS detector for similar events is $\sigma_x = (0.6 - 0.7)\ \sqrt{E_T}\ \mathrm{GeV}^{1/2}$ with no pile-up (i.e. extra simultaneous pp collision occurring at high-luminosity in the same bunch crossing) [5], which compares very well with the $\alpha = 0.63$ obtained with *Delphes*. Similarly, for an ATLAS-like detector, a value of 0.53 is obtained by *Delphes* for the α parameter, while the experiment expects it in the range [0.53 ; 0.57] [6].

6.3. τ-jet efficiency

Due to the complexity of their reconstruction algorithm, τ-jets have also to be checked. Table 7 lists the reconstruction efficiencies in *Delphes* for the hadronic τ-jets from $H, Z \to \tau^+\tau^-$. The mass of the Higgs boson is set successively to 140 and 300 GeV/c^2. The inclusive gauge boson productions ($pp \to HX$ and $pp \to ZX$) are performed with MadGraph/MadEvent and the τ lepton decay and further hadronisation are handled by *Pythia/Tauola*. All reconstructed τ-jets are 1−prong, and follow the definition described in section 3.3, which is very close to an algorithm of the CMS experiment [19]. At last, corresponding efficiencies published by the CMS and ATLAS experiments are quoted for comparison. The agreement is good enough at this level to validate the τ−reconstruction.

Table 7: Reconstruction efficiencies of τ-jets in $\tau^+\tau^-$ decays from Z or H bosons, in *Delphes*, CMS and ATLAS experiments [19, 6]. Two scenarios for the mass of the Higgs boson are investigated. Events generated with MadGraph/MadEvent and hadronised with *Pythia*. The decays of τ leptons is handled by the *Tauola* version embedded in *Pythia*.

	CMS	Delphes	ATLAS	Delphes
$Z \to \tau^+\tau^-$	38.2%	32.4 ± 1.8%	33%	28.6 ± 1.9%
$H(140) \to \tau^+\tau^-$	36.3%	39.9 ± 1.6%		32.8 ± 1.8%
$H(300) \to \tau^+\tau^-$	47.3%	49.7 ± 1.5%		43.8 ± 1.6%

7. Visualisation

When performing an event analysis, a visualisation tool is useful to convey information about the detector layout and the event topology in a simple way. The *Fast and Realistic OpenGL Displayer* FROG [20] has been interfaced in *Delphes*, allowing an easy display of the defined detector configuration [z].

Two and three-dimensional representations of the detector configuration can be used for communication purposes, as they clearly illustrate the geometric coverage of the different detector subsystems. As an example, the generic detector geometry assumed in this paper is shown in Fig. 2 and 11. The extensions of the central tracking system, the central calorimeters and both forward calorimeters are visible. Note that only the geometrical coverage is depicted and that the calorimeter segmentation is not taken into account in the drawing of the detector. Moreover, both the radius and the length of each sub-detectors are just display parameters and are not relevant for the physics simulation.

Deeper understanding of interesting physics processes is possible by displaying the events themselves. The visibility of each set of objects ($e^\pm$, $\mu^\pm$, $\tau^\pm$, jets, transverse missing energy) is enhanced by a colour coding. Moreover, kinematics information of each object is visible by a simple mouse action. As an illustration, an associated photoproduction of a W boson and a t quark is shown in Fig. 12. This corresponds to a $pp(\gamma p \to Wt)pX$ process, where the Wt couple is induced by an incoming photon emitted by one of the colliding proton [21]. This leading proton survives after photon

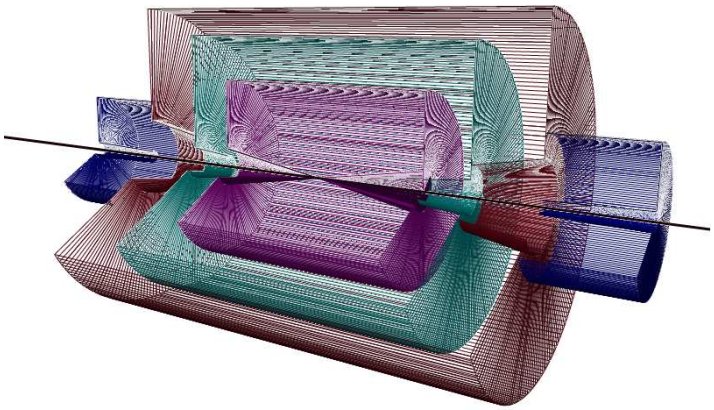

Figure 11: Layout of the generic detector geometry assumed in *Delphes*. Open 3D-view of the detector with solid volumes. Same colour codes as for Fig. 2 are applied. Additional forward detectors are not depicted.

emission and is present in the final state. As the energy and virtuality of the emitted photon are low, the surviving proton does not leave the beam and escapes from the central detector without being detected. The experimental signature is a lack of hadronic activity in the forward hemisphere where the surviving proton escapes. The t quark decays into a W boson and a b quark. Both W bosons decay into leptons ($W \to \mu\nu_\mu$ and $W \to e\nu_e$). The balance between the missing transverse energy and the charged lepton pair is clear, as well as the presence of an empty forward region. It is interesting to notice that the reconstruction algorithms build a fake τ-jet around the electron.

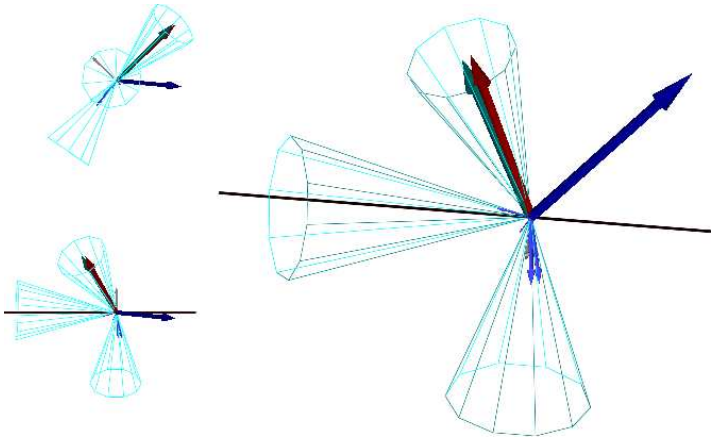

Figure 12: Example of $pp(\gamma p \to Wt)pY$ event display in different orientations, with $t \to Wb$. One W boson decays into a $\mu\nu_\mu$ pair and the second one into a $e\nu_e$ pair. The surviving proton leaves a forward hemisphere with no hadronic activity. The isolated muon is shown as the dark blue vector. Around the electron, in red, is reconstructed a fake τ-jet (green vector surrounded by a blue cone), while the reconstructed missing energy (in grey) is very small. One jet is visible in one forward region, along the beamline axis, opposite to the direction of the escaping proton.

