Dartmouth

TRIUMF

The lecture will begin shortly. Please mute your microphone until you are ready to speak.

PYTHIA8

Bootcamp

Part I

Stephen Mrenna
Fermilab[1]

October 25, 2017

[1] adapted from worksheet of T. Sjöstrand and S. Prestel

Notes for the Bootcamp 2017

There is an accompanying worksheet for the tutorial we run at MC4BSM and Summer Schools

You should view the tutorial sheet as a user reference. So do not be scared of the length of the document. We won't do it all.

Appendix A contains a brief summary of the event-record structure, and Appendix B some notes on simple histogramming and jet finding. Appendix C and D cover some technical details.

Objective: teach you the basics of how to use the PYTHIA 8.2 event generator to study various physics aspects.

Physics models are described in PYTHIA 8.2 and PYTHIA 6.4 manuals, and references within

Installation for the Bootcamp tutorial 2017

If you haven't done this, do it now while I'm talking!

You can find the virtual machine and instructions at
`http://www.slac.stanford.edu/~shoeche/cteq17/`

Your virtual machine already contains a very general installation of PYTHIA 8 (version 8.226).

The examples directory containing many example main programs can be found under `/opt/hep/share/Pythia8/examples`.

The `examples` directory that you should use for the bootcamp is `~/tutorials/mc/pythia`.

You will be asked to copy from one directory to the other

VM should be useful for S. Hoeche's SHERPA bootcamp

Bootcamp Overview

What it IS NOT

Pedagogical lectures on Physics models
See intro to S. Hoeche's SHERPA lectures and the above PYTHIA physics manuals

Displays of comparisons between predictions and data

What it IS

Hands-on examples to demonstrate PYTHIA as an investigative tool
More physics in an event record than in a textbook

The Code I

PYTHIA 8 is small, self-contained, and easy to install

Stand alone usage is good if you want to have it on your own laptop, or if you want to explore physics or debug code without any danger of destructive interference between different libraries.

You are expected to write the main program

We will go through several examples, some more flexible than others

The Code 2

While PYTHIA can be run stand alone, it can also be interfaced with a set of other libraries.

HEPMC is the standard format used by experimentalists to store generated events. Installation hints are provided in Appendix C.

RIVET uses HEPMC output to perform comparisons with analysis results, relying also on YODA

LHAPDF is the standard library for parton distribution functions

Further main programs included with the PYTHIA code provide examples of linking, e.g., to MADGRAPH, FASTJET, ROOT

The RIVET homepage has some nice scripts for installing many of these packages

I use HOMEBREW on a Mac

A “Hello World” program

We will now generate some QCD events at an early run of the LHC using PYTHIA standalone.

Please do the following:

```
cp /opt/hep/share/Pythia8/examples/main01.cc mymain01.cc
```

```
make mymain01
```

```
./mymain01 > mymain01.out
```

Please open `leafpad` and change the font to COURIER (this makes the output more readable)

