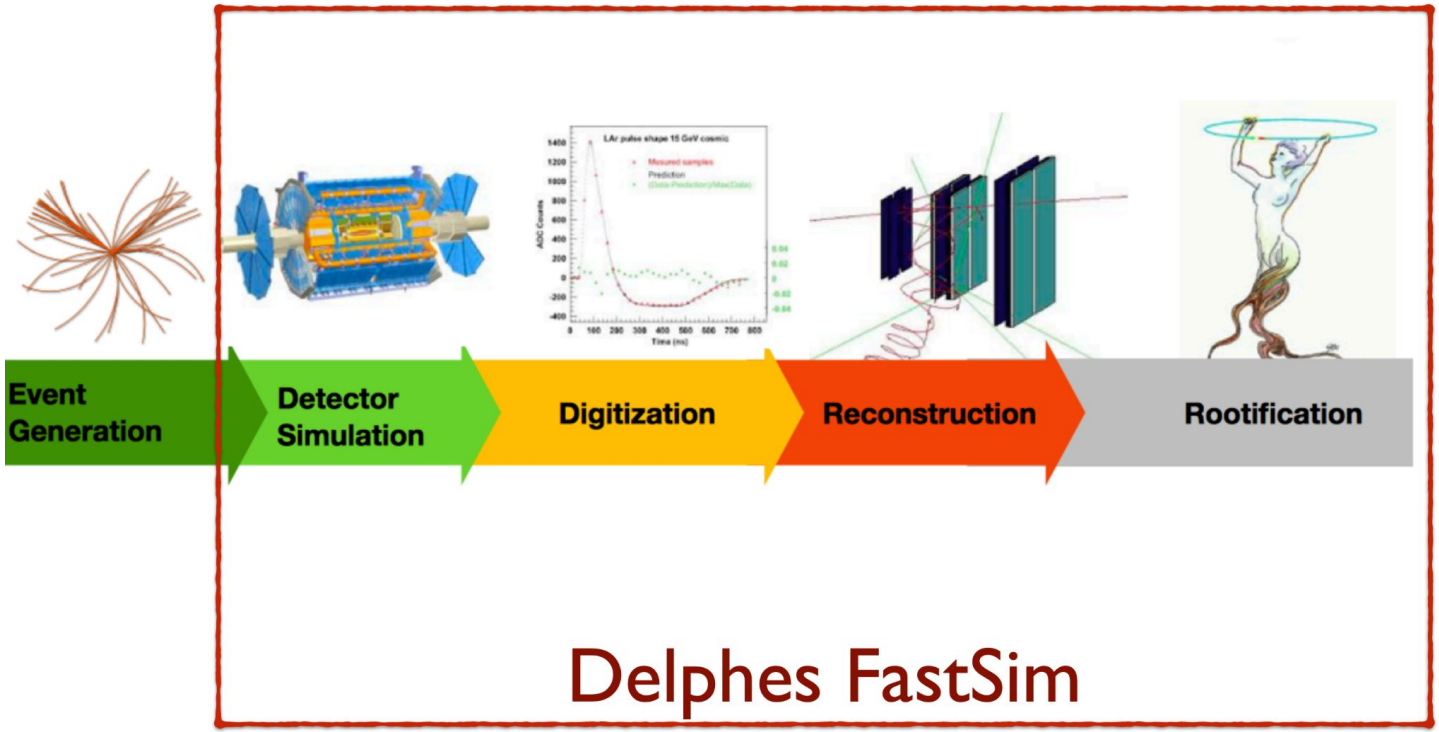
Delphes status and tuning

Michele Selvaggi (CERN)

In a nutshell

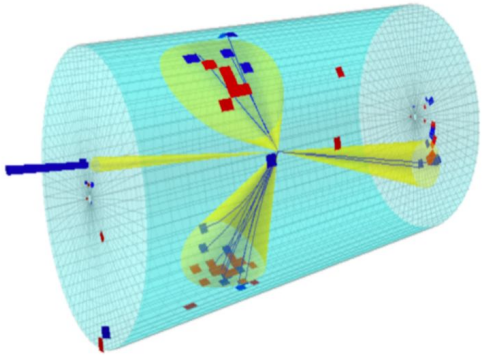


In a nutshell

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

- Includes:
  - pile-up
  - charged particle propagation in B field
  - EM/Had calorimeters
  - particle-flow

- Provides:
  - leptons, photons, neutral hadrons
  - jets, missing energy
  - heavy flavour tagging



- designed to deal with hadronic environment
- well-suited also for e+e- studies
- detector cards for: CMS (current/PhaseII) - ATLAS - LHCb - FCC-hh - ILD - CEPC - FCCee (IDEA/CLD) - MuCol