For comparison, Fig. 13 depicts an inclusive gluon pair production $pp \to ggX$. The event final state contains more jets, in particular along the beam axis, which is expected as the interacting protons are destroyed by the collision. Two muon candidates and large missing transverse energy are also visible.

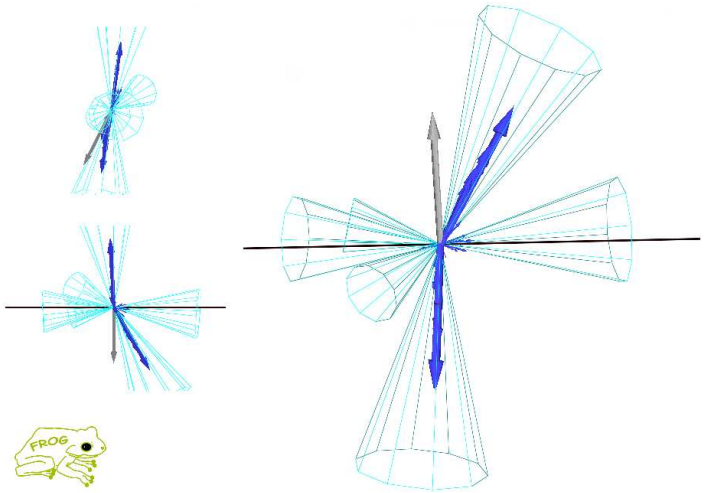

Figure 13: Example of inclusive gluon pair production $pp \to ggX$. Many jets are visible in the event, in particular along the beam axis. Two muons (in blue) are produced and the missing transverse energy is significant in this event (grey vector).

8. Conclusion and perspectives

We have described here the major features of the *Delphes* framework, introduced for the fast simulation of a collider experiment. This framework is a tool meant for feasibility studies in phenomenology, gauging the observability of model predictions in collider experiments.

Delphes takes as an input the output of event-generators and yields analysis-object data in the form of `TTree` in a `*.root` file. The simulation includes central and forward detectors to produce realistic observables using standard reconstruction algorithms. Moreover, the framework allows trigger emulation and 3D event visualisation.

Delphes has been developed using the parameters of the CMS experiment but can be easily extended to ATLAS and other non-LHC experiments, as at Tevatron or at the ILC. Further developments include a more flexible design for the subdetector assembly, a better b-tag description and possibly the implementation of an event mixing module for pile-up event simulation. This framework has already been used for several analyses [21, 22, 23], in particular in photon-induced interactions at the LHC.

Acknowledgements

The authors would like to thank Jerôme de Favereau, Christophe Delaere, Muriel Vander Donckt and David d'Enterria for useful discussions and comments, and Loic Quertenmont for support in interfacing FROG. We are also really grateful to Alice Dechambre and Simon de Visscher for being beta testers of the complete package. Part of this work was supported by the Belgian Federal Office for Scientific, Technical and Cultural Affairs through the Interuniversity Attraction Pole P6/11.

References

[1] J. Allison, et al., Nucl. Inst. & Meth. in **Phys. Res. A** 506 (2003) 250-303; **IEEE Trans. on Nucl. Sc.** 53:1 (2006) 270-278.
[2] *Delphes*, www.fynu.ucl.ac.be/delphes.html
[3] R. Brun, F. Rademakers, Nucl. Inst. & Meth. in **Phys. Res. A** 389 (1997) 81-86.
[4] P. Demin, (2006), unpublished. Now part of MadGraph/MadEvent.
[5] The CMS Collaboration, **CERN/LHCC** 2006-001.
[6] The ATLAS Collaboration, **CERN-OPEN** 2008-020, arXiv:0901.0512v1[hep-ex].
[7] X. Rouby, J. de Favereau, K. Piotrzkowski, **JINST** 2 P09005 (2007).
[8] M. Cacciari, G.P. Salam, **Phys. Lett. B** 641 (2006) 57.
[9] F. Abe et al. (CDF Coll.), **Phys. Rev. D** 45 (1992) 1448.
[10] G.C. Blazey, et al., arXiv:0005012[hep-ex].
[11] G.P. Salam, G. Soyez, **JHEP** 05 (2007) 086.
[12] S. Catani, Y.L. Dokshitzer, M.H. Seymour, B.R. Webber, **Nucl. Phys. B** 406 (1993) 187; S.D. Ellis, D.E. Soper, **Phys. Rev. D** 48 (1993) 3160.
[13] Y.L. Dokshitzer, G.D. Leder, S. Moretti, B.R. Webber, **JHEP** 08 (1997) 001; M. Wobisch, T. Wengler, arXiv:9907280[hep-ph].
[14] M. Cacciari, G.P. Salam, G. Soyez, **JHEP** 04 (2008) 063.
[15] C. Amsler et al. (Particle Data Group), **Phys. Lett. B** 667 (2008) 1.
[16] S. Ovyn, **Nucl. Phys. Proc. Suppl.** 179-180 (2008) 269-276.
[17] J. Alwall, et al., **JHEP** 09 (2007) 028.
[18] T. Sjostrand, S. Mrenna, P. Skands, **JHEP** 05 (2006) 026.
[19] R. Kinnunen, A.N. Nikitenko, **CMS NOTE** 1997/002.
[20] L. Quertenmont, V. Roberfroid, **CMS CR** 2009/028, arXiv:0901.2718v1[hep-ex].
[21] J. de Favereau de Jeneret, S. Ovyn, **Nucl. Phys. Proc. Suppl.** 179-180 (2008) 277-284; S. Ovyn, J. de Favereau de Jeneret, Nuovo Cimento B, arXiv:0806.4841[hep-ph].
[22] J. de Favereau et al, arXiv:0908.2020v1 [hep-ph] (2008), to be published in EPJ.
[23] S. de Visscher, J.M. Gerard, M. Herquet, V. Lemaître, F. Maltoni, arXiv:0904.0705[hep-ph].
[24] P. Lebrun, L. Garren, Copyright (c) 1994-1995 Universities Research Association, Inc.
[25] L.A. Garren, M. Fischler, cepa.fnal.gov/psm/stdhep/c++
[26] M. Dobbs and J.B. Hansen, **Comput. Phys. Commun.** 134 (2001) 41.
[27] J. Alwall, et al., **Comput. Phys. Commun.** 176:300-304,2007.