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
    if (!pythia.next()) continue;
    // Find number of all final charged particles and fill histogram.
    int nCharged = 0;
    for (int i = 0; i < pythia.event.size(); ++i)
      if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
        ++nCharged;
    mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout « mult;
  return 0;
}
```

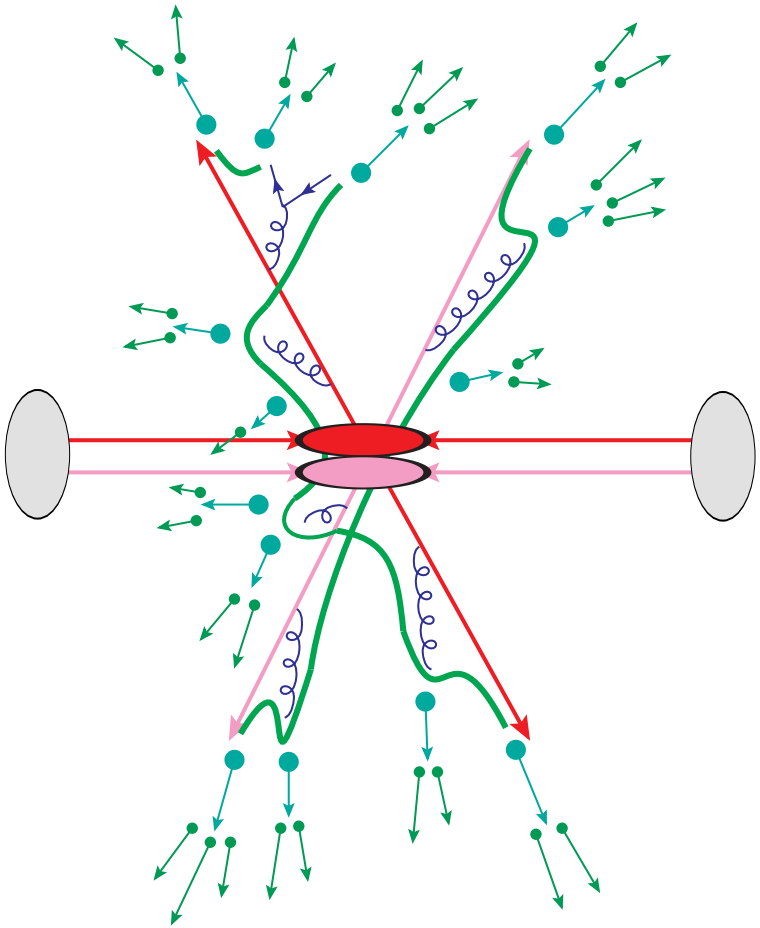
```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
    if (!pythia.next()) continue;
    // Find number of all final charged particles and fill histogram.
    int nCharged = 0;
    for (int i = 0; i < pythia.event.size(); ++i)
      if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
        ++nCharged;
    mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout « mult;
  return 0;
}
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
    if (!pythia.next()) continue;
    // Find number of all final charged particles and fill histogram.
    int nCharged = 0;
    for (int i = 0; i < pythia.event.size(); ++i)
      if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
        ++nCharged;
    mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout « mult;
  return 0;
}
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
    if (!pythia.next()) continue;
    // Find number of all final charged particles and fill histogram.
    int nCharged = 0;
    for (int i = 0; i < pythia.event.size(); ++i)
      if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
        ++nCharged;
    mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout « mult;
  return 0;
}
```

What is Pythia8?

A machine that generates realistic, high-energy particle collision event records using all the Standard Model physics we know.



Short-distance cross section:

μ_r^H, μ_f^H, PDF^H, α_s^H

Parton shower:

μ_q^{PS}, μ_r^{PS}, μ_f^{PS}, μ_{cut}^{PS}, PDF^{PS}, α_s^{PS}

Multiple interactions:

μ_q^{MPI}, PDF^{MPI}, α_s^{MPI} ...

String fragmentation:

$f(z)$, string p_T, tension

Beams:

Primordial k_T, remnants

Particle Decays:

BRs, MEs, BE correlations

...

```
2017-10-24 14:41       charged multiplicity

      9.00*10^ 0                   X     X
      8.70*10^ 0                   X     X
      8.40*10^ 0                   X     X
      8.10*10^ 0                   X     X
      7.80*10^ 0                   X     X
      7.50*10^ 0                   X     X
      7.20*10^ 0                   X     X
      6.90*10^ 0                   X     X
      6.60*10^ 0                   X     X
      6.30*10^ 0                   X     X
      6.00*10^ 0              X    X   X X   X
      5.70*10^ 0              X    X   X X   X
      5.40*10^ 0              X    X   X X   X
      5.10*10^ 0              X    X   X X 7 X
      4.80*10^ 0              X    X   X X X X
      4.50*10^ 0              X    X   X X X X
      4.20*10^ 0              X    X  3X X3X X
      3.90*10^ 0              X    X  XX XXX X
      3.60*10^ 0              X    X  XX XXX X
      3.30*10^ 0              X    X  XX XXX X
      3.00*10^ 0         X   XXXXXXXX XXXXXX X   X
      2.70*10^ 0         X   XXXXXXXX XXXXXX X   X
      2.40*10^ 0         X   XXXXXXXX XXXXXX X   X
      2.10*10^ 0        7X7 7XXXXXXXX7XXXXXX X7 7X 77   7
      1.80*10^ 0        XXX XXXXXXXXXXXXXXXX XX XX XX   X
      1.50*10^ 0        XXX XXXXXXXXXXXXXXXX XX XX XX   X
      1.20*10^ 0       3XXX3XXXXXXXXXXXXXXXX XX3XX XX3  X  33
      0.90*10^ 0       XXXXXXXXXXXXXXXXXXXXX XXXXX XXX  X  XX
      0.60*10^ 0       XXXXXXXXXXXXXXXXXXXXX XXXXX XXX  X  XX
      0.30*10^ 0       XXXXXXXXXXXXXXXXXXXXX XXXXX XXX  X  XX

