Table of Contents

How to Make ATLAS AlpGen production of W + n Jets

Introduction

This page is intended to give a step by step description of how to create your own AlpGen production of W + n Jet samples equivalent to the official ATLAS production. Additional information can be found in the AlpGen manual. The aim of this page is to provide the minimum short-cut instruction for truth level comparisons (such as scale variation). This guide is given for how to create an AOD file with the truth information and high level truth objects. The end result should be comparable with mc09_7TeV.107690.AlpgenJimmyWmunuNp0_pt20.merge.AOD.e511.

(1st step) Retrieving the AlpGen package

Users can get the source code of ALPGEN from the web page or from the afs, i.e. /afs/cern.ch/sw/lcg/external/MCGenerators/alpgen/2.1.3/share/alpgen-author/. Here, let us use the source code from the author's web page. In users working directory,

```
$ wget http://mlm.home.cern.ch/mlm/alpgen/V2.1/v213.tgz
$ tar zxvf v213.tgz
```

Now users can find directories named `"PROCESS"lib` or `"PROCESS"work`, other than the directories for core libraries (alplib, herlib, pylib), Documentation (DOCS), examples (validation). In this page we discuss the W+jets process. The user configuration files for W + nJets are in the `wjetwork/` directory, all the users work is carried out in this directory.

```
$ cd wjetwork
```

compile the code (here we use f77 compilation, if you want to use F90 see later section)

```
$ make gen
```

(2nd step) Running wjet out of the box

As a first test, let's run the alpgen without changing anything. If you have a problem here, you need to check what is wrong with your system. The instruction in this page does not include trouble shooting for different machines. Users should find the executable file `wjetgen`. The alpgen sample production is carried out in a two step operation. The operation mode is defined in the `input` file (the line with comment "imode")

Produce weighted events

First set the imode value = 1 (the value already set by default), then run as

```
$ ./wjetgen < input >& log_mode1
```

after about 1min, the job finishes and users would see the files named `w2j.*` (explanations for the meaning of each file are omitted here, full detail can be found in the manual). `w2j.wgt` is the weighted event file, it basically contains the random seeds and event weights.

Produce unweighted events

After just changing the imode to 2 in "input" file, try the same command

```
$ ./wjetgen < input >&  log_mode2
```

After this, users can find the unweighted event file, `w2j.unw` which is to be used as an input file for Herwig, or Pythia in the later section. The file `w2j.par` contains the information of configuration parameters and event generation (number of events, cross section, etc). There are other files created for monitoring of generation, users are suggested to check the manual for their meanings.

Please note that you will likely never just run a production of a fixed number of partons. In ATLAS we combine 0-5 partons.

(3rd step) Running wjet with ATLAS input

The current ATLAS default input file to create MC consistent with mc09_7TeV.107690.AlpgenJimmyWmunuNp0_pt20.merge.AOD.e511* is given below. This may be updated in the future and you are much more likely to find the updates here than on this page.

```
MODE            ! imode
PROCESS         ! label for files
0               ! start with: 0=new grid, 1=previous warmup grid, 2=previous generation grid
WARM 4        ! Nevents/iteration,  N(warm-up iterations)
EVENTS        ! Nevents generated after warm-up
*** The above 5 lines provide mandatory inputs for all processes
*** (Comment lines are introduced by the three asteriscs)
*** The lines below modify existing defaults for the hard process under study
*** For a complete list of accessible parameters and their values,
*** input 'print 1' (to display on the screen) or 'print 2' to write to file
njets JETS
ptjmin 15
drjmin 0.7
ih2 1                ! LHC
ebeam 3500.0         ! E beam
ndns 9               ! PDF CTEQ6L1
iqopt 1              ! Qscale, 1 is generator default for all the processes
qfac  1              ! Qscale factor
ickkw 1              ! enable jet-parton matching, determine scale of alpha_s
ktfac 1              ! ckkw alphs scale
ptlmin 0.            ! lepton min pt
metmin 0.0           ! missing et cut
etajmax 6.0          ! parton max eta
etalmax 10.0         ! lepton max eta
drlmin 0.0           ! min delta r between leptons
iwdecmode 2          ! W decay mode (2: mu)
cluopt 1             ! kt scale option. 1:kt propto pt, 2:kt propto mt
iewop 3              ! EW parameter scheme, (3= mw=80.419, mz=91.188, GF=1.16639^-5 hard coded)
iseed1 MYSEED1       ! first  random seed for weighting process
iseed2 MYSEED2       ! second random seed for weighting process
iseed1 MYSEED3       ! first  random seed for unweighting process
iseed2 MYSEED4       ! second random seed for unweighting process
```