Internal code references

[a] The standard Monte Carlo event structures `StdHEP` [25] and `HepMC` [26] can be used as an input. Besides, *Delphes* can also provide detector response for events read in "Les Houches Event Format" (LHEF [27]) and `*.root` files obtained from `*.hbook` using the `h2root` utility from the ROOT framework [3]. See the following classes: `HEPEVTConverter`, `HepMCConverter`, `LHEFConverter`, `STDHEPConverter` and `DelphesRootConverter`.
[b] The ROOT output files are created using the `ExRootAnalysis` utility [4]. Generator-level data are located under the `GEN` tree, the analysis data objects after reconstruction under the `Analysis` tree, and the results of the trigger emulation under the `Trigger` tree.
[c] Set the `FLAG_LHCO` variable to 1 or 0 in the detector card to switch on/off the creation of `*.lhco` output file.
[d] The list of particles considered as invisible is accessible in the `PdgParticle` class. This list currently contains the PIDs 12, 14, 16, 1000022, 1000023, 1000025, 1000035 and 1000045, in absolute values.
[e] The detector card is the `data/DetectorCard.dat` file. This file is parsed by the `SmearUtil` class.
[f] Detector and trigger cards for the ATLAS and CMS experiments are also provided in `data/` directory.
[g] The resolution terms in the detector card are named `ELG_Xyyy` or `HAD_Xyyy`, refering to electromagnetic and hadronic terms (resp.); `X` is replaced by `S`, `N`, `C` for the stochastic, noise and constant terms; and finally `yyy` is `cen` for central part, `ec` for end-caps, `fwd` for the forward calorimeters and `zdc` for the zero-degree calorimeters.
[h] See the `TrackPropagation` class.
[i] See the `TRACK_eff` and `TRACK_ptmin` terms in the detector card.
[j] The response of the detector is applied to the electromagnetic and the hadronic particles through the `SmearElectron` and `SmearHadron` methods in the `SmearUtil` class.
[k] To implement different ratios for other particles, see the `BlockClasses` class.
[l] As the detector is assumed to be cylindrical (e.g. symmetric in ϕ and with respect to the $\eta = 0$ plane), the detector card stores the number of calorimetric cells with $\phi = 0$ and $\eta > 0$ (default: 40 cells). For a given η, the size of the ϕ segmentation is also specified. See the `TOWER_number`, `TOWER_eta_edges` and `TOWER_dphi` variables in the detector card.
[m] All these processed data are located under the `Analysis` tree.
[n] See the `SmearMuon` method in the `SmearUtil` class.
[o] See the `IsolFlag` and `IsolPt` values in the `Electron` or `Muon` collections in the `Analysis` tree, as well as the `ISOL_PT` and `ISOL_Cone` variables in the detector card.
[p] Calorimetric isolation parameters in the detector card are `ISOL_Calo_ET` and `ISOL_Calo_Grid` in the detector card.
[q] These thresholds are defined by the `ZDC_gamma_E` and `ZDC_n_E` variables in the detector card.
[r] The choice is done by allocating the `JET_jetalgo` input parameter in the detector card.
[s] See the `PTCUT_jet` variable in the detector card.
[t] See the `JET_coneradius` and `JET_seed` variables in the detector card. The existing FastJet code has been modified to allow easy modification of the cell pattern in (η, ϕ) space. In following versions of *Delphes*, a new dedicated plug-in will be created on this purpose.
[u] Set `JET_Eflow` to 1 or 0 in the detector card in order to switch on or off the energy flow for jet reconstruction.
[v] Corresponding to the `BTAG_b`, `BTAG_mistag_c` and `BTAG_mistag_l` constants, for the efficiency of tagging of a *b*-jet, the efficiency of mistagging a *c*-jet as a *b*-jet, and the efficiency of mistagging a light jet (u,d,s,g) as a *b*-jet.
[w] See the following parameters in the detector card: `TAU_energy_scone` for R^{em}; `JET_M_seed` for min E_T^{cell}; `TAU_energy_frac` for C_τ; `TAU_track_scone` for R^{tracks}; `PTAU_track_pt` for min p_T^{tracks} and `TAUJET_pt` for min p_T.
[x] The trigger card is the `data/TriggerCard.dat` file. Default trigger files are also available for CMS-like and ATLAS-like detectors
[y] The resolution is defined by the `RP220_T_resolution` and `RP420_T_resolution` parameters in the detector card.
[z] To prepare the visualisation, the `FLAG_FROG` parameter should be equal to 1.

A. User manual

The available C++-code is compressed in a zipped tar file which contains everything needed to run the *Delphes* package, assuming a running ROOT installation. The package includes `ExRootAnalysis` [4], *Hector* [7], FastJet [8], and FROG [20], as well as the conversion codes to read standard StdHEP input files (`mcfio` and `stdhep`) [24] and HepMC [26]. In order to visualise the events with the FROG software, a few additional external libraries may be required, as explained in http://projects.hepforge.org/FROG/.

A.1. Getting started

In order to run *Delphes* on your system, first download its sources and compile them:

```
wget http://www.fynu.ucl.ac.be/users/s.ovyn/Delphes/files/Delphes_V_*.tar.gz
```

Replace the * symbol by the proper version number. Always refer to the download page on the *Delphes* website http://www.fynu.ucl.ac.be/users/s.ovyn/Delphes/download.html. Current version of Delphes for this manual is V 1.8 (July 2009).

```
me@mylaptop:~$ tar -xvf Delphes_V_*.tar.gz
me@mylaptop:~$ cd Delphes_V_*.*
me@mylaptop:~$ ./genMakefile.tcl > Makefile
me@mylaptop:~$ make
```

Due to the large number of external utilities, the number of printed lines during the compilation can be high. The user should not pay attention to possible warning messages, which are due to the external packages used by *Delphes*. When compilation is completed, the following message is printed:

```
me@mylaptop:~$ Delphes has been compiled
me@mylaptop:~$ Ready to run
```