        Contents
          *10^ 0  00000012321236333393246394506212302210020011000000000000000000000000000000000000000000000000000000000000000000
          *10^-1  00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
          *10^-2  00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
          *10^-3  00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

        Low edge  -
          *10^ 2  00000000000001111111111112222222222222333333333333444444444444455555555555566666666666667777777777
          *10^ 1  00123445678901223456678900123445678890122345667890012344567889012345667890012344567889012234566789
          *10^ 0  18642086420864208642086420864208642086420864208642086420864208642086420864208642086420864208642086

 Entries  =         100    Mean =  1.7430e+02    Underflow =  0.0000e+00    Low edge  = -5.0000e-01
 All chan =  1.0000e+02    Rms  =  6.5496e+01    Overflow  =  0.0000e+00    High edge =  7.9950e+02
```

```
2017-10-24 14:41       charged multiplicity

      9.00*10^ 0                   X     X
      8.70*10^ 0                   X     X
      8.40*10^ 0                   X     X
      8.10*10^ 0                   X     X
      7.80*10^ 0                   X     X
      7.50*10^ 0                   X     X
      7.20*10^ 0                   X     X
      6.90*10^ 0                   X     X
      6.60*10^ 0                   X     X
      6.30*10^ 0                   X     X
      6.00*10^ 0              X    X   X X   X
      5.70*10^ 0              X    X   X X   X
      5.40*10^ 0              X    X   X X   X
      5.10*10^ 0              X    X   X X 7 X
      4.80*10^ 0              X    X   X X X X
      4.50*10^ 0              X    X   X X X X
      4.20*10^ 0              X    X  3X X3X X
      3.90*10^ 0              X    X  XX XXX X
      3.60*10^ 0              X    X  XX XXX X
      3.30*10^ 0              X    X  XX XXX X
      3.00*10^ 0         X   XXXXXXXX XXXXXX X   X
      2.70*10^ 0         X   XXXXXXXX XXXXXX X   X
      2.40*10^ 0         X   XXXXXXXX XXXXXX X   X
      2.10*10^ 0        7X7 7XXXXXXXX7XXXXXX X7 7X 77   7
      1.80*10^ 0        XXX XXXXXXXXXXXXXXXX XX XX XX   X
      1.50*10^ 0        XXX XXXXXXXXXXXXXXXX XX XX XX   X
      1.20*10^ 0       3XXX3XXXXXXXXXXXXXXXX XX3XX XX3  X  33
      0.90*10^ 0       XXXXXXXXXXXXXXXXXXXXX XXXXX XXX  X  XX
      0.60*10^ 0       XXXXXXXXXXXXXXXXXXXXX XXXXX XXX  X  XX
      0.30*10^ 0       XXXXXXXXXXXXXXXXXXXXX XXXXX XXX  X  XX

        Contents
          *10^ 0  00000012321236333393246394506212302210020011000000000000000000000000000000000000000000000000000000000000000000
          *10^-1  00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
          *10^-2  00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
          *10^-3  00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

        Low edge  -
          *10^ 2  00000000000001111111111222222222222223333333333334444444444445555555555556666666666666777777777777
          *10^ 1  00123445678901223456789001234456788901223456678900123445678890122345667890012344567889012234566789
          *10^ 0  18642086420864208642086420864208642086420864208642086420864208642086420864208642086420864208642086

