

Jet clustering interface in FCC Analyses & b-tagging

Julie Munch Torndal
`julie.munch.torndal@cern.ch`

Niels Bohr Institute

April 26, 2021

Analysis of top-quark electroweak couplings to the photon and the Z boson in pair produced events

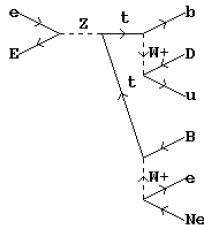
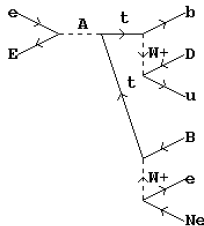
Signal:

Semileptonic channel

$$t\bar{t} \rightarrow b\bar{b}W^+W^- \rightarrow b\bar{b}q\bar{q}\ell\nu_\ell$$

Planned phase of FCC-ee @ $\sqrt{s} = 365\text{ GeV}$

Signature: 1 lepton + \cancel{E} + 4 jets (2 btags)




→ Jet studies!








Interface for later stage process adaptive jet clustering

- flexible "after burner" using FastJet
- dedicated more towards e^+e^- collisions
- run multiple jet reconstructions at once
- select input particles
- access to jet constituents

master [FCCAnalyses](#) / [analyzers](#) / [dataframe](#) /

Go to file Add file ...

 clementhelsens Update ReconstructedParticle.cc ✖ a065c10 on Mar 19 [History](#)

..		
 Algorithms.cc	cleanup in Algorithms	2 months ago
 Algorithms.h	cleanup in Algorithms	2 months ago
 CMakeLists.txt	Added Jade Plugin	2 months ago
 JetClustering.cc	Added Jade Plugin	2 months ago
 JetClustering.h	Added Jade Plugin	2 months ago
 JetClusteringUtils.cc	add protection for number of input constituents wrt number of exclusi...	2 months ago
 JetClusteringUtils.h	add protection for number of input constituents wrt number of exclusi...	2 months ago

Credit to Clement Helsen for setting up the interface

Jet Algorithms
k_t
Anti- k_t
Cambridge/Aachen
Generalised- k_t
Durham
Generalised- k_t for e^+e^-
Valencia
Jade



Recombination Schemes
E -scheme
p_t -scheme
p_t^2 -scheme
E_t -scheme
E_t^2 -scheme
Boost-invariant p_t -scheme
Boost-invariant p_t^2 -scheme
$E0$ -scheme
p -scheme

The jet definition depends on

- which partons are chosen to be combined into the jet
- how they are combined into the jet

Jet Algorithms		
k_t	clustering_kt	} suitable for pp collisions
Anti- k_t	clustering_antikt	
Cambridge/Aachen	clustering_cambridge	
Generalised- k_t	clustering_genkt	
Durham	clustering_ee_kt	} suitable for e^+e^- collisions
Generalised- k_t for e^+e^-	clustering_ee_genkt	
Valencia	clustering_Valencia	
Jade	clustering_Jade	

Plugins {

- The choice for the most suitable jet algorithm is specific to the analysis
- Plugins makes it possible to add your own favourite jet algorithm

Recombination schemes (coming soon)

Recombination Schemes
E -scheme
p_t -scheme
p_t^2 -scheme
E_t -scheme
E_t^2 -scheme
Boost-invariant p_t -scheme
Boost-invariant p_t^2 -scheme
$E0$ -scheme
p -scheme

- FastJet is focused towards hadron colliders and besides the E -scheme it does not have dedicated schemes for e^+e^- collisions.
- $E0$ - and p -scheme are external recombination schemes

Source: [1] and Pull request

E-scheme: Parton i and j are replaced by a pseudojet k with four-momentum

$$\mathbf{p}_k = \mathbf{p}_i + \mathbf{p}_j$$

- Lorentz invariant, energy and momentum conserved, non-zero mass for pseudojet k .

E0-scheme: The four-momentum of pseudojet k is rescaled to have zero invariant mass

$$E_k = E_i + E_j \quad , \quad \vec{p}_k = \frac{E_k}{|\vec{p}_i + \vec{p}_j|} \cdot (\vec{p}_i + \vec{p}_j)$$

- Not Lorentz invariant, only conserves energy.

