1st MCnet School, IPPP Durham
18–20 April 2007
updated October 2007

PYTHIA 8 Worksheet

Torbjörn Sjöstrand
CERN/PH and Department of Theoretical Physics, Lund University

1 Installation

Installing PYTHIA 8 standalone in your own directory is easy. Installing it as a generally available library and interfacing it with Rivet/RivetGun or other libraries is more complex, and is not covered in this introduction.

- Use your web browser to go to
  `http://www.thep.lu.se/~torbjorn/Pythia.html`.
- Right-click with your mouse on the link to
  `pythia8100.tgz`
  and decide where you want to download the file. (The left-click default location is installation-dependent, and could be your home directory or the Desktop one.)
- When in the directory where `pythia8100.tgz` was downloaded, type
  `tar xvfz pythia8100.tgz`
  This will create a new (sub)directory `pythia8100` where all files are unpacked.
- Move to this directory. Compile the program with
  `make`
  This will take about 3 minutes (of course computer-dependent).
- For test runs, move to the `examples` subdirectory. The programs `mainNN`, with `NN` from `01` through `17` (and a few more) can be executed with
  `make mainNN`
  `./mainNN.exe > outputfile`
  You can now study the contents of the `outputfile`. Example output is found in the `outNN` files.
- For the ambitious: to use all of the other main program you need to link also some external libraries, and in order to link them you have to know where they are located on your local computer. Given that, see the README files in the `pythia8100` directory and the `examples` subdirectory for details how to proceed.

2 A “Hello World” program

Assume you want to generate a $\mathrm{gg} \to \mathrm{t\bar{t}}$ event at the LHC.

To do this, first open a new file `mymain.cc` in the `examples` subdirectory. Then type the following lines (here interspersed with comments):

```
#include "Pythia.h"
```

to include the header file with classes and methods we need to use;

```
using namespace Pythia8;
```

to save us from having to type the `Pythia8::` qualifier all the time;

```
int main() {
```

to begin the main program;

```
Pythia pythia;
```

to declare an instance `pythia` of the main `Pythia` class;

```
pythia.readString("Top:gg2ttbar = on");
```

to send in a message to `pythia` to switch on the process `Top:gg2ttbar`;

```
pythia.init( 2212, 2212, 14000.);
```

to tell `pythia` to initialize for a collision process between two incoming protons (PDG code 2212) at 14000 GeV c.m. energy;

```
pythia.next();
```

to tell `pythia` to generate the next event (in this run the only one);

```
pythia.event.list();
```

to list the event information stored in the `event` member of `pythia`;

```
return 0; }
```

to finish the main program with an error-free return.

Next you need to edit the `Makefile` (the one in the `examples` subdirectory) to make it know what to do with `mymain.cc`. The lines

```
# Create an executable for one of the normal test programs
main00 main01 main02 main03 ...  main08 main09 main10 \
```

and the two next enumerate the main programs that do not need any external libraries. Edit the last of these lines to include also `mymain`:

```
 main21 main22 main23 main24 main25 main26 mymain: \
```

Now it should work as before with the other examples:

```
make mymain
./mymain.exe > outputfile
```

whereafter you can study `outputfile`.

3 A first realistic analysis

As a first realistic study, let us find the charged multiplicity distribution at LEP1. We can then take the above program and step-by-step modify it. (Rerun after each major modification.)

- To obtain an $\mathrm{e^+e^-}$ initial state, instead of a pp one, use that the PDG code for $\mathrm{e^-}$ is 11 and for $\mathrm{e^+}$ -11. The nominal c.m. energy for LEP1 is 91.2 GeV. The relevant

production process is now `WeakSingleBoson:ffbar2gmZ` (`f` here indicates that the process works with any incoming fermion species and `gmZ` that the process involves the quantum mechanical combination of γ^* and Z^0 exchange).