Speed

- Full simulation (GEANT):
  - **simulates** all **particle-detector interaction** (e.m/hadron showers, nuclear interaction, brem, conversions)

10^2-10^3 s/ev

- Experiment Fast Simulation (ATLAS, CMS ..)
  - **simplify** geometry, smear at the level of detector hits, frozen showers

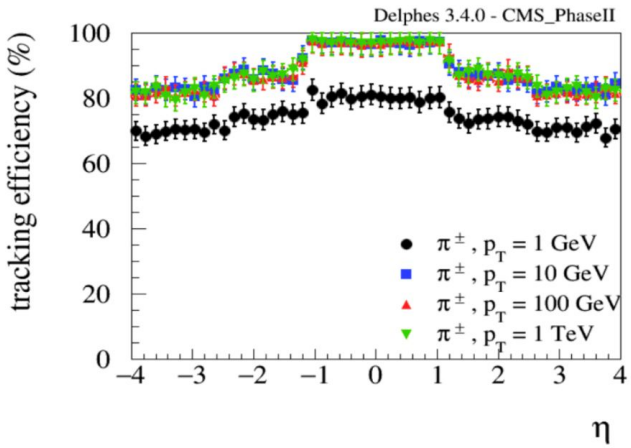
10-10^2 s/ev

- Parametric simulation (**Delphes**, PGS):
  - **parameterise detector response** at the particle level(efficiency, resolution on tracks, calorimeter objects)
  - reconstruct **complex objects** and observables(use particle-flow, jets, missing ET, pile-up ..)

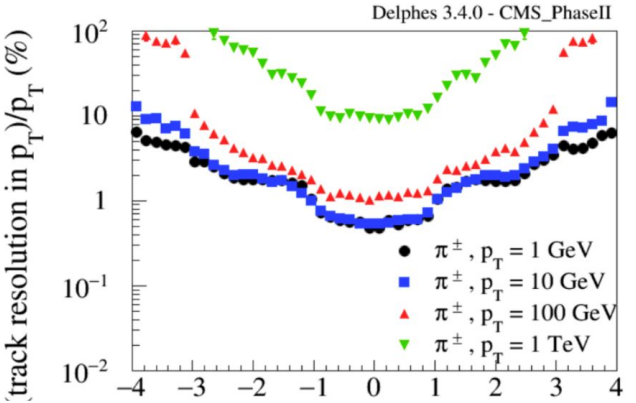
10^{-2} - 10^{-1} s/ev

- Ultra Fast:
  - from parton to detector object (smearing/lookup tables/ML)

In a nutshell - parameterisations



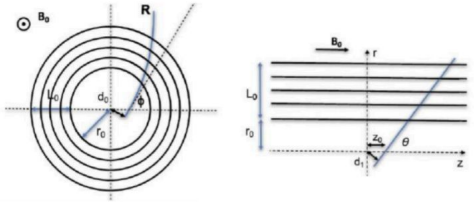

```
####################################
# Charged hadron tracking efficiency
####################################

module Efficiency ChargedHadronTrackingEfficiency {
  ## particles after propagation
  set InputArray  ParticlePropagator/chargedHadrons
  set OutputArray chargedHadrons
  # tracking efficiency formula for charged hadrons
  set EfficiencyFormula {
      (pt <= 0.2) * (0.00) + \
          (abs(eta) <= 1.2) * (pt > 0.2 && pt <= 1.0) * (pt * 0.96) + \
          (abs(eta) <= 1.2) * (pt > 1.0) * (0.97) + \
          (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 0.2 && pt <= 1.0) * (pt*0.85) + \
          (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 1.0) * (0.87) + \
          (abs(eta) > 2.5 && abs(eta) <= 4.0) * (pt > 0.2 && pt <= 1.0) * (pt*0.8) + \
          (abs(eta) > 2.5 && abs(eta) <= 4.0) * (pt > 1.0) * (0.82) + \
          (abs(eta) > 4.0) * (0.00)
  }
}
```