A.2. Running Delphes on your events

In this sub-appendix, we will explain how to use *Delphes* to perform a fast simulation of a general-purpose detector on your event files. The first step to use *Delphes* is to create the list of input event files (e.g. `inputlist.list`). It is important to notice that all the files comprised in the list file should have the same of extension (`*.hep`, `*.lhe`, `*.hepmc` or `*.root`). In the simplest way to run *Delphes*, you need this input file and you need to specify the name of the output file that will contain the generator-level data (`GEN` tree), the analysis data objects after reconstruction (`Analysis` tree), and the results of the trigger emulation (`Trigger` tree).

```
me@mylaptop:~$ ./Delphes inputlist.list OutputRootFileName.root
```

A.2.1. Setting up the configuration

The program is driven by two datacards (default cards are `data/DetectorCard.dat` and `data/TriggerCard.dat`) which allow the user to choose among a large spectrum of running conditions. Please note that if the user does not provide these datacards, the running will be done using the default parameters defined in the constructor of the class `RESOLution` (see next). If you choose a different detector or running configuration, you will need to edit the datacards accordingly. Detector and trigger cards are provided in the `data/` subdirectory for the CMS and ATLAS experiments.

1. **The detector card** It contains all pieces of information needed to run *Delphes*:

   - detector parameters, including calorimeter and tracking coverage and resolutions, transverse energy thresholds for object reconstruction and jet algorithm parameters.
   - six flags (`FLAG_bfield`, `FLAG_vfd`, `FLAG_RP`, `FLAG_trigger`, `FLAG_FROG` and `FLAG_LHCO`), should be set in order to configure the magnetic field propagation, the very forward detectors simulation, the use of very forward taggers, the trigger selection, the preparation for FROG display and the creation of an output file in `*.LHCO` text format (respectively).

   If no datacard is provided by the user, the default smearing and running parameters are used (corresponding to tables 1, 2).
   Definition of the sub-detector extensions:

```
CEN_max_tracker    2.5     // Maximum tracker coverage
CEN_max_calo_cen   1.7     // central calorimeter coverage
CEN_max_calo_ec    3.0     // calorimeter endcap coverage
CEN_max_calo_fwd   5.0     // forward calorimeter pseudorapidity coverage
CEN_max_mu         2.4     // muon chambers pseudorapidity coverage
```

Definition of the sub-detector resolutions:

```
# Energy resolution for electron/photon in central/endcap/fwd/zdc calos
# \sigma/E = C + N/E + S/\sqrt{E}, E in GeV
ELG_Scen          0.05     // S term for central ECAL
ELG_Ncen          0.25     // N term
ELG_Ccen          0.005    // C term
ELG_Sec           0.05     // S term for ECAL endcap
ELG_Nec           0.25     // N term
ELG_Cec           0.005    // C term
ELG_Sfwd          2.084    // S term for FCAL
ELG_Nfwd          0.       // N term
ELG_Cfwd          0.107    // C term
ELG_Szdc          0.70     // S term for ZDC
ELG_Nzdc          0.       // N term
ELG_Czdc          0.08     // C term

# Energy resolution for hadrons in central/endcap/fwd/zdc calos
# \sigma/E = C + N/E + S/\sqrt{E}, E in GeV
HAD_Scen          1.5      // S term for central HCAL
HAD_Ncen          0.       // N term
HAD_Ccen          0.05     // C term
HAD_Sec           1.5      // S term for HCAL endcap
HAD_Nec           0.       // N term
HAD_Cec           0.05     // C term
HAD_Sfwd          2.7      // S term for FCAL
HAD_Nfwd          0.       // N term
HAD_Cfwd          0.13     // C term
HAD_Szdc          1.38     // S term for ZDC
HAD_Nzdc          0.       // N term
HAD_Czdc          0.13     // C term

# Time resolution for ZDC/RP220/RP420
ZDC_T_resolution   0       // in s
RP220_T_resolution 0       // in s
RP420_T_resolution 0       // in s

# Muon smearing
MU_SmearPt        0.01     // transverse momentum Pt in GeV/c

# Tracking efficiencies
TRACK_ptmin       0.9      // minimal pT
TRACK_eff          90      // efficiency associated to the tracking (%)
```

Definitions related to the calorimetric cells:

```
# Calorimetric towers
TOWER_number         40
TOWER_eta_edges 0. 0.087 0.174 0.261 0.348 0.435 0.522 0.609 0.696 0.783
                0.870 0.957 1.044 1.131 1.218 1.305 1.392 1.479 1.566 1.653
                1.740 1.830 1.930 2.043 2.172 2.322 2.500 2.650 2.868 2.950
                3.125 3.300 3.475 3.650 3.825 4.000 4.175 4.350 4.525 4.700
                5.000

TOWER_dphi 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 10
           10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 20 20
```

`TOWER_eta_edges` is the list of the edges in η of all cells, in the $\eta > 0$ hemisphere (the detector is supposed to be symmetric with respect to the $\eta = 0$ plane, as well as around the z-axis). Starts with the lower edge of the most central tower (default: $\eta = 0$) and ends

with the higher edge of the most forward tower. TOWER_dphi lists the tower size in ϕ (in degree), assuming that all cells are similar in ϕ for a given η.
Thresholds applied for storing the reconstructed objects in the final collections:

```
# Thresholds for reconstructed objects, in GeV/c
PTCUT_elec       10.0
PTCUT_muon       10.0
PTCUT_jet        20.0
PTCUT_gamma      10.0
PTCUT_taujet     10.0

# Thresholds for reconstructed objects in ZDC, E in GeV
ZDC_gamma_E      20
ZDC_n_E          50
```

Definitions of variables related to the charged lepton isolation:

```
# Charged lepton isolation. Pt and Et in GeV
ISOL_PT          2.0  //minimal pt of tracks for isolation criteria
ISOL_Cone        0.5  //Cone  for isolation criteria
ISOL_Calo_Cone   0.4  //Cone for calorimetric isolation
ISOL_Calo_ET     2.0  //minimal tower E_T for isolation criteria. 1E99 means "off"
ISOL_Calo_Grid   3    //Grid size (N x N) for calorimetric isolation
```

