ATLAS Internal Note
ATLAS-SOFT-NO-XXX
September 30, 1998

Jet Finder Library : version 1.0

M. Bosman, I.C. Park
I.F.A.E, U.A.B., Barcelona
M. Cobal
CERN, Geneva, Switzerland
D. Costanzo, S. Lami, R. Paoletti
University of Pisa, INFN and Scuola Normale Superiore, Italy
G. Azuelos, K. Strahl
Univ. de Montréal, Montréal

September 30, 1998

Abstract

This note describes the conventional cone algorithm, two versions of the K_T algorithm, and a new clustering algorithm Mulguisin as they are implemented in the new ATLAS Jet Finder Library. Other algorithms, such as the CDF cone algorithm and topological clustering, are being tested and will be implemented in the next version.

Contents

1 Introduction

Many of the interesting physics signatures at the LHC involve jets in the final state: like resonances in invariant mass of multijet final states or jet ET distributions. The definition of a jet is not unique and the correspondence between parton energy and direction and the measured jet characteristics is influenced by many factors: parton fragmentatiom, final state radiation, underlying events, detector response and by the jet algorithm itself. To understand all those effects, it is useful to have access to different jet algorithms both in the framework of the fast simulation program ATLFAST [1] and of the detailed simulation DICE+ATRECON [2] [3]. Many jet algorithms have been used in past experiments and new ideas to cope with the specific environment of a high luminosity hadron collider like LHC may come up.

To provide easy access to many jet algorithms, a framework has been developed: a library of fortran routines, with well defined input and output common blocks. Interfaces to the ATLFAST and DICE+ATRECON packages are provided such that the choice of the jet algorithm and their parameters can be controlled by data cards.

This note reviews the jet algorithms that have been so far included in the library together with their input parameters. The interfaces to the two ATLAS packages are described in appendices.

2 Jet Algorithms

The final states of an hadron collision consist of a set of particles i with four-momenta p_i^μ, resulting from the primary hard process, hadronization, and decay of short-lived particles. In the granulated calorimeter, one only measures energy depositions p_{cell}^μ in cells, each covering a given range in pseudorapidity η and azimuthal angle ϕ. The values of each p_{cell}^μ represent the sum of the p_i^μ which that cell subtends. Calling each p_{cell}^μ a "protojet", the final state of a collision now consists of protojets, or cells, with transverse energy E_{Tcell} in two-dimensional $\eta - \phi$ space. The Jet Finder library is then a set of algorithms which make clusters based on the cells and define jets by certain catagory from the clusters. Here we summarize some well-defined jet algorithms which have been used in hadron collider experiments or have been developed recently.

2.1 Conventional Cone-Based Jet Algorithm

The conventional cone-based jet finding algorithm consists of

1. Finding a maximum E_T^{cell} cell in $\eta - \phi$ space and use the cell as a seed for a jet candidate if $E_T^{seed} > E_{threshold}^{seed}$, where $E_{threshold}^{seed}$ is a parameter for the jet definition.

2. Make a cone of radius R around the seed center and sum the E_T of cells inside the cone.

3. A cone with $E_T > E_{threshold}^{cone}$ is a jet, where $E_{threshold}^{cone}$ is another parameter of the jet definition.

4. Repeat the steps 1, 2, and 3 until there is no more seed found.

Remarks

There is no treatment for overlapping jets in this version.

2.2 CDF Fixed Cone Algorithm

The algorithm [1], is closely related to the algorithm used by the UA1 experiment, and corresponds closely to the definitions used in calculating QCD cross-sections.

