Detector simulation for LHC analyses recasting

Eric CONTE

The first MadAnalysis 5 workshop on LHC recasting @ Korea
20-27 August 2017
1. Introduction

2. General concepts on Delphes

3. CMS detector simulation with Delphes

4. Including pile-up effects

5. Validation & limitations of Delphes

6. Beyond Delphes: the tunes
1. Introduction
Monte-Carlo chain for recasting

One scenario of a given theoretical model

1. ME generator
2. Shower program
3. «Detector simulator»
4. Detector answer
Monte-Carlo chain for recasting

One scenario of a given theoretical model

- ME generator
- Shower program
- « Detector simulator »
- Detector answer

- Particle-Matter interaction
- Digitization
- Reconstruction
2 big categories of tools

realism

Full simulation:
- Particle-matter interactions are described by GEANT4
- Reconstruction algorithms

speed

Parametric simulation:
Functions between particles and reconstructed objects
2 big categories

realism

Full simulation:
• Particle-matter interactions are described by GEANT4
• Reconstruction algorithms

• ATLAS/CMS full simulation [private]

• ATLAS/CMS fast simulation [private]

Parametric simulation:
Functions between particles and reconstructed objects

• Delphes [public]

• Rivet, Gambit [soon]
Which tool for recasting?

List to Santa Klaus:

I would like a tool which is:

• Generic for handling the CMS & ATLAS detectors, for different kind of analyses
• Public
• Very fast: allowing to scan on huge parameter space (SUSY-like models)
• But realistic enough
• User-friendly
• Validated & has proved itself
Which tool for recasting?

List to Santa Klaus:

I would like a tool which is:

- Generic for handling the CMS & ATLAS detectors, for different kind of analyses
- Public
- Very fast: allowing to scan on huge parameter space (SUSY-like models)
- But realistic enough
- User-friendly
- Validated & has proved itself

Delphes!!!!!
2. General concepts of Delphes
What is Delphes?

• **DELPHES** is a very-fast-simulation for generic detector:
  - ATLAS & CMS detectors
  - Upgrade of ATLAS & CMS
  - LHCb
  - Future detectors: FCC

• Output in ROOT format

A question of linguistics:

What does ”Delphes” mean?

- JHEP 02 (2014) 057
Modular architecture

The detector simulation is split into modules.

→ Each module is devoted to a function.
Modular architecture

List of modules
Detector simulation is totally described by a card (text file in \texttt{tcl} language).

This card contains:
- the sequence of the modules that you need
- how they interact between themselves.

\begin{verbatim}
set ExecutionPath {
    ParticlePropagator
    ChargedHadronTrackingEfficiency
    ElectronTrackingEfficiency
    MuonTrackingEfficiency
    ChargedHadronMomentumSmearing
    ElectronMomentumSmearing
    MuonMomentumSmearing
    TrackMerger
    ECAL
    HCAL
    Calorimeter
    EventFlowMerger
    EventFlowFilter
    PhotonEfficiency
    PhotonIsolation
    ElectronFilter
    ElectronEfficiency
    ElectronIsolation
    ChargedHadronFilter
    MuonEfficiency
    MuonIsolation
    MissingET
    NeutrinoFilter
    GenJetFinder
    GenMissingET
    FastJetFinder
}
\end{verbatim}

Order of execution of the modules used in the simulation
Detector simulation is totally described by a card (text file in \texttt{tcl} language).

**This card contains:**
- the sequence of the modules that you need
- how they interact between themselves.

```tcl
module FastJetFinder FastJetFinder
{
    set InputArray EFlowMerger/eflow
    set OutputArray jets
    set JetAlgorithm 6
    set ParameterR 0.5
    set JetPTMin 20.0
}
```
Other informations

Requirements:

<table>
<thead>
<tr>
<th>Package</th>
<th>Utility</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT 6</td>
<td>Main framework &amp; data format</td>
<td>To be installed</td>
</tr>
<tr>
<td>TCL</td>
<td>Language of detector card</td>
<td>To be installed</td>
</tr>
<tr>
<td>FastJet</td>
<td>Jet-clustering algorithm, pile-up</td>
<td>Encapsulated in the delphes package</td>
</tr>
</tbody>
</table>

