LU TP 08–17
MCnet/08/09
September 2008

PYTHIA 8 Status Report*

Torbjörn Sjöstrand
Department of Theoretical Physics, Lund University

Abstract
PYTHIA 8, the C++ rewrite of the commonly-used PYTHIA event generator, is now available in a first full-fledged version 8.1. The older PYTHIA 6.4 generator in Fortran 77 is still maintained, for now, but users are strongly recommended to try out and move to the new version as soon as feasible.

1 Introduction

The "Lund Monte Carlo" family of event generators started in 1978 with the JETSET program. PYTHIA was begun a few years later, and the two eventually were joined under the PYTHIA label. Over the last 25 years the PYTHIA/JETSET program has been widely used to help understand the physics of high-energy collisions.

The program was from the onset written in Fortran 77, up to the current version 6.4 [1]. However, following the move of the experimental community to C++, a corresponding restart and rewrite was made for PYTHIA in 2004 – 2007, with most aspects cleaned up and modernized.

The first production quality release, PYTHIA 8.100, appeared towards the end of 2007 [2]. It was paced to arrive in time for LHC and therefore does not yet cover some physics topics. It has not yet caught on in the LHC experimental collaborations, however, and thus the older Fortran code is still maintained, even if at a reduced level.

2 Physics summary

Here follows a brief summary of the key physics aspects of PYTHIA 8.1, by topic.

Hard processes: The built-in library contains many leading-order processes, for the Standard Model almost all $2 \to 1$ and $2 \to 2$ ones and a few $2 \to 3$, beyond it a sprinkling of different processes, but not yet Supersymmetry or Technicolor. Parton-level events can also be input from external matrix-element-based generators, e.g. using Les Houches Event Files [3]. Also runtime interfaces are possible, and one such is provided to PYTHIA 6.4 for the generation of legacy processes. Resonance decays are included, often but not always with full angular correlations.

Parton showers: Transverse-momentum-ordered showers are used both for initial- and final-state radiation, the former based on backwards evolution. Implemented branchings are $q \to qg$, $g \to gg$, $g \to q\overline{q}$, $f \to f\gamma$ (f is a quark or lepton) and $\gamma \to f\overline{f}$. Recoils are handled in a dipole-style approach, but emissions are still associated with one emitting parton. Many processes include matching to matrix elements for the first (= hardest) emission; this especially concerns gluon emission in resonance decays.

Underlying events and minimum-bias events: PYTHIA implements a formalism with multiple parton–parton interactions, based on the standard QCD matrix elements for $2 \to 2$ processes, dampened in the $p_{\perp} \to 0$ limit. The collision rate is impact-parameter-dependent, and collisions are ordered in

* to appear in the proceedings of the HERA and the LHC workshop, 26–30 May 2008, CERN

decreasing $p_\perp$. Multiple interactions (MI) are therefore combined with initial- and final-state radiation (ISR and FSR) in one common sequence of decreasing transverse momenta $p_{\perp 1} > p_{\perp 2} > p_{\perp 3} \ldots$,

$$\begin{aligned} \left.\frac{\mathrm{d}\mathcal{P}}{\mathrm{d}p_\perp}\right|_{p_\perp = p_{\perp i}} &= \left(\frac{\mathrm{d}\mathcal{P}_{\mathrm{MI}}}{\mathrm{d}p_\perp} + \sum \frac{\mathrm{d}\mathcal{P}_{\mathrm{ISR}}}{\mathrm{d}p_\perp} + \sum \frac{\mathrm{d}\mathcal{P}_{\mathrm{FSR}}}{\mathrm{d}p_\perp} \right) \\ &\times \exp\left(- \int_{p_\perp}^{p_{\perp i-1}} \left(\frac{\mathrm{d}\mathcal{P}_{\mathrm{MI}}}{\mathrm{d}p'_\perp} + \sum \frac{\mathrm{d}\mathcal{P}_{\mathrm{ISR}}}{\mathrm{d}p'_\perp} + \sum \frac{\mathrm{d}\mathcal{P}_{\mathrm{FSR}}}{\mathrm{d}p'_\perp} \right) \mathrm{d}p'_\perp \right) , \end{aligned}$$

using the "winner takes all" Monte Carlo strategy. This leads to a competition, in particular between MI and ISR, for beam momentum. The beam remnants are colour-connected to the interacting subsystems, with a detailed modelling of the flavour and momentum structure, also for the parton densities to be used at each successive step. The framework also contains a model for colour reconnection, likely the least well understood aspect of this physics area, and therefore one that may require further development.

