Table of Contents

Generate, digitize and reconstruct two signal events with decays

Bd->J/Psi(mumu)Kshort

This tutorial shows you how to simulate, digitize and reconstruct signal events so that they can be analyzed in DaVinci. The versions of the code quoted are available in July 2013 and replace the version numbers with more recent ones if required. This is setup to generate Sim08 2012 events.

Using Gauss to simulate events

The slides that explain what the program Gauss does are here.

Setup a version of Gauss from the default installation at your site (or CERN)

Log in to lxplus or if it is installed locally any computer with access to the LHCb software. If CVMFS is available it is the recommended way to access the LHCb software, setup with

```
export VO_LHCB_SW_DIR=/cvmfs/lhcb.cern.ch
source $VO_LHCB_SW_DIR/lib/LbLogin.sh -c x86_64-slc5-gcc46-opt
```

this uses the code in slc5 compatibility mode as at the time of writing the slc6 versions are not available.

```
cd SomeDirectory
SetupProject Gauss v45r3
cp $GAUSSOPTS/Gauss-Job.py ./
emacs -nw Gauss-Job.py
```

The SetupProject sets the environment variables to run Gauss with the specific version. That line must always be executed each session before Gauss is run, although it is only needed once per terminal. `SomeDirectory` is a directory where the options, outputs and log files for the jobs will be stored. You should probably create this using `mkdir` if you do not have a suitable place already. The `emacs` line can be replaced with an editor of your choice.

Edit `Gauss-Job.py` to simulate the required events

Change the `5` in line

```
nEvts = 5
```

so that Gauss generates two events.

The output file name will be automatically generated by Gauss, override this if you want by uncommenting the appropriate lines.

Save the file and quit `emacs`

Find out the event type for Bd->J/Psi(mumu)Kshort

Go into the EvtGen web page linked from that of DecFiles and find out the event type for the decay you want to simulate. You will want to generate events with the decay products in the acceptance so pick one with

`DecProdCut` in the name (the table is at the end). For Bd->J/Psi(mumu)Kshort this means you will pick `11144103`. Clicking on the number of the decay will take you to the official file in python detailing how the decay will be generated.

Run `Gauss` and look at the monitoring output

To run Gauss you will use the command `gaudirun.py` and specify that you want

- a standard Gauss jobs with the 2012 geometry
- the event type you have chosen
- what is specific to your job (i.e. number of events, run and first event numbers, name of output file,...)

You will do so providing the appropriate arguments to the command, as in this case

```
gaudirun.py Gauss-Job.py $GAUSSOPTS/Gauss-2011.py $DECFILESROOT/options/11144103.py | tee BsJPsiK
```

Wait while Gauss configures itself, Pythia, EvtGen and GEANT4, then generates two events.

You should see a very long file produced, the bit where it tells you about the particles generated is here:

```
======================== Generators Statistics =====================
=                                                                  =
= Number of particles generated: 1330
= Number of events: 2
= Mean multiplicity: 665
=                                                                  =
= Number of pseudo stable particles generated: 1120
= Number of events: 2
= Mean pseudo stable multiplicity: 560
=                                                                  =
= Number of charged stable particles generated: 344
= Number of events: 2
= Mean charged stable multiplicity: 172
=                                                                  =
= Number of charged stable particles in LHCb eta 79
= Number of events: 2
= Mean charged stable multiplicity in LHCb eta: 39.5
=                                                                  =
====================================================================
```

showing the number of particle types created in each event. Note this is not the end of the log file and may have scrolled off the top of the screen. The `| tee` bit of above command means there is a copy of the output file in `BsJPsiKs-2evt.log`.

Look at the histogram file produced called something like `Gauss-11144103-2ev-20130724-histos.root` and check there are hits in the VELO, RICH etc. The number of hits is also listed in the output log file.

```
    Counter      |      #      |     sum      | mean/eff^* | rms/err^*  |
"#VeloPU MCHits |          2 |         121 |     60.500 |     35.500 |
"#MCRichTracks" |          2 |         227 |     113.50 |     63.500 |
"#MCRichSegment |          2 |         301 |     150.50 |     82.500 |
...
```