Definitions of variables related to the jet reconstruction:

```
# General jet variable
JET_coneradius   0.7  // generic jet radius
JET_jetalgo      1    // 1 for Cone algorithm,
                      // 2 for MidPoint algorithm,
                      // 3 for SIScone algorithm,
                      // 4 for kt algorithm
                      // 5 for Cambridge/Aachen algorithm
                      // 6 for anti-kt algorithm
JET_seed         1.0  // minimum seed to start jet reconstruction, in GeV
JET_Eflow        1    // Energy flow: perfect energy assumed in the tracker coverage.
                      // 1 is 'on' ; 0 is 'off'

# Tagging definition
BTAG_b           40    // b-tag efficiency (%)
BTAG_mistag_c    10    // mistagging (%)
BTAG_mistag_l    1     // mistagging (%)
```

Switches for options

```
# FLAGS
FLAG_bfield      1     //1 to run the bfield propagation else 0
FLAG_vfd         1     //1 to run the very forward detectors else 0
FLAG_RP          1     //1 to run the very forward detectors else 0
FLAG_trigger     1     //1 to run the trigger selection else 0
FLAG_FROG        1     //1 to run the FROG event display
FLAG_LHCO        1     //1 to run the LHCO
```

Parameters for the magnetic field simulation:

```
# In case BField propagation allowed
TRACK_radius      129   // radius of the BField coverage, in cm
TRACK_length      300   // length of the BField coverage, in cm
TRACK_bfield_x    0     // X component of the BField, in T
TRACK_bfield_y    0     // Y component of the BField, in T
TRACK_bfield_z    3.8   // Z component of the BField, in T
```

Parameters related to the very forward detectors

```
# Very forward detector extension, in pseudorapidity
# if allowed
VFD_min_zdc       8.3   // Zero-Degree neutral Calorimeter
VFD_s_zdc         140   // distance of the ZDC, from the IP, in [m]

#\textit{Hector} parameters
RP_220_s          220     // distance of the RP to the IP, in meters
RP_220_x          0.002   // distance of the RP to the beam, in meters
RP_420_s          420     // distance of the RP to the IP, in meters
RP_420_x          0.004   // distance of the RP to the beam, in meters
RP_beam1Card      data/LHCB1IR5_v6.500.tfs // beam optics file, beam 1
RP_beam2Card      data/LHCB2IR5_v6.500.tfs // beam optics file, beam 2
RP_IP_name        IP5     // tag for IP in \textit{Hector} ; 'IP1' for ATLAS
RP_offsetEl_x     0.097   // horizontal separation between both beam, in meters
RP_offsetEl_y     0       // vertical separation between both beam, in meters
RP_offsetEl_s     120     // distance of beam separation point, from IP
RP_cross_x        -500    // IP offset in horizontal plane, in micrometers
RP_cross_y        0       // IP offset in vertical plane, in micrometers
RP_cross_ang_x    142.5   // half-crossing angle in horizontal plane, in microrad
RP_cross_ang_y    0       // half-crossing angle in vertical plane, in microrad
```

Others parameters:

```
# In case FROG event display allowed
NEvents_FROG      100
# Number of events to process
NEvents           -1                    // -1 means 'all'

# input PDG tables
PdgTableFilename  data/particle.tbl     // table with particle pid,mass,charge,...
```

In general, energies, momenta and masses are expressed in GeV, GeV/c, GeV/c^2 respectively, and magnetic fields in T. Geometrical extension are often referred in terms of pseudorapidity η, as the detectors are supposed to be symmetric in ϕ. From version 1.8 onwards, the number of events to run is also be included in the detector card (`NEvents`). For version 1.7 and earlier, the parameters related to the calorimeter endcaps (`CEN_max_calo_ec`, `ELG_Sec`, `ELG_Nec`, `ELG_Cec`, `HAD_Sec`, `HAD_Nec` and `HAD_Cec`) did not exist in the detector cards; in addition, some other variables had different names (`HAD_Scen` was `HAD_Sfcal`, `HAD_Ncen` was `HAD_Nfcal`, `HAD_Ccen` was `HAD_Cfcal`, `HAD_Sfwd` was `HAD_Shf`, `HAD_Nfwd` was `HAD_Nhf`, `HAD_Cfwd` was `HAD_Chf`). However, these cards are still completely compatible with new versions of *Delphes*. In such a case, the calorimeter endcaps are simply assumed to be located at the edge of the central calorimeter volumes, with the same resolution values.

2. **The trigger card**

This card contains the definitions of all trigger-bits. Cuts can be applied on the transverse momentum p_T of electrons, muons, jets, τ-jets, photons and the missing transverse energy. The following codes should be used so that *Delphes* can correctly translate the input list of trigger-bits into selection algorithms:

Trigger code	*Corresponding object*
`ELEC_PT`	electron
`IElec_PT`	isolated electron
`MUON_PT`	muon
`IMuon_PT`	isolated muon
`JET_PT`	jet
`TAU_PT`	τ-jet
`ETMIS_PT`	missing transverse energy
`GAMMA_PT`	photon
`Bjet_PT`	b-jet

Each line in the trigger datacard is allocated to exactly one trigger-bit and starts with the name of the corresponding trigger. Logical combination of several conditions is also possible. If the trigger-bit requires the presence of multiple identical objects, the order of their p_T thresholds is very important: they must be defined in *decreasing* order. The transverse momentum p_T is expressed in GeV/c. Finally, the different requirements on the objects must be separated by a `&&` flag. The default trigger card can be found in the data repository of *Delphes* (`data/TriggerCard.dat`), as well as for both CMS and ATLAS experiments at the LHC. An example of trigger table consistent with the previous rules is given here:

```
SingleJet                  >> JET_PT: '200'
DoubleElec                 >> ELEC_PT: '20' && ELEC_PT: '10'
SingleElec and Single Muon  >> ELEC_PT: '20' && MUON_PT: '15'
```

A.2.2. Running the code

First, create the detector and trigger cards (`data/DetectorCard.dat` and `data/TriggerCard.dat`).
Then, create a text file containing the list of input files that will be used by *Delphes* (with extension `*.lhe`, `*.hepmc`, `*.root` or `*.hep`). To run the code, type the following command (in one line)

```
me@mylaptop:~$ ./Delphes inputlist.list OutputRootFileName.root
                         data/DetectorCard.dat data/TriggerCard.dat
```

As a reminder, typing the `./Delphes` command simply displays the correct usage:

```
me@mylaptop:~$ ./Delphes
 Usage: ./Delphes input_file output_file [detector_card] [trigger_card]
 input_list - list of files in Ntpl, StdHep, HepMC or LHEF format,
 output_file - output file.
 detector_card - Card containing resolution variables for detector simulation (optional)
 trigger_card - Card containing the trigger algorithms (optional)
```

A.3. Getting the Delphes information

A.3.1. Contents of the Delphes ROOT trees

The *Delphes* output file (`*.root`) is subdivided into three *trees*, corresponding to generator-level data, analysis-object data and trigger output. These *trees* are structures that organise the output data into *branches* containing data (or *leaves*) related with each others, like the kinematics properties (E, p_x, η, ...) of a given particle.