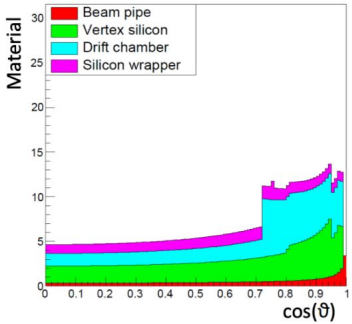

```
set ResolutionFormula {    (abs(eta) >= 0.0000 && abs(eta) < 0.2000) * (pt >= 0.0000 && pt < 1.0000) * (0.00457888) + \
   (abs(eta) >= 0.0000 && abs(eta) < 0.2000) * (pt >= 1.0000 && pt < 10.0000) * (0.004579 + (pt-1.000000)* 0.000045) + \
   (abs(eta) >= 0.0000 && abs(eta) < 0.2000) * (pt >= 10.0000 && pt < 100.0000) * (0.004983 + (pt-10.000000)* 0.000047) + \
   (abs(eta) >= 0.0000 && abs(eta) < 0.2000) * (pt >= 100.0000) * (0.009244+pt/100.000000) + \
   (abs(eta) >= 0.2000 && abs(eta) < 0.4000) * (pt >= 0.0000 && pt < 1.0000) * (0.00505011) + \
   (abs(eta) >= 0.2000 && abs(eta) < 0.4000) * (pt >= 1.0000 && pt < 10.0000) * (0.005050 + (pt-1.000000)* 0.000033) + \
   (abs(eta) >= 0.2000 && abs(eta) < 0.4000) * (pt >= 10.0000 && pt < 100.0000) * (0.005343 + (pt-10.000000)* 0.000043) + \
   (abs(eta) >= 0.2000 && abs(eta) < 0.4000) * (pt >= 100.0000) * (0.009172+pt/100.000000) + \
   (abs(eta) >= 0.4000 && abs(eta) < 0.6000) * (pt >= 0.0000 && pt < 1.0000) * (0.00510573) + \
   (abs(eta) >= 0.4000 && abs(eta) < 0.6000) * (pt >= 1.0000 && pt < 10.0000) * (0.005106 + (pt-1.000000)* 0.000023) + \
   (abs(eta) >= 0.4000 && abs(eta) < 0.6000) * (pt >= 10.0000 && pt < 100.0000) * (0.005317 + (pt-10.000000)* 0.000042) + \
   (abs(eta) >= 0.4000 && abs(eta) < 0.6000) * (pt >= 100.0000) * (0.009077+pt/100.000000) + \
   (abs(eta) >= 0.6000 && abs(eta) < 0.8000) * (pt >= 0.0000 && pt < 1.0000) * (0.00578020) + \
   (abs(eta) >= 0.6000 && abs(eta) < 0.8000) * (pt >= 1.0000 && pt < 10.0000) * (0.005780 + (pt-1.000000)* -0.000000) + \
   (abs(eta) >= 0.6000 && abs(eta) < 0.8000) * (pt >= 10.0000 && pt < 100.0000) * (0.005779 + (pt-10.000000)* 0.000038) + \
   (abs(eta) >= 0.6000 && abs(eta) < 0.8000) * (pt >= 100.0000) * (0.009177+pt/100.000000) + \
   (abs(eta) >= 0.8000 && abs(eta) < 1.0000) * (pt >= 0.0000 && pt < 1.0000) * (0.00728723) + \
   (abs(eta) >= 0.8000 && abs(eta) < 1.0000) * (pt >= 1.0000 && pt < 10.0000) * (0.007287 + (pt-1.000000)* -0.000031) + \
   (abs(eta) >= 0.8000 && abs(eta) < 1.0000) * (pt >= 10.0000 && pt < 100.0000) * (0.007011 + (pt-10.000000)* 0.000038) + \
   (abs(eta) >= 0.8000 && abs(eta) < 1.0000) * (pt >= 100.0000) * (0.010429+pt/100.000000) + \
   (abs(eta) >= 1.0000 && abs(eta) < 1.2000) * (pt >= 0.0000 && pt < 1.0000) * (0.01045117) + \
   (abs(eta) >= 1.0000 && abs(eta) < 1.2000) * (pt >= 1.0000 && pt < 10.0000) * (0.010451 + (pt-1.000000)* -0.000051) + \
   (abs(eta) >= 1.0000 && abs(eta) < 1.2000) * (pt >= 10.0000 && pt < 100.0000) * (0.009989 + (pt-10.000000)* 0.000043) + \
```

New features (highlights)

Track Covariance

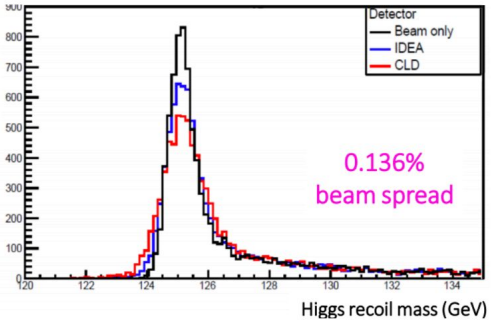


$$\vec{\alpha} = (D,\, \varphi_0,\, C,\, z_0,\, \lambda)$$



Track Smearing

- Simple tracker geometry implementation, **including material**
- Computes **full covariance matrix** (in present Delphes we have "diagonal" smearing in the 5 tracking parameters)
- Can be used for studying impact of material and realistic **HF tagging** simulation

