Atlfast Requirements Document

P.Clarke H. Phillips P. Sherwood

April 9, 2001

Version: -0.6
Status: Draft
v.-0.3: Complies with Athena-Atlfast User Requirements Report recommendations
v.-0.4: Includes AtlfastF code fragments.
v.-0.5: After UCL phone meeting 2/2/01.
v.-0.6: Incorporate table for correspondence with original User Requirements Reviewed document.

Contents

1 Introduction

1.1 Scope of the Requirements Document

This document specifies the requirements for the Atlas fast simulation program (Athena-Atlfast). It describes the purpose of the program, the environment it runs in, the input and the output objects.

The document is intended for use during the design, implementation and test phases of the program. It is also intended to provide a base for discussions on the role that Athena-Atlfast is to play among the Atlas physics, simulation and framework communities.

This document supercedes the original User Requirements document presented to the Atlfast User Requirements Review committee. Appendix C maps the user requirements of the original document into the functional requirements of this documents.

1.2 Organisation

In the remainder of this section, acronym definitions and references are given. The purpose of the program is described.

Section 2 describes the relation to the framework in which Athena-Atlfast is to run, and an account of how the fast simulation is performed.

Section 3 gives details on the quantities to be calculated, the configuration of the program, the output of the program and the time and space performance requirements.

Appendix A gives the parts of AtlfastF that specify the default functionality of parts of Athena-Atlfast.

Appendix B gives a list of items desirable for future versions of Athena-Atlfast.

Appendix C shows the corrispondence between the user requirements of the Original User Requirements document and the functional requirements expressed in Section 3 of this document.

1.3 Acronym Expansion and Definitions

Algorithm A self-contained process that can be scheduled by ATHENA

Athena The ATLAS offline software framework

Athena-Atlfast This version of Atlfast. An OO program written in C++.

Atlfast Generic term for the ATLAS smear-mode Monte Carlo.

AtlfastF Fortran version of Atlfast, used to produce the TDR

AtlfastTemp A rewrite of Atlfast in C++ by E. Richter-Was.

AtlfastSTL A rewrite of AtlfastF in C++ with heavy use of the STL library by E. Richter-Was.

HepMC An environment for storing and manipulating the output of physics even generators.

OO Object Oriented

Pythia A program used to generate a physics event. Outputs a list of four-vectors and other information about virtual and real particles of the event.

TDR ATLAS Detector and Physics Performance Technical Design Report LHCC 99-14/15.

TES ATHENA transient event store.

1.4 References

- http://atlasinfo.cern.ch/Atlas/GROUPS/PHYSICS/HIGGS/Atlfast.html
- http://www.hep.ucl.ac.uk/atlas/atlfast/
- AtlfastCode/doc in CVS repository
- Atlfast 2.0 - ATLAS Internal Note ATL-PHYS-98-131
- AtlfastTemp Package - ATLAS Internal Note ATL-COM-PHYS-2000-023
- HepMC - User Manual Version 1.0 - http://cern.ch/HepMC

1.5 Overview

Athena-Atlfast is an OO fast simulation program for performing reconstruction algorithms on four-vectors representing particles produced by event generator programs.

The detector response to the passage of the particles is simulated.

A number of reconstructed quantities are calculated using the detector response for use in subsequent physics analyses. These quantities include lists of jets, isolated electrons, isolated muons and global event quantities.

A physics analysis can be performed directly using these lists. Alternatively a PAW n-tuple containing information on the the reconstructed quantities is produced (this tuple is part of the ATLAS Combined Tuple), to be analysed at a later time.

2 General Description

2.1 Perspective

Athena-Atlfast is to run in the Atlas Athena Framework. ATHENA will provide all external connections.

It will process events generated by event generator programs (such as Pythia, Isajet or Herwig). The generators will be interfaced to the HepMC classes. Athena-Atlfast will access the generator data by reading HepMC objects.

Athena-Atlfast will use a number of services provided by ATHENA. These include:

- steering services to schedule the Athena-Atlfast algorithms,
- histogram services.
- ntuple Services
- message logging services.
- other Persistency Services.

No services provided at present, or in the short-term future will be duplicated in Athena-Atlfast.

2.2 Activity

Athena-Atlfast will work in the ATHENA framework. The framework provides an initialisation phase, an event loop and a finalisation phase.

During the event loop, a list of four-vectors which result from the simulation of the collision of two or more beam particles (a physics event) is made available to Athena-Atlfast via the HepMC interface. Each of these four-vectors represents a particle, either one which is virtual (an intermediate step in the calculation of detectable particles), short lived (one that is in principle detectable, but which decays before interacting with the detector), or long-long lived (one which travels to the detector, or would do so if there were no magnetic field). Depending on their nature, long-lived particles may or may-not be detectable.

The four-vector lists can be generated by a number of different event generators, such as Pythia, Herwig or Isajet. Each of which has its own native output format, and is interfaced to Athena-Atlfast via HepMC.

Alternatively, the generators may have been run at an earlier time, and the events are read in from a persistent store.

Single particle generators will also be used, either as a debugging tool for Athena-Atlfast code, or for detector simulation studies.

Athena-Atlfast processes the four-vectors, to simulate the detector response, and to calculate reconstructed physics quantities.

2.2.1 Detector Simulation

The interaction of particles in the detector is simulated. The detector response is output, and is later used by reconstruction algorithms.

An example of this is jet-finding: the energy of a four-vector representing a final state particle is assigned to a calorimeter cell. For the present, the model is simply that the energy of the particle striking a cell is assigned to that cell, where a cell is simply an element of a regular $\eta - \phi$ grid. Future modelling of the energy process will be more complex, and will include modelling of longitudinal shower development, and spreading of energy across cell boundaries.

The jet-finder algorithms combines cells to make clusters, which in turn are combined to form jets.

2.2.2 Simultaneous Detector Simulation and Reconstruction

Here the simulation of the detector and the reconstruction is performed in a single step.

An example is the the creation of reconstructed particles: the reconstructed particle has a momentum obtained by smearing the original four-momentum according to an appropriate resolution function. The detector response (hits in the tracker) and the reconstruction algorithm (the tracking program) are reduced to the action of smearing a four-vector.

2.2.3 Reconstruction Algorithms

Reconstruction algorithms use the detector response to calculate quantities to be used in physics analysis programs. These quantities include the jet activity in the event and track isolation. Global event quantities such as the missing momentum vector, and event shape variables are also calculated.

The program is written in a modular, structured manner that allows easy replacement or extension of reconstruction algorithms and the detector simulation in a manner where the supplier of the new functionality needs to know little of the internal structure of Athena-Atlfast. This requirement refers to changes in calculation strategy, rather than parameter passing to a strategy already in place. Examples include a change of smearing function, or a change of jet-finder algorithm.

Athena-Atlfast will include the detector response and reconstruction algorithms developed for AtlfastF.

The program is to work within the ATHENA framework, which is also the framework used by the Generator Interface group. This allows the use of different generators as the Generator Interface group introduces them into ATHENA.

By using the framework, Athena-Atlfast can also be used as a "trigger" for running other more complex algorithms that conform to the framework.

The reconstructed quantities, which are the entities such as jets, isolated charged leptons and missing transverse momentum can be used directly by analysis programs running within Athena-Atlfast. Alternative ways of using these quantities is to store the information in ntuples for later analysis using PAW or ROOT.

Other long term storage methods for the reconstructed quantities are available through the ATHENA persistency services.