 Entries  =         100    Mean =  1.7430e+02    Underflow =  0.0000e+00    Low edge  = -5.0000e-01
 All chan =  1.0000e+02    Rms  =  6.5496e+01    Overflow  =  0.0000e+00    High edge =  7.9950e+02
```

The Event Record

History & Physics (demonstrates divide & conquer)

Many copies of the “same” particle may exist, but only positive status codes for the final state.

e.g. a top quark produced in the hard interaction, initially with positive status code. Then later a shower branching $\mathrm{t} \to \mathrm{tg}$ occurs, the new t and g are added at the bottom of the then-current event record, but the old t is not removed. It is marked as decayed with a negative status code. At any stage of the shower there is thus only one “current” copy of the top. After the shower, when the final top decays, $\mathrm{t} \to \mathrm{bW}^+$, also that copy receives a negative status code.

Note how the mother/daughter indices tie together the various copies.

Run Diagnostics

During the run you may see messages of three kinds:

- a *warning* is a minor problem that is automatically fixed by the program, at least approximately;
- an *error* is a bigger problem, that is normally still automatically fixed by the program, by backing up and trying again;
- an *abort* is such a major problem that the current event could not be completed; in such a rare case `pythia.next()` is `false` and the event should be skipped.

Look out for aborts. During event generation, a problem message is printed only the first time it occurs (except for a few special cases).

`pythia.stat()` will then tell you how many times each problem was encountered over the entire run.

Input files

With the `mymain01.cc` structure developed above it is necessary to recompile the main program for each minor change

Parameters can be put in special input "card" files that are read by the main program.

The file can contain one command per line, of the type

```
variable = value
```

All valid variables are listed in the online manual.

The name of the input file can be hardcoded in the main program.

OR use a command-line argument:

`int main(){` → `int main(int argc, char* argv[]){`

`pythia.readString(...)` → `pythia.readFile(argv[1]);`

Running the input card example

Please do the following:

```
cp /opt/hep/share/Pythia8/examples/main03.cc mymain03.cc

cp /opt/hep/share/Pythia8/examples/main03.cmnd .

make mymain03

./mymain03 > mymain03.out
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

int main() {

  // Generator.
  Pythia pythia;

  // Shorthand for the event record in pythia.
  Event& event = pythia.event;

  // Read in commands from external file.
  pythia.readFile("main03.cmnd");
--cut--
  // Extract settings to be used in the main program.
  int nEvent = pythia.mode("Main:numberOfEvents");
  int nAbort = pythia.mode("Main:timesAllowErrors");

  // Initialize.
  pythia.init();

  // Begin event loop.
  int iAbort = 0;
  for (int iEvent = 0; iEvent < nEvent; ++iEvent) {
    // Generate events. Quit if many failures.
    if (!pythia.next()) {
      if (++iAbort < nAbort) continue;
      cout « " Event generation aborted prematurely, owing to error!\n";
      break;
    }
--cut--
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

int main() {

  // Generator.
  Pythia pythia;

  // Shorthand for the event record in pythia.
  Event& event = pythia.event;

  // Read in commands from external file.
  pythia.readFile("main03.cmnd");
--cut--
  // Extract settings to be used in the main program.
  int nEvent = pythia.mode("Main:numberOfEvents");
  int nAbort = pythia.mode("Main:timesAllowErrors");

  // Initialize.
  pythia.init();

  // Begin event loop.
  int iAbort = 0;
  for (int iEvent = 0; iEvent < nEvent; ++iEvent) {
    // Generate events. Quit if many failures.
    if (!pythia.next()) {
      if (++iAbort < nAbort) continue;
      cout « " Event generation aborted prematurely, owing to error!\n";
      break;
    }
--cut--
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

int main() {

  // Generator.
  Pythia pythia;

  // Shorthand for the event record in pythia.
  Event& event = pythia.event;

  // Read in commands from external file.
  pythia.readFile("main03.cmnd");
--cut--
  // Extract settings to be used in the main program.
  int nEvent = pythia.mode("Main:numberOfEvents");
  int nAbort = pythia.mode("Main:timesAllowErrors");

  // Initialize.
  pythia.init();

