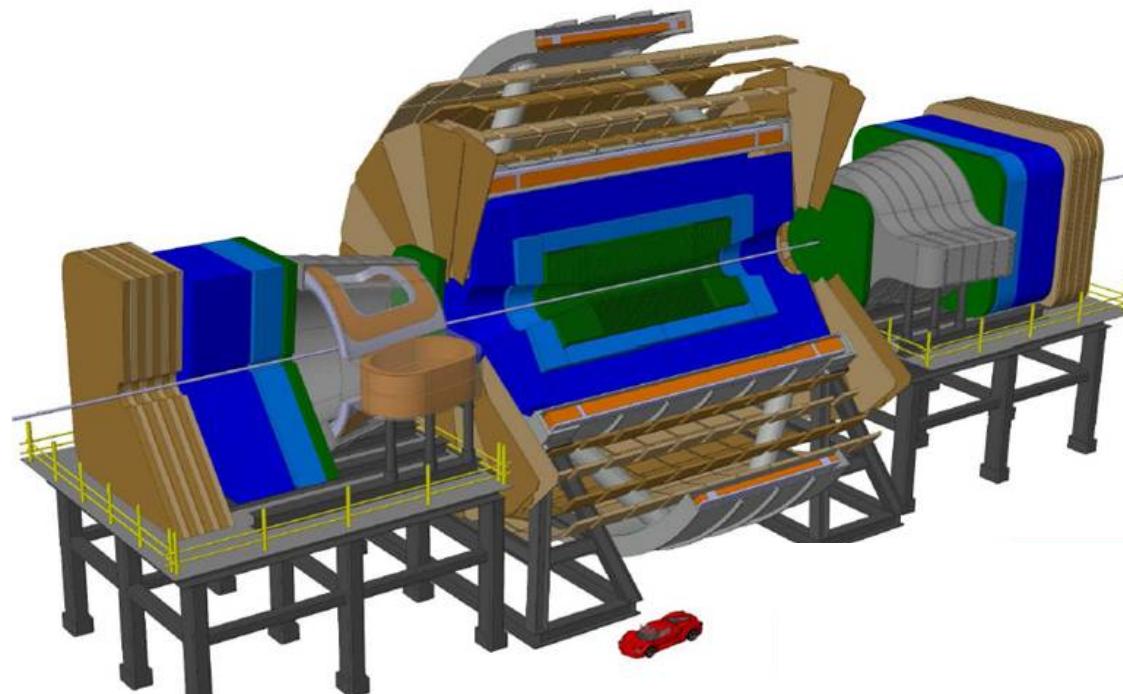


DELPHES description of default FCC detector

Filip Moortgat (CERN)



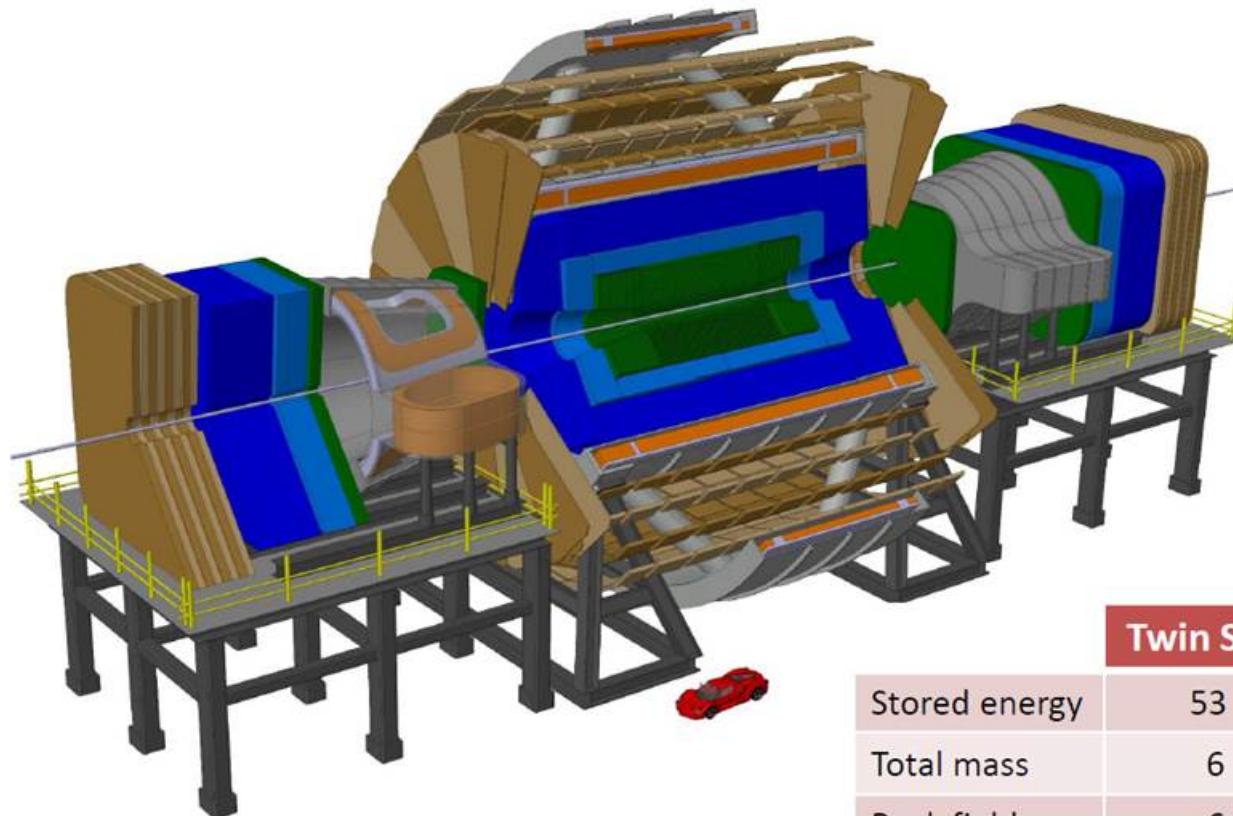
With Heather Gray, Zbynek Drasal, Clement Helsens and Werner Riegler