p-scheme: The four-momentum is constructed to have zero invariant mass

$$\vec{p}_k = \vec{p}_i + \vec{p}_j \quad , \quad E_k = |\vec{p}_k|$$

- Not Lorentz invariant, only conserves momentum.

```
#build pseudo jets from momentum components and energy
.Define("pseudo_jets", "JetClusteringUtils::set_pseudoJets(RP_px, RP_py, RP_pz, RP_e)")

#run jet clustering with all reconstructed particles.
#jade_algorithm, R=0.5, exclusive clustering, exactly 4 jets, sorted by E, E0-scheme
.Define("FCCAnalysesJets_jade", "JetClustering::clustering_jade(0.5, 2, 4, 1, 10)(pseudo_jets)")

#get the jets out of the struct
.Define("jets_jade", "JetClusteringUtils::get_pseudoJets(FCCAnalysesJets_jade)")
#get the jets constituents out of the struct
.Define("jetconstituents_jade", "JetClusteringUtils::get_constituents(FCCAnalysesJets_jade)")
#get some variables
.Define("jets_jade_px", "JetClusteringUtils::get_px(jets_jade)")
.Define("jets_jade_py", "JetClusteringUtils::get_py(jets_jade)")
.Define("jets_jade_pz", "JetClusteringUtils::get_pz(jets_jade)")
.Define("jets_jade_btag", "JetClusteringUtils::get_btag(jets_jade, Particle, 0.80)")

/* Structure to keep useful informations for the jets*/
struct FCCAnalysesJet{
    ROOT::VecOps::RVec<fastjet::PseudoJet> jets;
    std::vector<std::vector<int>> constituents;
};
```

Arguments for jet definition:

- ① Jet cone radius
- ② Clustering
 - 0=inclusive clustering,
 - 1=exclusive clustering with dcut,
 - 2=exclusive clustering to exactly njets,
 - 3=exclusive clustering up to exactly njets,
 - 4=exclusive clustering with ycut.
- ③ Cut-value depending on clustering
- ④ Ordering of returned jets
 - 0=sorted by p_t ,
 - 1=sorted by E .
- ⑤ Recombination scheme
- ⑥ Additional input parameters specific to jet algorithm
 - see JetClustering.h

b-tagging a la Delphes (coming soon)

In Delphes, a jet is b-tagged if a b-parton lies within a cone radius of $\Delta R < 0.5$.
An efficiency formula is applied e.g. in `delphes/cards/delphes_card_IDEA.tcl`:

```
# efficiency formula for b-jets  
add EfficiencyFormula {5} {0.80}
```

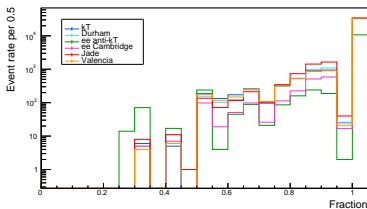
In FCCAnalyses, a jet is b-tagged if a b-parton lies within a vector angle < 0.3 .
It takes a flat efficiency (for now).

```
.Define("jets_jade_btag", "JetClusteringUtils::get_btag(jets_jade, Particle, 0.80)")
```

OBS: Only works for samples generated with Pythia since b-partons are selected from their status code (71-79) and PDG.

Reco jets

Largest fraction of b-hadron decay products in one jet

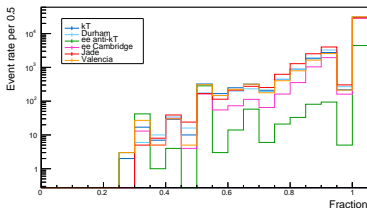


Events with full separation of b-hadron decay products:

- k_T : 76.3 ± 0.6 %
- Durham: 79.2 ± 0.6 %
- ee anti- k_T : 12.3 ± 0.2 %
- ee Cambridge: 61.7 ± 0.6 %
- Jade: 73.8 ± 0.6 %
- Valencia: 76.9 ± 0.6 %

Particle jets

Largest fraction of b-hadron decay products in one jet



Events with full separation of b-hadron decay products:

- k_T : 62.2 ± 0.6 %
- Durham: 62.8 ± 0.6 %
- ee anti- k_T : 3.71 ± 0.14 %
- ee Cambridge: 42.5 ± 0.5 %
- Jade: 54.3 ± 0.5 %
- Valencia: 64.1 ± 0.6 %

b-tagging a la Delphes:

- Exploit MC and match jet to b-hadron instead of b-parton

b-tagging using b-hadron decay products:

- Energy sharing or track sharing?
- Proto-jets – force all decay products into the same jet

b-tagging with vertexing:

- Realistic b-tagging requires vertexing
- Primary vertex fitters are available in FCCAnalyses
- Secondary vertex finding tools are not available yet
- Two suggestions:
 - 1 Apply vertex fitter to constituents of a jet and use distance to the IP (0,0,0). Assumes good separation of decay products in the jets and does not account for potential tertiary vertex.
 - 2 Use perfect seeding from MC to look for secondary vertices with MC to RP associations. See [Analysis of \$B_c\$ / \$B^+\$ to tau nu](#) presented by Clement Helsens at FCC-ee Physics Performance meeting, April 19, 2021.