- MODE is 1 to produce weighted events then 2 to unweight the events
- PROCESS should be changed to the correct process ie w2j for W + 2 Partons
- WARM and EVENTS must be chosen by the user (Note: we recommend you to get x-sec uncertainty is smaller than 1%. If not, please increase these parameters.)
- JETS is the number of partons to be generated with the W and must correspond to the PROCESS
- MYSEED values should be random number inputs. In ALPGEN, the random numbers are calculated by the function described in the paper, F.JAMES, COMP PHYS COMM 60(1990)329. (see alpgen.f

in alplib). The users are supposed to provide two sets for random seeds in 5 digits (4 numbers in total), for example, for weighted event generation 12345/67890, for unweighted event generation 12345/67890. The numbers could be any from 1-99999 which would produce independent random numbers in a period of 2x10^18 for 32bit machine. When users want to accelerate the event production rate, it can be easily done by running multiple jobs with different random seeds. One practical example is to use seed1=12345+10*i, seed2=67890-10*i (i=0,1,2...) for i-th jobs.Seed 3,4 could be equal to Seed 1,2.

- Concerning PDF choice, since CTEQ6L1(ndns 9) is used as default at ATLAS, you need to make a link or copy parameter file from alplib/pdfdat/, for example, ln -s your path/alplib/pdfdat/cteq6l1.tbl cteq6l1 .

The idea is to produce adequate number of events by setting EVENTS correctly. This takes a little trial and error. You must produce events for each JET [0,5] by first running

```
 ./wjetgen < input >& log_mode1
```

with MODE=1 then again with MODE=2 to unweight. This means 10 executions of wjetgen for a complete w + Jets samples if you do not split your jobs.

(4th step) running ALPGEN Athena interface

After the previous step, the files named 'w*j_unw.par' and 'w*j.unw' are created.

What Athena and Herwig want as an input is somewhat different than what you are given as an output. Firstly the file names are wrong so you must change them. For example in the 2 parton sample:

```
mv w2j.unw alpgen.unw_events
mv w2j_unw.par inparmAlpGen.dat
```

Secondly what is now called `inparmAlpGen.dat` is missing some parameters. The following lines should be added to `inparmAlpGen.dat` just before `************** end parameters` line in the file.

```
501    20.        ! min ETCLUS used for parton-jet matching (Normally ETCLUS = ptjmin + 5)
502    0.7        ! min RCLUS value for parton-jet matching
503    6.         ! max ETACLUS value for parton-jet matching
504    1          ! 0 inclusive 1 exclusive
```

IMPORTANT NOTE: The value for 504 determines if the MLM matching should be inclusive or exclusive. This means that the 5 JETS sample should have this set to 0.