  // Begin event loop.
  int iAbort = 0;
  for (int iEvent = 0; iEvent < nEvent; ++iEvent) {
    // Generate events. Quit if many failures.
    if (!pythia.next()) {
      if (++iAbort < nAbort) continue;
      cout « " Event generation aborted prematurely, owing to error!\n";
      break;
    }
--cut--
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

int main() {

  // Generator.
  Pythia pythia;

  // Shorthand for the event record in pythia.
  Event& event = pythia.event;

  // Read in commands from external file.
  pythia.readFile("main03.cmnd");
--cut--
  // Extract settings to be used in the main program.
  int nEvent = pythia.mode("Main:numberOfEvents");
  int nAbort = pythia.mode("Main:timesAllowErrors");

  // Initialize.
  pythia.init();

  // Begin event loop.
  int iAbort = 0;
  for (int iEvent = 0; iEvent < nEvent; ++iEvent) {
    // Generate events. Quit if many failures.
    if (!pythia.next()) {
      if (++iAbort < nAbort) continue;
      cout « " Event generation aborted prematurely, owing to error!\n";
      break;
    }
--cut--
```

```
! 1) Settings used in the main program.
Main:numberOfEvents = 1000         ! number of events to generate
Main:timesAllowErrors = 3          ! how many aborts before run stops

! 2) Settings related to output in init(), next() and stat().
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = off ! list changed particle data
Next:numberCount = 100             ! print message every n events
Next:numberShowInfo = 1            ! print event information n times
Next:numberShowProcess = 1         ! print process record n times
Next:numberShowEvent = 0           ! print event record n times

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 2212                   ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212                   ! second beam, p = 2212, pbar = -2212
Beams:eCM = 14000.                 ! CM energy of collision

! 4) Settings for the hard-process generation.

! Example 1: QCD + prompt photon production; must set pTmin.
HardQCD:all = on                   ! switch on all QCD jet + jet processes
PromptPhoton:all = on              ! swich on gamma + jet and gamma + gamma
PhaseSpace:pTHatMin = 50.          ! minimal pT scale in process
--cut--
! 6) Other settings. Can be expanded as desired.
#Tune:preferLHAPDF = off           ! use internal PDFs when LHAPDF not linked
Tune:pp = 6                        ! use Tune 4Cx
ParticleDecays:limitTau0 = on      ! set long-lived particle stable ...
ParticleDecays:tau0Max = 10        ! ... if c*tau0 > 10 mm
```

```
! 1) Settings used in the main program.
Main:numberOfEvents = 1000         ! number of events to generate
Main:timesAllowErrors = 3          ! how many aborts before run stops

! 2) Settings related to output in init(), next() and stat().
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = off ! list changed particle data
Next:numberCount = 100             ! print message every n events
Next:numberShowInfo = 1            ! print event information n times
Next:numberShowProcess = 1         ! print process record n times
Next:numberShowEvent = 0           ! print event record n times

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 2212                   ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212                   ! second beam, p = 2212, pbar = -2212
Beams:eCM = 14000.                 ! CM energy of collision

! 4) Settings for the hard-process generation.

! Example 1: QCD + prompt photon production; must set pTmin.
HardQCD:all = on                   ! switch on all QCD jet + jet processes
PromptPhoton:all = on              ! swich on gamma + jet and gamma + gamma
PhaseSpace:pTHatMin = 50.          ! minimal pT scale in process
--cut--
! 6) Other settings. Can be expanded as desired.
#Tune:preferLHAPDF = off           ! use internal PDFs when LHAPDF not linked
Tune:pp = 6                        ! use Tune 4Cx
ParticleDecays:limitTau0 = on      ! set long-lived particle stable ...
ParticleDecays:tau0Max = 10        ! ... if c*tau0 > 10 mm
```

```
! 1) Settings used in the main program.
Main:numberOfEvents = 1000         ! number of events to generate
Main:timesAllowErrors = 3          ! how many aborts before run stops

! 2) Settings related to output in init(), next() and stat().
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = off ! list changed particle data
Next:numberCount = 100             ! print message every n events
Next:numberShowInfo = 1            ! print event information n times
Next:numberShowProcess = 1         ! print process record n times
Next:numberShowEvent = 0           ! print event record n times

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 2212                   ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212                   ! second beam, p = 2212, pbar = -2212
Beams:eCM = 14000.                 ! CM energy of collision

! 4) Settings for the hard-process generation.