Extra programs:

- **EVE** (former FROG): program of event visualisation
- **DelphesAnalysis**: reading Delphes ROOT file with Python

3. CMS detector simulation with Delphes
Dataflow diagram

Official dataflow diagram:

Dataflow diagram

Official dataflow diagram:


Obsolete with Delphes 3.4.1
Process Part 1: tracking

Stable particles

ParticlePropagator

Muon

Efficiency

MomentumSmearing

Stable particles

Electron

Efficiency

EnergySmearing

Muon

Efficiency

Charged Hadrons

MomentumSmearing

Track
Process Part 1: tracking

In the real life:

In Delphes, there is no real tracking or vertexing.
Process Part 1: tracking

In the real life:

In Delphes, there is no real tracking or vertexing.
Process Part 1: tracking

→ Charged particles are propagated in the magnetic field until they reach calorimeters. We apply the following movement equations in a cylinder:

\[
\frac{d\hat{p}}{dt} = q(\hat{\nu} \times \vec{B})
\]

→ The result of this computation is to compute the position @ first calorimeter layer.
Process Part 1: tracking

ParticlePropagator

module ParticlePropagator ParticlePropagator
{
    # input
    set InputArray Delphes/stableParticles

    # outputs
    set OutputArray stableParticles
    set ChargedHadronOutputArray chargedHadrons
    set ElectronOutputArray electrons
    set MuonOutputArray muons

    # radius of the magnetic field coverage, in m
    set Radius 1.29
    # half-length of the magnetic field coverage, in m
    set HalfLength 3.00

    # magnetic field
    set Bz 3.8
}

PS: for ATLAS

set Radius 1.15
set HalfLength 3.51
set Bz 2.0

Definition of the cylindrical volume of the tracker.

Magnitude of the magnetic field
Process Part 1: tracking

Efficiency

```cpp
module Efficiency MuonTrackingEfficiency {
  set InputArray ParticlePropagator/muons
  set OutputArray muons

  # tracking efficiency formula for muons
  set EfficiencyFormula {
    (abs(eta) <= 1.5) * (pt > 0.1) && pt <= 1.0) * (0.00) +
    (abs(eta) <= 1.5) * (pt > 1.0) * (0.99) +
    (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1) && pt <= 1.0) * (0.70) +
    (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0) * (0.98) +
    (abs(eta) > 2.5) * (0.00)
  }
}
```

MomentumSmearing

```cpp
module MomentumSmearing MuonMomentumSmearing {
  set InputArray MuonTrackingEfficiency/muons
  set OutputArray muons

  # set ResolutionFormula {resolution formula as a function of eta and pt}
  # resolution formula for muons
  set ResolutionFormula {
    (abs(eta) <= 0.5) * (pt > 0.1) * sqrt(0.01^2 + pt^2*1.0e-4^2) +
    (abs(eta) > 0.5 && abs(eta) <= 1.5) * (pt > 0.1) * sqrt(0.015^2 + pt^2*1.5e-4^2) +
    (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1) * sqrt(0.025^2 + pt^2*3.5e-4^2)}
}
```

Resolution on pT
Process Part 2: calorimetry

Stable particles → ECAL

Track → eflowTracks → HCAL

ECAL:
- ecalTower
- hcalTower

HCAL:
- eflowTracks
- ChargedHadronFilter
- ElectronFilter

Charged Hadrons → Electrons

Neutral Hadrons → Eflow Neutral Hadrons

Towers: eflowPhoton
Process Part 2: calorimetry

ECAL   \hspace{1cm} HCAL

= inheritance from the same module: Calorimeter

1) Segmentation of the calorimeter into cells
Process Part 2: calorimetry

ECAL

HCAL

= inheritance from the same module: Calorimeter

2) Energy fraction absorbed by the calorimeter

**ECAL case**

```plaintext
add EnergyFraction {0} {0.0}
# energy fractions for e, gamma and p10
add EnergyFraction {11} {1.0}
add EnergyFraction {22} {1.0}
add EnergyFraction {111} {1.0}
# energy fractions for muon, neutrinos and neutralinos
add EnergyFraction {12} {0.0}
add EnergyFraction {13} {0.0}
add EnergyFraction {14} {0.0}
add EnergyFraction {16} {0.0}
add EnergyFraction {100022} {0.0}
add EnergyFraction {100023} {0.0}
add EnergyFraction {100025} {0.0}
add EnergyFraction {100035} {0.0}
add EnergyFraction {100045} {0.0}
# energy fractions for K0 short and Lambda
add EnergyFraction {310} {0.3}
add EnergyFraction {3122} {0.3}
```