The 2 parton `inparmAlpGen.dat` file should now look like

```
 W + 2 jets
 W-> ell nu
 ====================================
 Generation cuts for the partonic event sample:
      Light jets:
 ptmin=  15. |etamax|=  6. dR(j-j)>  0.7
      Leptons:
 ptmin(lep)=  0. |etamax|=  10. Et(miss)>  0. dR(l-j)>  0.
************** run parameters
   3 ! hard process code
   0.000   4.700 174.300  80.419  91.188 120.000 ! mc,mb,mt,mw,mz,mh
 2  1.  ! ih2
 3  3500.  ! ebeam
 4  9.  ! ndns
 5  1.  ! iqopt
 6  1.  ! qfac
```

```
 7  1.  ! ickkw
 8  1.  ! ktfac
 10  2.  ! njets
 30  15.  ! ptjmin
 33  0.  ! ptlmin
 34  0.  ! metmin
 40  6.  ! etajmax
 43  10.  ! etalmax
 50  0.7  ! drjmin
 55  0.  ! drlmin
 90  10002.  ! iseed1
 91  20002.  ! iseed2
 151  2.  ! iwdecmod
 160  1.  ! cluopt
 190  12345.  ! iseed3
 191  67890.  ! iseed4
 501    20.        ! min ETCLUS used for parton-jet matching
 502    0.7        ! min RCLUS value for parton-jet matching
 503    6.         | max ETACLUS value for parton-jet matching
 504    1         ! 0 inclusive 1 exclusive
************** end parameters
            1373.86476306              46.94481202  ! Crosssection +- error (pb)
 56  0.040760926 ! unwtd events, lum (pb-1)
```

If you will use an official script, that is, Evgen_try.py to make evgen files and later will ask the official production, please use the following naming convention;

```
mv w2j.unw group09.phys-gener.alpgen.XXXXXX.WenuNp2_7TeV.TXT.v1._00001.events
mv w2j_unw.par group09.phys-gener.alpgen.XXXXXX.WenuNp2_7TeV.TXT.v1._00001.dat
tar zcvf  group09.phys-gener.alpgen.XXXXXX.WenuNp2_7TeV.TXT.v1._00001.{dat,events}
```

The above new 4 lines should be added to *.dat. The 6 digit (XXXXXX) is dataset id. For the official production, please get it from MC production manager of each group. For the local production, please decide it, say 900001, by yourself.

You can find many concrete examples used in the official production on the grid. Please try dq2-ls group09.phys-gener.alpgen*/ and get one(some) of them by dq2-get.

Also you can find another instruction on it here. (DON'T register any dataset with group*.phys-gener.* by yourself. Please contact MC production managers of each group.)

(5tht step) running ALPGEN Athena interface to do Shower MC

The official joboptions (jobOptions.AlpgenHerwig.py) and (jobOptions.AlpgenPythia.py) are unreliable. At the time of the writing of this page they point to files with the wrong parameters set. Many parameters are set by hand to be consistent with mc09_7TeV.107690.AlpgenJimmyWmunuNp0_pt20.merge.AOD.e511. There are official jobOptions for this (MC9.117680.AlpgenPythiaWenuNp0_pt20.py and MC9.107680.AlpgenJimmyWenuNp0_pt20.py)but they do not explicitly state all the parameters used.

Below are given jobOptions files which will run out of the box and produce an AOD-like file with all the truth information. Many parameters are set here by hand to be consistent with the official production. These jobOptions additionally produce useful offline truth objects. These will give the user truth jets and truth MET that may be useful for your work. Please note the similarities between Herwig and Pythia. After setting up athena the user should be able to execute the jobOptions

```
athena.py jobOptions.AlpgenHerwig.py
```