2.3 User Characteristics

The intended users for Athena-Atlfast are physicists wishing to carry out a analysis of ATLAS data with a program that provides fast detector simulation.

If the physicist decides to generate events, rather than read events from a file, it is expected that she will understand the control options for the event generator.

The physicist will schedule the reconstruction algorithms, and control parameters for those algorithms. Default values will be provided so that no additional input is needed for the program to run.

Athena-Atlfast will be written in C++. The user will have the option of writing her analysis program in C++, or using the ntuples provided to perform an analysis outside of Atlfast++ using a PAW ntuple.

A user manual that describes all user settable values, how to access the reconstructed objects and the reconstruction classes as well a simple test job will be provided.

2.4 General Constraints

AtlfastRequirement 1 *Athena-Atlfast will run under version Red Hat release 6.1 (or later) of LINUX .*

AtlfastRequirement 2 *version 1.3.2 or later of the ATLAS framework of ATHENA*

AtlfastRequirement 3 *will be written in C++.*

AtlfastRequirement 4 *will adhere to the ATLAS coding conventions.*

AtlfastRequirement 5 *will adhere to the ATLAS agreed interfaces.*

AtlfastRequirement 6 *four-vector input will use the HEPMC interface.*

3 Specific Requirements

3.1 Functional Requirements

The following subsections give the details necessary to create the design specifications of Athena-Atlfast.

The modular design of Athena-Atlfast is intended to make changes to existing algorithms straight forward. The initial set of these algorithms will be those developed in AtlfastF. The detailed specification of this section is for the AtlfastF algorithms (see appendix A).

3.1.1 Particles

The HepMC data structure will be scanned for the presence of a a particle identified as an elecron, positron, muon, anti-muon or photon.

If the four-vector passes user-definable acceptance cuts (the acceptance is a boolean function of the four-vector), a reconstructed particle is made by smearing the generator four-vector with a smearing function. The smearing is user-definable, and may depend on the luminosity being simulated, and the isolation of the track.

The default acceptance (given by AtlfastF) is:

- electrons - p_t>0.0
- photons - p_t>0.0

- muons - p_t>0.5

The following information about the particles will be available:

- The smeared four-vector
- PDG id
- The generator history (includes ancestor and children four-vector information). This is a user requirement on an Athena-Atlfast quantity, but will be satisfied by general history storage mechanisms of ATHENA. Athena-Atlfast particles must be able to exist even in the absence of the generator particles, *eg* if they are retrieved from a persistent store, and no generator information has been kept.

AtlfastRequirement 7 *Electrons, muons and photons will be reconstructed as Reconstructed Particles*

AtlfastRequirement 8 *Reconstructed Particles will be reconstructed in a user defianble p_t, η, ϕ region, with default limits given by the ATlfastTemp default values*

AtlfastRequirement 9 *The Reconstructed Particle will have a PDG id code,*

AtlfastRequirement 10 *The following functions of the four-vector will be available: p_t, η, ϕ, p_x, p_y, p_z.*

AtlfastRequirement 11 *A list of each type of particle will be made available to the user each event.*

AtlfastRequirement 12 *The history (ie generator four-vector for the particle) will be traceable. Athena- Atlfast will be able to function in the absence of history information.*

3.1.2 Tracks

The generator four-vectors and the point of origin of the track will be used to calculate the track helix parameters.

AtlfastRequirement 13 *The following information about tracks will be available:*

- *five helix parameters* $(\phi_0, \cot\theta, \frac{charge}{p_t}, z_0, r_0)$

- *specific attributes:* η*,* p_t*,* d_0*,* z_0
- *covariance matrix*
- *the generator four-vector used to calculate the track. Athena-Atlfast tracks must be able to exist even in the absence of the generator four-vectors.*

AtlfastRequirement 14 *A list of tracks will be made available to the user each event.*

3.1.3 Struck Cells

Particles, represented by generator four-vectors, deposit energy in the calorimeter. The cell struck by the four-vector is found by projecting the vector from its point of origin through the magnetic field to the calorimeter surface.

The default algorithm will set the cell energies to zero at the beginning of the event. For each particle hitting a cell, an energy will be added to the cell energy, The amount of energy will be specifiable either as a fraction of the total energy, or as a fixed quantity (to handle minimum ionising particles).

The default behaviour, given by AtlfastF, is that all particles give all their energy to the cell, except muons and neutrinos, which give none.

AtlfastRequirement 15 *Atlfast will provide a cells which represent Calorimeter cells.*

AtlfastRequirement 16 *The Cells will be laid out in a uniform* $\eta-\phi$ *rectangular grid in three regions, to represent the forward and backward endcaps, and the barrel region of the calorimeter. The granularity of the grid will be user setable with default values* $\Delta\eta=0.1$ *in the barrel, and* $\Delta\eta=0.2$ *in the endcaps. The default* ϕ *granularities will be such that there are 64 cells in* ϕ *in the barrel region to give* 2π *coverage, and 32 cells in the endcaps for the same* ϕ *coverage.*

AtlfastRequirement 17 *Cells with provide the following functionality.*

- *geometry information (*η*,* ϕ*,* $\Delta\eta$*,* $\Delta\phi$*) for the cell.*
- *the energy deposited in the cell*
- *the list of particles that strike the cell Athena-Atlfast cells must be able to exist even in the absence of the generator four-vectors.*

AtlfastRequirement 18 *A list of struck cells will be made available to the user each event. In compiling this list, it will be possible to select the cells on the basis of geometry and energy.*

Appendix B gives some possible future requirements.

3.1.4 Isolation

Cells will be used to test the isolation of electrons, muons and photons. Isolation is a measure of the energy deposited in a cone centered on the the generator four-vector for the particle.

The explicit default isolation algorithms are given in AtlfastF (see Appendix A), and is briefly described here.

If the reconstructed particle is of type that deposits energy in the calorimeter (electron, positron or photon), and if there is one or more clusters lying inside a cone centered on the direction of the four-vector of the particle, the cluster with smallest ΔR is associated with the particle.

Cluster isolation is tested. If there are clusters lying within a cone centered on the direction of the four-vector of the particle, the sum of the p_t of these clusters and cells, with any clusters associated to a Reconstructed Particle excluded, gives the isolation sum.

If the isolation sum exceeds a user settable threshold, the particle is isolated. Otherwise, the particle is unisolated.

Cell isolation is then tested. The p_t sum is calculated for cells lying within a cone centered on the direction of the four-vector of the particle. If the particle is of type that deposits in the calorimeter, the pt of the particle is subtracted from the sum. If the sum exceeds a user settable threshold, the particle is isolated. Otherwise, the particle is unisolated.

The correct smear function to be applied to go from generator four-vectors to particles depends on the isolation of the particle.

AtlfastRequirement 19 *Reconstructed Particle isolation will be carried out using the above algorithm.*

AtlfastRequirement 20 *The check on whether the type of particle deposits energy in the calorimeter will be gauranteed to be the same as the check performed when the cell p_t is calulated, to ensure that inivisible particles are handled consistantly.*

AtlfastRequirement 21 *The user will be presented with a lists of isolated electrons, muons and photons.*

3.1.5 Jets

In general, jet-finders build jets from tracks and cells.

The default jet-finding algorithm builds jets from cells not in clusters, clusters and muons. It does not use tracks.

The default algorithm is as in AtlfastF (see Appendix A).

Jets will be assigned a flavour tag.