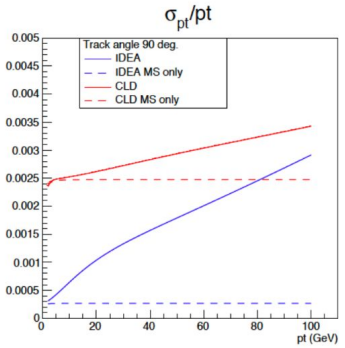


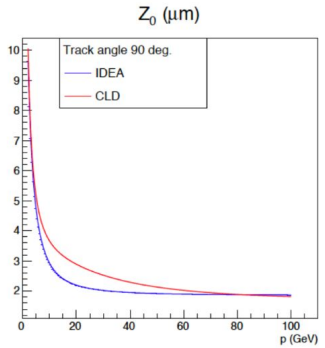
Bedeschi, Gouskos, MS, [2202.03285]

Track Covariance

```
########################################
# Smearing for charged tracks
########################################

module TrackCovariance TrackSmearing {

    set InputArray TrackMergerPre/tracks
    set OutputArray tracks

    ## minimum number of hits to accept a track
    set NMinHits 6

    ## magnetic field
    set Bz $B

    ## uses https://raw.githubusercontent.com/selvaggi/FastTrackCovariance/master/GeoIDEA_BASE.txt
    set DetectorGeometry {
```

# barrel	name	zmin	zmax	r	w (m)	X0	n_meas	th_up (rad)	th_down (rad)	reso_up (m)	reso_down (m)	flag
1	PIPE	-100	100	0.015	0.001655	0.2805	0	0	0	0	0	0
1	VTXLOW	-0.12	0.12	0.017	0.00028	0.0937	2	0	1.5708	3e-006	3e-006	1
1	VTXLOW	-0.16	0.16	0.023	0.00028	0.0937	2	0	1.5708	3e-006	3e-006	1
1	VTXLOW	-0.16	0.16	0.031	0.00028	0.0937	2	0	1.5708	3e-006	3e-006	1
1	VTXHIGH	-1	1	0.32	0.00047	0.0937	2	0	1.5708	7e-006	7e-006	1
1	VTXHIGH	-1.05	1.05	0.34	0.00047	0.0937	2	0	1.5708	7e-006	7e-006	1

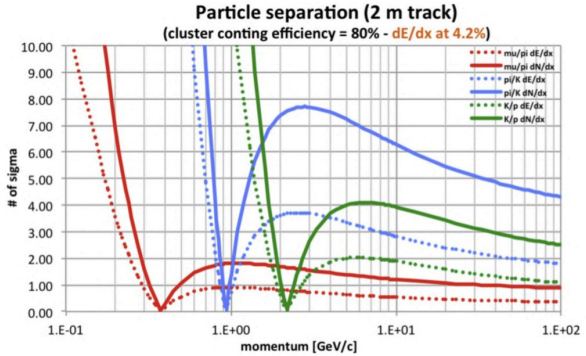




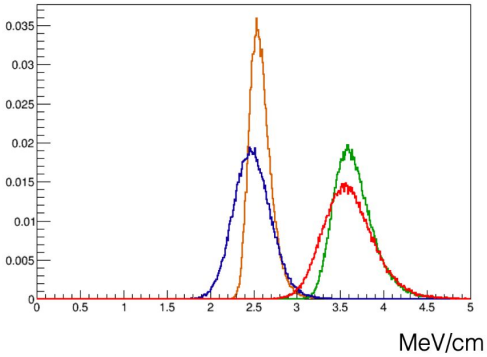
TrackCovariance module

- **Requires:**
  - Geometry input
    - cylinder coaxial
    - planar disks
  - Magnetic field

Hadron ID (dEdX, dNdx)




```
module EnergyLoss EnergyLoss {
  add InputArray ChargedHadronMomentumSmearing/chargedHadrons
  add InputArray ElectronMomentumSmearing/electrons
  add InputArray MuonMomentumSmearing/muons


  # absolute resolution per measurement (normalized in MeV/cm)
  # CMS pixel detector performance is reproduceable with r = 0.4
  # dedicated dEdX detector can achieve r = 0.0 or below (i.e better than Landau)

  #set Resolution 0.4
  set Resolution 0.2

  # fraction of measurements to ignore when computing truncated mean
  # suggested range [0.4-0.6]

  set TruncatedMeanFraction 0.5

  # detector properties (active fraction = nhits*thickness/L)
  set Thickness 100E-6
  set ActiveFraction 0.0006666

  # Silicon properties, for other materials:
  # cf. http://pdg.lbl.gov/2014/AtomicNuclearProperties/properties8.dat

  set Z 14.
  set A 28.0855
  set rho 2.329

  # material polarisation correction parameters
  set a 0.1492
  set m 3.2546
  set x0 0.2015
  set x1 2.8716
  set I 173.0
  set c0 4.4355
```