Introduction



- DELPHES seems the most appropriate simulation package for generic FCC-hh physics studies at the moment.
- FCCSW (FCC software framework) is now interfaced to include DELPHES interface, plus other detector simulation options
- Iterated on a default DELPHES implementation of a generic FCC detector in the FCC-hh Detector meetings (W. Riegler)
- Received useful comments from Daniel, Paolo, Andre, Sergei, Ana, Mike, Thibaud and others. Thanks for the feedback!

Default FCC detector



FCC Air core Twin solenoid and Dipoles

State of the art high stress / low mass design.

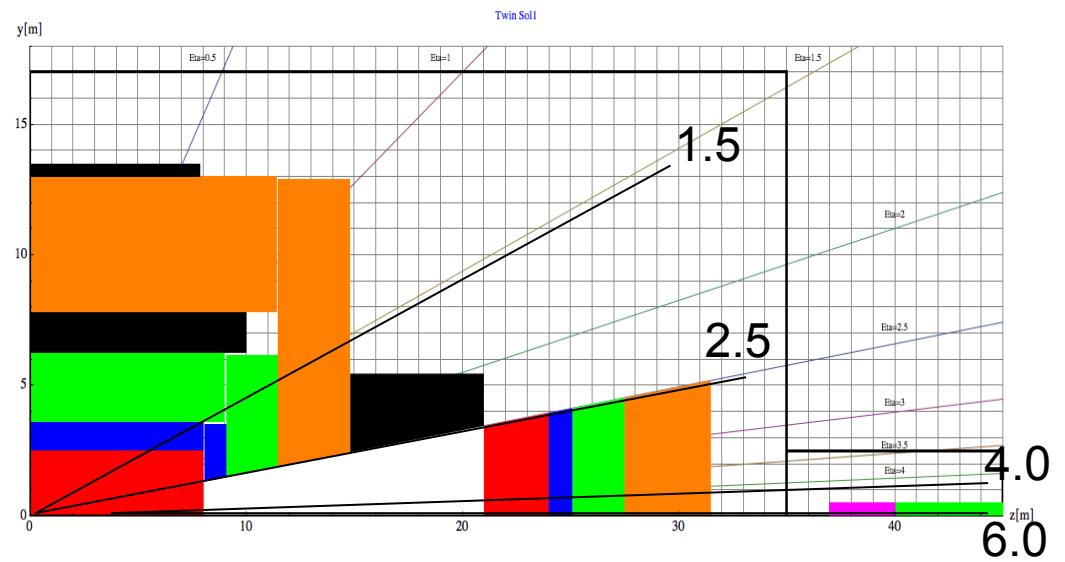
	Twin Solenoid	Dipole
Stored energy	53 GJ	2 x 1.5 GJ
Total mass	6 kt	0.5 kt
Peak field	6.5 T	6.0 T
Current	80 kA	20 kA
Conductor	102 km	2 x 37 km
Bore x Length	12 m x 20 m	6 m x 6 m

Detector geometry



4 regions in eta:

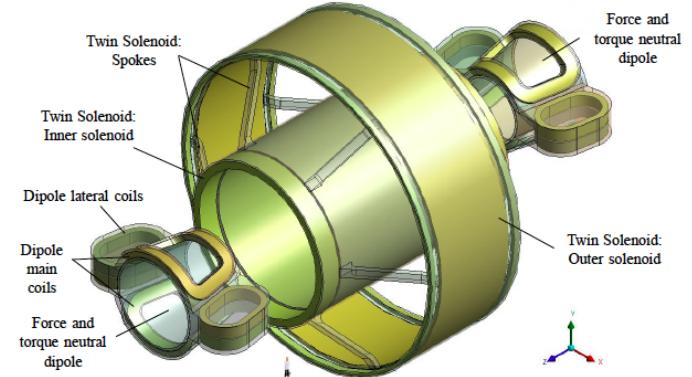
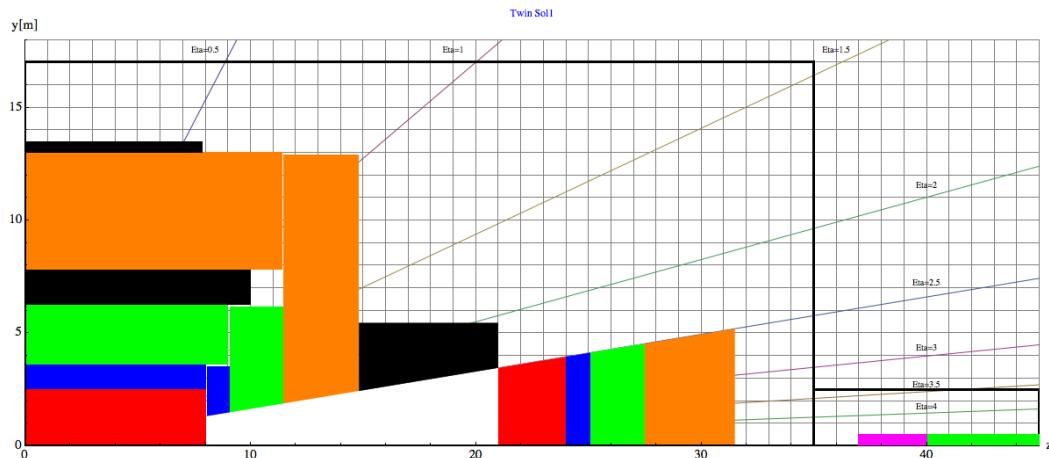
- Barrel: $0 < \eta < 1.5$
- Endcap: $1.5 < \eta < 2.5$
- Forward: $2.5 < \eta < 4.0$
- Very forward: $4.0 < \eta < 6.0$



Magnet choice



First choice: pick Twin Solenoid option as baseline



Delphes card:

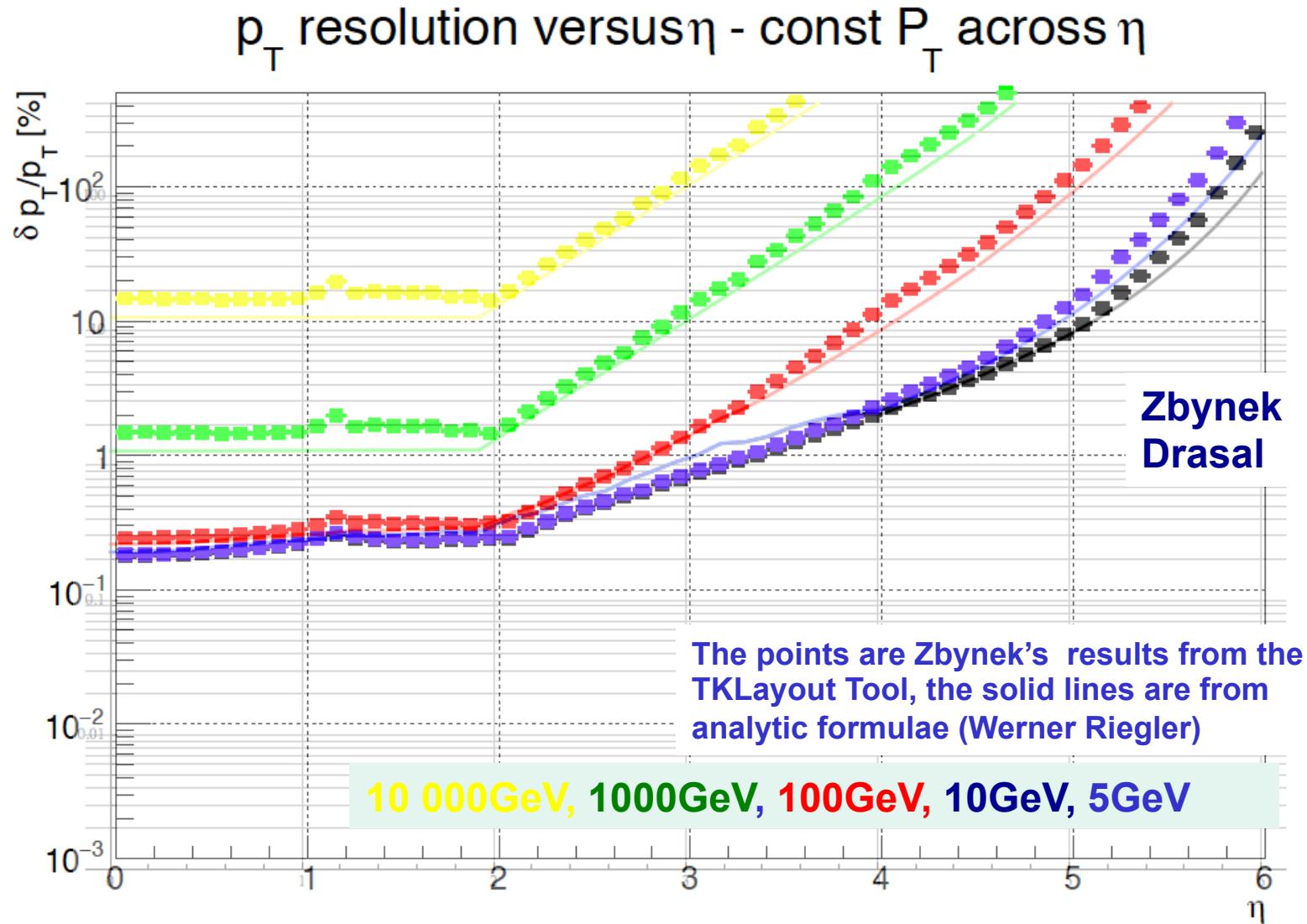
radius of the magnetic field coverage, in m
set Radius 6.0