Herwig

Herwig with Jimmy is used as the default generator for the shower and fragmentation in ATLAS.

```
##############################################################
#
# Job options file
#
#==============================================================
#--------------------------------------------------------------
# General Application Configuration options
#--------------------------------------------------------------
import AthenaCommon.AtlasUnixGeneratorJob

from AthenaCommon.AppMgr import theApp
from AthenaCommon.AppMgr import ServiceMgr

# make sure we are loading the ParticleProperty service
from PartPropSvc.PartPropSvcConf import PartPropSvc
ServiceMgr += PartPropSvc()
ServiceMgr.MessageSvc.OutputLevel = INFO

#--------------------------------------------------------------
# Event related parameters
#--------------------------------------------------------------
# Number of events to be processed (default is 10)
theApp.EvtMax = -1
#--------------------------------------------------------------
# Algorithms Private Options
#--------------------------------------------------------------
import random
seed1=random.uniform(100000000, 999999999)
seed2=random.uniform(100000000, 999999999)
seed3=random.uniform(100000, 999999)
seed4=random.uniform(1000000, 9999999)

from AthenaServices.AthenaServicesConf import AtRndmGenSvc
ServiceMgr += AtRndmGenSvc()
ServiceMgr.AtRndmGenSvc.Seeds = ["HERWIG " + str(int(seed1))+" " + str(int(seed2)), "HERWIG_INIT
ServiceMgr.AtRndmGenSvc.Seeds += ["PHOTOS "  + str(int(seed1))+" " + str(int(seed2)), "PHOTOS_INI
ServiceMgr.AtRndmGenSvc.Seeds += ["TAUOLA "  + str(int(seed1))+" " + str(int(seed2)), "TAUOLA_INI

from AthenaCommon.AlgSequence import AlgSequence
topSequence=AlgSequence()

from Herwig_i.Herwig_iConf import Herwig
topSequence += Herwig()

topSequence.Herwig.HerwigCommand = [
    # mass
    "rmass 6 172.5",    # PDG2007 TOP mass
    "rmass 198 80.403", # PDG2007 W mass
    "rmass 199 80.403", # PDG2007 W mass
    "rmass 200 91.1876",# PDG2007 Z0 mass
    "gamw 2.141",       # PDG2007 W width
    "gamz 2.4952",      # PDG2007 Z0 width
    # main switch
    "jmueo 1",          # Underlying event option (2->2 QCD)
    "msflag 1",         # turn on multiple interactions
    "jmbug 0",
    # U.E. tuning
    "ptjim 4.06",       # for 7TeV
    "jmrad 73 1.8",     #  Inverse proton radius squared
    "prsof 0",          #  soft underlying event off (Herwig parameter)
    # PDF
    "modpdf 10042",     #  CTEQ6L1 (LO fit with LO alpha_s)
```

```
    "autpdf HWLHAPDF",  #  external PDF library
    # Fragmentation
    "clpow 1.20",       # to fix the ratio of mesons to baryons in B decays
    "pltcut 0.0000000000333",  # to make Ks and Lambda stable
    "ptmin 10.",        # (D=10.) min. pT in hadronic jet production
    # Others
    "mixing 1",         # (D=1) include neutral B meson mixing
    "maxpr 5"           # print out event record
    ]

topSequence.Herwig.HerwigCommand += [ "iproc alpgen", "taudec TAUOLA",  ]

# ... Tauola
from Tauola_i.Tauola_iConf import Tauola
topSequence += Tauola()
topSequence.Tauola.TauolaCommand = [
    "tauola polar 1",
    "tauola radcor 1",
    "tauola phox 0.01",
    "tauola dmode 0",
    "tauola jak1 0",
    "tauola jak2 0"
    ]
from MC09JobOptions.TauolaEvgenConfig import evgenConfig

# ... Photos
from Photos_i.Photos_iConf import Photos
topSequence += Photos()
topSequence.Photos.PhotosCommand = [
    "photos pmode 1",
    "photos xphcut 0.01",
    "photos alpha -1.",
    "photos interf 1",
    "photos isec 1",
    "photos itre 0",
    "photos iexp 1",
    "photos iftop 0"
    ]
from MC09JobOptions.PhotosEvgenConfig import evgenConfig

# truth jet/met

from RecExConfig.RecConfFlags import jobproperties
jobproperties.RecConfFlags.AllowBackNavigation = True
from JetRec.JetRecFlags import jobproperties as jobpropjet
jobpropjet.JetRecFlags.inputFileType = "GEN"

from ParticleBuilderOptions.AODFlags import AODFlags
AODFlags.allSetOff()
AODFlags.MissingEtTruth = True
AODFlags.TruthParticleJet = True
AODFlags.McEventKey="GEN_EVENT"
AODFlags.Print()

from AthenaCommon.GlobalFlags import GlobalFlags
GlobalFlags.Luminosity.set_zero()

# This allows JetsFromTruthTool to read GEN_EVENT input
from JetRec.JetRecFlags import jetFlags
jetFlags.inputTruthColl_RECO = AODFlags.McEventKey

# This is necessary for making truth jets,
# as RecFlags.doTruth is also tested
from RecExConfig.RecFlags import rec
rec.doTruth=True

# The function that makes the truth jets, with appropriate
```

```
# arguments
from JetRec.JetRec_defaults import make_StandardJets
make_StandardJets(doTowerJet = False, doTopoJet = False, doTruthJet = True, doLocalCalib = False)

include( "ParticleBuilderOptions/MissingEtTruth_jobOptions.py" )
METPartTruth.TruthCollectionName=AODFlags.McEventKey
topSequence.METAlg+=METPartTruth


#--------------------------------------------------------------
# Pool Persistency
#--------------------------------------------------------------
from AthenaPoolCnvSvc.WriteAthenaPool import AthenaPoolOutputStream

StreamAOD = AthenaPoolOutputStream("StreamAOD")
from AthenaCommon.DetFlags import DetFlags

include ("AthenaPoolCnvSvc/WriteAthenaPool_jobOptions.py")
include ("RecExPers/RecoOutputAODList_jobOptions.py")
include ("RecAthenaPool/RecAthenaPool_joboptions.py" )


StreamAOD.ItemList += ["EventInfo#*", "McEventCollection#*" ]
StreamAOD.ItemList +=["2101#*", "133273#*"]
StreamAOD.ForceRead=True
StreamAOD.OutputFile = "AlpgenHerwig.root"

#--------------------------------------------------------------
# Ntuple service output
#--------------------------------------------------------------
#==============================================================
#
# End of job options file
#
###############################################################
```