Hadronization: The Lund model for string fragmentation is used to describe the transition from coloured partons to colour singlet hadrons. Subsequent hadronic decays are usually described isotropic in phase space, but in some cases matrix-element information is inserted. It is also possible to link to external decay packages, e.g. for τ or B decays. A model for Bose–Einstein effects is included, but is off by default.

3 Program evolution

The above physics description largely also applies to PYTHIA 6.4. There are some differences to be noted, however.

Many old features have been definitely removed. Most notably this concerns the framework for independent fragmentation (a strawman alternative to string fragmentation) and the older mass-ordered showers (that still are in use in many collaborations, but do not fit so well with the new interleaved MI/ISR/FSR description).

Features that have been omitted so far, but should appear when time permits, include ep, γp and $\gamma\gamma$ beam configurations and a set of SUSY and Technicolor processes.

New features, relative to PYTHIA 6.4 include

- the interleaved MI/ISR/FSR evolution (6.4 only interleaved MI and ISR),
- a richer mix of underlying-event processes, no longer only QCD jets but also prompt photons, low-mass lepton pairs and J/ψ,
- possibility to select two hard processes in an event,
- possibility to use one PDF set for the hard process and another for MI/ISR, and
- updated decay data.

Major plans for the future include a new model for rescattering processes in the MI machinery, and new facilities to include matrix-element-to-parton-shower matching.

In addition minor improvements are introduced with each new subversion. Between the original 8.100 and the current 8.108 the list includes

- possibility to have acollinear beams, beam momentum spread and beam vertex spread,
- updated interfaces to several external packages,
- improved possibility to run several `Pythia` instances simultaneously,
- code modifications to compile under gcc 4.3.0 with the `-Wshadow` option, and
- some minor bug fixes.

4 Program structure

The structure of the PYTHIA 8 generator is illustrated in Fig. 1. The main class for all user interaction is called `Pythia`. It calls on the three classes