The default algorithm, given by AtlfastF is as follows. Jets are assigned a flavour label if there is a nearby τ-meson or heavy quark. The default algorithm examines the generated particle list for the presence of a τ-meson or heavy quark within a fixed angle to the jet direction. Quarks are to be considered after they have completed gluon radiation, as this changes the quark direction. The angle between the labelling particle and the jet is flavour-dependent.

AtlfastRequirement 22 *Jet Construction will be carried out using the above algorithm.*

If there is no τ-meson or heavy quark close to the jet, the jet is given a u-quark label. If there is one labelling particle close to the jet, the jet is labelled by that particle. If there are more than one labelling particle close by, the labelling precedence (highest precedence first) is b-quark, c-quark and τ-meson.

AtlfastRequirement 23 *Jet tagging will be carried out using the above algorithm.*

AtlfastRequirement 24 *The following information about jets will be available:*

- *the four-vector.*
- *the components from which it is made (cells, particles, tracks).*
- *the generator history (includes ancestor and children four-vector information).*

AtlfastRequirement 25 *A list of jets will be made available to the user each event.*

3.1.6 Global Event Information

Information that is associated with the event as a whole will be made available to the user. Some of the this information is calculated from the four-vectors, and some is of a more administrative nature.

AtlfastRequirement 26 *The following global event data will be made available.*

- *event number*
- *run number*
- *generator(s) specification: name and version of the generator*
- *event weight*
- *measured missing* p_t*.*
- *true missing momentum (calculated from generated four-vectors, excluding those corresponding to particles that do not interact in the detector.)*
- *event shape variables: sphericity, acoplanarity.*

AtlfastRequirement 27 *The measured missing* p_t *will be given by the sum of the* p_t *of*

- *isolated Reconstructed Electrons*
- *isolated Reconstructed Photons*
- *all Reconstructed Muons*
- *Jets*
- *Clusters not used in Jet building*
- *Cells not used in clusters.*

In the missing p_t *calculation the cells will be smeared with the same smearer used to produce the jets.*

3.2 Configuration

3.2.1 Athena Framework

The steering will be done using the mechanisms provided by the ATHENA Framework. In general an ATHENA job will have steering that controls actions (such as event generation) that occurs chronologically before Athena-Atlfast is run, the Athena-Atlfast part of the job, and any programs that are to run after Athena-Atlfast. Steering would include the selection of which event generator(s) to run, which detector simulation and which reconstruction algorithm parameter values to use. Default steering options will be provided so that Athena-Atlfast will provide sensible physics output without requiring user interaction.

The rest of this subsection will be concerned with the steering of Athena-Atlfast only.

The current mechanism of carrying out this task is via the job options file. This mechanism has severe limitations in that the type of the parameters passed to from the jobOptions file to Athena-Atlfast currently is limited to certain native types (integer or double).

Another important limitation is that locations of algorithm inputs and outputs are maintained explicitly.

3.2.2 Functionality Configuration

AtlfastRequirement 28 *The following functionality can be chosen by the user:*

- *electron creation*
- *muon creation*
- *photon creation*
- *cell creation*
- *jet creation*
- *isolated electron creation*
- *isolated muon creation*
- *isolated photon creation*
- *standard Histograms*
- *standard Ntuples*

AtlfastRequirement 29 *It will possible to run more than one version of Athena-Atlfast in the same job.*

3.2.3 Parameter Configuration

Steering Parameter		*Default Value*
Luminosity	LHC luminosity (high/low)	1
ParticleType	electron/muon/photon	–
mcMinimumPt	acceptance cut on MCParticles	0.0
mcMaximumEta	acceptance cut on MCParticles	100.0
MinimumPt	acceptance cut on RecParticles	5.0
MaximumEta	acceptance cut on RecParticles	2.5
DoSmearing	flag for smearing	true
MCeventLocation	location in TES for input MCPart	–
OutputLocation	location in TES of RecPart	–
MuonSmearKey	flag for smearing function (one of four)	1
RandSeed	value for smearer	12345

Table 1: The following table shows default values which can be changed by the user

3.2.4 Configuration Inputs

AtlfastRequirement 30 *The configuration information will be supplied via the the ATHENA configuration service.*

3.2.5 Configuration Processing

AtlfastRequirement 31 *The configuration information will be used to determine which Athena-Atlfast algorithms to run.*

AtlfastRequirement 32 *The algorithms will access the parameters set in the configuration file.*

AtlfastRequirement 33 *The number of configuration files will be small. The user will be able to modiify user settable quantities in a single location.*

3.2.6 Configuration Outputs

The framework copies the configuration information to the log file.

AtlfastRequirement 34 *The different parts of Athena-Atlfast will print out a formatted version of the parameters to the log file.*

3.3 Athena-Atlfast Output

At the end of the Athena-Atlfast job, the following items will have been output.

- Log file with configuration information, and information from the initialisation, event-loop and finalisation phases from the Athena-Atlfast algorithms.
- Entities created during the event loop, and written to the persistent store
- HBOOK files of histograms and ntuples.

The standard ntuple is a PAW ntuple. The quantities to be stored in the ntuple can be found in the AtlfastF code (see Appendix A).

AtlfastRequirement 35 *Atlfast will output the ATlfast part of the Standard Ntuple*

A Appendix: AtlfastF code fragments referred to in this document

The following code fragments were obtained by a cvs checkout.

cvs checkout -d atlfast -r atlfast-00-02-04 offline/atlfast

The files cited below are to be found in the directory atlfast/atlfast/atlfast.F, unless otherwise stated. This path is relative to the directory in which the CVS co command was performed.

A.1 Electron Smearing

```
      REAL FUNCTION RESELE(KEYLUM,ENE,PT,ETA,PHI)
C     *******************************************
C     parametrizes electron  resolution for low and high luminosity
C     parametrization from L. Poggioli
C         KEYLUM=1   -- low  luminosity
C         KEYLUM=2   -- high luminosity
C
      IMPLICIT NONE
      INTEGER KEYLUM
      REAL ENE, PT, ETA, PHI
      REAL AA, BB
      REAl SIGPU, RPILUP, SIGPH, SIGPH1

 10           call rannor(aa,bb)
              sigph1 = aa*0.12/sqrt(ene)
              if(1.+sigph1.le..0) go to 10
              sigph=sigph1
              if(abs(eta).lt.1.4) then
 11           call rannor(aa,bb)
              sigph1 =  aa*0.245/PT
     #                  + bb*0.007
              if(1.+sigph1.le..0) go to 11
              else
 12           call rannor(aa,bb)
              sigph1 = aa* 0.306*((2.4-abs(eta))+0.228)/ene
     #                 + bb* 0.007
              if(1.+sigph1.le..0) go to 12
```

```
            endif
            sigph=sigph+sigph1
            IF(KEYLUM.EQ.2) THEN
             if(abs(eta).lt.0.6)rpilup=0.32
             if(abs(eta).gt.0.6.and.abs(eta).lt.1.4)rpilup=0.295
             if(abs(eta).gt.1.4)rpilup=0.27
 13          call rannor(aa,bb)
             sigpu=(aa*rpilup/pt)
             if(1.+sigpu.le..0) go to 13
             sigph=sigpu+sigph
            ENDIF
             if(1.+sigph.le..0) go to 10

         RESELE=SIGPH

      END
```