In case you will use an official script Evgen_try.py, you need to prepare job option file, for example, MC9.XXXXXX.AlpgenJimmyWenuNp2.py Then run this script;

```
Evgen_try.py 7000 XXXXXX 1 -1 1 MC9.XXXXXX.AlpgenJimmyWenuNp2.py evgen._00001.pool.root NONE NONE
```

You can find many examples of job option files on SVN, for example, MC9.107682.AlpgenJimmyWenuNp2_pt20.py.

In case of Alpgen, the present official script produces one evgen file with 500 events. This is proper for filtered datasets but if not, 5000 events is reasonable. (this is default for most of generators, Pythia and so on.) To change this limitation, please add "evgenConfig.minevents=5000" in the end of job option file. You can find some example on SVN, for example MC9.109660.AlpgenQcdJx1Np2_pt20.py.

Pythia

Pythia version used: is 6.4.21. Although Pythia is not the default generator for the shower and fragmentation in ATLAS it is just as valid as Herwig.

```
##############################################################
#
# Job options file
#
#==============================================================
#--------------------------------------------------------------
# General Application Configuration options
#--------------------------------------------------------------
```

```
import AthenaCommon.AtlasUnixGeneratorJob

from AthenaCommon.AppMgr import theApp
from AthenaCommon.AppMgr import ServiceMgr

# make sure we are loading the ParticleProperty service
from PartPropSvc.PartPropSvcConf import PartPropSvc
ServiceMgr += PartPropSvc()
ServiceMgr.MessageSvc.OutputLevel = INFO

#--------------------------------------------------------------
# Event related parameters
#--------------------------------------------------------------
# Number of events to be processed (default is 10)
theApp.EvtMax = -1
#--------------------------------------------------------------
# Algorithms Private Options
#--------------------------------------------------------------
import random
seed1=random.uniform(1000000, 9999999)
seed2=random.uniform(100000000, 999999999)
seed3=random.uniform(100000, 999999)
seed4=random.uniform(1000000, 9999999)

from AthenaServices.AthenaServicesConf import AtRndmGenSvc
ServiceMgr += AtRndmGenSvc()
ServiceMgr.AtRndmGenSvc.Seeds = ["PYTHIA " + str(int(seed1))+" " + str(int(seed2)), "PYTHIA_INIT
ServiceMgr.AtRndmGenSvc.Seeds += ["PHOTOS "  + str(int(seed1))+" " + str(int(seed2)), "PHOTOS_INI
ServiceMgr.AtRndmGenSvc.Seeds += ["TAUOLA "  + str(int(seed1))+" " + str(int(seed2)), "TAUOLA_INI

from AthenaCommon.AlgSequence import AlgSequence
topSequence=AlgSequence()

from Pythia_i.Pythia_iConf import Pythia
topSequence += Pythia()

# use AMBT1 tune
Pythia.Tune_Name="ATLAS_20100001"

topSequence.Pythia.PythiaCommand = [
    # initializations
    "pyinit pylisti 12",
    "pyinit pylistf 1",
    "pystat 1 3 4 5",
    "pyinit dumpr 1 5",
    # mass
    "pydat2 pmas 6 1 172.5",    # PDG2007 TOP mass
    "pydat2 pmas 24 1 80.403",  # PDG2007 W mass
    "pydat2 pmas 23 1 91.1876", # PDG2007 Z0 mass
