Monte Carlo School
20 – 23 April 2009

HepMCAnalysis Exercises

These exercises will use the HepMCAnalysis package to analyse events generated with either different MC generators or from different processes.

The aim of the exercise is to separate SM background (here W + jets) from SUSY signal using standard observables. You should calculate these observables from the event record and fill histograms using the UserAnalysis class from the HepMCAnalysis package. Events are read in using the HepMCReader application from the examples section of the HepMCAnalysis package. The format is the HepMC ASCII format.

For further documentation of the HepMCAnalysis package see http://hepmcanalysistool.desy.de/. The online documentation of the classes can be found here: http://hepmcanalysistool.desy.de/Doxygen/html/index.html

In the baseAnalysis class of the HepMCAnalysis package a few helper functions are implemented to make your life easier. The most important ones are described here. See the documentation for full details. The UserAnalysis class derives from the baseAnalysis class, so all discussed member functions and data members from the baseAnalysis class are available in the UserAnalysis class.

- **Histograms:** You can easily book a 1d ROOT histogram using the `baseAnalysis::initHist()` member function:
  TH1D* initHist(string name, string title, string xlabel, int nrBins=100, double xmin=0., double xmax=100.)

  All histograms booked via `initHist()` will be written automatically to a ROOT file at the end of the job.

- **JetFinder:** For every analysis class the SISCone jet finder from the FastJet package is run. It uses 1.0 for the cone and 0.1 as overlap threshold. It only finds jets in the central region of the event. This means $|\eta| < 2.5$.
  The jets are stored in the `baseAnalysis::m_inclusive_jets` vector. The size of the vector gives directly the number of jets. Each jet is of class `fastjet::PseudoJet`. This class has a similar interface as the `HepMC::FourVector` class. For example the

member function `eta()` returns the pseudorapidity (η) of the jet/particle and the member function `perp()` the transverse momentum.

- Missing transverse energy: Missing E_T is defined as the transverse component of the sum of all stable, invisible particles:

  $$E_T^{mis} = \sqrt{(\sum_i^{invis} p_x)^2 + (\sum_i^{invis} p_y)^2}$$

  In the SM these are only the three neutrinos (and their anti particles). In case of SUSY it is the lightest supersymmetric particle (LSP). This is either the lightest neutralino in mSUGRA models or the gravitino in GMSB models.

  You can calculate missing ET using the `baseAnalysis::FindMissingEt()` member function. The result is stored in the `baseAnalysis::etMissTruth` data member.

- Effective mass: The effective mass is defined as the scalar sum of the missing E_T and the transverse momentum of the first leading jets:

  $$m_{eff} = E_T^{mis} + \sum_{jet=1}^{4} p_T^{jet}$$

  Usually, the ordering of jets is done in p_T. The jet with the highest p_T come first.

- In order to separate SM background from SUSY signal we will apply cuts on the following quantities:
  - missing E_T
  - number of jets
  - p_T and η of the first three jets

Installation

1. Installation of the analysis classes from the HepMCAnalysis package:
   a) go to the directory, where you want to work, e. g. ~/school
   b) untar the HepMCAnalysis source from the MC school AFS space:
   ```
   > tar -xzvf
    /afs/desy.de/group/alliance/mcg/public/mc_schools/2009/hep
   mcanalysis/MCS_HepMCAnalysis.tar.gz
   ```
   c) compile the analysis classes
   ```
   > cd HepMCAnalysis
   > source setup.sh
   > gmake
   ```

2. Installation of the HepMCReader application:

   a) go to the directory, where you have the HepMCAnalysis installed, e. g. ~/school/HepMCAnalysis
   b) setup the HepMCAnalysis environment:
   ```
   > source setup.sh
   ```
   c) go to the hepmcreader sub directory in the example directory
   ```
   > cd examples/hepmcreader
   ```
   d) compile the HepMCReader application, which gives the reader.exe executable
   ```
   > gmake
   ```
   e) all configuration, i. e. definition of input files, are done in the `Process.config` steering file. Run the default:
   ```
   > ./reader.exe Process.config
   ```

3. For large scale event processing we have prepared event files in HepMC format for the signal (0.5 million SUSY gluino events) and background (1million W + jets events) for all used generators. They are located in `/afs/desy.de/project/mcschool/data01-data07` in different subdirectories for different generators. The integrated cross section for these files/processes are written into the log files, which are stored in the same directories as the HepMC files.
   The `Process.config` files are in the example directory of the MC school AFS space:

   ```
   /afs/desy.de/group/alliance/mcg/public/mc_schools/2009/examples
   ```

   For example the file Process_Herwig++_W.config contains the config for the HepMCReader application to read 1 million W + jets events from Herwig++.

Exercises

In this exercise you will modify the UserAnalysis class from the HepMCAnalysis package. The header file UserAnalysis.h is locate in the include directory of the package. The source code of the class implementation UserAnalysis.cc in the src directory of the package. If you followed the default instructions they are here:

~/school/HepMCAnalysis/include/UserAnalysis.h

~/school/HepMCAnalysis/src/UserAnalysis.cc

If you need a new data member, e. g. pointer to a histogram, write it into the header file. Histograms need to be booked in the `Init()` member function. This member function is called once before the event loop. All other code, e. g. calculating variables and filling histograms should go into the `Process()` member function, which is called in the event loop for every event. If you modify either the header or the source file, you need to compile the HepMCAnalysis package again. See the *Instruction* section on the previous page for details. If you modify the header file you also need to compile the HepMCReader application again. See the *Instruction* section on the previous page for details.

