pythia_tutorials_1

In this tutorial we will learn pythia8. We will assume that you already have pythia8 installed and working properly (if not visit their documentation). Since this might be a new subject, you must remember these things before learning anything new, which is why I made a summary of protocols and practices that will make you more focused and make your learning session more effective and efficient. You can find this summary, which is optional to read, at the bottom of this tutorial.

Starting with CERN-LCGAPP-2007-04, we see that pythia models the complexity of particle collisions by breaking each event into many smaller tasks such as hard scattering, MPI, etc. in MC.

The most important program element is the **Event** class, which has two objects: * **process**: initial matrix element ** **event** : more general, cover incoming beams to final hadrons.

OK so let's go to the worksheet.

The correct worksheet for your version is found in share/Pythia8/pdfdoc/worksheet8200.pdf

So, after making sure it works fine (I had to recompile mine, just `./configure && make -j 8` for stadalone, go to /examples directory, do `make main01` and then `./main01` to get an example running. Open this file in your editor to see what's in it.

An important aspect of the displayed event record is that many duplicates of the same particle may exist, but only those with positive status code make it to the final state. For example if you have a branching $q \to qg$ then q is initially positive status code but then becomes negative and the new q and g are now positive.

Now make your own code. If you have version `8.eCall this `main200.cc` as opposed to `mymain.cc` which is instricted in the worksheet, which uses version `8.1` while my installation is is `8.244`. This is unless you want to change the `Makefile` to include your different naming scheme `mymin`, because the their Makefile allows for complication for any file starting with `main` and they end at 113 or something, so why not just start with 200.

Move all the code files in this tutorial in the `exampels` directory of your pythia8.2 installation. If you have version `8.2`, edit the `examples/Makefile`. To allow the compilation of one extra file, include

`mymain01 : mymain01.cc $(PREFIX_LIB)/libpythia8.a $(CXX) $< -o $@ $(CXX_COMMON) $(GZIP_INC) $(GZIP_FLAGS)`

To allow for compilation of `mymain01.cc-mymain05.cc`, include

`mymain01 mymain02 mymain03 mymain04 mymain05: $$@.cc\ $(PREFIX_LIB)/libpythia8.a $(CXX) $< -o $@ $(CXX_COMMON) $(GZIP_INC) $(GZIP_FLAGS)`

in the `Makefile`.

Now include the following in your `mymain01.cc`

mymain01.cc

```
// Headers and Namespaces.
#include "Pythia8/Pythia.h" // Include Pythia headers.
using namespace Pythia8;
// Let Pythia8:: be implicit.
int main() {
// Begin main program.
// Set up generation.
Pythia pythia;
// Declare Pythia object
pythia.readString("Top:gg2ttbar = on"); // Switch on process.
pythia.readString("Beams:eCM = 8000."); // 8 TeV CM energy.
pythia.init(); // Initialize; incoming pp beams is default.
// Generate event(s).
pythia.next(); // Generate an(other) event. Fill event record.
return 0;
}
// End main program with error-free return.
```

and do `make mymain01 && ./mymain01`

All this printed information shows all the particles for one event (sheesh)! Now go to `Appendix A` to look at what all this printed information is in the event record.

So essentially the event record is listed ordered chronologically in the order that the particles were produced, where particles that decay are successively given a negative status. A bunch of info is given, and these values could be accessed for each particle by doing `event[i].method()` where the method could be status, etc. An important thing is the `id`, which is the PDG id for different particles, and an antiparticle is given the same number but with a negative sign. `status` codes as we discussed earlier are positive for particles that are added, but remain positive if they make it as final state particles. The different numbers describe *why* each particle was added. `isFinal()` method returns `true` if they are positive status code, i.e. final state particle, and returns `false` if they are not final state particles.

Now let's do a more complicated example, `mymain02.cc`:

we can add many `pythia.readString` calls to include many more processes

mymain02.cc: Doing an event loop with 5 events, with an extra process

```
// Headers and Namespaces.
#include "Pythia8/Pythia.h" // Include Pythia headers.
using namespace Pythia8;
// Let Pythia8:: be implicit.
int main() {
// Set up generation.
Pythia pythia;
// Declare Pythia object
pythia.readString("Top:gg2ttbar = on"); // Switch on process.
pythia.readString("Beams:eCM = 8000."); // 8 TeV CM energy.
pythia.init(); // Initialize; incoming pp beams is default.
// Generate 5 events.
for (int iEvent=0; iEvent < 5; ++iEvent) {

pythia.next(); // Generate an(other) event. Fill event record.
}
return 0;
}
```

This runs to generate 5 events. However, it shows only one event record. To show all 5 event records in succession you'll have to scroll up a lot), include `pythia.readString("Next:numberShowEvent = 5");` next to the readstring commands at the top.

mymain03.cc: Accessing each particle within the event loop

```
// Headers and Namespaces.
#include "Pythia8/Pythia.h" // Include Pythia headers.
using namespace Pythia8;
// Let Pythia8:: be implicit.
int main() {
// Set up generation.
Pythia pythia;
// Declare Pythia object
pythia.readString("Top:gg2ttbar = on"); // Switch on process.
pythia.readString("Beams:eCM = 8000."); // 8 TeV CM energy.
// pythia.readString("Next:numberShowEvent = 5");
pythia.init(); // Initialize; incoming pp beams is default.
// Generate 5 events.
for (int iEvent=0; iEvent < 5; ++iEvent) {

pythia.next(); // Generate an(other) event. Fill event record.
 //START PARTICLE LOOP
 for (int i = 0; i < pythia.event.size(); ++i) {

cout << "i = " << i << ", id = " << pythia.event[i].id() << endl;
}

}
return 0;
}
```