half-length of the magnetic field coverage, in m
set HalfLength 11.5

magnetic field
set Bz 6.0

Note: dipole field not yet implemented in Delphes.
Will follow later.
If this feature is critical for you, you probably need a more detailed simulation anyway.

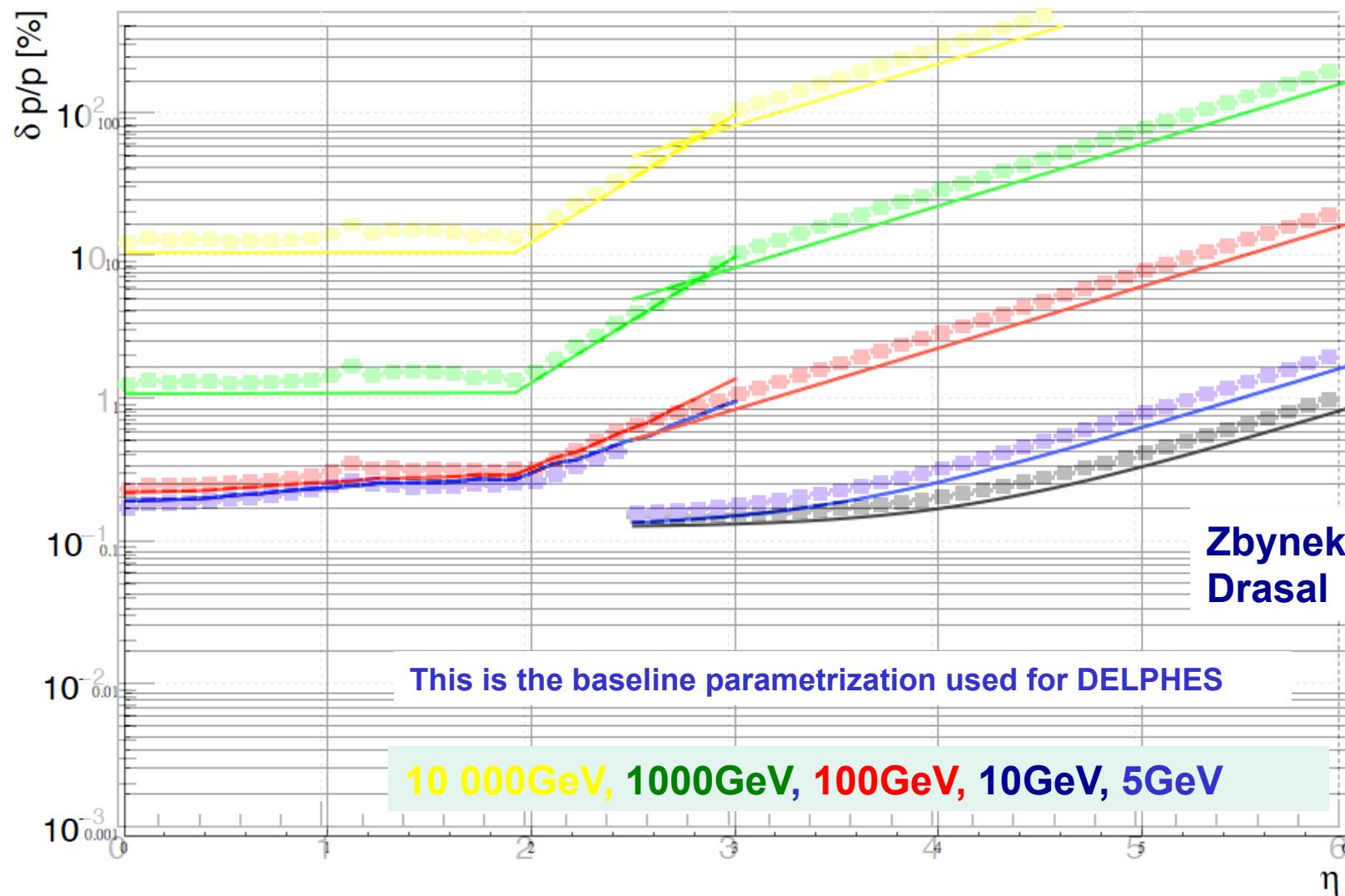
Tracking: solenoid only



Tracking: solenoid + dipole



Total (Dipole+Central) p_T resolution versus η - const P_T across η



Calorimeters



Simple calorimeter (no longitudinal segmentation at the moment):

ECAL: granularity : 0.0125×0.0125 for $\eta < 2.5$,
 0.025×0.025 for $\eta < 4.0$,
 0.05×0.05 for $\eta < 6.0$

```
# set ECalResolutionFormula {resolution formula as a function of eta and energy}
set ResolutionFormula {
    (abs(eta) <= 4.0) * sqrt(energy^2*0.01^2 + energy*0.10^2) +
    (abs(eta) > 4.0 && abs(eta) <= 6.0) * sqrt(energy^2*0.01^2 + energy*0.10^2)}
```

i.e. stochastic term: 10%, constant term: 1%

HCal: granularity : 0.05×0.05 for $\eta < 2.5$,
 0.1×0.1 for $\eta < 4.0$,
 0.2×0.2 for $\eta < 6.0$

```
# set HCalResolutionFormula {resolution formula as a function of eta and energy}
set ResolutionFormula {
    (abs(eta) <= 4.0) * sqrt(energy^2*0.03^2 + energy*0.50^2) +
    (abs(eta) > 4.0 && abs(eta) <= 6.0) * sqrt(energy^2*0.05^2 + energy*1.00^2)}
```

i.e. stochastic term: 50% (100%), constant term: 3% (5%)