**HCAL case**

```plaintext
# default energy fractions {abs(PDG code)} {Fecal Fical}
add EnergyFraction {0} {1.0}
# energy fractions for e, gamma and p10
add EnergyFraction {11} {0.0}
add EnergyFraction {22} {0.0}
add EnergyFraction {111} {0.0}
# energy fractions for muon, neutrinos and neutralinos
add EnergyFraction {12} {0.0}
add EnergyFraction {13} {0.0}
add EnergyFraction {14} {0.0}
add EnergyFraction {16} {0.0}
add EnergyFraction {100022} {0.0}
add EnergyFraction {100023} {0.0}
add EnergyFraction {100025} {0.0}
add EnergyFraction {100035} {0.0}
add EnergyFraction {100045} {0.0}
# energy fractions for K0 short and Lambda
add EnergyFraction {310} {0.7}
add EnergyFraction {3122} {0.7}
```

**WARNING:** if you had exotic particle in your sample, declare its EnergyFractions.
Process Part 2: calorimetry

ECAL

HCAL

= inheritance from the same module: Calorimeter

3) Smearing of cell energy

\[
\left( \frac{\sigma}{E} \right)^2 = \left( \frac{S(\eta)}{\sqrt{E}} \right)^2 + \left( \frac{N(\eta)}{E} \right)^2 + C(\eta)^2
\]

ECAL case

```c
set ResolutionFormula {
    (abs(eta) <= 1.5) * (1-0.6*eta^2) * sqrt(energy^2*0.008^2 + energy*0.11^2 + 0.60^2) +
    (abs(eta) > 1.5 && abs(eta) <= 2.5) * (2.16 + 5.6*(abs(eta)-2)^2) * sqrt(energy^2*0.008^2 + energy*0.11^2 + 0.40^2) +
    (abs(eta) > 2.5 && abs(eta) <= 5.0) * sqrt(energy^2*0.107^2 + energy*2.08^2))
}
```


HCAL case

```c
# set HCalResolutionFormula {resolution formula as a function of eta and energy}
set ResolutionFormula {
    (abs(eta) <= 3.0) * sqrt(energy^2*0.050^2 + energy*1.50^2) +
    (abs(eta) > 3.0 && abs(eta) <= 5.0) * sqrt(energy^2*0.130^2 + energy*2.70^2))
}
```

+ Min value on energy cell
Process Part 2: calorimetry

= inheritance from the same module: Calorimeter

4) Two kinds of output collection

- Calorimeter information: towers 
  `ecalTower, hcalTower`

- Calorimeter + tracker information: particle (or energy) flow

<table>
<thead>
<tr>
<th></th>
<th>With track</th>
<th>Without track</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECAL</td>
<td><code>eflowElectron</code></td>
<td><code>eflowPhoton</code></td>
</tr>
<tr>
<td>HCAL</td>
<td><code>eflowChargedHadron</code></td>
<td><code>eflowNeutralHadron</code></td>
</tr>
</tbody>
</table>

Characteristic of objects are corrected:
- tracking provides good measurement of momenta at low energy
- calorimeter provides good measurement of momenta at high energy
Process Part 3: e, µ, γ

- Electrons
  - Efficiency
  - Isolation

- Muons
  - Efficiency
  - Isolation

- Photons
  - Efficiency
  - Isolation

- Eflow
  - EflowFilter

- Eflow without leptons
Process Part 3: \( e, \mu, \gamma \)