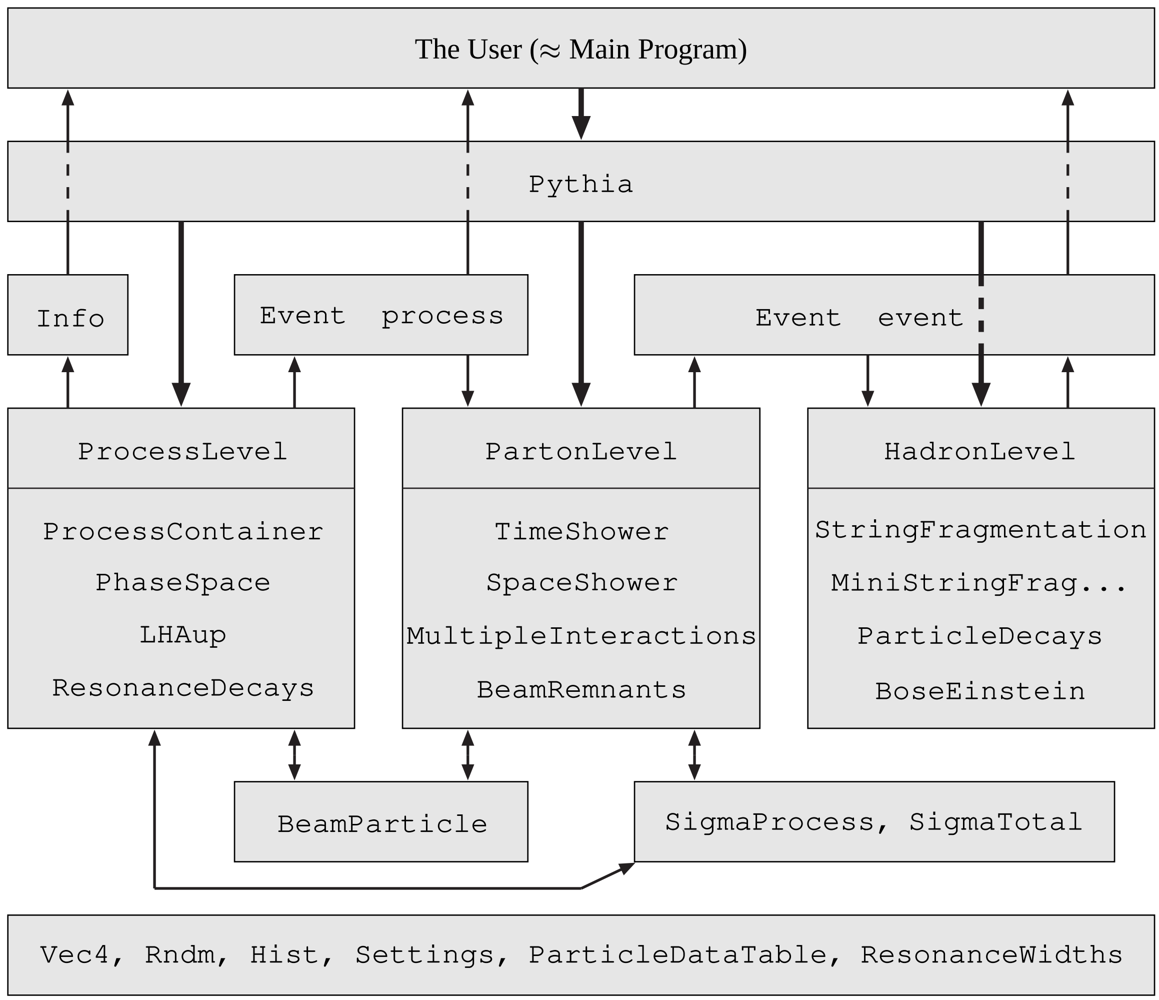

Fig. 1: The relationship between the main classes in PYTHIA 8. The thick arrows show the flow of commands to carry out different physics tasks, whereas the thinner show the flow of information between the tasks. The bottom box contains common utilities that may be used anywhere. Obviously the picture is strongly simplified.

- `ProcessLevel`, for the generation of the hard process, by sampling of built-in matrix elements or input from an external program,
- `PartonLevel`, for the additional partonic activity by MI, ISR, FSR and beam remnants, and
- `HadronLevel`, for the transition from partons to hadrons and the subsequent decays.

Each of these, in their turn, call on further classes that perform the separate kinds of physics tasks.

Information is flowing between the different program elements in various ways, the most important being the event record, represented by the `Event` class. Actually, there are two objects of this class, one called `process`, that only covers the few partons of the hard process above, and another called `event`, that covers the full story from the incoming beams to the final hadrons. A small `Info` class keeps track of useful one-of-a-kind information, such as kinematical variables of the hard process.

There are also two incoming `BeamParticles`, that keep track of the partonic content left in the beams after a number of interactions and initial-state radiations, and rescales parton distributions accordingly.

The process library, as well as parametrisations of total, elastic and diffractive cross sections, are used both by the hard-process selection machinery and the MI one.

The `Settings` database keeps track of all integer, double, boolean and string variables that can be changed by the user to steer the performance of PYTHIA, except that `ParticleDataTable` is its own separate database.

Finally, a number of utilities can be used just about anywhere, for Lorentz four-vectors, random numbers, jet finding, simple histograms, and for a number of other "minor" tasks.

5 Program usage

When you want to use PYTHIA 8 you are expected to provide the main program. At least the following commands should them be used:

- `#include "Pythia.h"` to gain access to all the relevant classes and methods,
- `using namespace Pythia8;` to simplify typing,
- `Pythia pythia;` to create an instance of the generator,
- `pythia.readString("command");` (repeated as required) to modify the default behaviour of the generator (see further below), or alternatively
- `pythia.readFile("filename");` to read in a whole file of commands, one per line,
- `pythia.init();` to initialize the generator, with different optional arguments to be used to set incoming beam particles and energies,
- `pythia.next();` to generate the next event, so this call would be placed inside the main event generation loop,
- `pythia.statistics();` to write out some summary information at the end of the run.

The `pythia.readString(...)` and `pythia.readFile(...)` methods are used to modify the values stored in the databases, and it is these that in turn govern the behaviour of the program. There are two main databases.