10 measurements x100 um silicon

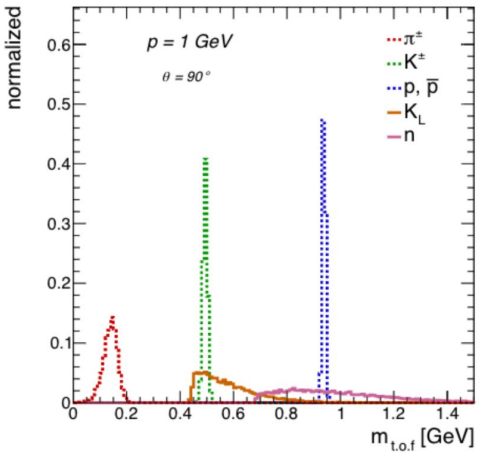
beta = 1 (Landau)
beta = 0.75 (Landau)
beta = 1 (LanGauss)
beta =0.75 (LanGauss)

PID:

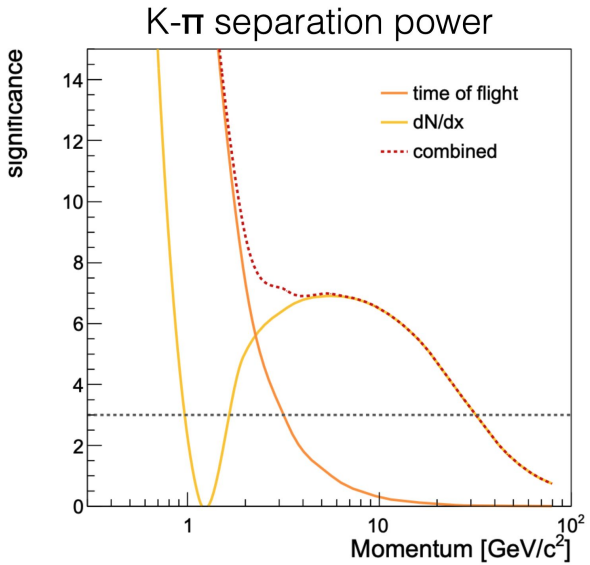
- dN/dx method:
  - param. from Garfield (vs βγ)
- 4 gas mixes implemented
- parameterisation

Hadron ID (ToF)

- Implemented in the IDEA card for testing for both charged and neutrals:




```
########################################
#   Time Smearing MIP
########################################

module TimeSmearing TimeSmearing {
  set InputArray ClusterCounting/tracks
  set OutputArray tracks

  # assume constant 30 ps resolution for now
  set TimeResolution {
                      (abs(eta) > 0.0 && abs(eta) <= 3.0)* 30E-12
                     }
}


########################################
#   Time Of Flight Measurement
########################################

module TimeOfFlight TimeOfFlight {
  set InputArray TimeSmearing/tracks
  set VertexInputArray TruthVertexFinder/vertices

  set OutputArray tracks

  # 0: assume vertex time tV from MC Truth (ideal case)
  # 1: assume vertex time tV = 0
  # 2: calculate vertex time as vertex TOF, assuming tPV=0

  set VertexTimeMode 0
}
```



Delphes CMS Phase II Tuning (for Snowmass 2021)

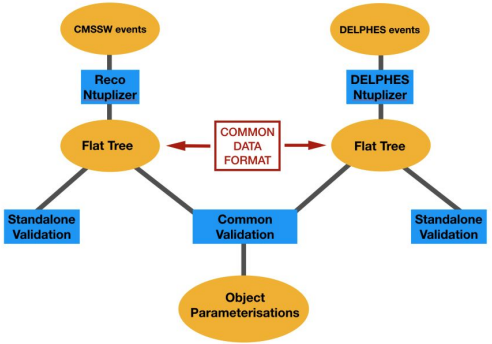
In collaboration with
Julie Hogan, Sandya Jain and Oguz Guzel

Context and caveats

- Produce physics projection for the HL-LHC in the context of the SnowMass effort
  - Used in many (> 20) official CMS results