! Example 1: QCD + prompt photon production; must set pTmin.
HardQCD:all = on                   ! switch on all QCD jet + jet processes
PromptPhoton:all = on              ! swich on gamma + jet and gamma + gamma
PhaseSpace:pTHatMin = 50.          ! minimal pT scale in process
--cut--
! 6) Other settings. Can be expanded as desired.
#Tune:preferLHAPDF = off           ! use internal PDFs when LHAPDF not linked
Tune:pp = 6                        ! use Tune 4Cx
ParticleDecays:limitTau0 = on      ! set long-lived particle stable ...
ParticleDecays:tau0Max = 10        ! ... if c*tau0 > 10 mm
```

```
! 1) Settings used in the main program.
Main:numberOfEvents = 1000         ! number of events to generate
Main:timesAllowErrors = 3          ! how many aborts before run stops

! 2) Settings related to output in init(), next() and stat().
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = off ! list changed particle data
Next:numberCount = 100             ! print message every n events
Next:numberShowInfo = 1            ! print event information n times
Next:numberShowProcess = 1         ! print process record n times
Next:numberShowEvent = 0           ! print event record n times

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 2212                   ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212                   ! second beam, p = 2212, pbar = -2212
Beams:eCM = 14000.                 ! CM energy of collision

! 4) Settings for the hard-process generation.

! Example 1: QCD + prompt photon production; must set pTmin.
HardQCD:all = on                   ! switch on all QCD jet + jet processes
PromptPhoton:all = on              ! swich on gamma + jet and gamma + gamma
PhaseSpace:pTHatMin = 50.          ! minimal pT scale in process
--cut--
! 6) Other settings. Can be expanded as desired.
#Tune:preferLHAPDF = off           ! use internal PDFs when LHAPDF not linked
Tune:pp = 6                        ! use Tune 4Cx
ParticleDecays:limitTau0 = on      ! set long-lived particle stable ...
ParticleDecays:tau0Max = 10        ! ... if c*tau0 > 10 mm
```

```
! 1) Settings used in the main program.
Main:numberOfEvents = 1000         ! number of events to generate
Main:timesAllowErrors = 3          ! how many aborts before run stops

! 2) Settings related to output in init(), next() and stat().
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = off ! list changed particle data
Next:numberCount = 100             ! print message every n events
Next:numberShowInfo = 1            ! print event information n times
Next:numberShowProcess = 1         ! print process record n times
Next:numberShowEvent = 0           ! print event record n times

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 2212                   ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212                   ! second beam, p = 2212, pbar = -2212
Beams:eCM = 14000.                 ! CM energy of collision

! 4) Settings for the hard-process generation.

! Example 1: QCD + prompt photon production; must set pTmin.
HardQCD:all = on                   ! switch on all QCD jet + jet processes
PromptPhoton:all = on              ! swich on gamma + jet and gamma + gamma
PhaseSpace:pTHatMin = 50.          ! minimal pT scale in process
--cut--
! 6) Other settings. Can be expanded as desired.
#Tune:preferLHAPDF = off           ! use internal PDFs when LHAPDF not linked
Tune:pp = 6                        ! use Tune 4Cx
ParticleDecays:limitTau0 = on      ! set long-lived particle stable ...
ParticleDecays:tau0Max = 10        ! ... if c*tau0 > 10 mm
```

```
! 1) Settings used in the main program.
Main:numberOfEvents = 1000         ! number of events to generate
Main:timesAllowErrors = 3          ! how many aborts before run stops

! 2) Settings related to output in init(), next() and stat().
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = off ! list changed particle data
Next:numberCount = 100             ! print message every n events
Next:numberShowInfo = 1            ! print event information n times
Next:numberShowProcess = 1         ! print process record n times
Next:numberShowEvent = 0           ! print event record n times

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 2212                   ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212                   ! second beam, p = 2212, pbar = -2212
Beams:eCM = 14000.                 ! CM energy of collision

! 4) Settings for the hard-process generation.