Electrons



```
# tracking efficiency formula for electrons  
  
set EfficiencyFormula { (pt <= 10) * (0.00) + \  
  (abs(eta) <= 1.5) * (pt > 10 && pt <= 50) * (0.80) + \  
  (abs(eta) <= 1.5) * (pt > 50) * (0.90) + \  
  (abs(eta) > 1.5 && abs(eta) <= 4) * (pt > 10 && pt <= 50) * (0.80) + \  
  (abs(eta) > 1.5 && abs(eta) <= 4) * (pt > 50) * (0.90) + \  
  (abs(eta) > 4 && abs(eta) <= 6) * (pt > 10 && pt <= 50) * (0.70) + \  
  (abs(eta) > 4 && abs(eta) <= 6) * (pt > 50) * (0.80) + \  
}
```

Identification
efficiency included

70-90% efficiency for
Pt > 10 GeV

Track-based isolation
PTMin 0.5

```
# resolution formula for electrons  
  
set ResolutionFormula {  
  (abs(eta) <= 1.5) * sqrt(energy^2*0.01^2 + energy*0.10^2) + \  
  (abs(eta) > 1.5 && abs(eta) <= 2.5) * sqrt(energy^2*0.01^2 + energy*0.10^2) + \  
  (abs(eta) > 2.5 && abs(eta) <= 4.0) * sqrt(energy^2*0.01^2 + energy*0.10^2) + \  
  (abs(eta) > 4.0 && abs(eta) <= 6.0) * sqrt(energy^2*0.01^2 + energy*0.10^2)}  
}
```

= ECAL resolution



Muons

```
set EfficiencyFormula {      (pt <= 5) * (0.00) + \
                          (abs(eta) <= 1.5) * (pt > 5 ) * (0.99) + \
                          (abs(eta) > 1.5 && abs(eta) <= 4.0) * (pt > 5) * (0.99) + \
                          (abs(eta) > 4.0) * (0.00)}
```