    ]

topSequence.Pythia.PythiaCommand += [
    "pyinit user alpgen",
    "pydat1 parj 90 20000.", # Turn off FSR.
    "pydat3 mdcy 15 1 0", # Turn off tau decays.
    "pypars mstp 143 1" # matching
    ]

# ... Tauola
from Tauola_i.Tauola_iConf import Tauola
topSequence += Tauola()
topSequence.Tauola.TauolaCommand = [
    "tauola polar 1",
    "tauola radcor 1",
    "tauola phox 0.01",
    "tauola dmode 0",
    "tauola jak1 0",
```

```
    "tauola jak2 0"
    ]
from MC09JobOptions.TauolaEvgenConfig import evgenConfig

# ... Photos
from Photos_i.Photos_iConf import Photos
topSequence += Photos()
topSequence.Photos.PhotosCommand = [
    "photos pmode 1",
    "photos xphcut 0.01",
    "photos alpha -1.",
    "photos interf 1",
    "photos isec 1",
    "photos itre 0",
    "photos iexp 1",
    "photos iftop 0"
    ]
from MC09JobOptions.PhotosEvgenConfig import evgenConfig

# truth jet/met

from RecExConfig.RecConfFlags import jobproperties
jobproperties.RecConfFlags.AllowBackNavigation = True
from JetRec.JetRecFlags import jobproperties as jobpropjet
jobpropjet.JetRecFlags.inputFileType = "GEN"

from ParticleBuilderOptions.AODFlags import AODFlags
AODFlags.allSetOff()
AODFlags.MissingEtTruth = True
AODFlags.TruthParticleJet = True
AODFlags.McEventKey="GEN_EVENT"
AODFlags.Print()

from AthenaCommon.GlobalFlags import GlobalFlags
GlobalFlags.Luminosity.set_zero()

# This allows JetsFromTruthTool to read GEN_EVENT input
from JetRec.JetRecFlags import jetFlags
jetFlags.inputTruthColl_RECO = AODFlags.McEventKey

# This is necessary for making truth jets,
# as RecFlags.doTruth is also tested
from RecExConfig.RecFlags import rec
rec.doTruth=True

# The function that makes the truth jets, with appropriate
# arguments
from JetRec.JetRec_defaults import make_StandardJets
make_StandardJets(doTowerJet = False, doTopoJet = False, doTruthJet = True, doLocalCalib = False)

include( "ParticleBuilderOptions/MissingEtTruth_jobOptions.py" )
METPartTruth.TruthCollectionName=AODFlags.McEventKey
topSequence.METAlg+=METPartTruth


#--------------------------------------------------------------
# Pool Persistency
#--------------------------------------------------------------
from AthenaPoolCnvSvc.WriteAthenaPool import AthenaPoolOutputStream

StreamAOD = AthenaPoolOutputStream("StreamAOD")
from AthenaCommon.DetFlags import DetFlags

include ("AthenaPoolCnvSvc/WriteAthenaPool_jobOptions.py")
include ("RecExPers/RecoOutputAODList_jobOptions.py")
include ("RecAthenaPool/RecAthenaPool_joboptions.py" )
```

```
StreamAOD.ItemList += ["EventInfo#*", "McEventCollection#*" ]
StreamAOD.ItemList +=["2101#*", "133273#*"]
StreamAOD.ForceRead=True
StreamAOD.OutputFile = "AlpgenPythia.root"

#--------------------------------------------------------------
# Ntuple service output
#--------------------------------------------------------------
#==============================================================
#
# End of job options file
#
###############################################################
```