A.2 Muon Smearing

```
      REAL FUNCTION RESMUO(KEYMUO,KEYFUN,PT,ETA,PHI)
C     **************************************
C     parametrizes muon resolution for
C     KEYFUN=1 parametrization from C. Guyot
C     KEYFUN=2 parametrization from M.Virchaux & L.Chevalier
C         KEYMUO=1   -- muon system stand-alone
C         KEYMUO=2   -- only tracker
C         KEYMUO=3   -- combined
C
      IMPLICIT NONE
      INTEGER KEYMUO, KEYFUN
      REAL PT, ETA, PHI, SIGMA
      REAL RESOMU, AA, BB
      REAL SIGMU,SIGMUON,ATRAK,WMUON,SIGTRAK,WTRAK,SIGTR,CORR,WTOT

          IF(KEYMUO.EQ.1) THEN
 11         CALL RANNOR(AA,BB)
            SIGMA=RESOMU(KEYFUN,ABS(ETA),PHI,PT)/100.
            SIGMA=AA*SIGMA
            IF(1.+SIGMA.LE..0) GO TO 11
          ELSEIF(KEYMUO.EQ.2) THEN
```

```
 12         CALL RANNOR(AA,BB)
            ATRAK=0.0005
            ATRAK=ATRAK*(1.+ABS(ETA)**10/7000.)
            SIGMA=ATRAK*PT*AA + 0.012 * BB
            SIGMA=SIGMA*(1.+ABS(ETA)**10/7000.)
            IF(1.+SIGMA.LE..0) GO TO 12
          ELSEIF(KEYMUO.EQ.3) THEN
 13         CALL RANNOR(AA,BB)
            SIGMUON=RESOMU(KEYFUN,ABS(ETA),PHI,PT)/100.
            SIGMU=AA*SIGMUON
            IF(1.+SIGMU.LE..0) GO TO 13
 14         CALL RANNOR(AA,BB)
            ATRAK=0.0005
            ATRAK=ATRAK*(1.+ABS(ETA)**10/7000.)
            SIGTRAK=SQRT((ATRAK*PT)**2+0.012**2)
            SIGTR = ATRAK*PT*AA+0.012*BB
            IF(1.+SIGTR.LE..0) GO TO 14
            WMUON = 1.0/SIGMUON**2
            WTRAK = 1.0/SIGTRAK**2
            WTOT  = WMUON+WTRAK
            CORR  = (WMUON*(1.0+SIGMU)+WTRAK*(1.0+SIGTR))/WTOT
            SIGMA = CORR-1.0
          ELSEIF(KEYMUO.EQ.4) THEN
 15         CALL RANNOR(AA,BB)
            SIGMUON=RESOMU(KEYFUN,ABS(ETA),PHI,PT)/100.
            SIGMU=AA*SIGMUON
            IF(1.+SIGMU.LE..0) GO TO 15
 16         CALL RANNOR(AA,BB)
            ATRAK=0.0005
            ATRAK=ATRAK*(1.+ABS(ETA)**10/7000.)
            SIGTRAK=SQRT((ATRAK*PT)**2+0.012**2)
            SIGTR = ATRAK*PT*AA+0.012*BB
            IF(1.+SIGTR.LE..0) GO TO 16
            WMUON = 1.0/SIGMUON**2
            WTRAK = 1.0/SIGTRAK**2
            WTOT  = WMUON+WTRAK
ccc            CORR  = (WMUON*(1.0+SIGMU)+WTRAK*(1.0+SIGTR))/WTOT
            call norran(aa)
            CORR  = aa/sqrt(wtot)+1.0
            SIGMA = CORR-1.0
          ELSEIF(KEYMUO.EQ.5) THEN
```

```
17         CALL RANNOR(AA,BB)
          sigma=0.02
          SIGMA=AA*SIGMA
          IF(1.+SIGMA.LE..0) GO TO 17
        ELSEIF(KEYMUO.EQ.6) THEN
          sigmuon=0.02
          sigtrak=0.02
          WMUON = 1.0/SIGMUON**2
          WTRAK = 1.0/SIGTRAK**2
          WTOT  = WMUON+WTRAK
          call norran(aa)
          CORR  = aa/sqrt(wtot)+1.0
          SIGMA = CORR-1.0
        ENDIF
        RESMUO=SIGMA

      END
```

A.3 Photon Smearing

```
      REAL FUNCTION RESPHO(KEYLUM,ENE,PT,ETA,PHI)
C     ******************************************
C     parametrizes photon  resolution for low and high luminosity
C     parametrization from L. Poggioli and F. Gianotti
C         KEYLUM=1   -- low  luminosity
C         KEYLUM=2   -- high luminosity
C
      IMPLICIT NONE
      INTEGER KEYLUM
      REAL ENE, PT, ETA, PHI
      REAL AA, BB
      REAl SIGPU, RPILUP, SIGPH, SIGPH1

 10         call rannor(aa,bb)
            sigph1 = aa*0.10/sqrt(ene)
            if(1.+sigph1.le..0) go to 10
            sigph=sigph1
            if(abs(eta).lt.1.4) then
 11         call rannor(aa,bb)
            sigph1 =  aa*0.245/PT
```

```
    #                + bb*0.007
             if(1.+sigph1.le..0) go to 11
             else
 12          call rannor(aa,bb)
             sigph1 = aa* 0.306*((2.4-abs(eta))+0.228)/ene
    #               + bb* 0.007
             if(1.+sigph1.le..0) go to 12
             endif
             sigph=sigph+sigph1
             IF(KEYLUM.EQ.2) THEN
              if(abs(eta).lt.0.6)rpilup=0.32
              if(abs(eta).gt.0.6.and.abs(eta).lt.1.4)rpilup=0.295
              if(abs(eta).gt.1.4)rpilup=0.27
 13           call rannor(aa,bb)
              sigpu=(aa*rpilup/pt)
              if(1.+sigpu.le..0) go to 13
              sigph=sigpu+sigph
             ENDIF
              if(1.+sigph.le..0) go to 10

          RESPHO=SIGPH

      END

      REAL FUNCTION RESTHE(ENE,ETA,THETA)
C     *******************************************
C     parametrizes smearing in photon position  theta
C     parametrization from F. Gianotti
C
      IMPLICIT NONE
      REAL PI
      DATA PI /3.141592653/
      REAL ENE, ETA, THETA
      REAL AA, BB
      REAL SIGPH2

 12         call rannor(aa,bb)
            if(abs(eta).lt.0.8)sigph2 = aa*0.065/sqrt(ENE)
            if(abs(eta).ge.0.8.and.abs(eta).lt.1.4)
    #                         sigph2 = aa*0.050/sqrt(ENE)
            if(abs(eta).ge.1.4.and.abs(eta).lt.2.5)
```

```
#                        sigph2 = aa*0.040/sqrt(ENE)
       if(abs(eta).ge.2.5)sigph2 = 0.0
       if(abs(theta+sigph2).gt.PI) go to 12
       RESTHE=SIGPH2

      END
```