Resolution : same as tracking

99% efficiency for
Pt > 5 GeV and eta < 4.0

Photons

```
set EfficiencyFormula {      (pt <= 10.0) * (0.00) + \
                          (abs(eta) <= 1.5) * (pt > 10.0) * (0.95) + \
                          (abs(eta) > 1.5 && abs(eta) <= 4.0) * (pt > 10.0) * (0.90) + \
                          (abs(eta) > 4.0 && abs(eta) <= 6.0) * (pt > 10.0) * (0.80) + \
                          (abs(eta) > 6.0) * (0.00) }
```

Resolution & isolation : same as electrons.

Jet & MET



Then:

jets:

- default Anti-Kt (FastJet) JetAlgorithm 7 (includes substructure variables)
- using Eflow objects as input
- default cone 0.4
- default $\text{PT}>30 \text{ GeV}$

MET = negative vector sum of Eflow objects

B-tagging



- Suggest using parametrised efficiency and fake rate
- set DeltaR 0.4
- set BPTMin 20.0
- set BEtaMax 2.5
- add EfficiencyFormula {1,2,3} {0.001}
- add EfficiencyFormula {4} {0.05}
- add EfficiencyFormula {5} {0.70}

Efficiency for b's: 70%, charm: 5%, light quarks: 0.1%

Added a
separate tune
for charm

add EfficiencyFormula {1,2,3} {0.01}

add EfficiencyFormula {5} {0.10}

add EfficiencyFormula {4} {0.25}

Efficiency for charm: 25%, b's: 10%, light quarks: 1%

Tau-tagging



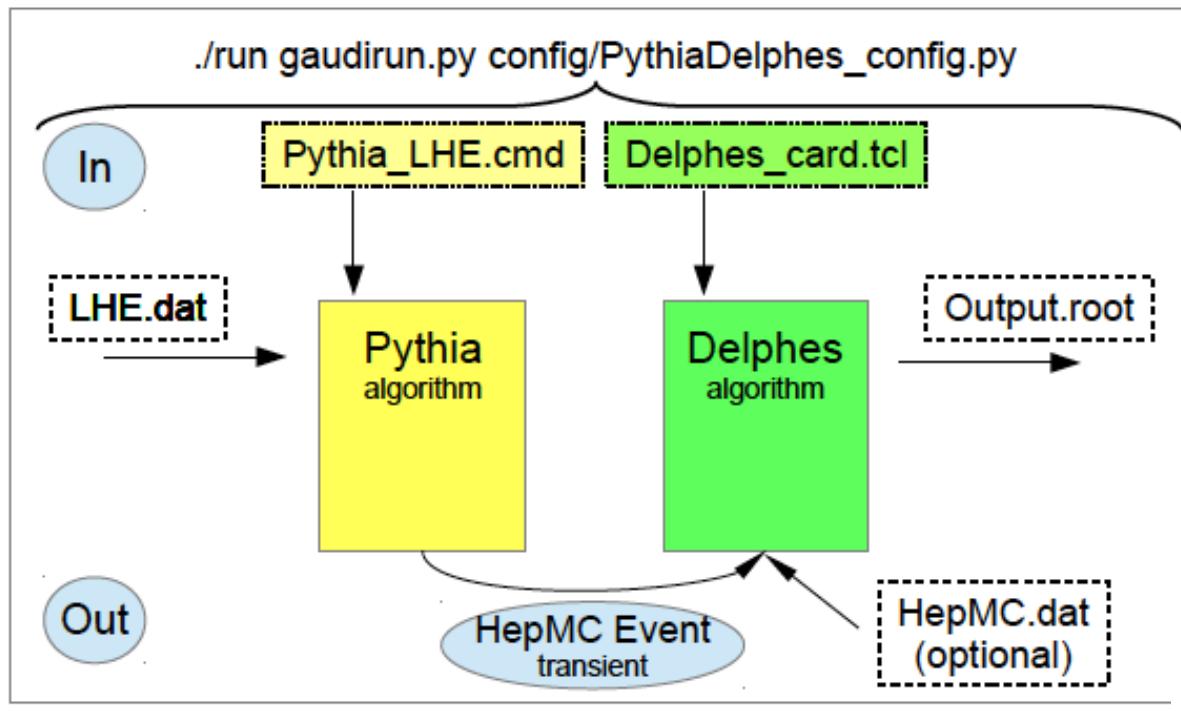
Parametrized description (40% efficiency, 0.1% fake rate, 0.5% electron fakes)

```
module TauTagging TauTagging {  
    set ParticleInputArray Delphes/allParticles  
    set PartonInputArray Delphes/partons  
    set JetInputArray JetEnergyScale/jets  
  
    set DeltaR 0.4  
  
    set TauPTMin 10.0  
  
    set TauEtaMax 2.5  
  
    # add EfficiencyFormula {abs(PDG code)} {efficiency formula as a function of eta and pt}  
  
    # default efficiency formula (misidentification rate)  
    add EfficiencyFormula {0} {0.001}  
    add EfficiencyFormula {11} {0.005}  
    # efficiency formula for tau-jets  
    add EfficiencyFormula {15} {0.4}  
}
```