**Efficiency**

Applying efficiency corresponding to identification

```plaintext
module Efficiency MuonEfficiency {
    set InputArray MuonMomentumSmearing/muons
    set OutputArray muons

    # efficiency formula for muons
    set EfficiencyFormula { (pt <= 10.0) * (0.00) +
                            (abs(eta) <= 1.5) * (pt > 10.0) * (0.95) +
                            (abs(eta) > 1.5 && abs(eta) <= 2.4) * (pt > 10.0) * (0.95) +
                            (abs(eta) > 2.4) * (0.00)}
}

module Efficiency ElectronEfficiency {
    set InputArray ElectronFilter/electrons
    set OutputArray electrons

    # efficiency formula for electrons
    set EfficiencyFormula { (pt <= 10.0) * (0.00) +
                            (abs(eta) <= 1.5) * (pt > 10.0) * (0.95) +
                            (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 10.0) * (0.85) +
                            (abs(eta) > 2.5) * (0.00)}
}

module Efficiency PhotonEfficiency {
    set InputArray ECal/eflowPhotons
    set OutputArray photons

    # efficiency formula for photons
    set EfficiencyFormula { (pt <= 10.0) * (0.00) +
                            (abs(eta) <= 1.5) * (pt > 10.0) * (0.95) +
                            (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 10.0) * (0.85) +
                            (abs(eta) > 2.5) * (0.00)}}
```
Process Part 3: e, μ, γ

Isolation

Applying an isolation criterion to leptons & photons wrt jets

Δ\( R_{max} \)

2 methods

UsePTSum=1
Scalar sum of track PT < threshold

UsePTSum=0 [default]
Scalar sum of track PT / lepton PT < threshold
Process Part 3: $e, \mu, \gamma$

Isolation

Applying an isolation criterion to leptons & photons wrt jets

```
module Isolation_MuonIsolation {
    set CandidateInputArray MuonEfficiency/muons
    set IsolationInputArray EFlowFilter/eflow
    set OutputArray muons
    set DeltaRMax 0.5
    set PTMin 0.5
    set PTRatioMax 0.25
}
```

- **Size of the isolation cone**
- **Remove muons with low PT**
- **Threshold on the ratio**
Process Part 4: jets

Stable particles

NeutrinoFilter

GenJetFinder

GenJets

FastJetFinder

JetEnergyScale

JetFlavourAssociation

b-tagging

Tau-tagging

Jets

eflow

FatJetFinder

FatJets
Process Part 4: jets

Stable particles

- NeutrinoFilter
  - Remove invisible particle from the stable particles.
- GenJetFinder
  - Apply a jet clustering algorithm
    - 1 CDFJetClu
    - 2 MidPoint
    - 3 SIScone
    - 4 kt
    - 5 Cambridge/Aachen
    - 6 antikt
  - And remove low-PT jets
- GenJets
  - Reconstructed jets with a perfect calorimeter
Process Part 4: jets

Stable particles

\[ \text{NeutrinoFilter} \]

\[ \text{GenJetFinder} \]

\[ \text{GenJets} \]

module PdgCodeFilter NeutrinoFilter {
  set InputArray Delphes/stableParticles
  set OutputArray filteredParticles
  add PdgCode {12}
  add PdgCode {14}
  add PdgCode {16}
  add PdgCode {-12}
  add PdgCode {-14}
  add PdgCode {-16}
}

module FastJetFinder GenJetFinder {
  set InputArray NeutrinoFilter/filteredParticles
  set OutputArray jets
  set JetAlgorithm 6
  set ParameterR 0.5
  set JetPTMin 20.0
}
Process Part 4: jets

- **eflow**
  - **FastJetFinder**
  - **JetEnergyScale**
  - **JetFlavourAssociation**
  - **b-tagging**
  - **Tau-tagging**

**Apply a jet —clustering algorithm**
- 1 CDFJetClu
- 2 MidPoint
- 3 SIScone
- 4 kt
- 5 Cambridge/Aachen
- 6 antikt
And remove low-PT jets