Algorithm description

1. A list of towers above a fixed E_T is created, to be used as seeds for the jet finder
2. Preclusters are formed from an unbroken chain of contiguous seed towers with a continuously decreasing tower E_T. If a tower is outside a window of 7x7 towers surrounding the seed, it is used to form a new pre-cluster. These preclusters are used as a starting point for cone clustering. Using the true tower segmentation, the preclusters are grown into clusters.
3. The E_T weighted centroid of the precluster is found and a cone in the $\eta - \phi$ space of radius R is formed around the centroid.
4. All towers with E_T greater than a threshold E_o are incorporated into the cluster. A tower is included in a cluster if its centroid is inside the cone. Otherwise it is excluded. The tower centroid is calculated from the E_T weighted centroid of the EM and hadronic compartments.
5. A new cluster center is calculated from the set of towers within the clustering cone, again using an E_T weighted centroid, and a new cone is drawn about this position.

Remarks

- The process of recomputing a centroid and finding new or deleting old towers is iterated until the tower list remains unchanged.
- The algorithm handles the overlap of two clusters. This an important feature, particularly for final-state gluon radiation where the gluon can merge into the jet. There are four possible overlap conditions.
    1. If two clusters are distinct, they are left alone
    2. If one cluster is completely contained in another, the smaller of the two is dropped
    3. If the towers have some finite overlap, then an overlap fraction is defined as the sum of the E_T of the towers in common, divided by the E_T of the smaller cluster. If the fraction is above a cutoff then the two clusters combined, otherwise the clusters are both kept. In this last case each tower in the overlap region is

[1] CDF, which is now using the fixed cone algorithm, explored in the past several techniques to identify jets. Some information can be obtained from [4] [5].

assigned to the cluster closest in the $\eta - \phi$ space. After the clusters are uniquely assigned to towers, the centroids are re-computed.

The process of centroid computation and tower reshuffling is iterative, and ends when the tower lists remain unchanged.

2.3 E_T Dependent Cone Algorithm (CDF)

Algorithm description

It is similar to the CDF fixed cone algorithm, but the cone radius varies as:

$$\Delta R = Min(\frac{12.0}{E_T}, 0.6).$$

This algorithm is based on the concept that a jet is composed of particles with a limited energy transverse to a central axis, and therefore the central axis is related to the jet E_T. Due to the gluon bremstrahlung, jets can broaden at higher E_T complicating matters.

Remarks

This algorithm is not yet implemented in the Jet Finder Library.

2.4 K_T Algorithm (version Pisa)

A better description of the K_T algorithm can be found in [6]. An application of the K_T algorithm can be found in the ATLAS internal note [7].

Algorithm description

1. Define a list of jets. This is just a list of the four-momenta of hit calorimeter cells.

2. For each jet, i, define:

$$D_{iB} = p_{ti}^2$$

and for each pair of jets, i, j, define:

$$d_{ij} = min(p_{ti}, p_{tj})^2 R_{ij}^2 / R_{cut}^2,$$
$$R_{ij}^2 = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2.$$

p_{ti}, η_i and ϕ_i are, respectively the transverse momentum, rapidity and azimuth of jet i. R_{cut} is a parameter somehow related to the cluster dimension.

3. Find the smallest number of $\{d_{ij}, D_{iB}\}$, and call it d_n, where n is the number of jets remaining.

4. If $d_n = d_{ij}$, then merge jets i and j to give a single jet with four momentum

$$p_{(ij)} = p_i \oplus p_j$$

where $\oplus$ is some operation defining the recombination criterium which can be chosen in different ways:

(a) The covariant E-scheme (the four-momenta of the two jets are summed up)

$$p_{(ij)} = p_i + p_j$$

(b) The p_t weighted scheme

$$\begin{aligned} p_{t(ij)} &= p_{ti} + p_{tj}, \\ \eta_{(ij)} &= (p_{ti}\eta_i + p_{tj}\eta_j)/p_{t(ij)}, \\ \phi_{(ij)} &= (p_{ti}\phi_i + p_{tj}\phi_j)/p_{t(ij)}. \end{aligned}$$

5. If $d_n = d_B$ then merge jet i with the "beam jets" and the jet i is removed from the list of jets used in further part of the procedure

6. Decide if another iteration is needed. If yes go back to step 2, otherwise the algorithm can stop. This stopping procedure will be defined later in different ways.