No muon fake rate at the moment

Delphes in FCCSW

- FCCSW → based on modular structure of Gaudi framework
- Strategy:
 - modularize Pythia & Delphes as Gaudi algorithms
 - use configuration files to control processing sequence



**Zbynek
Drasal**

FCCSW configuration



- FCCSW configuration:
 - Run sequence of Gaudi framework controlled through a Python script
 - Use the following file: \${FCCSW} /config/PythiaDelphes_config.py
 - Variables to be arranged:
 - NEvents → events to be simulated
 - PythiaconfFile → Pythia configuration, called command, file (use either Pythia_LHEinput.cmd or Pythia_standard.cmd)
 - DelphesCard → Delphes TCL configuration file (use official Delphes card)
 - DelphesHepMCInFile → Delphes input file (use "" to read HepMC directly from Pythia, i.e. from transient data store)
 - DelphesRootOutFile → Delphes output file (define name of Delphes output file)
If file not defined the data goes to transient data store → will be useful once the FCC event-data output is defined

Zbynek
Drasal

Tutorial



- For additional information follow the HowTo? tutorial at FCC Twiki page:
 - <https://twiki.cern.ch/twiki/bin/view/FCC/FccPythiaDelphes>

TWiki > FCC Web > CommonTools > FccSoftware > FccPythiaDelphes (2015-09-15, ZbynekDrasal)

FCC Pythia + Delphes Analysis (Tutorial)

Contents

- + FCC Pythia + Delphes Analysis (Tutorial)
 - ↳ Overview
 - ↳ Installation Procedure
 - ↳ How to Run?
 - ↳ How to Analyze Data?
 - ↳ Example: Read data from Delphes output ROOT file

Overview

A small tutorial on how to study FCC-hh benchmark channels within the [FCCSW framework](#) with [Pythia](#) generator and [Delphes](#) simulation.

Input/Output:

- The Pythia input is required through [LHE](#) (Les Houches Event) data file together with a special Pythia config file or just standard Pythia config file.
- The Pythia output is through transient memory data store using [HepMC](#) event data format.
- The Delphes input is preferably from event data store, but can be in principal read-in from [HepMC](#) data file.
- The Delphes output is through Delphes ROOT tree writer, so via [ROOT](#) file.
- Support for FCC event-data format is under development --> will be available soon!

Zbynek
Drasal

FCC event database



Clement Helsens has set up a database that collects FCC samples:

<https://test-fcc.web.cern.ch/test-FCC/FCCEvents.php>

It contains two types of samples:

- LHE files
- generated events in FCC event data format (MC truth only)

	name	nevents	nfiles	outputdir	mainprocess	finalstates
1	B-4p-0-1	91000	10	/eos/fcc/hh/generation/snowmass/FCCEvents/v0_0/B_4p/B-4p-0-1_100TEV/	vector boson + jets	V+0J
2	BB-4p-0-300	55000	10	/eos/fcc/hh/generation/snowmass/FCCEvents/v0_0/BB_4p/BB-4p-0-300_100TEV/	divector boson + jets	V+nJ
3	BB-4p-1400-2900	55000	10	/eos/fcc/hh/generation/snowmass/FCCEvents/v0_0/BB_4p/BB-4p-1400-2900_100TEV/	divector boson + jets	V+nJ
4	BB-4p-2900-5300	46177	10	/eos/fcc/hh/generation/snowmass/FCCEvents/v0_0/BB_4p/BB-4p-2900-5300_100TEV/	divector boson + jets	V+nJ
5	BB-4p-300-1400	55000	10	/eos/fcc/hh/generation/snowmass/FCCEvents/v0_0/BB_4p/BB-4p-300-1400_100TEV/	divector boson + jets	V+nJ
6	BB-4p-5300-8800	31498	10	/eos/fcc/hh/generation/snowmass/FCCEvents/v0_0/BB_4p/BB-4p-5300-8800_100TEV/	divector boson + jets	V+nJ
7	BB-4p-8800-100000	9392	10	/eos/fcc/hh/generation/snowmass/FCCEvents/v0_0/BB_4p/BB-4p-8800-100000_100TEV/	divector boson + jets	V+nJ
8	BBB-4p-0-1200	55000	10	/eos/fcc/hh/generation/snowmass/FCCEvents/v0_0/BBB_4p/BBB-4p-0-1200_100TEV/	tri-vector + jets, Higgs associated + jets	(VVV+nJ),

Plan is to run these events through Delphes and provide the ROOT output.
Please help in the validation and provide feedback!