**Apply correction to jet energy**

**Match jets to partons and determine the « true »**

**B-tagging id and mis-id**

**tau-tagging id and mis-id**

**Jets**
Process Part 4: jets

eflow

FastJetFinder

JetEnergyScale

JetFlavourAssociation

b-tagging

Tau-tagging

Jets

arXiv:1211.4462
Process Part 4: jets

Collection devoted to boosted objects

tops in a single jet

ParameterR is bigger than the normal one. Therefore these «fat» jets has a substructure.
Process Part 4: jets

```
module FastJetFinder FatJetFinder {
    set InputArray EFlowMerger/eflow
    set OutputArray jets
    set JetAlgorithm 6
    set ParameterR 0.8
    set ComputeNsubjettiness 1
    set Beta 1.0
    set AxisMode 4
    set ComputeTrimming 1
    set RTrim 0.2
    set PtFracTrim 0.05
    set ComputePruning 1
    set ZcutPrun 0.1
    set RcutPrun 0.5
    set RPrun 0.8
    set ComputeSoftDrop 1
    set BetaSoftDrop 0.0
    set SymmetryCutSoftDrop 0.1
    set ROSoftDrop 0.8
    set JetPTMin 200.0
}
```

- Probing jet substructure with N-subjettiness algo with N=1,2,3,4,5
- Trimming algo
- Pruning algo
- SoftDrop algo
Process Part 5: output

Towers, Tracks, eflowPhotons, eflowTracks, eflowNeutralHadrons, GenJet, FatJet, MissingEt, ScalarHT, GenParticle

TreeWriter

UniqueObjectFinder

electrons  muons  photons  jets
Jet collection can contain photons, electrons & muons.

→ Cleaning collections by removing redundancies.
Process Part 5: output

TreeWriter

```plaintext
module TreeWriter TreeWriter
{
    add Branch Delphes/allParticles Particle GenParticle
    add Branch TrackMerger/tracks Track Track
    add Branch Calorimeter/towers Tower Tower
    add Branch HCal/eflowTracks EFlowTrack Track
    add Branch ECal/eflowPhotons EFlowPhoton Tower
    add Branch HCal/eflowNeutralHadrons EFlowNeutralHadron Tower
    add Branch GenJetFinder/jets GenJet Jet
    add Branch GenMissingET/momentum GenMissingET MissingET
    add Branch UniqueObjectFinder/jets Jet Jet
    add Branch UniqueObjectFinder/electrons Electron Electron
    add Branch UniqueObjectFinder/photons Photon Photon
    add Branch UniqueObjectFinder/muons Muon Muon
    add Branch FatJetFinder/jets FatJet Jet
    add Branch MissingET/momentum MissingET MissingET
    add Branch ScalarHT/energy ScalarHT ScalarHT
}
```

List of all (temporary or final) collection of objects saved in the ROOT files
## Process Part 5: output