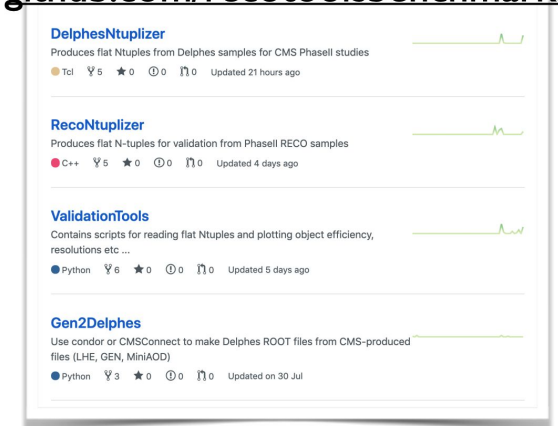
- Dedicated group of 5-10 worked on producing a Delphes CMS Phase II card based on CMS Phase II simulation performance
  - Tuning focussed on "standard" objects (leptons, jets/MET, flavour tagging)
    - i.e "prompt physics"
  - by construction, projected performance VERY conservative, probably out-dated now
    - Since based on "current" reco performance at the time (very limited person-power for optimisation of Phase II reco algorithms in CMS)

Validation Suite



- **Ntuplizers** produce flat nTuples with basic information on kinematics, isolation and ID for all physics objects.
  - Can be produced by running FS/Delphes event reconstruction on GEN-SIM samples
  - Can be used as common format for FS/ Delphes
    - for fullSim/Delphes comparison
    - for physics analysis

github.com/recotoolsbenchmarks

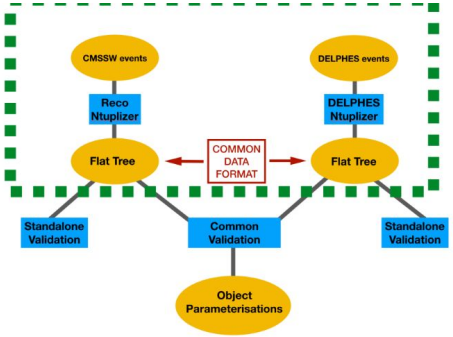


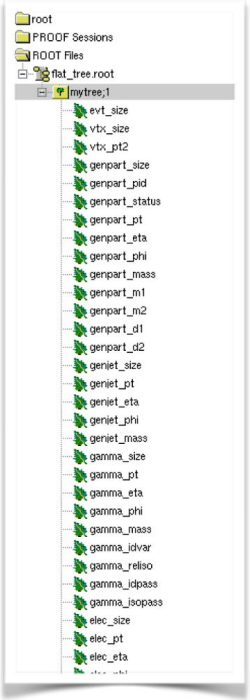
- **ValidationTools** contain utilities for reading flat tree format and producing comparisons
- **Gen2Delphes** contains scripts for submission of Delphes production campaigns (using cms-connect)

A common data format for FullSim/Delphes

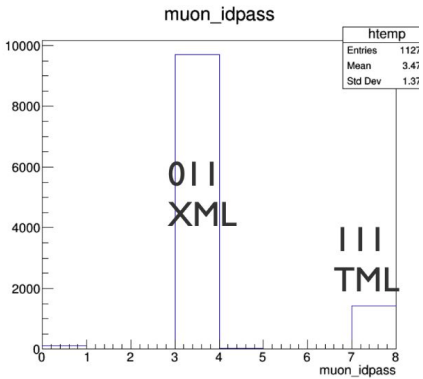
github.com/recotoolsbenchmarks/DelphesNtuplizer

github.com/recotoolsbenchmarks/RecoNtuplizer





- Contains **gen** and **reco** information of most common objects **according to recommended recipes**:
  - P4
  - Id variable
  - Iso variables
  - Id/Tags