A.4 Standard Ntuple Filling

The following fragment was extracted from the file atlfast/atlfast/atlfastntup.F.

```
* $Id: AtlfastReq.tex,v 1.3 2001/04/08 17:20:46 sherwood Exp $
* $Name:  $
* #include <commons/config.h>
      SUBROUTINE ATLFNTUP(MODE)
C     ************************
*
      IMPLICIT NONE
      INTEGER MODE
#include <atlfast/maknout.inc>
#include <atlfast/atlfevent.inc>
#include <atlfast/process.inc>
#include <atlfast/cluster.inc>
#include <atlfast/jetall.inc>
#include <atlfast/ptmiss.inc>
#include <atlfast/isoele.inc>
#include <atlfast/isomuo.inc>
#include <atlfast/noisomuo.inc>
#include <atlfast/isophot.inc>
#include <atlfast/miscelaus.inc>
#include <atlfast/triger.inc>
#include <atlfast/tracks.inc>
#include <atlfast/ptracks.inc>
#include <atlfast/atlfinfo.inc>
#include <atlfast/pmissing.inc>
#include <atlfast/pleptons.inc>
#include <atlfast/pphotons.inc>
#include <atlfast/ppjets.inc>
#include <atlfast/pmuxs.inc>
#include <atlfast/phistory.inc>
```

```
#include <atlfast/bphysics.inc>
#include <atlfast/ptrigger.inc>
*
* internal variables
      INTEGER KC
      INTEGER KDUM, KTRGDUM, KLEP, NJETB, NJETC, NPAR
      INTEGER IDUM, ILEP
      REAL ETMAX, IMAX
      REAL  PLEP, PDUM
      DIMENSION PLEP(100,5),KLEP(100,5)
      DIMENSION PDUM(100,5),KDUM(100,5),KTRGDUM(100)
      INTEGER I, II
      INTEGER ISTAT,ICYCLE
*
* initialisation and booking
      IF(MODE.EQ.-1) THEN
*
        CALL HCDIR('//PAWC',' ')
        CALL HROPEN(50,'SLUGRZ','ATLFAST.NTUP','N',1024,ISTAT)
        IF(ISTAT.NE.0) print *,' Error from HROPEN!!! ISTAT=',ISTAT
*
        CALL HBNT  (3333,'ATLFAST','D')
        CALL HBNAME(3333,'ATLFINFO',ISUB,'ISUB,'//
     +                  'NEL,NMU,NMUX,NPH,JETB,JETC,JETL,'//
     +                  'CIRCJ, CIRCE, THRUST, OBLAT')
        CALL HBNAME(3333,'PMISSING',PMISS,'PMISS(2)')
        CALL HBNAME(3333,'PLEPTONS',NLEP,'NLEP[0,12],KFLEP(NLEP),'//
     +                  'KTRGLEP(NLEP),'//
     +                  'PXLEP(NLEP),PYLEP(NLEP),PZLEP(NLEP),'//
     +                  'EELEP(NLEP)')
        CALL HBNAME(3333,'PPHOTONS',NPHO,'NPHO[0,12],KFPHO(NPHO),'//
     +                  'PXPHO(NPHO),PYPHO(NPHO),PZPHO(NPHO),'//
     +                  'EEPHO(NPHO)')
        CALL HBNAME(3333,'PPJETS',NJETA,'NJETA[0,20],KFJET(NJETA),'//
     +                  'PXJET(NJETA),PYJET(NJETA),PZJET(NJETA),'//
     +                  'EEJET(NJETA)')
        CALL HBNAME(3333,'PMUXS',NONMUX,'NONMUX[0,12],KFMUX(NONMUX),'//
     +                  'KTRGMUX(NONMUX),'//
     +                  'PXMUX(NONMUX),PYMUX(NONMUX),PZMUX(NONMUX),'//
     +                  'EEMUX(NONMUX)')
        CALL HBNAME(3333,'PTRACKS',NTRA,'NTRA[0,500],'//
```

```
     +                    'KPTRA(NTRA),KFTRA(NTRA),'//
     +                    'KPM1TRA(NTRA),KFM1TRA(NTRA),'//
     +                    'KPM2TRA(NTRA),KFM2TRA(NTRA),'//
     +                    'KPM3TRA(NTRA),KFM3TRA(NTRA),'//
     +                    'KPM4TRA(NTRA),KFM4TRA(NTRA),'//
     +                    'KPM5TRA(NTRA),KFM5TRA(NTRA),'//
     +                    'KPM6TRA(NTRA),KFM6TRA(NTRA),'//
     +                    'D0TRACRU(NTRA), Z0TRACRU(NTRA),'//
     +                    'PHITRACRU(NTRA),'//
     +                    'COTTRACRU(NTRA),PTINVTRACRU(NTRA),'//
     +                    'D0TRA(NTRA), Z0TRA(NTRA),PHITRA(NTRA),'//
     +                    'COTTRA(NTRA),PTINVTRA(NTRA),'//
     +                    'CORR11(NTRA),CORR21(NTRA),CORR31(NTRA),'//
     +                    'CORR41(NTRA),CORR51(NTRA),CORR22(NTRA),'//
     +                    'CORR32(NTRA),CORR42(NTRA),CORR52(NTRA),'//
     +                    'CORR33(NTRA),CORR43(NTRA),CORR53(NTRA),'//
     +                    'CORR44(NTRA),CORR54(NTRA),CORR55(NTRA),'//
     +                    'EFFTRA(NTRA),ISTATRA(NTRA)')
        CALL HBNAME(3333,'PHISTORY',NPART,'NPART[0,40],KFPAR(NPART),'//
     +                    'PXPAR(NPART),PYPAR(NPART),PZPAR(NPART),'//
     +                    'EEPAR(NPART)')
        CALL HBNAME(3333,'BPHYSICS',NBPHYS,'NBPHYS[0,100],'//
     +                    'KPBPHYS(NBPHYS),KFBPHYS(NBPHYS),'//
     +                    'PXBPHYS(NBPHYS),PYBPHYS(NBPHYS),'//
     +                    'PZBPHYS(NBPHYS),EEBPHYS(NBPHYS),'//
     +                    'VXBPHYS(NBPHYS),VYBPHYS(NBPHYS),'//
     +                    'VZBPHYS(NBPHYS)')
        CALL HBNAME(3333,'PTRIGGER',TGALL,'TGALL,TGEM1,TGPH1,TGEM2,'//
     +                    'TGMU1,TGMU2,TGEMU,TGJT1,TGJT3,TGJT4')
*
* analysis and ntuple filling
      ELSEIF(MODE.EQ.0) THEN
*
*       process ID in PYTHIA
        ISUB=IPROCESS
*       isolated electrons
        IDUM=0
        DO I=1,NELE
           IDUM=IDUM+1
           KDUM(IDUM,2)=KELE(I,2)
           KTRGDUM(IDUM)=100
```

```
            KDUM(IDUM,5)=1
            PDUM(IDUM,3)=PELE(I,3)
            PDUM(IDUM,4)=PELE(I,4)
            PDUM(IDUM,5)=PELE(I,5)
         ENDDO
*        isolated muons
         DO I=1,NMUO
            IDUM=IDUM+1
            KDUM(IDUM,2)=KMUO(I,2)
            KDUM(IDUM,5)=1
            KTRGDUM(IDUM)=KMUOTRG(I)
            PDUM(IDUM,3)=PMUO(I,3)
            PDUM(IDUM,4)=PMUO(I,4)
            PDUM(IDUM,5)=PMUO(I,5)
         ENDDO
*        all leptons (order in ET)
         NLEP=IDUM
         DO ILEP=1,NLEP
            ETMAX=0
            DO 160 IDUM=1,NLEP
               IF(KDUM(IDUM,5).EQ.0) GOTO 160
               IF(PDUM(IDUM,5).LT.ETMAX) GOTO 160