Remarks

There are three different ways for stopping procedures:

- **Inclusive way**: jets with d_n=d_{iB} are not merged with the beam jets, but instead are considerd as output of the algorithm. Procedure is then stopped when all the jets are labelled as beam jets. As a result we have a list of jets ordered in transverse momentum.
- **NSTOP way**: Stopping when n non-beam jets are left.
- d_{cut} **way**: Stopping when $d_n > d_{cut}$ The cutoff d_{cut} is dependent by the physics process we are considering, i.e. by the typical p_t of the jets involved in analysis.

2.5 K_T Algorithm (version Montréal)

The algorithm is based on [8, 9], as proposed by Catani *et al.* [10]. It has essentially the same features as the algorithm described in the section 2.4. The algorithm uses two parameters: R_{cut} and E_T^{cut}.

Algorithm description

1. Initially, each cell is considered a protojet.

2. For every possible pair i and j of protojets, one defines their "closeness" d_{ij}:

$$d_{ij} = \min(E_{Ti}^2, E_{Tj}^2)\left((\Delta\eta_{ij})^2 + (\Delta\phi_{ij})^2\right)$$

3. For every protojet i, one defines d_{ib}, the "closeness" to the beam direction

$$d_{ib} = E_{Ti}^2 R_{\rm cut}^2$$

where $R_{\rm cut}$ is a parameter.

4. One defines d_{min} to be the smallest of all d_{ib} and d_{ij}'s.

5. If d_{min} is a d_{ij}, protojets i and j are merged into a new protojet k according to

$$\begin{aligned} E_{Tk} &= E_{Ti} + E_{Tj} \\ \eta_k &= (\eta_i E_{Ti} + \eta_j E_{Tj})/E_{Tk} \\ \phi_k &= (\phi_i E_{Ti} + \phi_j E_{Tj})/E_{Tk} \end{aligned}$$

and protojets i and j are removed from the list.

6. If d_{min} is a d_{ib}, protojet i is not mergeable. It is added to the list of clusters and removed from the list of protojets.

7. If there are protojets left, go back to step 2

8. A cut on the minimum E_T of the clusters is then applied: only clusters having $E_T > E_T^{\rm cut}$ are kept.

Remarks

- The algorithm produces clusters where all opening angles within each one are $< R_{\rm cut}$ and all opening angles between clusters are $> R_{\rm cut}$
- The parameter $R_{\rm cut}$ must be roughly 1.35 times larger than the R parameter of the cone algorithm, if one wants to have the same number of jets on output [8] (This has been verified at $\sqrt{s} = 1.8$ TeV, but not at LHC energies).
- Note that this subroutine does *not* fill the array of seed cells, as there is no well defined seed cell in this algorithm.
- The clusters are produced initially in increasing order of E_T with every cell which is hit assigned to a cluster, but after the E_T cut the subroutine reverses the order to decreasing E_T and leaves some cells unassigned to clusters.
- A mapping is made of which cell belongs to which cluster in the array `iuseC`, and the number of cells associated with each cluster is kept in the array `multJ`.
- The speed of this algorithm is inversely proportional to the cube of the number of initial protojets.

2.6 Sliding Window Algorithm

The Sliding Window Algorithm is the algorithm available as the default option in ATRECON.

2.7 Mulguisin Algorithm

Mulguisin Algorithm was first introduced in the ATLAS Jet/ETmiss meeting held on Nov. 25th, 1997. The algorithm is neither a varient of the conventional cone algorithm nor a varient of the Durham algorithm. The algorithm has some improved features like jet direction optimization and optimization of jet size.

Algorithm description

1. Find the maximum E_T cell and define it as the first cluster. Set the initial cluster size equal to the resolution (R_0).

2. Find the cell with the next biggest E_T.

3. Calculate the distance between the cell and the nearest cluster.

   If the distance is smaller than the cluster size, *i.e.* inside the cluster region, combine the cell with the cluster. After the clustering one can recalculate the cluster center by weighting the cluster and the cell with E_T, and one can also set the cluster size equal to the real jet dispersion, *i.e.*, distance between the cluster center and the farthest cell.