Here is the exhaustive list of *branches* availables in these *trees*, together with their corresponding physical objet and `ExRootAnalysis` C++ class name:

GEN `Tree`		
Particle	generator particles from HEPEVT	`GenParticle`
Trigger `Tree`		
TrigResult	Acceptance of different trigger-bits	`TRootTrigger`
Analysis `Tree`		
Tracks	Collection of tracks	`TRootTracks`
CaloTower	Calorimetric cells	`TRootCalo`
Electron	Collection of electrons	`TRootElectron`
Photon	Collection of photons	`TRootPhoton`
Muon	Collection of muons	`TRootMuon`
Jet	Collection of jets	`TRootJet`
TauJet	Collection of jets tagged as τ-jets	`TRootTauJet`
ETmis	Transverse missing energy information	`TRootETmis`
ZDChits	Hits in the Zero Degree Calorimeters	`TRootZdcHits`
RP220hits	Hits in the first proton taggers	`TRootRomanPotHits`
FP420hits	Hits in the next proton taggers	`TRootRomanPotHits`

The third column shows the names of the corresponding classes to be written in a ROOT tree. The bin number in the unique leaf in the `trigger` tree (namely, `TrigResult.Accepted`) corresponds to the trigger number in the provided list. In addition, the result of the global trigger decision upon each event (i.e. the logical `OR` of all trigger conditions) is stored in the first bin (number 0) of this leaf. In `Analysis` tree, all classes except `TRootTracks`, `TRootCalo`, `TRootTrigger`, `TRootETmis` and `TRootRomanPotHits` inherit from the class `TRootParticle` which includes the following data members (stored as *leaves* in *branches* of the *trees*):

Most common leaves

```
float E;     // particle energy in GeV
float Px;    // particle momentum vector (x component) in GeV/c
float Py;    // particle momentum vector (y component) in GeV/c
float Pz;    // particle momentum vector (z component) in GeV/c
float PT;    // particle transverse momentum in GeV/c
float Eta;   // particle pseudorapidity
float Phi;   // particle azimuthal angle in rad
```

In addition to their kinematics, some additional properties are available for specific objects:

Leaves in the `Particle` branch (`GEN` tree)

```
int PID;        // particle HEP ID number
int Status;     // particle status
int M1;         // particle 1st mother
int M2;         // particle 2nd mother
int D1;         // particle 1st daughter
int D2;         // particle 2nd daughter
float Charge;   // electrical charge in units of e
float T;        // particle vertex position (t component, in mm/c)
float X;        // particle vertex position (x component, in mm)
float Y;        // particle vertex position (y component, in mm)
float Z;        // particle vertex position (z component, in mm)
float M;        // particle mass in GeV/c^2
```

Additional leaves in `Electron` and `Muon` branches (`Analysis` tree)

```
int Charge       // particle Charge
bool IsolFlag    // stores the result of the tracking isolation test
float IsolPt     // sum of all track pt in isolation cone (GeV/c)
float EtaCalo    // particle pseudorapidity when entering the calo
float PhiCalo    // particle azimuthal angle in rad when entering the calo
float EHoverEE   // hadronic energy over electromagnetic energy
float EtRatio    // calo Et in NxN-cell grid around the muon over the muon Et
```

Additional leaf in the `Jet` branch (`Analysis` tree)

```
bool Btag        // stores the result of the b-tagging
int NTracks      // number of tracks associated to the jet
float EHoverEE   // hadronic energy over electromagnetic energy
```

Leaves in the `Tracks` branch (`Analysis` tree)

```
float Eta        // pseudorapidity at the beginning of the track
float Phi        // azimuthal angle at the beginning of the track
float EtaOuter   // pseudorapidity at the end of the track
float PhiOuter   // azimuthal angle at the end of the track
float PT         // track transverse momentum in GeV/c
float E          // track energy in GeV
float Px         // track momentum vector (x component) in GeV/c
float Py         // track momentum vector (y component) in GeV/c
float Pz         // track momentum vector (z component) in GeV/c
float Charge     // track charge in units of e
```

Leaves in the `CaloTower` branch (`Analysis` tree)

```
float Eta      // pseudorapidity of the cell
float Phi      // azimuthal angle of the cell in rad
float E        // cell energy in GeV
float E_em     // electromagnetic component of the cell energy in GeV
float E_had    // hadronic component of the cell energy in GeV
float ET       // cell transverse energy in GeV
```

Leaves in the `ETmis` branch (`Analysis` tree)

```
float Phi      // azimuthal angle of the transverse missing energy in rad
float ET       // transverse missing energy in GeV
float Px       // x component of the transverse missing energy in GeV
float Py       // y component of the transverse missing energy in GeV
```

The hits in very forward detector (ZDC, RP220, FP420) have some common data. In particular, the `side` variable tells in which detector (left:-1 or right:+1 of the interaction point) the hit has been seen. Moreover, some generator level data is provided for information, as the correspondance with the contents of the `GEN` tree is not possible. These generator-level data correspond to the particle kinematics (energy, momentum, angle) and identification (pid).

Common leaves for ZDC, RP220, FP420

```
float T          // time of flight in s
float E          // measured/smeared energy in GeV
int side         // -1 or +1
```

Generator level data

```
int pid;         // particle ID
float genPx;     // particle momentum vector (x component) in GeV/c
float genPy;     // particle momentum vector (y component) in GeV/c
float genPz;     // particle momentum vector (z component) in GeV/c
float genPT;     // particle transverse momentum in GeV/c
float genEta;    // particle pseudorapidity
float genPhi;    // particle azimuthal angle in rad
```

Additional leaves in the `ZDChits` branch (`Analysis` tree)

```
int hadronic_hit  // 0(is not hadronic) or 1(is hadronic)
```

Additional leaves in the `RP220hits` and `FP420hits` branches (`Analysis` tree)

```
flaot S      // detector position from IP in m
float X      // hit horizontal position in m
float Y      // hit vertical position in m
float TX     // hit horizontal angle in rad
float TY     // hit vertical angle in rad
float q2     // reconstructed momentum transfer in GeV^2
```

The hit position is computed from the center of the beam position, not from the edge of the detector.