- `Settings` come in four kinds, boolean `flags`, integer `modes`, double-precision `parms`, and string `words`. In each case a change requires a statement of the form `task:property = value`, e.g. `TimeShower:pTmin = 1.0`.
- `ParticleDataTable` stores particle properties and decay tables. To change the former requires a statement of the form `id:property = value`, where `id` is the identity code of the particle, an integer. The latter instead requires the form `id:channel:property = value`, where `channel` is a consecutive numbering of the decay channels of a particle.

Commands to the two databases can be freely mixed. The structure with strings to be interpreted also allows some special tricks, like that one can write `on` instead of `true` and `off` instead of `false`, or that the matching to variable names in the databases is case-insensitive.

Information about all settings and particle data can be found in the online manual, which exists in three copies. The `xml` one is the master copy, which is read in when an instance of the generator is created, to set up the default values that subsequently can be modified. The same information is then also provided in a copy translated to more readable `html` format, and another copy in `php` format. The interactivity of the latter format allows a primitive graphical user interface, where a file of commands can be constructed by simple clicking and filling-in of boxes.

The online manual contains more than 60 interlinked webpages, from a program overview to some reference material, and in between extensive descriptions how to set up run tasks, how to study the output, and how to link to other programs. In particular, all possible settings are fully explained.

6 Trying it out

If you want to try out PYTHIA 8, here is how:

- Download `pythia8108.tgz` (or whatever is the current version when you read this) from `http://www.thep.lu.se/~torbjorn/Pythia.html`

- `tar xvfz pythia8108.tgz` to unzip and expand.
- `cd pythia8108` to move to the new directory.
- `./configure ...` is only needed to link to external libraries, or to use options for debug or shared libraries, so can be skipped in the first round.
- `make` will compile in $1-3$ minutes (for an archive library, same amount extra for a shared one).
- The `htmldoc/pythia8100.pdf` file contains A Brief Introduction [2].
- Open `htmldoc/Welcome.html` in a web browser for the full manual.
- Install the `phpdoc/` directory on a webserver and open `phpdoc/Welcome.php` in a web browser for an interactive manual.
- The `examples` subdirectory contains > 30 sample main programs: standalone, link to libraries, semi-internal processes, ...
- These can be run by `make mainNN` followed by `./mainNN.exe > outfile`.
- A `Worksheet` contains step-by-step instructions and exercises how to write and run main programs.

Note that PYTHIA is constructed so it can be run standalone, and this is the best way to learn how it works. For an experimental collaboration it would only be a piece in a larger software puzzle, and so a number of hooks has been prepared to allow various kinds of interfacing. The price to pay for using them is a more complex structure, where e.g. the origin of any errors is less easy to hunt down. Several aspects, such as the access to settings and particle data, should remain essentially unchanged, however.

7 Outlook

PYTHIA 6.4 is still maintained, with a current version 6.418 that weighs in at over 77,000 lines of code (including comments and blanks) and has a 580 page manual [1], plus update notes and sample main programs. No further major upgrades will occur with this program, however, and we intend to let it gradually die.

Instead PYTHIA 8.1 should be taking over. Currently it is smaller than its predecessor, with "only" 53,000 lines of code and a puny 27 page manual [2], but with much further online documentation and a big selection of sample main programs. It already contains several features not found in 6.4, and will gradually become the obvious version to use.

The LHC collaborations are strongly encouraged to accelerate the transition from 6.4 to 8.1, e.g. by serious tests with small production runs, to find any remaining flaws and limitations.

Acknowledgements

The PYTHIA 8 program was made possible by a three-year "sabbatical" with the SFT group at CERN/EP. This unwavering support is gratefully acknowledged. Mikhail Kirsanov and other members of the GENSER group has provided further help with Makefiles and other technical tasks. The work was supported in part by the European Union Marie Curie Research Training Network MCnet under contract MRTN-CT-2006-035606.

References

[1] Sjöstrand, T. and Mrenna, S. and Skands, P., JHEP **05**, 026 (2006).

[2] Sjöstrand, T. and Mrenna, S. and Skands, P., Comput. Phys. Commun. **178**, 852 (2008).

[3] Alwall, J. and others, Comput. Phys. Commun. **176**, 300 (2007).