Pythia_i: An interface between Pythia and Athena release 8.0.0 and later

Ian Hinchliffe (I_Hinchliffe@lbl.gov), G. Stavropoulos (George.Stavropoulos@cern.ch)

October 18, 2004

This package runs Pythia from within Athena, puts the events into the transient store in HepMC format. See the documentation on GenModule for general information. The note refers only to Pythia specific material. The External/Pythia package is used to set up the paths to the Pythia library which is now pointing to the release maintained by Genser. This works with pythia6.xxx only. The current version is 6.221.

A version of 6.221 with the LHAPDF structure function package is also available. To use this one replace "Pythia_i" by "PythiaLha_i" and the algorthm name "Pythia" by "PythiaLha".

The module is activated from the jobOptions service.
See the examples in **Pythia_i/share/jobOptions.pythia.py** , **Pythia_i/share/jobOptions.pythiatest.py** and **Pythia_i/share/jobOptions.pythia_lhapdf.py**

The pythia parameters are set from the job options service. The default makes $t\bar{t}$ events with a top mass of 175 GeV. The default jobOptions.pythia.py file will have been copied to your TestRelease area when you set up athena under CMT. **Note that all parameters passed to Pythia are in the units specified in the Pythia manual. In particular, energies and masses are in GeV, not the standard atlas units of MeV.** The following is needed if you wish to run Pythia

```
theAppDLLs  += [ "Pythia_i"]
theAppTopAlg = ["Pythia"]
```

The parameters are passed via the following line in the jobOptions.py file.

```
Pythia.PythiaCommand = ["common_block_name variable index value",
"common_block_name1 variable1 index1 value1"]
```

Each quoted string sets one parameter. You can have as many as you like seperated by commas. **common_block_name** must be one of the following common block names and must be in lower case.
pydatr
pydat1
pydat2
pydat3
pypars
pymssm
pysubs
pyint2

pyinit
pystat
The pyinit deals with parameters associated with the job setup, while the pystat one deals with the setting of the print level of the PYSTAT routine (called at the end of the run). An error message is returned if the common block is not one of these. The job continues to run but the behaviour may not be what you expect. **variable** must be the variable name that you are adjusting. Valid names are as follows (lower case is required)
for pydatr – mrpy and rrpy
for pydat1 – mstu paru mstj and parj. Do not change mstu(11) (see below)
for pydat2 – kchg pmas parf and vckm
for pydat3 – mdcy mdme brat and kfdp
for pymssm – imss and rmss
for pypars – mstp parp msti and pari
for pysubs – msel msub kfin and ckin
for pyint2 – iset kfpr coef and icol
The following do not correspond to a common block but the parsing is similar
pyinit access variable that are either passed in the call to PYINIT or to variables that control the listing. The choices are pyinit – pbar (changes one of the incoming particles to pbar), user (changes the process selection to user code for connection to external processes, a string then specifies the specific procees, see discussion of external processes below), FIXT which switches to fixed target operation, win (changes the center of mass energy, units are GeV) output (controls redirection of output) pylisti (gives the number to be passed to pylist on the setup) pylistf (gives the number to be passed to pylist on the dumped events) dumpr (is two integers specifying the range of events to be written out). **Settings in the pyinit groups should come before others. This is essential if you are using an external process such as comphep or AcerMC**
pystat sets the print level of the PYSTAT routine (called at the end of the run).

Again an error message is returned if the common block is not one of these. The job continues to run but the behaviour may not be what you expect.

The remainder of the values in the " " specify the indices (if any) of the variable and the value that you are setting it to. The range of the indices is as described in the Pythia manual (Do not try to be clever and offset them). You must specify the indices and value completely. The number of quantities that you must provide depends on the variable. The order is "(first_index) (second_index) value"; if there is no corresponding index, omit it. There is currently minimal error checking here so you will get junk or a core dump if you make a mistake. The variable and common block names are parsed and checked. A message is sent to the Athena LOG if an error is detected but the job does not abort.

Examples should make it clear

"pysubs msel 13" will turn on Z/gamma+jet production
"pysubs ckin 3 18." will set the minimum p_T to 18 GeV
"pypars mstp 43 2" wiil turn off the photon and Z/photon interference.
"pyinit win 1800." changes the center of mass energy to 1800 GeV
Note that the entries are seperated by a single space and that reals must have a decimal point.
"pyinit pylisti 12" dumps all the particle and decay data (see pythia manual) after initialization.
"pyinit output junk.dat" causes all the pythia output to dump into a file junk.dat (it resets mstu(11))
"pyinit pylistf 1 " dumps the complete event record for the specified events

"pyinit dumpr 3 12 " causes events 6 through 12 to be written out
"pystat 3 5 7" sets the PYSTAT print levels. Pystat is called as many times as integers after pystat. In the example, pystat will be called with prin-level 3 followed by a call with print-level 5 and then by a call with print-level 7. The default is to call pystat once with print-level 1.
"pydat2 pmas 4000011 1 1000." will set the mass of particle with **KF code** 4000011 to 1000 GeV.
WARNING: Someone has to give the KF and NOT the KC code. The KF code is converted (via a call to PYCOMP) to the KC code internally in Pythia.cxx

The jobOptions.tex file that has this example is contained in **Pythia_i/share/jobOptions.pythia.py**

Default Parameters

The Pythia libarary built by Stan Thompson has the parameters as set by the Pythia authors. The ATLAS settings are set inside Pythia.cxx. The default initialization corresponds to *pp* collsions at 14 TeV, PYLIST is called as PYLIST(11) after initialization. The current set of defaults is the ones in the Pythia release 6.203 with the following exceptions.