A.4. Deeper description of jet algorithms

In this section, we briefly describe the differences between the six jet algorithms interfaced in *Delphes*, via the FastJet utiliy [8]. Jet algorithms differ in their sensitivity to soft particles or collinear splittings, and in their computing speed performances. The first three belong to the cone algorithm class while the last three are using a sequential recombination scheme. For all of them, the calorimetric cells are used as inputs for the jet clustering.

Cone algorithms

1. *CDF Jet Clusters* [9]: Basic cone reconstruction algorithm used by the CDF experiment in Run II). All cells lying in a circular cone around the jet axis with a transverse energy E_T higher than a given threshold are used to seed the jet candidates. This algorithm is fast but sensitive to both soft particles and collinear splittings.

2. *CDF MidPoint* [10]: Cone reconstruction algorithm developed for the CDF Run II to reduce infrared and collinear sensitivities compared to purely seed-based cone by adding 'midpoints' (energy barycentres) in the list of cone seeds.

3. *Seedless Infrared Safe Cone* [11]: The SISCone algorithm is simultaneously insensitive to additional soft particles and collinear splittings, and fast enough to be used in experimental analysis.

Recombination algorithms

The three sequential recombination jet algorithms are safe with respect to soft radiations (*infrared*) and collinear splittings. They rely on recombination schemes where calorimeter cell pairs are successively merged. The definitions of the jet algorithms are similar except for the definition of the *distances d* used during the merging procedure. Two such variables are defined: the distance d_{ij} between each pair of cells (i, j), and a variable d_{iB} (*beam distance*) depending on the transverse momentum of the cell i. The jet reconstruction algorithm browses the calorimetric cell list. It starts by finding the minimum value $d_{\min}$ of all the distances d_{ij} and d_{iB}. If $d_{\min}$ is a d_{ij}, the cells i and j are merged into a single cell with a four-momentum $p^\mu = p^\mu(i) + p^\mu(j)$ (*E-scheme recombination*). If $d_{\min}$ is a d_{iB}, the cell is declared as a final jet and is removed from the input list. This procedure is repeated until no cells are left in the input list. Further information on these jet algorithms is given here below, using k_{ti}, y_i and ϕ_i as the transverse momentum, rapidity and azimuth of calorimetric cell i and $\Delta R_{ij} = \sqrt{(y_i - y_j)^2 + (\phi_i - \phi_j)^2}$ as the jet-radius parameter:

4. *Longitudinally invariant k_t jet* [12], with $d_{ij} = \min(k_{ti}^2, k_{tj}^2) \times \frac{\Delta R_{ij}^2}{R^2}$ and $d_{iB} = k_{ti}^2$,

5. *Cambridge/Aachen jet* [13], with $d_{ij} = \frac{\Delta R_{ij}^2}{R^2}$ and $d_{iB} = 1$,

6. *Anti k_t jet* [14], where hard jets are exactly circular in the (y, ϕ) plane: $d_{ij} = \min(1/k_{ti}^2, 1/k_{tj}^2) \times \frac{\Delta R_{ij}^2}{R^2}$ and $d_{iB} = \frac{1}{k_{ti}^2}$.

A.5. Running an analysis on your Delphes events

To analyse the ROOT ntuple produced by *Delphes*, the simplest way is to use the `Analysis_Ex.cpp` code which is coming in the `Examples` repository of *Delphes*. Note that all of this is optional and done to facilitate the analyses, as the output from *Delphes* is viewable with the standard ROOT `TBrowser` and can be analysed using the `MakeClass` facility. As an example, here is a simple overview of a `myoutput.root` file created by *Delphes*:

```
me@mylaptop:~$ root -l myoutput.root
root [0]
Attaching file myoutput.root as _file0...
root [1] .ls
TFile**         myoutput.root
 TFile*         myoutput.root
  KEY: TTree    GEN;1   Analysis tree
  KEY: TTree    Analysis;1      Analysis tree
  KEY: TTree    Trigger;1       Analysis tree
root [2] TBrowser t;
root [3] Analysis->GetEntries()
(const Long64_t)200
root [4] GEN->GetListOfBranches()->ls()
OBJ: TBranchElement  Event       Event_ : 0 at: 0x9108f30
OBJ: TBranch         Event_size  Event_size/I : 0 at: 0x910cfd0
OBJ: TBranchElement  Particle    Particle_ : 0 at: 0x910c6b0
OBJ: TBranch  Particle_size  Particle_size/I : 0 at: 0x9111c58
root [5] Trigger->GetListOfLeaves()->ls()
OBJ: TLeafElement TrigResult_          TrigResult_ : 0 at: 0x90f90a0
OBJ: TLeafElement TrigResult.Accepted  Accepted[TrigResult_] : 0 at: 0x90f9000
OBJ: TLeafI       TrigResult_size      TrigResult_size : 0 at: 0x90fb860
```

The `.ls` command lists the current keys available and in particular the three *tree* names. `TBrowser t` launches a browser and the `GetEntries()` method outputs the number of data in the corresponding *tree*. The list of *branches* or *leaves* can be displayed with the `GetListOfBranches()` and `GetListOfLeaves()` methods, pointing to the `ls()` one. In particular, it is possible to shown only parts of the output, using wildcard characters (`*`):

```
root [6] Analysis->GetListOfLeaves()->ls("*.E")
OBJ: TLeafElement       Jet.E           E[Jet_] : 0 at: 0xa08bc68
OBJ: TLeafElement       TauJet.E        E[TauJet_] : 0 at: 0xa148910
OBJ: TLeafElement       Electron.E      E[Electron_] : 0 at: 0xa1d8a50
OBJ: TLeafElement       Muon.E          E[Muon_] : 0 at: 0xa28ac80
OBJ: TLeafElement       Photon.E        E[Photon_] : 0 at: 0xa33cd88
OBJ: TLeafElement       Tracks.E        E[Tracks_] : 0 at: 0xa3cced0
```

```
OBJ: TLeafElement       CaloTower.E     E[CaloTower_] : 0 at: 0xa4ba188
OBJ: TLeafElement       ZDChits.E       E[ZDChits_] : 0 at: 0xa54a3c8
OBJ: TLeafElement       RP220hits.E     E[RP220hits_] : 0 at: 0xa61e648
OBJ: TLeafElement       FP420hits.E     E[FP420hits_] : 0 at: 0xa6d0920
```