- Some of the energy is lost by photon radiation off the incoming $\mathrm{e}^{\pm}$, but published data is usually corrected for this, so use `pythia.readString(...)` to switch off such radiation, `"PDF:lepton = off"`.
- To give some variety of event types, introduce a loop over 5 events (to begin with), i.e. `pythia.next()` should be executed this many times.
- For the analysis you need to be able to access the information in the event record, as shown by the `pythia.event.list()` listing. To loop over all particles in it, use
  ```
  for (int i = 0; i < pythia.event.size(); ++i) {
  ```
  with a matching closing curly bracket further down.
- Inside the loop you can access the properties of particle `pythia.event[i]`. For instance, the method `id()` returns the PDG identity code, so if you insert
  ```
  cout << "i = " << i << " id = " << pythia.event[i].id() << endl;
  ```
  you should get a table of identity codes that matches the one in the event listing.
- The event listing contains all partons and particles, traced through a number of intermediate steps. Since we want to study the observable charged multiplicity distribution, only final-state particles are of interest. The `isFinal()` method returns `true` for final-state particles and `false` for the others. Correspondingly `isCharged()` tells whether the particle is charged or not. Modify the `cout << ...` statement to print only for final charged particles. Confirm that the selection works as it should. (Non-final particles have negative `status` codes and their names enclosed in brackets in the event listing.)
- Introduce a counter `nChg` for the charged multiplicity, initially set zero for each new event, and updated by one for each final charged particle. Print the value of `nChg` for each event.
- We now want to generate more events, say 10 000, to view the shape of the multiplicity distribution. You then need to remove the printing for each event. In its place book a histogram
  ```
  Hist nCharged("charged multiplicity", 100, -0.5, 99.5);
  ```
  before the event loop, store the value of `nChg`
  ```
  nCharged.fill(nChg);
  ```
  for each event, and write out the histogram
  ```
  cout << nCharged;
  ```
  after the event loop.
- As you notice, there is a peak at zero or two particles, coming from decays to neutrinos or leptons. To get rid of these, use `pythia.readString(...)` to switch off all decays of the Z^0, `"23:onMode = off"`, and then switch back on the ones of any of the five lighter quarks, `"23:onIfAny = 1 2 3 4 5"`.

4 Some further examples

1. Study the distribution of $\ln(1/x_p)$ for final charged particles in the above $\mathrm{e}^+\mathrm{e}^-$ case. Here $x_p = 2|\mathbf{p}|/E_{\mathrm{cm}}$. You can obtain the $|\mathbf{p}|$ value of a particle with the `pAbs()` method. Study how the position of the peak changes if you vary the energy. A variation roughly like

$$\ln(1/x_p)_{\max} \approx \mathrm{const} + \frac{1}{2}\ln E_{\mathrm{cm}}$$

   is suggested by perturbative QCD with coherence effects included.

2. Instead of quark production, consider that of a $\mu^+\mu^-$ pair, using the `onIfAny` command with 13, the μ^- code. Owing to photon radiation the $\mu^+\mu^-$ pair mass can be below the nominal 91.2 GeV energy. Plot this distribution, using the methods `px()`, `py()`, `pz()` and `e()` to extract momenta and energies for the particles. Note that the muons can appear in several copies, and it is the last one of each that is at the end of all photon emissions. Study the distribution both with and without `"PDF:lepton = off"` to observe the additional effects of radiation off the incoming $\mathrm{e}^{\pm}$.

3. Return to LHC physics and study minimum-bias events, i.e. the process `SoftQCD:minBias`. Plot the expected distribution of the charged multiplicity per unit of pseudorapidity η, $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta$, using the `eta()` method to obtain η values. Figure out how far out your histogram needs to go in η to catch (almost) all particles. Note that you can simply multiply your histogram by a double-precision number to normalize it appropriately. Warning: LHC events are more time-consuming than LEP ones, so reduce statistics.

4. The prediction for the above distribution depends on a number of model details. These are described e.g. if you go to the online manual found on

   `http://www.thep.lu.se/~torbjorn/php8100/Welcome.php`

   and click on the "Multiple Interactions" link in the index. Assume you want to change a few of these (for now more or less at random). Then use the instructions on the "Save Settings" page in the index to save a file with all the changes you want to do. Read in the instructions stored in this file using the `pythia.readFile("filename")` method, which automatically calls on `pythia.readString(...)` to interpret each line in the file. The file should be stored in the normal `examples` subdirectory, or else you need to provide the full file path along with the name. After the `readString(...)` and `readFile(...)` commands you can insert `pythia.settings.listChanged()` to check that the program got it right.

5. Pick some of the currently implemented processes, using the "Process Selection" webpages and impose relevant "Phase Space Cuts". Print integrated cross sections by inserting `pythia.statistics()` at the end of the run.