- In common block PYPARS: MSTP(2)=1; MSPT(33)=0; MSTP(128)=2; MSPT(82)=4; PARP(82)=1.
- In common block PYDAT1: MSTJ(11)=3; MSTJ(27)=1; PARJ(55)=-0.006
- In common block PYDAT2: PMASS(6,1)=175.
- In common block PYDATR: MRPY(1)=19780503
- In common block PYSUBS: MSEL=6

If you select MSPT(81)=0 then you will get the following warning:
!!!!!!!!!! WARNING ON PYTHIA LOOPING !!!!!!!!
YOU HAVE SWITCHED OFF MULTIPLE INTERACTIONS, mstp(81) = 0
THE DEFAULT ATLAS MULTIPLE INTERACTIONS SCENARIO, mstp(82) = 4
CHANGED TO mstp(82) = 1, BECAUSE PYTHIA IS LOOPING WHEN
mstp(81) = 0 and mstp(82) > 2

Random Numbers

Pythia.cxx is using the AtRndmGenSvc Athena Service to provide to Pythia (via the pyr function, found in Pythia_i/src/pyr.F) the necessary random numbers. This service is using the RanecuEngine of CLHEP, and is based on the "stream" logic, each stream being able to provide an idependent sequence of random numbers. Pythia.cxx is using two streams: PYTHIA_INIT and PYTHIA. The first stream is used to provide random numbers for the initialization phase of Pythia and the second one for the event generation. The user can set the initial seeds of each stream via the following option in the jobOption file.

```
AtRndmGenSvc.Seeds = [‘‘PYTHIA_INIT 2345533 9922199’’, ‘‘PYTHIA 5498921 659091’’]
```

The above sets the seeds of the PYTHIA_INIT stream to 2345533 and 9922199 and of the PYTHIA one to 5498921 and 659091. If the user will not set the seeds of a stream then the AtRndmGenSvc will use default values.

The seeds of the Random number service are saved for each event in the HepMC Event record and they are printed on screen by DumpMC. In this way an event can be reproduced easily. The user has to rerun the job by simply seting the seeds of the PYTHIA stream (the seeds of the PYTHIA_INIT stream should stay the same) to the seeds of that event.

Additionaly the AtRndmGenSvc is dumping into a file (AtRndmGenSvc.out) the seeds of all the streams at the end of the job. This file can be read back by the service if the user set the option

```
AtRndmGenSvc.ReadFromFile = true
```

(default = false). In this case the file AtRndmGenSvc.out is read and the streams saved in this file are created with seeds as in this file. The name of the file to be read can be set by the user via the option

```
AtRndmGenSvc.FileToRead = MyFileName
```

The above file is also written out when a job crashes. **This last option (when job crashing) is currently not working, waiting for a modification in Athena**.

The **Pythia_i/share/jobOptions.pythia.py** contains the above options.

User modifications

Two types of user modifications are common

- If you are trying to replace an existing routine that is in the Pythia library this is straightforward. Assume that you are trying to replace test.f that exists in Pythia. Check out Pythia_i under CMT, (use cmt co -r Pythia_i-xx-xx-xx Generators/Pythia_i where xx-xx-xx is the version in the release that you are running against), put your version of test.f into the /src area of the checked out code. Then in the /cmt area edit the requirements file and add test..f into the list of files that get complied. Note that each generator has its own library. You must therefore put your file in the right place. For Pythia, here is the example.

- If you want to access "External Process". This is done in Pythia by setting "USER"(see section 9.9 of the 6.2 Pythia manual). External procesess usually read a file containting events. At present four externals are available. The first is CompHep, the second is AcerMC, the third is Alpgen and the fourth is a dummy that users can adapt to their needs by following the CompHep example.
  "pyinit user comphep"
  sets the USER state in the call to PYINIT and then loads the comphep example. This will run without modification. If the user wants to process another type of compHep event, the following procedure must be followed. Modify the file inparmCompHep.dat that you will find in your run area. You specify the name or your event file here. No Athena recompilation is needed.

  To hadronize **Alpgen** generated events with Pythia, you need to link the file with the unweighted events produced by Alpgen to the file alpgen.unw_events. Then you only need to run athena with the jobOptions file jobOptions.AlpgenPythia.py by typing in the prompt
  athena jobOptions.AlpgenPythia.py
  It is possible to hadronize the same events using Herwig (look at the Herwig documentation)

  To hadronize **AcerMC** generated events with Pythia, you only need to run athena with the jobOptions file jobOptions.AcerMCPythia.py by typing in the prompt

athena jobOptions.AcerMCPythia.py
It is possible to hadronize the same events using Herwig (look at the Herwig documentation)

To add your own external, create your own inituser.f and useuser.f, put these filenames into the cmt/requirements so that they are built into libPythia_i.so and then rebuild the Pythia_i package (cmt make) and then set
"pyinit user user"