To draw a particular leaf, either double-click on the corresponding name in the `TBrowser` or use the `Draw` method of the corresponding *tree*.

```
root [7] Trigger->Draw("TrigResult.Accepted");
```

Mathematical operations on several *leaves* are possible within a given *tree*, following the C++ syntax:

```
root [8] Analysis->Draw("Muon.Px * Muon.Px");
root [9] Analysis->Draw("sqrt(pow(Muon.E,2) -  pow(Muon.Pz,2) + pow(Muon.PT,2))");
```

Finally, to prepare an deeper analysis, the `MakeClass` method is useful. It creates two files (`*.h` and `*.C`) with automatically generated code that allows the access to all branches and leaves of the corresponding tree:

```
root [10] Trigger->MakeClass()
Info in <TTreePlayer::MakeClass>: Files: Trigger.h and
        Trigger.C generated from TTree: Trigger
```

For more information, refer to ROOT documentation. Moreover, an example of code (based on the output of `MakeClass`) is provided in the `Examples/` directory.

To run the `Examples/Analysis_Ex.cpp` code, the two following arguments are required: a text file containing the input *Delphes* `root` files to run, and the name of the output `root` file.

```
me@mylaptop:~$ ./Analysis_Ex input_file.list output_file.root
```

One can easily edit, modify and compile (`make`) changes in this file.

A.5.1. Adding the trigger information

The `Examples/Trigger_Only.cpp` code permits to run the trigger selection separately from the general detector simulation on output *Delphes* root files. A *Delphes* `root` file is mandatory as an input argument for the `Trigger_Only` routine. The new *tree* containing the trigger result data will be appended to this file. The trigger datacard is also necessary. To run the code:

```
me@mylaptop:~$ ./Trigger_Only input_file.root data/TriggerCard.dat
```

A.6. Running the FROG event display

- If the `FLAG_FROG` was switched on in the smearing card, two files have been created during the running of *Delphes*: `DelphesToFROG.vis` and `DelphesToFROG.geom` . They contain all the needed pieces of information to run FROG.
- To display the events and the geometry, you first need to compile FROG. Go to the `Utilities/FROG` and type `make`. This compilation is done once for all, with this geometry (i.e. as long as the `*vis` and `*geom` files do not change).
- Go back into the main directory and type

  ```
  me@mylaptop: $ ./Utilities/FROG/FROG
  ```

A.7. LHCO file format

The `*LHCO` file format is a text-ASCII data format briefly discussed here. An exhaustive description is provided on http://v1.jthaler.net/olympicswiki. This section is based on this webpage. Only final high-level objects are available in the `LHCO` format, and their properties are arranged in columns. Each row corresponds to an object in the event and all events are written after each other. Comment-lines starts with a hash # symbol.

```
#  typ   eta     phi     pt     jmas    ntrk    btag   had/em    dum1   dum2
0          57     0
1   0   1.392  -2.269  19.981   0.000   0.000   0.000   4.605   0.000  0.000
2   3   1.052   2.599  29.796   3.698  -1.000   0.000   0.320   0.000  0.000
3   4   1.542  -2.070  84.308  41.761   7.000   0.000   1.000   0.000  0.000
4   4   1.039   0.856  58.992  34.941   1.000   0.000   1.118   0.000  0.000
5   4   1.052   2.599  29.796   3.698   0.000   0.000   0.320   0.000  0.000
6   4   0.431  -2.190  22.631   3.861   0.000   0.000   1.000   0.000  0.000
7   6   0.000   0.845  62.574   0.000   0.000   0.000   0.000   0.000  0.000
```

Each row in an event starts with a unique number (i.e. in first column). Row `0` contains the event number (here: `57`) and some trigger information (here: `0`. This very particular trigger encoding is not implemented in *Delphes*.). Subsequent rows list the reconstructed high-level objects. Each row is organised in columns, which details the object kinematics as well as more specific information, such as isolation criteria or *b*-tagging.

1st column (`#`). The first column is the line number in the event. Each event starts with a 0 and contains as many lines as needed to list all high-level objects.

2nd column (`typ`). The second column gives the object identification code, or *type*. The different object types are:

- `0` for a photon (γ)
- `1` for an electron ($e^{\pm}$)
- `2` for a muon ($\mu^{\pm}$)
- `3` for a hadronically-decaying tau (τ-jet)
- `4` for a jet
- `6` for a missing transverse energy (E_T^{miss})

Object type `5` is not defined. An event always ends with the row corresponding to the missing transverse energy (type `6`).

3rd (`eta`) and 4th (`phi`) columns. The third and forth columns gives the object pseudorapidity η and azimuth ϕ. This latter quantity is expressed in radians, ranging from $-\pi$ to π.

5th (`pt`) and 6th (`jmass`) columns. The fifth column provides the object transverse momentum (p_T in GeV/c) or energy (E_T in GeV), while the invariant mass (M in GeV/c^2) is in the sixth column.

7th column (`ntrk`). The seventh column reports the total number of tracks associated to the objects. This is `0` for photons, `± 1` for charged leptons including taus (where the sign reports the lepton measured charge) and a positive number (`≥ 0`) for jets.

8th column (`btag`). The eighth column tells whether a jet is tagged as a *b*-jet (`1`) or not (`0`). This is always `0` for electrons, photons and missing transverse energy. For muons, the closest jet in searched for, in terms of ΔR. The integer-part of the quoted number is the row-number (column 1) of this jet.

9th column (`had/em`). For jets, electrons and photons, the ninth column is the ration between hadronic and electromagnetic energies in the calorimetric cells associated to the object. This is always `0` for missing transverse energy. For muons, this number (`aaa.bb`) reports two values related to the muon isolation (section 3.1). The integer part (`aaa`) is transverse momentum sum P_T (in GeV/c) and the fractional part (`bb`) is the energy ratio ρ_μ.

10th and 11th columns (`dum1` and `dum2`). The last two columns are currently not used.

Warning. Inherently to the data format itself, the `*LHCO` output contains only a fraction of the available data. Moreover, dealing with text file may have various drawbacks, such as the output file size and the time needed for its creation. Whenever possible, working on the `*root` output file should be preferred.