   If the distance is bigger than the cluster size, *i.e.* outside the cluster region, define the cell as a new cluster and set the initial cluster size equal to R_0.

4. Repeat the step 2) until there is no cell left.

Remarks

The algorithm can behave like three different algorithms by switching on/off the recalculation of cluster center and the resetting of cluster size.

- Mode 0: The cluster seed is fixed and cluster size is fixed as R_0 (behaves like the conventional cone algorithm), *i.e.*,

$$\begin{aligned} E_{T,k+1} &= E_{T,k}^{clus} + E_{T,k}^{cell} \\ \eta_{k+1}^{clus} &= \eta_k^{clus} \\ \phi_{k+1}^{clus} &= \phi_k^{clus} \\ R_{k+1}^{clus} &= R_k, \end{aligned}$$

  where k denotes the clustering step (not cluster number).

- Mode 1: The cluster direction is recalculated for each clustering step, but the cluster size is fixed as R_0, *i.e.*,

$$\begin{aligned} E_{T,k+1} &= E_{T,k}^{clus} + E_{T,k}^{cell} \\ \eta_{k+1}^{clus} &= (E_{T,k}^{clus}\eta_k^{clus} + E_{T,k}^{cell}\eta_k^{cell})/(E_{T,k}^{clus} + E_{T,k}^{cell}) \\ \phi_{k+1}^{clus} &= (E_{T,k}^{clus}\phi_k^{clus} + E_{T,k}^{cell}\phi_k^{cell})/(E_{T,k}^{clus} + E_{T,k}^{cell}) \\ R_{k+1}^{clus} &= R_k, \end{aligned}$$

- Mode 2: The cluster direction is recalculated for each clustering step and the cluster size is reset to the distance between the cluster center and the farthest cell in the cluster if the distance is larger that the resolution, *i.e.*,

$$\begin{aligned} E_{T,k+1} &= E_{T,k}^{clus} + E_{T,k}^{cell} \\ \eta_{k+1}^{clus} &= (E_{T,k}^{clus}\eta_k^{clus} + E_{T,k}^{cell}\eta_k^{cell})/(E_{T,k}^{clus} + E_{T,k}^{cell}) \\ \phi_{k+1}^{clus} &= (E_{T,k}^{clus}\phi_k^{clus} + E_{T,k}^{cell}\phi_k^{cell})/(E_{T,k}^{clus} + E_{T,k}^{cell}) \\ R_{k+1}^{clus} &= MAX(R_k, \delta), \end{aligned}$$

2.8 Nearest Neighbor Algorithm (CDF)

Algorithm description

1. A seed tower above a threshold is used to start.
2. All contiguous towers around the seed one with energies above a shoulder threshold are found.
3. The eight nearest towers to the seed tower are examined, and if their E_T is less than $f \times E_T$(seed) they are merged (ie: the energy is added into the cluster). The f parameter, or *parent-daughter ratio*, is a fixed fraction.
4. The merged tower will be themselves used as seeds to search for other contiguous towers, as described above. Jets are therefore defined around local maxima in the transverse energy
5. If a tower is not merged in this process, it will be the seed for another cluster

Remarks

The algorithm stops when all the towers above threshold are examined.

2.9 Pairwise Merging Algorithm (CDF)

Algorithm description

1. Starts with the clusters from the nearest neighbor algorithm
2. Tries to pair them, looking at which pair are the closest in the $\eta - \phi$ space

Remarks

The algorithm stops when all the towers within some minimum distance are merged together.