### List of Arrays

<table>
<thead>
<tr>
<th>Array name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delphes/allParticles</td>
<td>All generated particles</td>
</tr>
<tr>
<td>Delphes/stableParticles</td>
<td>Final state particles (Status==1)</td>
</tr>
<tr>
<td>Delphes/partons</td>
<td>Decayed particles or partons produced in shower (Status==2)</td>
</tr>
<tr>
<td>ParticlePropagator/stableParticles</td>
<td>All propagated particles</td>
</tr>
<tr>
<td>ParticlePropagator/chargedHadrons</td>
<td>Propagated charged hadrons</td>
</tr>
<tr>
<td>ParticlePropagator/electrons</td>
<td>Propagated electrons</td>
</tr>
<tr>
<td>ParticlePropagator/photons</td>
<td>Propagated photons</td>
</tr>
<tr>
<td>ChargedHadronTrackingEfficiency/chargedHadrons</td>
<td>Propagated charged hadrons that pass the efficiency selection</td>
</tr>
<tr>
<td>ChargedHadronMomentumSmearing/chargedHadrons</td>
<td>Tracks with momentum smeared according to the charged hadrons momentum resolution</td>
</tr>
<tr>
<td>ElectronTrackingEfficiency/electrons</td>
<td>Propagated electrons that pass the efficiency selection</td>
</tr>
<tr>
<td>ElectronEnergySmearing/electrons</td>
<td>Tracks with energy smeared according to the electron energy resolution</td>
</tr>
<tr>
<td>MuonTrackingEfficiency/photons</td>
<td>Propagated muons that pass the efficiency selection</td>
</tr>
<tr>
<td>MuonMomentumSmearing/photons</td>
<td>Tracks with momentum smeared according to the muon momentum resolution</td>
</tr>
<tr>
<td>TrackMerger/tracks</td>
<td>Combination of charged hadrons and electrons</td>
</tr>
<tr>
<td>Calorimeter/towers</td>
<td>All calorimeter towers</td>
</tr>
<tr>
<td>Calorimeter/photons</td>
<td>Calorimeter towers associated with the photons</td>
</tr>
<tr>
<td>Calorimeter/eFlowTracks</td>
<td>Tracks output from the energy flow algorithm</td>
</tr>
<tr>
<td>Calorimeter/eFlowPhotons</td>
<td>Photon output from the energy flow algorithm</td>
</tr>
<tr>
<td>Calorimeter/eFlowNeutralHadrons</td>
<td>Neutral hadron output from the energy flow algorithm</td>
</tr>
<tr>
<td>EFlowMerger/eFlow</td>
<td>Combination of tracks, calorimeter towers and muons required for jet finding</td>
</tr>
<tr>
<td>ElectronEfficiency/electrons</td>
<td>Electrons that pass the efficiency selection</td>
</tr>
<tr>
<td>ElectronIsolation/electrons</td>
<td>Isolated electrons</td>
</tr>
<tr>
<td>PhotonEfficiency/photons</td>
<td>Photons that pass the efficiency selection</td>
</tr>
<tr>
<td>PhotonIsolation/photons</td>
<td>Isolated photons</td>
</tr>
<tr>
<td>MuonEfficiency/photons</td>
<td>Muons that pass the efficiency selection</td>
</tr>
<tr>
<td>MuonIsolation/photons</td>
<td>Isolated muons</td>
</tr>
<tr>
<td>FastJetFinder/jets</td>
<td>Reconstructed jets</td>
</tr>
<tr>
<td>MissingET/momentum</td>
<td>Missing transverse energy</td>
</tr>
<tr>
<td>ScalarHT/energy</td>
<td>Scalar sum of transverse momenta and energy of all reconstructed objects</td>
</tr>
<tr>
<td>UniqueObjectFinder/photons</td>
<td>Uniquely identified photons</td>
</tr>
<tr>
<td>UniqueObjectFinder/electrons</td>
<td>Uniquely identified electrons</td>
</tr>
<tr>
<td>UniqueObjectFinder/jets</td>
<td>Uniquely identified jets</td>
</tr>
</tbody>
</table>

4. Including pile-up effects
What is pile-up?

The phenomenon

Several interactions per bunch crossing

The consequences on the reconstruction:

- reduced efficiency
- worsened resolution (jets, MET)
- degraded isolation
- fake tracks, jets

2 ways to simulate this effect:

- Full simulation:
  - Injecting several interactions
  - Pileup subtraction @ reco
- Smearing & efficiency functions
  \(\rightarrow\) no fake & to be tuned @ hand
Process Part 1: tracking

Stable particles

PileUpMerger

Stable particles

ParticlePropagator

Muon

Efficiency

MomentumSmearing

Vertices

Stable particles

Muon

Track

Electron

Efficiency

EnergySmearing

Charged Hadrons

Efficiency

MomentumSmearing
Process Part 1: tracking

PileUpMerger

Superimposing to your signal event several MinBias events

```java
module PileUpMerger PileUpMerger
{
    set InputArray Delphes/stableParticles
    set ParticleOutputArray stableParticles
    set VertexOutputArray vertices

    # pre-generated minbias input file
    set PileUpFile MinBias.pileup

    # average expected pile up
    set MeanPileUp 50

    # maximum spread in the beam direction in m
    set 2VertexSpread 0.25

    # maximum spread in time in s
    set TVertexSpread 800E-12

    # vertex smearing formula f(z,t) (z,t need to be respectively given in m,s)
    set VertexDistributionFormula {exp(-t^2/160e-12^2/2) * exp(-(z^2/0.053^2/2))}
}
```

MinBias samples. Events are chosen randomly by Delphes.

A file « MinBias.pileup » with 1,000 events is provided by Delphes.

Average number of interactions (superimposing $N$ interactions according to a Poisson law).