Here the particle in the event is `pythia.event[i]`, and you can access any method for the particle, e.g. `id()`, `status(),` etc. You can also fill histograms by first booking (initiallyzing) it with

```
Hist pT("transverse momentum", 100, 0., 200.);
```

and then filling it in the particle loop with `pT.fill(pythia.event[i].pT());` and then you can print it *after* your event loop with

```
cout << pT
```

Now, for using **input files, or "cards"**, which are files contatning all the parameters and seetings for the pythia program. With this, all the `pythia.readString()` commands are replaced by `pythia.readFile(filename)`.

using an input file "card"

mymain05.cmnd

````
Beams:idA = 2212 !first incoming beam is a 2212, i.e. a proton.
Beams:idB = 2212 !second beam is also a proton
Beams:eCM = 8000. the cm energy of collisions.
Top:gg2ttbar = on ! switch on the process g g -> t tbar.
Top:qqbar2ttbar = on !switch on the process q qbar -> t tbar.```

## mymain05.cc

```c
// Headers and Namespaces.
#include "Pythia8/Pythia.h" // Include Pythia headers.
using namespace Pythia8;
// Let Pythia8:: be implicit.
int main(int argc, char* argv[]) {
// Begin main program.
// Set up generation.
Pythia pythia;
// Declare Pythia object
pythia.readFile(argv[1]);

pythia.init(); // Initialize; incoming pp beams is default.
// Generate event(s).
pythia.next(); // Generate an(other) event. Fill event record.
return 0;
}
````

and then to run it with this card, while putting the output in `myout05` do

`./mymain05 mymain05.cmnd > myout05`

End of tutorial 1.

Optional: My Quick Summary on Practices/Routines for Enhanced Learning and Focus and Efficiency

- You have to get **motivated** and have a reason for learning this. It's going to be hard, but if you don't have a reason to motivate you, you will give up easily in the learning session.
````

But also realize that learning happens when things are difficult, stressful and uncomforatble! This is how new neural pathways get formed. Potential reasons for learning this could be: to give skills that would make you more employable to reach financial freedom, to discover the fundamental laws of physics, to help find laws/theories which will lead to technologies that will improve all our lives, not to let down your professor or embarass yourself in from of him, to be proud of yourself for accomplishing something difficult no matter what it is, to have these skills so that you can help others and teach them with it, to be more educated and reach enlightenment, etc.

- You have to be very focused, **put away all distractions, and only learn for the appropriate amount of time, which is 1:30 hrs**. This is the length of the ultradian cycle, and this is how much humans can stay focused on a task before losing attention. Make sure you're in a quiet place or with music playing, but that this is all you're doing.
- **Have goals of this learning session.** Remember that smart work > hard work but that hard work is more important. This means that longer time may not be as productive, because you may be distracted, stressed, etc.
    - My goad for this learning session: To generate parton level and particle level data for jets with pythia.
        - If I get this down I will match them using ΔR.
- **Make sure you're well rested** (have gotten 8 hours of sleep). This will ensure that you have greater willpower. Also make sure you are learning in the time of day where you tend to be most focused, which is mirrored by your cyrcadian cycle, which determines the amount of cortisol, dopamine, etc. in your body at any given time, which tends to be in the morning or daytime for most people.
- There is literature on the effect of a type of meditation which lasts 17 minutes prior to your working session that should considerably increase your focus in your session. The meditation is the following: **for 17 minutes you close your eyes and not try to think of anything outside of you in particular, instead focus on your breathing, your state (interroception), the sensations in your body** (e.g. what parts of your body are touching what surfaces, how do they feel, etc.).
- The more you focus on a small visual target, not moving your eyes, and not blinking, the more dopamine that will be released in your body and hence the more focused you'll be. A good practice is to **focus on a small visual target (e.g. the corner of your laptop) for 60 seconds, without blinking** (or with as little blinking as possible) before your session. With this you'll have more focus. Vice versa: the more you widen your gaze and look around and blink a lot the more relaxed and less focued you'll be.
- **Fidgeting or moving your legs, tapping your foot whilst working will also increase your focus.**

- **Other things like excercise, diet, balance, cold shower, meditation/NSDR, smell or other sensational triggers, etc. are also helpful.**
- **Make sure you have good posture (being upright with chest out, and such that you can have deep breaths)
- **Remember to deep breathe often during your session.**
- **YOU HAVE TO DO PRACTICE PROBLEMS, NOT JUST READ.**
- **Being in a well lit environment also makes you more alert.**
- **Accept everything and know that you've done your best.** You may not accomplish your goal, but know that you've done everything in your power, which is all that matters because it's the only thing in your control. There could be things beyond your control, e.g. you're naturally not as good in processing scientific information. That shouldn't bother you because in this session you've learned some new things and we know that the brain is plastic, so you will be better in the future,