3 Jet Finder Library

3.1 Organization of Input and Output

Input COMMON

The actual inputs of each of the jet finders described above are the P_T of each cell. Since many cells are not hit, or have energy deposition below the P_T cut, it is advantageous in terms of CPU time to define three one-dimensional arrays for E_T, η and ϕ of the cells hit, rather than loop over all the cells.

```
INTEGER   ImaxC,InumC
PARAMETER (ImaxC=10000)
INTEGER   IuseC(ImaxC),IdumC(ImaxC)
REAL      RentC(ImaxC),RetaC(ImaxC),RphiC(ImaxC)
COMMON /JF_CELL/ InumC,IuseC,RentC,RetaC,RphiC
```

ImaxC	Maximum number of calorimeter cells
InumC	Number of fired cells
RentC(i)	ET of the i'th cell
RetaC(i)	Eta of the cell center
RphiC(i)	Phi of the cell center

Output

The most general outputs for any kind of clustering algorithm may be the list of jets containing E_T, η, and ϕ. These informations are saved in the COMMON /JF_CLUS/.

The COMMON /JF_SEED/ is intended for keeping the jet seed information for the case of cone-like algorithm. It can be used for saving the information of the maximum E_T cell of a cluster in the case of K_T algorithm.

```
INTEGER   ImaxJ,InumJ
PARAMETER (ImaxJ=1000)
INTEGER   IuseJ(ImaxJ),IdumJ(ImaxJ)
REAL      RentJ(ImaxJ),RetaJ(ImaxJ),RphiJ(ImaxJ),RdumJ(ImaxJ)
REAL      RentS(ImaxJ),RetaS(ImaxJ),RphiS(ImaxJ),RdumS(ImaxJ)
COMMON /JF_CLUS/ InumJ,RentJ,RetaJ,RphiJ,RdumJ
COMMON /JF_SEED/       RentS,RetaS,RphiS,RdumS
```

ImaxJ	Maximum number of jets
InumJ	Number of clusters found
RentJ(i)	ET of the i'th cluster (sum of ET of the cells included)
RetaJ(i)	Eta of the cluster (weighted by ET of the cells included)
RphiJ(i)	Phi of the cluster (weighted by ET of the cells included)
RdumJ(i)	dummy
RentS(i)	ET of the seed for the i'th cluster
RetaS(i)	Eta of the seed
RphiS(i)	Phi of the seed
RdumS(i)	dummy

Additional COMMON blocks

In order not to tie the jet finder library to a particular geometry, the pseudorapidity coverage, η_{min} to η_{max}, and the azimuthal angle coverage, ϕ_{min} to ϕ_{max}, are set in a common block. Though the azimuthal angle coverage are normally full 2π, the latter is necessary to pass the ϕ angle definition[2]. , *i.e.*, whether $0 \to 2\pi$ or $-\pi \to \pi$

The calorimeter is divided into cells of $\delta\eta \times \delta\phi$ grid with binning N_η and N_ϕ. For example, in ATLAS, 0.1×0.1 cell is often used, which corresponds 100 bins in case of pseudo-rapidity coverage of $|\eta| < 5.0$. Either the number of bins or the cell size is needed to be used for some calculations, for example, translation a bin number to η and ϕ or *vice versa*.

The above calorimeter variables are grouped and form a FORTRAN COMMON, /JF_INFO/.

```
INTEGER   IbinEta,IbinPhi
REAL      RminEta,RmaxEta,RminPhi,RmaxPhi
COMMON /JF_INFO/ IbinEta,RminEta,RmaxEta,IbinPhi,RminPhi,RmaxPhi
```

In order to reduce the pile-up effects and also to reduce CPU time consumation, it is usual that a minimum P_T cut of a cell is applied before running a jet algorithm ($P_{T,min}^{cell}$). Certain jet algorithms use a P_T threshold cut for the jet initiator ($P_{T,min}^{seed}$). Furthermore, a minimum cluster transverse momentum $P_{T,min}^{cluster}$ is often required. These jet definition variables form a FORTRAN COMMON, /JF_PARA/.

```
REAL
|  RminETcell,RminETseed,RminETclus,
|  Rdum0,Rdum1,Rdum2,Rdum3
COMMON /JF_PARA/
|  RminETcell,RminETseed,RminETclus,
|  Rdum0,Rdum1,Rdum2,Rdum3
```

Subroutine Arguments

Parameters related to a specific algorithm are to be interfaced by the subroutine arguments. For example, the cone size of the cone jet algorithm is passed by the FORTRAN subroutine argument.