To ease the start, all necessary histograms are already booked in the `Init()` member function of the UserAnalysis class. If you need a histogram, go through the list and pick up the name of the pointer to the histogram to access it. Check the histogram for the number of jets for a full example.

Input files:

When you develop your code, you usually run over a small amount of events to check if you code is doing what you want. The `Process.config` file from the example/hepmcreader directory contains a selection of input files from the different generators and processes. Look for the lines starting with InputFileNames and comment out the unnecessary lines using the hash symbol (#). You can also use files which you have produced on your own in the exercises. See the previous section for processing large amount of events.

Now to the exercises:

1. If you have not done yet, process 10 events with the reader.exe application and the default UserAnalysis class.

2. Modify the `Process()` member function of the UserAnalysis class in such a way, that the full event is dumped to the screen (the `Process()` member function is located in src/UserAnalysis.cc).

Hint: See documentation of `HepMC::GenEvent` class

3. Extract the jet quantities (number of jets, eta and p_T of the first three jets) and fill them into histograms. The number of jets is already implemented into the UserAnalysis class. For more details read the JetFinder part in the *Introduction* section on the first page.

   Hints: Be aware of units. The default units for the HepMC format is MeV and mm. But not all generators keep to this default. GeV and mm are also quite common.

4. Calculate missing E_T and fill it into a histogram.

   Hints: HepMCAnalysis can do most of the job for you: Search through the HepMCAnalysis documentation for the `baseAnalysis::FindMissingEt` member function and the `baseAnalysis::etMissTruth` data member. You need to call the member function, which will then calculate missing E_T and put it into the `etMissTruth` variable.

5. Calculate the effective mass and fill it into a histogram.

   Hint: The definition of the effective mass is given in the Introduction section on page 2.

6. **It's plotting time!** Run your code over a large amount of events and look at the histograms in ROOT. If you have already histograms for SM background and SUSY signal, try to overlay the corresponding histograms and either normalise them to unity or the cross section.

7. We would like to separate the SUSY signal from the SM background. Apply the following cuts and fill all quantities after the cuts again:
   - missing $E_T > 100$ GeV
   - at least three jets in the central region ($|\eta| < 2.5$)
   - for the first three jets $p_T > 50$ GeV

   Hints: The jet finder used in the HepMCAnalysis package applies already a cut of $|\eta| < 2.5$ on each jet. Nothing to do for you here.

8. **It's plotting time!** Run your code again over a large amount of events and look at the histograms with the observables after cuts. Now it is essential that you have used SM background and SUSY signal samples. Overlay the corresponding histograms and normalise them to the cross section.
   - Are the cuts sufficient to separate signal from background?
   - What is the SM background rejection rate?
   - What is the SUSY signal efficiency?

Example Solution

If you are stuck or want to see how a problem can be solved, have a look at the example solution (UserAnalysis.h and UserAnalysis.cc) in the following directory:

```
/afs/desy.de/group/alliance/mcg/public/mc_schools/2009/examples
```

You can run with this version of the UserAnalysis class, if you copy the header file in the HepMCAnalaysis/include directory and the source file into the HepMCAnalaysis/src directory. Don't forget to save your version of the UserAnalysis class before and compile the HepMCAnalysis package again.

Further Exercises for Experts

If you have time or want to learn more about HepMC or the physics, here are more exercises:

1. Implement your own calculation of missing E_T without and fill it into a histogram. Compare it on a event by event basis with the missing E_T from the `baseAnalysis` class.

   Hints: Calculate missing E_T on your own: Loop over all particles in the event and search for stable (status code 1) and invisible particles (neutrinos and the lightest neutralino). An example of the loop is given in the *HepMC in 5 seconds* sheet. For accessing the status code and PDG id of a particle see the `HepMC::GenParticle` class documentation. For the PDG id of neutrinos and the neutralino check the Monte Carlo Numbering Scheme in the *HepMC in 5 minutes* sheet.

2. Check, which W decays are simulated in the W+jet samples in the different generators. Loop again over all particles in the event and search Ws. You can get to the daughters via the decay vertex of the mother particle (W). Loop over all daughters and search either for electron, muon or tau. If none is found it must decay hadronically.

   Hints: See the `HepMC::GenParticle` documentation for accessing the PDG id and the decay vertex of a particle. See the `HepMC::GenVertex` documentation for accessing all out going particles (daughters) from a vertex.
   Not all generators write the same kind of information into the event record. For example, in Herwig++ the same W can be written more than once into the event record. The same W decays into a W into a W and finally into a muon and a neutrino. This can be even longer for Pythia8. Sherpa doesn't write the W at all into the event record and you have to look for a vertex where you have a lepton and the corresponding neutrino in the list of output particles. The best is, that you print and look through one or two events before.

Also for this exercises we have prepared example solutions. Have a look at the header and source file of the UserAnalysis class in

```
/afs/desy.de/group/alliance/mcg/public/mc_schools/2009/examples/experts
```

You can run with this version of the UserAnalysis class, if you copy the header file in the HepMCAnalaysis/include directory and the source file into the HepMCAnalaysis/src directory. Don't forget to save your version of the UserAnalysis class before and compile the HepMCAnalysis package again.