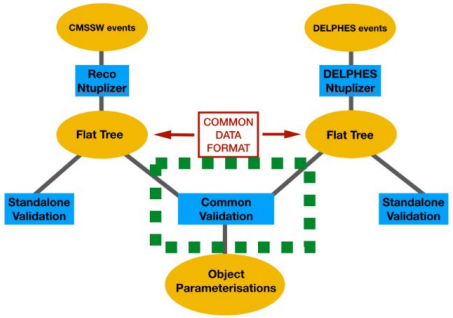


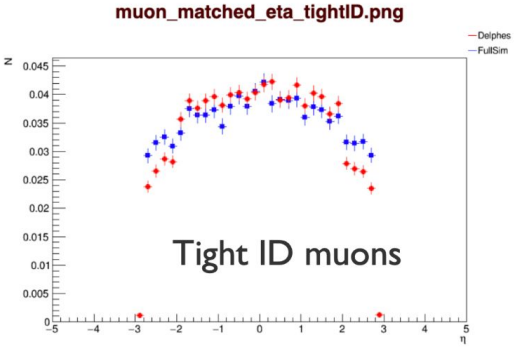
bitwise information on ID/tagging

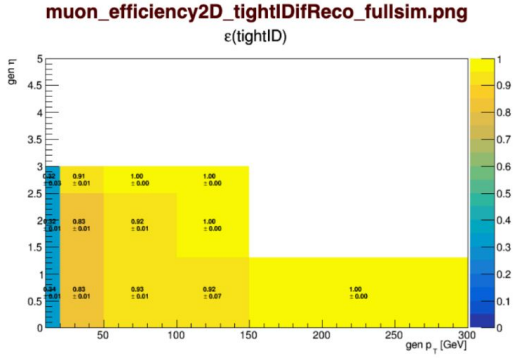
Validation Tools

github.com/recotoolsbenchmarks/ValidationTools

- Contains:
  - scripts for easily **accessing the flat tree content** (NtupleDataFormat class)
  - scripts for plotting for **kinematic information**, id variables
  - templates for **computing object efficiencies** (fake rate), resolutions, responses

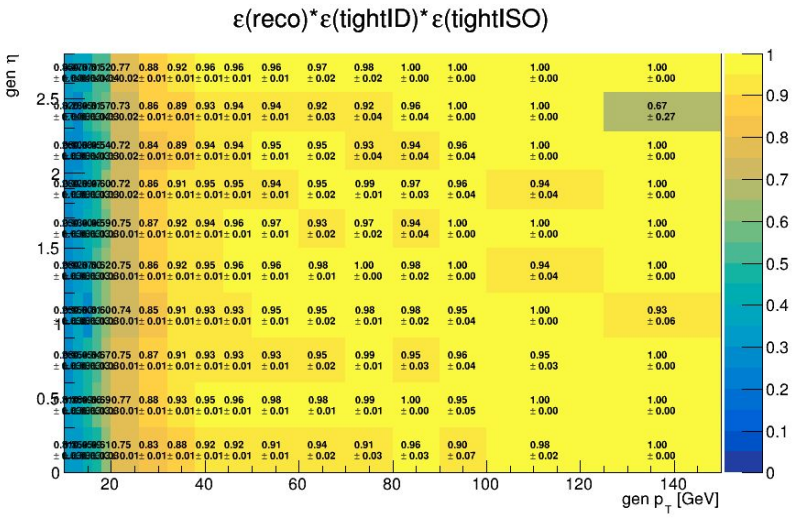


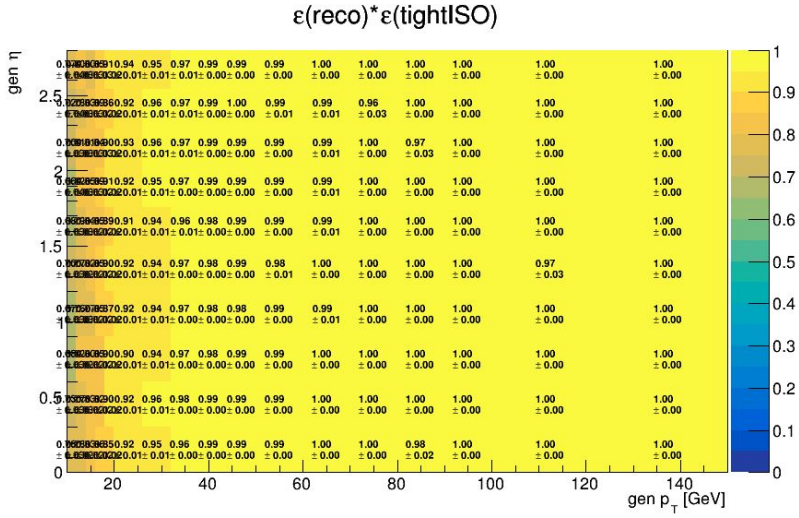




Efficiencies

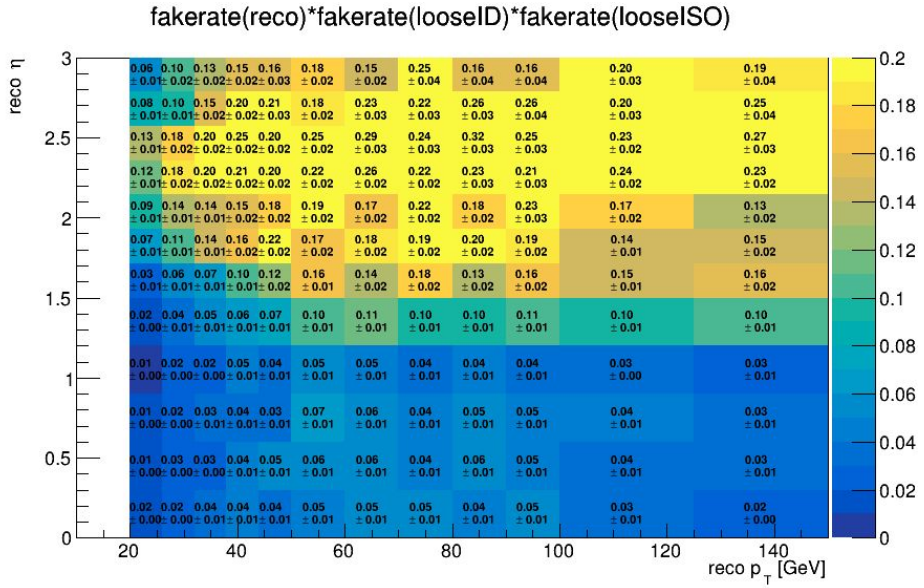
- Delphes natively applies reco efficiencies, and isolation
  - encode in ID all residual inefficiencies
  - For each working point (L/M/T), ID efficiency in Delphes derived as :
    - eff_delphes (ID) = eff_fullsim (reco x iso x ID) / eff_delphes (reco x iso)