3.2 Usage

Some of algorithms described above have been implemented in the Jet Finder Library. The algorithms have been collected and modified in order to be used transparently with the unique IO scheme described in the previous section.

To use a jet finder routine, one needs to do the following steps:

- Link with the JetFinder library currently kept in the following directory when you make your executable.

[2]For example, ATRECON uses $0 \to 2\pi$ but ATLFAST uses $-\pi \to \pi$

`~parkic/public/jetfinder/libjetfinder.a`

- Fill /JF_CELL/.
- Call a Jet Finder program with proper arguments as described in the followings.
- The results are saved in /JF_CLUS/.

Current available algorithms and their program names are

`CONE_JET`

Program:
Conventional Cone-Based Algorithm

Usage:

```
CALL CONE_JET(Rcone,ETseedcut,ETcluscut)
```

Arguments:

Rcone	(REAL)	Cone size
ETseedcut	(REAL)	Minimum ET for a jet seed
ETcluscut	(REAL)	Minimum ET to be a jet

`MULGUISIN`

Program:
Mulguisin algorithm

Usage:

```
CALL MULGUISIN(Mode,Resolution,ETseedcut,ETcluscut)
```

Arguments:

Mode	(INTEGER)	Running Mode
Resolution	(REAL)	Minimum distance to separate jets
ETseedcut	(REAL)	Minimum ET for a jet seed
ETcluscut	(REAL)	Minimum ET to be a jet

`KTCLUS`

Program:
K_T Algorithm - Pisa version

Usage:

```
CALL KTCLUS(Nstop,Conesize,Dcut,TowerCut,ETcell,ETclus)
```

Arguments:

Nstop	(INTEGER)	Minimu number of jets to be found
Conesize	(REAL)	Cone size
Dcut	(REAL)	
TowerCut	(REAL)	ET threshold for a cell
ETcell	(REAL)	at least a cell must have ET larger than this.
ETclus	(REAL)	Minimum ET to be a jet

`KTJET`

Program:
K_T Algorithm - Montreal version

Usage:

```
CALL KTJET(Rcut,ETcut)
```

Arguments:

Rcut	(REAL)	R parameter
ETcut	(REAL)	Minimum ET to be defined as a jet

CDFJET

Program:
CDF Fixed Cone Algorithm

Usage:

```
CALL CDF(Rjet,PTcut,ETstop,ECcut,ECcut_pr,CellFRcut)
```

Arguments:

Rjet	(REAL)	Cone size
PTcut	(REAL)	Minimum ET for a jet seed
ETstop	(REAL)	Minimum ET to be a jet
ECcut	(REAL)	
ECcut_pr	(REAL)	
CellFRcut	(REAL)	

4 PAW based Event Display: `jf_look.exe`

We developed a utility program for calorimeter display. This program was originally designed for the purpose of debugging the Jet Finder routine in PAW session, but it has now become a fully stand alone program. To use it one needs to create a CWN (Colurm Wise Ntuple) by using of the subroutine `jf_hbook` provided together with Jet Finder Library.

5 Comments

Some jet algorithms which are not available in the current version are being tested and will be implemented in the next version.

All existing jet finding algorithms are supposed to be used with the combined calorimeter after calibration, but in principle they can be run with detector cell by cell information. This feature is not possible if an algorithm works with 2-dimensional $\eta-\phi$ map for example the original CDF Cone algorithm.

References

[1] "ATLFAST 1.0: A package for particle-level analysis",
Elzbieta Richter-Was, Daniel Froidevaux, and Luc Poggioli,
ATLAS-PHYS-NO-079

[2] "DICE",
ATLAS-SOFT-16

[3] "ATRECON",
ATLAS-SOFT-XX

[4] "Proceedings of the *Workshop on Calorimetry for the Superconducting Supercollider*", Tuscaloosa, Alabama (1989), talk from J. Huth,

[5] F. Abe et al., Phys. Rev. **D**, vol 45 (1992), p. 1448.