Digitize the two decays made in Gauss

Setup access to a version of Boole

```
SetupProject Boole v26r6
```

Create `Boole-2012.py` to setup Boole to digitize the events generated earlier

Write the file `Boole-2012.py` containg

```
from Configurables import Boole
Boole().DataType  = "2012"
from Configurables import LHCbApp
LHCbApp().DDDBtag   = "Sim08-20130503-1"
LHCbApp().CondDBtag = "Sim08-20130503-1-vc-md100"
inputFiles = ['Gauss-11144103-2ev-20130724.sim'] # wants a list of files
from GaudiConf import IOHelper
IOHelper('ROOT').inputFiles(inputFiles)
# name the ouput file the same as the first input file with Gauss->Boole
# DatasetName also sets histogram output file name
Boole().DatasetName = inputFiles[0].replace('Gauss','Boole').replace('.sim','')
```

Change the input data file to the `Gauss-11144103-2ev-20130724.sim` file you generated earlier in Gauss (note the date part will be different to this one!). Also you must always make the DDDBtag and CondDBtag values match as these control the detector geometry and alignment. This must be the same in Gauss and Boole, the values used in Gauss can be found from the file $GAUSSOPTS/Gauss-2012.py which you used earlier and is also in the Gauss log file near the top. Check the values above are still correct.

Run Boole and check the output

Run Boole with `gaudirun.py Boole-2012.py | tee BsJPsiKs-2evt_Boole.log`

Look at the histograms produced by Boole in `BsJPsiKs-2evt-histos.root`, check there are entries in the histograms.

Use Moore to simulate the trigger for the events

Use the same instructions as for Gauss and Boole to setup Moore version v14r8p1. Note in general you have to match a TCK version to a specific Moore version. One way to find compatible versions is to check the bookkeeping for what was used in official productions.

Setup an configuration file for Moore, which sets the trigger TCK to configure a consistent trigger: Save the following to a file called `Moore-2012.py` (correct the input file name to the one Boole created).

```
from Configurables import Moore
# match the DDDB and CondBD tags to the events
Moore().DDDBtag   = "Sim08-20130503-1"
Moore().CondDBtag = "Sim08-20130503-1-vc-md100"

# Load the files that configure the full L0 & HLT emulation
# The TCK version number sets the trigger conditions
# this is the same as adding the files to the gaudirun.py command
from Gaudi.Configuration import importOptions
importOptions("$APPCONFIGOPTS/Moore/MooreSimProductionWithL0Emulation.py")
importOptions("$APPCONFIGOPTS/Conditions/TCK-0x409f0045.py")
importOptions("$APPCONFIGOPTS/Moore/DataType-2012.py")
importOptions("$APPCONFIGOPTS/L0/L0TCK-0x0045.py")

fileList = ['Boole-11144103-2ev-20130724.digi']
```

```
Moore().inputFiles = fileList
# output filename is same as input file with Boole->Moore (both .digi)
Moore().outputFile = Moore().inputFiles[0].replace('Boole','Moore')
```

then run with

```
gaudirun.py Moore-2012.py | tee BsJPsiKs-2evt_Moore.log
```

Note the output will be very long as we have a lot of trigger lines.

Use Brunel to reconstruct the events digitized with Boole

Setup Brunel version v44r5.

The Brunel options required to run the job are the values of DDDBtag and CondDBtag, the input file name and type and the required output from Brunel.

We can write a file that sets all of this:

```
from Gaudi.Configuration import *
from Configurables import Brunel, LHCbApp
LHCbApp().DDDBtag   = "Sim08-20130503-1"
LHCbApp().CondDBtag = "Sim08-20130503-1-vc-md100"

inputFiles = ['Moore-11144103-2ev-20130724.digi']
from GaudiConf import IOHelper
IOHelper('ROOT').inputFiles(inputFiles)
# sets output and histogram file names
Brunel().DatasetName = inputFiles[0].replace('Moore','Brunel').replace('.digi','')
Brunel().DataType = '2012'  # sets the 2011 configuration of Brunel
Brunel().InputType = "DIGI" # input has the format digi
Brunel().WithMC    = True   # use the MC truth information in the digi file
```

Save the above in `Brunel-2012.py` to set up Brunel.

Then run Brunel with `gaudirun.py Brunel-2012.py | tee BsJPsiKs-2evt_Brunel.log`

Again check the output log file and the histograms produced.

-- DavidHutchcroft - 24-Jul-2013

This topic: LHCb > SimDigiReconTutorial
Topic revision: r19 - 2014-05-19 - NathanaelFarley

Copyright &© 2008-2021 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
or Ideas, requests, problems regarding TWiki? use Discourse or Send feedback