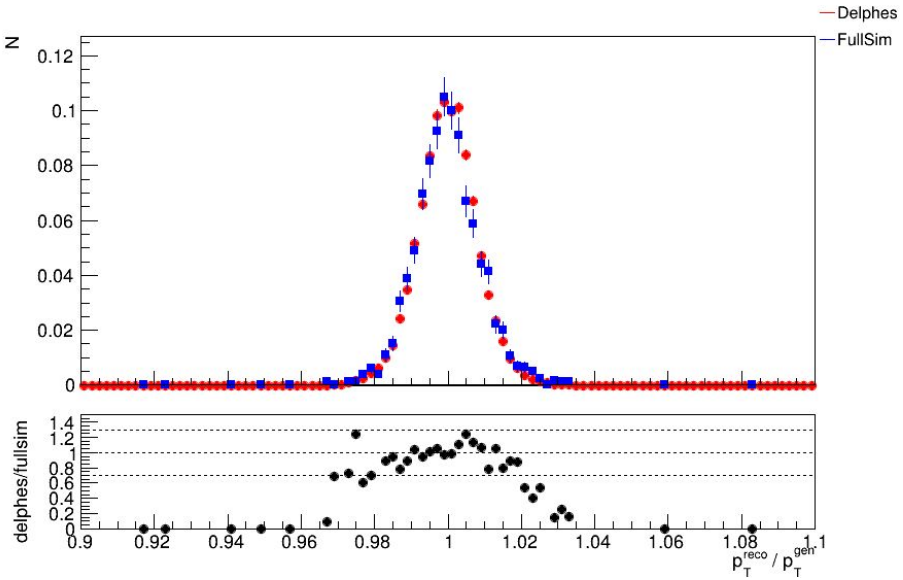
Fake-rate

- defined as fraction of reco-jets incorrectly ID'ed as e/mu/gamma
  - this definition allows to include pile-up contribution (not stored in gen-jets)
  - assumes recojets well-modeled in Delphes



Resolution - scale

- Apply scale μ(full-sim) / μ(delphes)
- Apply residual resolution to match full sim
  - σ(smear) = sqrt(σ(full-sim)^2- σ(delphes)^2)



Procedure

- Start with aggressive (ID efficiency and resolution) baseline detector assumptions
- Compute correction factors, fake rate and resolution difference comparing fast/full sim
  - Over wide range of objects (jet/MET, e/mu/gamma, flavor tagging) and kinematics
  - NOTE: requires large (number ~10M and variety) MC samples
    - e.g at the time of SnowMass:
      - electrons/muons: DYToLL
      - Photons: H → gamma gamma
      - jets/ (e/mu/gamma fakes): QCD_Pt-15to3000
      - MET: DYToLL
      - Btag: TTbar
      - Tautag: Htautau
  - This can be optimized and agreed upon via smart choice of kinematics /processes
- Apply correction efficiency/smearing in Delphes by appending new module instances
  - Benefits from the modularity of Delphes
- Iteratively (fine-)tune if residual corrections to be applied (typical turnaround ~2-3 days)
  - assuming benchmark MC is already produced

Conclusions

- Framework for "auto" Delphes tuning Delphes card parameterisation was developed during the Snowmass 2021 effort =:
    - all steps are fully automatized
    - requires "some" amount of fine tuning card by hand, due to poor stats in some bins, fit instabilities etc
    - turnaround from new card to closure validation ~ 2days
- All this work can be re-vamped and re-utilized if person-power allocated to this task
- If experimental collaborations are open to having an official Delphes parameterisation,
    - benchmark processes event generation/full simulation per experiment is needed
    - what is required is essentially a public list of full sim samples in a commonly agreed flat format described above, with skimmed MC information