Piece of advice:

$N_{\text{minbias}} > \text{average}$
Process Part 1: tracking

PileUpMerger

Superimposing to your signal event several MinBias events

```plaintext
module PileUpMerger PileUpMerger
{
    set InputArray Delphes/stableParticles
    set ParticleOutputArray stableParticles
    set VertexOutputArray vertices

    # pre-generated minbias input file
    set FileUpFile MinBias.pileup

    # average expected pile up
    set MeanPileUp 50

    # maximum spread in the beam direction in m
    set ZVertexSpread 0.25

    # maximum spread in time in s
    set TVertexSpread 800E-12

    # vertex smearing formula f(z,t) (z,t need to be respectively given in m,s)
    set VertexDistributionFormula {exp(-(t^2/160e-12^2/2))*exp(-(z^2/0.053^2/2))}
}
```
Process Part 1: tracking

PileUpMerger

Superimposing to your signal event several MinBias events

```java
module PileUpMerger
{
    set InputArray Delphes/stableParticles
    set ParticleOutputArray stableParticles
    set VertexOutputArray vertices

    # pre-generated minbias input file
    set PileUpFile MinBias.pileup

    # average expected pile up
    set MeanPileUp 50

    # maximum spread in the beam direction in m
    set ZVertexSpread 0.25

    # maximum spread in time in s
    set TVertexSpread 800E-12

    # vertex smearing formula f(z,t) (z,t need to be respectively given in m,s)
    set VertexDistributionFormula {exp(-(t^2/160e-12^2/2))*exp(-(z^2/0.053^2/2))}
}
```

Add new particles to the collection of stable particles
Process Part 3: jets

- eflow
  - Rho
    - FastJetFinder
      - PileUpJetID
        - JetPileUpSubtractor
          - JetEnergyScale
            - JetFlavourAssociation
              - b-tagging
                - Tau-tagging
                  - Jets
Use the **GridMedianBackgroundEstimator** approach [FastJet package] for computing:

- $\rho = \text{median density of pile-up contamination (for each eta and phi region)}$
Change: compute area for each jet during the jet-clustering algorithm
[arXiv:0707.1378], [arXiv:0802.1188]
Process Part 3: jets

PileUpJetID

Compute jet id variables

module PileUpJetID PileUpJetID {
    set JetInputArray FastJetFinder/jets
    set TrackInputArray HCal/eflowTracks
    set NeutralInputArray NeutralTowerMerger/towers

    set VertexInputArray PileUpMerger/vertices
    # assume perfect pile-up subtraction for tracks with |z| > fZVertexResolution
    # Z vertex resolution in m
    set ZVertexResolution 0.0001

    set OutputArray jets

    set Use Constituents 0
    set ParameterR 0.5

    set JetPTMin 20.0

    NCharged
    number of charged constituents

    NNeutrals
    number of neutral constituents

    Beta
    (sum pt of charged pile-up constituents)/(sum pt of charged constituents)

    BetaStar
    (sum pt of charged constituents coming from hard interaction)/(sum pt of charged constituents)

    MeanSqDeltaR
    average distance (squared) between constituent and jet weighted by pt (squared) of constituent.

    PTD
    average pt between constituent and jet weighted by pt of constituent.

    FracPt[i]
    (sum pt of constituent within a ring 0.1"i<DeltaR<0.1"(i+1) )/(sum pt)
Process Part 3: jets

JetPileUpSubtractor

Neutral subtraction of pile-up

```
module JetPileUpSubtractor JetPileUpSubtractor {
    set JetInputArray PileUpJetID/jets
    set RhoInputArray Rho/rho
    set OutputArray jets
    set JetPTMin 20.0
}
```

- Jet correction: \( p_T \rightarrow p_T - \rho \times \text{area.pT} \)
- Remove jets with \( p_T < \text{JetPTMin} \)
Process Part 4: e, μ, γ

- Electrons
  - TrackPileUpSubtractor
  - Efficiency
  - Isolation

- Muons
  - TrackPileUpSubtractor
  - Efficiency
  - Isolation

- Photons
  - TrackPileUpSubtractor
  - Efficiency
  - Isolation

- Eflow
  - EflowFilter
  - Eflow without leptons
Process Part 3: jets

TrackPileUpSubtractor

Charged subtraction of pile-up

```javascript
module TrackPileUpSubtractor TrackPileUpSubtractor {
    add InputArray HCal/eflowTracks       eflowTracks
    add InputArray ElectronFilter/electrons electrons
    add InputArray MuonMomentumSmearing/muons muons
    set VertexInputArray PileUpMerger/vertices
    set ZVertexResolution {0.0001}
}
```