(6th step) Calculating a cross section

At the end of the output file `log_mode2` the user should see something like the following:

```
 starting scan/unweighting of  43513. events
 Crosssection(pb)=  1313.52144+-  13.9189497
 Generated  4007 unweighted events, lum=  3.0505783pb-1
```

- The first number (43513) is the number of unweighted events generated by AlpGen. It is of no use here.
- The second number is the AlpGen cross section. For split jobs this should be averaged.
- The third number (4007) is the number of AlpGen events created before MLM matching.
- The MLM efficiancy is the number of events in the AOD divided by the the number of AlpGen events created before MLM matching
- The actual cross section is the Alpgen cross section * eff(MLM)

It is strongly advised that you check the Alpgen cross section and eff(MLM) against the numbers for the official production these can be found here.

(7th step) Systematic Variations

At this point the user will have created a sample equivalent to the official one. It is common to now vary the input parameters that are not well understood theoretically to set bounds on systematic uncertainties. Common variations are:

Factorization Scale Functional Form:

- Funtion which determines the scale used in the PDF
- iqopt = 1 (M_V^2+ m_T^2), 2 (M_V^2), 3 (M_V^2+P_{TV}^2)
- Default: iqopt = 1

Factorization Scale Factor:

- Multiplicitive factor to Factorization Functional Form
- Default: qfac = 1.

$_s$-reweighting for additional radiation:

- Enable the CKKW matching scheme reweighting.
- This is also what should be used for the MLM matching as in the case of AlpGen
- Default: ickkw=0 (ie off)

$_s$ Renormalization scale used in reweighting when ickkw=1 is specified:

- Proportionality factor for reweighting factor determined from the k_T at each local vertex
- Default: ktfac = 1 (implies k_T = sqrt(t) z (1-z) > Q_{min} is the branch cut)

P_T Parton Cut:

- Minimum P_T to meet the definition of a hard parton.
- ie the ME will not generate partons with P_T less than this value
- Default: ptjmin = 15 GeV

P_T Jet Cut:

- Minimum P_T to meet the definition of a Jet for parton-jet MLM matching.
- Default ETCLUS = ptjmin + 5

Delta-R Parton Cut:

- Minimum Parton separation in Delta-R.
- Default: drjmin = 0.7

Delta-R Jet Cut:

- Minimum jet separation in Delta-R to meet the definition of a Jet for parton-jet MLM matching.
- Default: RCLUS = drjmin

Clarification of different scales

If you run in ickkw=0 mode (no matching) you only have qfac, which is the scale factor common to both renormalization and factorization scales. When you run in ickkw=1 mode, iqopt and qfac govern the factorization scale (namely the pdf scale), while the scale in $_s$ is entirely driven by ktfac. The functional form is the one of the ckkw prescription (namely proportional to the k_T of each local vertex), while ktfac is an overall factor. The ME generation is however not entirely independent of ktfac, since we have to do the "$_s$ reweighting" in order to go from the ME calculation with all powers of $_s$ evaluated at the same scale, to the case where each power of $_s$ has the scale of the given local vertex. To optimize the unweighting efficiency, information of the value of ktfac is already used at the generation level.

Choosing an appropriate variation is hard to quantify. For scale factors like ktfac and qfac a factor of 2 is taken as an estimate on one &sigma.

Previous studies

An examples of such studies are:

- Comparative study of various algorithms for the merging of parton showers and matrix elements in hadronic collisions
- Ongoing Studies

Further Information

- There is additional information on the following page: AlpgenFAQ . Any further question can also be posted there.
- A useful talk can be found here here.

-- KeithEdmonds - 16-Jul-2010

This topic: Main > AtlasAlpgenWJets
Topic revision: r22 - 2010-11-03 - KeithEdmonds