[6] H. Seymour, Zeit. fur Phys. C62 127 (1993)

[7] "Particle Level Study of $W \to q\bar{q}$ and Jet Algorithms", S. Kuhlmann, S. Bettellei, A. Bocci, D. Costanzo, R. Paoletti, S. Lami, ATLAS-PHYS-NO-106

[8] S. D. Ellis and D. E. Soper, Phys. Rev. **D48** 1993 3160

[9] M. H. Seymour, RAL-97-026, hep-ph/9707338

[10] S. Catani, Yu. L. Dokshitzer, M. H. Seymour, and B. R. Webber, Nucl. Phys. **B406** (1993) 187

[11] "A new jet finder : Mulguisin Algorithm", I.C. Park, ATLAS-PHYS note in preparation

[12] "PYTHIA 5.7 and JETSET 7.4 Physics and Manual", Torbjorn Sjostrand, LU TP 95-20, August 1995

A How to run JetFinder with ATLFAST

From ATLFAST version 2.0, JetFinder library (v1.0) became a part of the ATLFAST package. Just install ATLFAST version 2.0 and turn on `LPAR(13)`. You need to modify the data card `jetfinder.dat` according to you choice of jet algorithm. Here is an example data card to run K_T algorithm.

Example card to run KTCLUS

```
* Jet Finder Switch
       2         ---  IdxJF  ... use KTCLUS
* Cone-Based Jet Algorithm
     0.7         ---  Cone size
     2.0         ---  Seed ET cut
    20.0         ---  Cluster ET cut
* Mulguisin Algorithm
       2         ---  Run Mode
     0.3         ---  Resolution
     2.0         ---  Seed ET cut
    20.0         ---  Cluster ET cut
* KT Algorithm (version PISA) <--- will be used
       0         ---  Nstop
     0.7         ---  Cone size
 10000.0         ---  Dcut value
     1.0         ---  TowCut value
     2.0         ---  ET threshold of a maximum ET cell
    10.0         ---  ET threshold of a cluster
* Sliding Window Algorithm
* KT Algorithm (version Monreal)
     0.7         ---  R parameter
    20.0         ---  Minimum ET to be defined as a jet
```

B How to run JetFinder with ATRECON

From 98_2, `jetfinder.cmz` will be provided. User just need to provide the following card in the job directory. If the card does not exist the default algorithm, "Sliding Window", will be used.

Example card to run KTJET

```
*------
*------ ATLAS Jet Finder Library controlling card
*------
*.
*. This card file generates ZEBRA banks for control ATLAS JetFinder
*. The bank JFAL is the actual switch for the algorithm
*.
*.      0 - the conventional cone-based algorithm
*.      1 - Mulguisin algorithm
*.      2 - K_T algorithm of Pisa
*.      3 - Sliding Window Algorithm (default)
*.      4 - K_T algorithm of Montreal
*.      5 - Topological Clustering of CDF
*.
*PRINT
*DO JFAL         #. Algorithm switch
    4            #. select K_T algorithm of Montreal

*DO JF0P         #. Parameters for Conventional cone-based algorithm
    0.4          #. Cone size

*DO JF1P         #. Parameters for Mulguisin
    2            #. Mulguisin mode
    0.3          #. Mulguisin initial cone size
    2.0          #. Minimum jet seed ET
   20.0          #. Minimum required ET to be a jet

*DO JF2P         #. Parameters for KTCLUS
    4            #. Nstop
    0.7          #. Cone size
  100.0          #. Dcut
    2.0          #. TowCut
    2.0          #. Minimum jet seed ET
   20.0          #. ET threshold of a jet

*DO JF4P         #. Parameters for KTJET
    0.7          #. R parameter
   20.0          #. Minimum cluster ET to be a jet
```