Filtering the tracks:

- For $|z| < Z\text{VertexResolution} \rightarrow$ The hard interaction vertex cannot be distinguished from pile-up vertices.
- For $|z| > Z\text{VertexResolution} \rightarrow$ The hard interaction vertex can be distinguished from pile-up vertices → REMOVE
Isolation of leptons or photons

Without pileup

```plaintext
module Isolation MuonIsolation {
  set CandidateInputArray MuonEfficiency/muons
  set IsolationInputArray EFlowFilter/eflow
  set OutputArray muons
  set DeltaRMax 0.5
  set PTMin 0.5
  set PTRatioMax 0.25
}
```

With pileup

```plaintext
module Isolation MuonIsolation {
  set CandidateInputArray MuonEfficiency/muons
  set IsolationInputArray EFlowFilter/eflow
  set RhoInputArray Rho/rho
  set OutputArray muons
  set DeltaRMax 0.5
  set PTMin 0.5
  set PTRatioMax 0.25
}
```

**Isolation with pile-up**

If UseRhoCorrection is set to true the rho-corrected variable is used for isolation:

```
IsolationVarRhoCorr = sumChargedNoPU + max(sumNeutral - max(rho, 0.0)*\pi*R^2, 0.0)
```

Otherwise the beta variable is used:

```
IsolationVar = sumChargedNoPU + max(sumNeutral - 0.5*sumChargedPU, 0.0)
```
Warning to recasters

- Pile-up simulation ***increases intensively the time processing.***
- **Pile-up simulation requires an extra MC sample**: a MinBias sample. Delphes provides a 1k sample but usually not enough for large production. To generate carefully! More details here: https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook/PileUp #RunningDelpheswithPile-Up1

**Suggestion to recasters:**

implementing & debugging your analysis

FIRST without pile-up
5. Validation & limitations of Delphes
Validation

PT resolution of reconstructed muons
Validation

ParticleFlow validation

Jet energy resolution

MET resolution
Validation

Energy resolution of reconstructed electrons & photons

Validation

Energy resolution of reconstructed electrons & photons

Disagreement below 20 GeV for electrons
The limitations of Delphes

• The **simulation of the trigger** (online selection) is missing. BUT usually the offline selection includes the effects of the trigger.

• There is no tracker simulation:
  - **No reconstructed vertices**
  - No fake tracks
  - No track quality

• No noise in the calorimeters

• **No photon conversion** (but included for LHCb simulation)

• **No fake muons, electrons or photons**

• **Does not convient for exotics topology.**
  Example of long-lived particles:
  - No displaced vertices
  - No displaced tracks
  - No tracks dispearing
6. Beyond Delphes: the tunes
Different tunes of Delphes

• The Delphes development model is community-based.

  People are encouraged to:
  - Tune their detector cards according to their usage
  - Develop their own modules
  - Modify the code if necessary

• Proliferation of tunes of Delphes. Some example:

  • CMS or ATLAS tunes of Delphes [private]

  • Rivet tune of Delphes:
    • Improving ATLAS simulation realism
    • Adding a lot of tags in the data format

  • MadAnalysis 5 tune(s) of the Delphes cards
    • Reducing the size of Delphes ROOT file
    • Isolation @ analysis level
    • Displaced tracks [new]
In order to recast LHC analyses, we need a very-fast & realistic detector simulation.

**Delphes (current release: 3.4.1)** suits very well:
- Generic simulation, in particular ATLAS & CMS can handled.
- Modular architecture & initially community-based.
- Simulation of pile-up effects & subtraction based on the Jet Area method.

It is important in this workshop that people:
- check (and potentially tune) the content of the Delphes card according to their analysis.
- know that Delphes has its limitations.

**Missing points in this talk:**
- How to install Delphes?
- How to run Delphes?
- How to analyze Delphes output?

➔ MadAnalysis 5 will facilitate your life [see you before the lunch]