               IMAX=IDUM
               ETMAX=PDUM(IDUM,5)
 160        CONTINUE
            KDUM(IMAX,5)=0
            DO II=1,5
               KLEP(ILEP,II)=KDUM(IMAX,II)
               KTRGLEP(ILEP)=KTRGDUM(IMAX)
               PLEP(ILEP,II)=PDUM(IMAX,II)
            ENDDO
         ENDDO
         NJETB=0
         NJETC=0
         DO I=1,NJET
          IF(ABS(KJET(I,2)).EQ.5) NJETB=NJETB+1
          IF(ABS(KJET(I,2)).EQ.4) NJETC=NJETC+1
         ENDDO
*
         NEL      = NELE
         NMU      = NMUO
```

```
      NMUX     = NMUOX
      NPH      = NPHOT
      JETB     = NJETB
      JETC     = NJETC
      JETL     = NJET-NJETB-NJETC
      PMISS(1) = PXMISS
      PMISS(2) = PYMISS
      CIRCJ    = 0.0
      CIRCE    = CIRCEVE
      THRUST   = THRUSTA
      OBLAT    = OBL
*
*     isolated leptons
      NLEP=MIN(MAXLEP,NLEP)
      DO I=1,NLEP
         KFLEP(I)=KLEP(I,2)
         KTRGLEP(I)=KTRGLEP(I)
         PXLEP(I)=PLEP(I,5)*COS(PLEP(I,4))
         PYLEP(I)=PLEP(I,5)*SIN(PLEP(I,4))
         PZLEP(I)=PLEP(I,5)*SINH(PLEP(I,3))
         EELEP(I)=PLEP(I,5)*COSH(PLEP(I,3))
      ENDDO
*     isolated photons
      NPHO=NPHOT
      NPHO=MIN(MAXPHO,NPHO)
      DO I=1,NPHO
         KFPHO(I)=KPHOT(I,2)
         PXPHO(I)=PPHOT(I,5)*COS(PPHOT(I,4))
         PYPHO(I)=PPHOT(I,5)*SIN(PPHOT(I,4))
         PZPHO(I)=PPHOT(I,5)*SINH(PPHOT(I,3))
         EEPHO(I)=PPHOT(I,5)*COSH(PPHOT(I,3))
      ENDDO
*     all jets
      NJETA=MIN(MAXJET,NJET)
      DO I=1,NJETA
         KFJET(I)=KJET(I,2)
         PXJET(I)=PJET(I,5)*COS(PJET(I,4))
         PYJET(I)=PJET(I,5)*SIN(PJET(I,4))
         PZJET(I)=PJET(I,5)*SINH(PJET(I,3))
         EEJET(I)=PJET(I,5)*COSH(PJET(I,3))
      ENDDO
```

```
*       non-isolated muons
        NONMUX = NMUX
        NONMUX=MIN(MAXMUX,NONMUX)
        DO I=1,NONMUX
           KFMUX(I)=KMUOX(I,2)
           KTRGMUX(I)=KMUOXTRG(I)
           PXMUX(I)=PMUOX(I,5)*COS(PMUOX(I,4))
           PYMUX(I)=PMUOX(I,5)*SIN(PMUOX(I,4))
           PZMUX(I)=PMUOX(I,5)*SINH(PMUOX(I,3))
           EEMUX(I)=PMUOX(I,5)*COSH(PMUOX(I,3))
        ENDDO
*       charged tracks
        NTRA=MIN(NTRAC,MAXTRA)
        DO I=1,NTRA
         KPTRA(I)   = KPTRAC(I)
         KFTRA(I)   = KFTRAC(I)
         KPM1TRA(I) = KPM1TRAC(I)
         KFM1TRA(I) = KFM1TRAC(I)
         KPM2TRA(I) = KPM2TRAC(I)
         KFM2TRA(I) = KFM2TRAC(I)
         KPM3TRA(I) = KPM3TRAC(I)
         KFM3TRA(I) = KFM3TRAC(I)
         KPM4TRA(I) = KPM4TRAC(I)
         KFM4TRA(I) = KFM4TRAC(I)
         KPM5TRA(I) = KPM5TRAC(I)
         KFM5TRA(I) = KFM5TRAC(I)
         KPM6TRA(I) = KPM6TRAC(I)
         KFM6TRA(I) = KFM6TRAC(I)
         D0TRACRU(I)    = D0TRACCRU(I)
         Z0TRACRU(I)    = Z0TRACCRU(I)
         PHITRACRU(I)   = PHITRACCRU(I)
         COTTRACRU(I)   = COTTRACCRU(I)
         PTINVTRACRU(I)= PTINVTRACCRU(I)
         D0TRA(I)   = D0TRAC(I)
         Z0TRA(I)   = Z0TRAC(I)
         PHITRA(I)  = PHITRAC(I)
         COTTRA(I)  = COTTRAC(I)
         PTINVTRA(I)= PTINVTRAC(I)
         CORR11(I)  = CORRR11(I)
         CORR21(I)  = CORRR21(I)
         CORR31(I)  = CORRR31(I)
```

```
         CORR41(I)  = CORRR41(I)
V         CORR51(I)  = CORRR51(I)
         CORR22(I)  = CORRR22(I)
         CORR32(I)  = CORRR32(I)
         CORR42(I)  = CORRR42(I)
         CORR52(I)  = CORRR52(I)
         CORR33(I)  = CORRR33(I)
         CORR43(I)  = CORRR43(I)
         CORR53(I)  = CORRR53(I)
         CORR44(I)  = CORRR44(I)
         CORR54(I)  = CORRR54(I)
         CORR55(I)  = CORRR55(I)
         ISTATRA(I) = ISTATRAC(I)
         EFFTRA(I)  = EFFTRAC(I)
        ENDDO
c....store B_0, Lambda_b, B_0^s,D_0, D_S, K_0, Lambda, J/Y,e,mu
        NBPHYS=0
        IF(NTRAC.EQ.0) GOTO 1000
        DO I=1,N
        KC=ABS(K(I,2))
         IF((KC.EQ.511.OR.KC.EQ.531.OR.
     #       KC.EQ.421.OR.KC.EQ.431.OR.
     #       KC.EQ.130.OR.KC.EQ.310.OR.
     #       KC.EQ.443.OR.
     #       KC.EQ.5122.OR.KC.EQ.3122.OR.
     #       KC.EQ.11.OR.KC.EQ.13)
     #       .AND.K(I,1).NE.21)  THEN
           NBPHYS=NBPHYS+1
           KPBPHYS(NBPHYS)=I
           KFBPHYS(NBPHYS)=K(I,2)
           PXBPHYS(NBPHYS)=P(I,1)
           PYBPHYS(NBPHYS)=P(I,2)
           PZBPHYS(NBPHYS)=P(I,3)
           EEBPHYS(NBPHYS)=P(I,4)
           VXBPHYS(NBPHYS)=V(I,1)
           VYBPHYS(NBPHYS)=V(I,2)
           VZBPHYS(NBPHYS)=V(I,3)
         ENDIF
        ENDDO
 1000   CONTINUE
*       event history
```

```
        NPAR=0
        DO I=1,N
           IF(K(I,1).EQ.21) THEN
             IF(NPAR.LT.MAXPAR) THEN
              NPAR=NPAR+1
              KFPAR(NPAR)=K(I,2)
              PXPAR(NPAR)=P(I,1)
              PYPAR(NPAR)=P(I,2)
              PZPAR(NPAR)=P(I,3)
              EEPAR(NPAR)=P(I,4)
             ENDIF
           ENDIF
        ENDDO
        NPART=NPAR
*       trigger info
        TGALL=0.
        TGEM1=0.
        TGPH1=0.
        TGEM2=0.
        TGMU1=0.
        TGMU2=0.
        TGEMU=0.
        TGJT1=0.
        TGJT3=0.
        TGJT4=0.
        IF(TRGALL) TGALL=1.
        IF(TRGEM1) TGEM1=1.
        IF(TRGPH1) TGPH1=1.
        IF(TRGEM2) TGEM2=1.
        IF(TRGMU1) TGMU1=1.
        IF(TRGMU2) TGMU2=1.
        IF(TRGEMU) TGEMU=1.
        IF(TRGJT1) TGJT1=1.
        IF(TRGJT3) TGJT3=1.
        IF(TRGJT4) TGJT4=1.
*
*       fill ntuple
        CALL HFNT(3333)
*
* output the ntuple
      ELSEIF(MODE.EQ.1) THEN
```

```
*
        CALL HCDIR('//SLUGRZ',' ')
        CALL HROUT(0,Icycle,' ')
        CALL HREND('SLUGRZ')
*
      ENDIF
*
      END
```