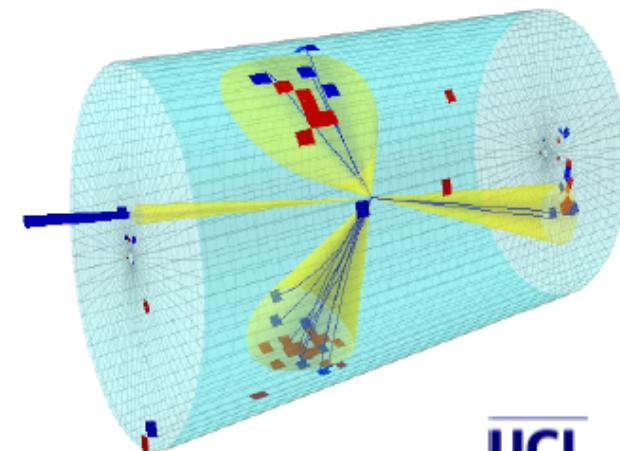
BACKUP



DELPHES



- **Delphes** is a **modular framework** that simulates the response of a multipurpose detector in a **parameterized** fashion
- **Includes:**
 - pile-up
 - charged particle **propagation** in magnetic field
 - electromagnetic and hadronic **calorimeters**
 - **muon** system
- **Provides:**
 - leptons (electrons and muons)
 - photons
 - jets and missing transverse energy (particle-flow)
 - taus and b's
- Website and manual: <https://cp3.irmp.ucl.ac.be/projects/delphes>
- Paper: [JHEP 02 \(2014\) 057](https://arxiv.org/abs/1402.0571)



M. Selvaggi,
A. Mertens,
P. Demin

Ingredients



What do we need for a DELPHES description of an FCC detector?

- basic geometry
 - radius of tracking system (before the magnet or calorimetry)
 - length of magnetic field coverage
 - magnetic field intensity
 - tracking coverage in eta/theta
 - eta/phi granularity of hcal and ecal
- momentum resolution formula for charged tracks
- energy resolution for electrons and photons
- momentum resolution for muons
- impact parameters resolution (optional)
- identification and mis-identification efficiency for particles: muons, electron, pions, kaons, ...
- neutral hadron energy fraction lost in hcal and ecal (sum =1)
- energy resolution formula for jets (optional)
- b-tag efficiency (optional)

Twin Solenoid design



"System"	"ATLAS"	"CMS"	"Twin Solo"
"DrTracker"	1.23	1.29	2.5
"DrCoil0"	0.05	0	0
"DrEcal"	1	0.48	1.1
"DrHcal"	1.97	1.23	2.6
"DrCoil"	0	0.312	1.73
"DrMuon"	6	4.2	3.57
"DrCoil2"	0	0	1.1
"LTracker"	2.8	2.9	8
"LEecal"	1	0.98	1.1
"LEhcal"	2.6	1.8	2.6
"LEmuon"	17	5.18	3.57
"LUtracker"	0	0	0
"LDipole"	0	0	0
"DrDipole"	0	0	0
"LDtracker"	0	0	0
"LFecal"	0	0	0
"LFhecal"	0	4	0
"LFmuon"	0	0	0
"EtaForward"	10	3	10
"Lstar"	23	23	40
"LTAS"	3	3	3
"Lcavern"	23.5	23.5	35
"Rcavern"	15	15	17
"Rtunnel"	2	2	2.5
"Rtriplet"	0.5	0.5	0.5
"zmax"	45	45	45
"ymax"	18	18	18
"eps"	0.1	0.1	0.1
"BBarrel"	2	3.8	6
"SigBarrel"	$20 * 10^{-6}$	$30 * 10^{-6}$	$20 * 10^{-6}$
"NBarrel"	15	15	15
"ForwardBFlag"	False	False	False
"ToroidFlag"	False	False	False

Jets/MET



Two options:

- A simple smearing of jet and MET (made up from genparticles)
- Or an implementation of a simple calorimeter
 - granularity
 - resolution
 - fraction of energy deposit per particle

Output: towers (full deposit) or EflowTowers (full deposit minus charged deposit)

First option: - easy to change the jet resolution
- but no way to study jet substructure, PU mitigation, ...

Second option: - can also do substructure, PU mitigation, ...
- but less direct to change resolution