A.5 Jet Algorithms

The following code is in file atlfastjet.F

```
* $Id: AtlfastReq.tex,v 1.3 2001/04/08 17:20:46 sherwood Exp $
* $Name:  $
V* #include <commons/config.h>
*****************************************************************
      SUBROUTINE JetFinder(Mode)
*****************************************************************
*
* Author: I.C. Park, IFAE-UAB Barcelona - 19971204
*
* Arguments:
*
*       IdxJF    /I     : Jet Finder Index
*                         0 - Conventional cone based algorithm
*                         1 - Mulguisin algorithm
*                         2 - Pisa KT algorithm
*                         3 - Sliding Window algorithm (not yet)
*                         4 - Montreal KT algorithm
*
*       Ipar(10) /I     : Integer parameters
*       Rpar(10) /R     : Real parameter
*
* Description :
*
*       This is a steering subroutine to interface a jet finder to
*       ATLFAST.
*       The routine must be called in MAKCLU in ATLFAST.
*
```

```
*       Currently we have 4 different Jet finders.
*       - Conventional cone-base algorithm
*       - Mulguisin algorithm by I.C. Park
*       - KT algorithm provided by D. Costanzo
*       - KT algorithm provided by K. Strahl & Georges Azuelos
*
* Note :
*
*       Conventional cone-base algorithm will not be called by this routine
*       because MAKCLU is this.
*
************************************************************************
      LOGICAL   First/.TRUE./
      CHARACTER*20 CTITJF(8)
      DATA      CTITJF/
     |          '  Cone-Jet Algorithm',' Mulguisin Algorithm',
     |          'KT Clustering (Pisa)','      Sliding Window',
     |          '   KT Jet (Montreal)',' Topological Cluster',
     |          ' reserved           ',' reserved           '/
*
*       ATLFAST interface (input /CELLS/, output /CLUSTER/)
*
#include <atlfast/maknout.inc>
#include <atlfast/cells.inc>
#include <atlfast/cluster.inc>
      INTEGER KDUM
      REAL    PDUM
      DIMENSION KDUM(100,5),PDUM(100,5)
      INTEGER LUNCARD
      DATA    LUNCARD/90/
      CHARACTER*40 JF_TITLE,JF_NAME(10)
      INTEGER IPAR(10),ITMP(10,10)
      REAL    RPAR(10),RTMP(10,10)
      INTEGER IdxJF
      SAVE IdxJF,IPAR,RPAR
*
*       Jet Finder information
*
#include <atlfast/jf_info.inc>
#include <atlfast/jf_para.inc>
*
```

```
*       Jet Finder Input
*
#include <atlfast/jf_cell.inc>
*
*       Jet Finder Output
*
#include <atlfast/jf_clus.inc>
#include <atlfast/jf_seed.inc>

*= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

      IF (Mode .EQ. -1) THEN
*
      IdxJF=-1
*
      OPEN(LUNCARD,FILE='data/jet_data/jetfinder.dat',
     |     STATUS='OLD',ERR=999)
*       Jet Finder Switch
      READ(LUNCARD,'(A40)') JF_TITLE
      READ(LUNCARD,'(I10)') IdxJF
*       cone-based jet
      READ(LUNCARD,'(A40)') JF_NAME(1)
      READ(LUNCARD,'(F10.4)') RTMP(1,1)
      READ(LUNCARD,'(F10.4)') RTMP(2,1)
      READ(LUNCARD,'(F10.4)') RTMP(3,1)
*       Mulguisin
      READ(LUNCARD,'(A40)') JF_NAME(2)
      READ(LUNCARD,'(I10)') ITMP(1,2)
      READ(LUNCARD,'(F10.4)') RTMP(1,2)
      READ(LUNCARD,'(F10.4)') RTMP(2,2)
      READ(LUNCARD,'(F10.4)') RTMP(3,2)
*       KT pisa
      READ(LUNCARD,'(A40)') JF_NAME(3)
      READ(LUNCARD,'(I10)') ITMP(1,3)
      READ(LUNCARD,'(F10.4)') RTMP(1,3)
      READ(LUNCARD,'(F10.4)') RTMP(2,3)
      READ(LUNCARD,'(F10.4)') RTMP(3,3)
      READ(LUNCARD,'(F10.4)') RTMP(4,3)
      READ(LUNCARD,'(F10.4)') RTMP(5,3)
*       Sliding Window
      READ(LUNCARD,'(A40)') JF_NAME(4)
```

```
*       KT Montreal
      READ(LUNCARD,'(A40)') JF_NAME(5)
      READ(LUNCARD,'(F10.4)') RTMP(1,5)
      READ(LUNCARD,'(F10.4)') RTMP(2,5)
      CLOSE(LUNCARD)
      DO I=1,10
        IPAR(I)=ITMP(I,IdxJF+1)
        RPAR(I)=RTMP(I,IdxJF+1)
      ENDDO
      RETURN
  999 CONTINUE
      WRITE(*,*)'----- JF_READPAR 0 -----'
      WRITE(*,*)' Error happens when open data card file '
      WRITE(*,*)'----- JF_READPAR 1 -----'
      RETURN

      ELSE

      if (First) then
        write(NOUT,'(A40)')'+--------------------------------------+'
        write(NOUT,'(A40)')'|     ATLAS Jet Finder Library v.1.0   |'
        write(NOUT,'(A40)')'+--------------------------------------+'
        write(NOUT,'(10X,A20,10X)')CTITJF(IdxJF+1)
        write(NOUT,'(A40)')'+--------------------------------------+'
        First = .FALSE.
      endif

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*       copy arrays /CELLS/ to /JF_CELL/
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      InumC=0
      do i=NCMIN,NCMAX
        InumC=InumC+1
        IuseC(InumC)=0
        RetaC(InumC)=PCELL(i,3)
        RphiC(InumC)=PCELL(i,4)
        RentC(InumC)=PCELL(i,5)
      enddo

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
*       call a JetFinder depending on the user card
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      IF (IdxJF .EQ.0) THEN

        Rcone   = Rpar(1)
        ETseed0 = Rpar(2)
        ETcone0 = Rpar(3)

        call cone_jet(Rcone,ETseed0,ETcone0)

      ELSEIF (IdxJF .EQ. 1) THEN        ! Mulguisin algorithm

        Method  = Ipar(1)
        Resol   = Rpar(1)
        ETseed0 = Rpar(2)
        ETcone0 = Rpar(3)

        call Mulguisin(Method,Resol,ETseed0,ETcone0)

      ELSEIF (IdxJF .EQ. 2) THEN        ! Pisa KT algorithm

        Nstop   = Ipar(1)
        consiz  = Rpar(1)
        dcut    = Rpar(2)
        TowCut  = Rpar(3)
        ETseed0 = Rpar(4)
        ETcone0 = Rpar(5)

        call KTCLUS(Nstop,consiz,dcut,TowCut,ETseed0,ETcone0)

      ELSEIF (IdxJF .EQ. 3) THEN        ! Sliding Window algorithm

c        Ncand     = Ipar(1)
c        Nclus     = Ipar(2)
c        ETcand    = Rpar(13)
c        ETclus    = Rpar(10)
c        Assym_eta = Rpar(901)
c        Assym_phi = Rpar(902)x

c       call SlidWind(Ncand,Nclus,ETcand,ETclus,Assym_eta,Assym_phi)
```

```
      ELSEIF (IdxJF .EQ. 4) THEN        ! Montreal KT algorithm

        Rcut    = RPAR(1)
        ETcut   = RPAR(2)

        call KTJET(Rcut,Etcut)

      ELSE
         return
      ENDIF

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*       overwrite /CELLS/ with /JF_CELL/
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      DO I=NCMIN,NCMAX
        KCELL(I,5) = 1
        IF (IuseC(I-NCMIN+1) .NE. 0) KCELL(I,5) = 0
      ENDDO

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*       overwrite /CLUSTER/ with /JF_CLUS/
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      NCLU=0
      do ij=1,InumJ
        NCLU=NCLU+1
        KCLU(NCLU,1)=ij         ! always i
        KCLU(NCLU,2)=98         ! why 98?
        Nused=0
        Npart=0
        do ic=1,InumC
          if (IuseC(ic).eq.ij) then
            Nused=Nused+1
            Npart=Npart+KCELL(ic+NCMIN,4)
          endif
        enddo
        KCLU(NCLU,3)=Nused      ! # of cells in the jet
        KCLU(NCLU,4)=Npart      ! # of particles in the jet
        KCLU(NCLU,5)=1          ! always 1
        PCLU(NCLU,1)=RetaS(ij)
```

```
        PCLU(NCLU,2)=RphiS(ij)
        PCLU(NCLU,3)=RetaJ(ij)
        PCLU(NCLU,4)=RphiJ(ij)
        PCLU(NCLU,5)=RentJ(ij)
      enddo

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*       sort /CLUSTER/ in falling ET sequence
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      NDUM=NCLU
      DO IDU=1,NCLU
        ETMAX=0
        DO IMU=1,NCLU
          IF(KCLU(IMU,5).NE.0) THEN
            IF(PCLU(IMU,5).GE.ETMAX) THEN
              IMAX=IMU
              ETMAX=PCLU(IMU,5)
            ENDIF
          ENDIF
        ENDDO
        KCLU(IMAX,5)=0
        DO II=1,5
          KDUM(IDU,II)=KCLU(IMAX,II)
          PDUM(IDU,II)=PCLU(IMAX,II)
        ENDDO
      ENDDO
      NCLU=0
      DO I=1,NDUM
        IF(PDUM(I,5).GT.0.0) THEN
          NCLU=NCLU+1
          DO II=1,5
             KCLU(NCLU,II)=KDUM(I,II)
             PCLU(NCLU,II)=PDUM(I,II)
          ENDDO
          KCLU(NCLU,1) = NCLU
          KCLU(NCLU,5) = 1
        ENDIF
      ENDDO

      ENDIF
```

```
*= = =  = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
      RETURN
      END
*-----------------------------------------------------------------------

      real function Rangle(x,y)
      parameter (pi=3.141592653)
      Rangle=0.0
      r=sqrt(x**2+y**2)
      if (r.lt.1e-20) return
      if (abs(x)/r .lt. 0.8) then
        Rangle=sign(acos(x/r),y)
      else
        Rangle=asin(y/r)
        if (x .lt.0.0 .and. Rangle .ge. 0.0) then
          Rangle=pi-Rangle
        elseif (x .lt. 0.0) then
          Rangle=-pi-Rangle
        endif
      endif
      end

      real function eta_ie(ie,Neta,EtaMin,EtaMax)
*      Neta=(EtaMax-EtaMin)/Deta
      eta_ie=EtaMin+(real(ie)-0.5)*(EtaMax-EtaMin)/float(Neta)
      end

      real function phi_ip(ip,Nphi)
      parameter (pi =3.141592653, pi2=6.283185307)
      phi_ip=(real(ip)-0.5)*pi2/float(Nphi)-pi
      end

      integer function ie_eta(eta,Neta,EtaMin,EtaMax)
      ie_eta=int((eta+EtaMax)/2.0/EtaMax*float(Neta))+1
      end

      integer function ip_phi(phi,Nphi)
      parameter (pi =3.141592653, pi2=6.283185307)
      ip_phi=int((phi+PI)/2.0/PI*float(Nphi))+1
      end
```

```
integer function ietaphi(eta,Neta,EtaMin,EtaMax,phi,Nphi)
ietaphi=Nphi*ie_eta(eta,Neta,EtaMin,EtaMax)+ip_phi(phi,Nphi)
end
```

B Appendix: Desirable Features

The purpose of this appendix is to provide a location for requirements which have not been fully worked out.

They are either user requirements which need more user definition to be attended to, or interesting ideas which are important not to loose.

As these ideas have not been fully specified, they are not forward-traceable to the design phase.

The review process should periodically examine these ideas to ascertain if further information is available, which would allow them to become fully fledged requirements.

- Particle reconstruction: efficiency and purity. To be implemented need to know in more details what they mean.
- Atlfast will fulfill requirements to interface with ATLAS display tools, but display tools have to be defined.
- The Athena-Atlfast event size will be small data sets, but working definition of small has yet to be defined.
- Performance requirements: Athena-Atlfast will run in a time that is less than 2.0 times the time to generate events, when the generator will be Pythia xx.xx, the physics channel will be specifies by the Pythia parameter MS=?? (pp→??) with non standard parameters = ??.
- L1 Trigger Simulation.
- L2 Trigger Simulation.
- Weights for jet-labels
- Output classes will share an interface with corresponding classes for full simulation.
- Cells: different cell sizes.
- Cells: parameterisation of longitudinal and lateral energy deposition.

- Cells: out of time vectors (pile-up+pulse electronic pulse shape)
- Secondary vertices
- Hits

C Appendix: Corrispondence between the original User Requirements document and Atlfast Requirements Document

This appendix is to provides the correspondence between the requirements of the user requirements document that went before the Atlfast User Requirements review and the present document.

According to the recommndations of the commitee, some the reponsability for some of the requirements have been allocated to programs external to Atlfast.

USER REQS	*ATL REQ DOC*
UR1	ARD7, ARD16, ARD28
UR2	Appendix B
UR3	section 3
UR4	ARD10, ARD11, ARD21
UR5	ARD24, ARD25, ARD26
UR6	ARD24, ARD25, ARD26
UR7	ARD14
UR8	ARD37
UR9	-
UR10	ARD5
UR11	-
UR12	Appendix B
UR13	Appendix B
UR14	ARD5
UR15	ARD2
UR16	ARD35
UR17	-
UR18	HepMC
UR19	Athena
UR20	Appendix B
UR21	ARD21
UR22	ARD31
UR23	Athena
UR24	Athena
UR25	Athena
UR26	-
UR27	Athena