! Example 1: QCD + prompt photon production; must set pTmin.
HardQCD:all = on                   ! switch on all QCD jet + jet processes
PromptPhoton:all = on              ! swich on gamma + jet and gamma + gamma
PhaseSpace:pTHatMin = 50.          ! minimal pT scale in process
--cut--
! 6) Other settings. Can be expanded as desired.
#Tune:preferLHAPDF = off            ! use internal PDFs when LHAPDF not linked
Tune:pp = 6                        ! use Tune 4Cx
ParticleDecays:limitTau0 = on      ! set long-lived particle stable ...
ParticleDecays:tau0Max = 10        ! ... if c*tau0 > 10 mm
```

You are now free to play with further options in the input file, such as:

- `PartonLevel:FSR = off`
  switch off final-state radiation.
- `PartonLevel:ISR = off`
  switch off initial-state radiation.
- `PartonLevel:MPI = off`
  switch off multiparton interactions.
- `Tune:pp = 3` (or other values between 1 and 17)
  different combined tunes, in particular to radiation and multiparton interactions parameters. In part this reflects that no generator is perfect, and also not all data is perfect, so different emphasis will result in different optima.
- `Random:setSeed = on`
  `Random:seed = 123456789`
  all runs by default use the same random-number sequence, for reproducibility, but you can pick any number between 1 and 900,000,000 to obtain a unique sequence.

For instance, check the importance of FSR, ISR and MPI on the charged multiplicity of events by switching off one component at a time.
The possibility to use command-line input files is further illustrated e.g. in `main16.cc` and `main42.cc`.

Building your own run card

The online manual also exists in an interactive variant, where you semi-automatically can construct a file with all the command lines you wish to have.

You can use the one at `http://home.thep.lu.se/Pythia/pythia82php/Welcome.php`

Full instructions are provided on the “Save Settings” page.

Typical BSM Theorist Question

I made a UFO file and generated an LHE file with MG. How do I give this to an experimentalist or run it through Delphes?

Most common use-case for Pythia

Passing of parton information based on LH Accord

Passing of BSM quantum numbers and properties based on SLH Accord

Largest consumer of person-hours of theorists and experimentalists in the MC community

Running the input LHEF example

Please do the following:

```
cp /opt/hep/share/Pythia8/examples/main11.cc mymain11.cc

cp /opt/hep/share/Pythia8/examples/ttbar.lhe .

make mymain11

gzip ttbar.lhe

./mymain11 > mymain11.out
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {

  // You can always read an plain LHE file,
  // but if you ran "./configure -with-gzip" before "make"
  // then you can also read a gzipped LHE file.
#ifdef GZIPSUPPORT
  bool useGzip = true;
#else
  bool useGzip = false;
#endif
  cout « " useGzip = " « useGzip « endl;

  // Generator. We here stick with default values, but changes
  // could be inserted with readString or readFile.
  Pythia pythia;

  // Initialize Les Houches Event File run. List initialization information.
  pythia.readString("Beams:frameType = 4");
  if (useGzip) pythia.readString("Beams:LHEF = ttbar.lhe.gz");
  else         pythia.readString("Beams:LHEF = ttbar.lhe");
  pythia.init();
--cut--

  // Begin event loop; generate until none left in input file.
  for (int iEvent = 0; ; ++iEvent) {

    // Generate events, and check whether generation failed.
    if (!pythia.next()) {

      // If failure because reached end of file then exit event loop.
      if (pythia.info.atEndOfFile()) break;
--cut--
}
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {

  // You can always read an plain LHE file,
  // but if you ran "./configure -with-gzip" before "make"
  // then you can also read a gzipped LHE file.
#ifdef GZIPSUPPORT
  bool useGzip = true;
#else
  bool useGzip = false;
#endif
  cout « " useGzip = " « useGzip « endl;

  // Generator. We here stick with default values, but changes
  // could be inserted with readString or readFile.
  Pythia pythia;

  // Initialize Les Houches Event File run. List initialization information.
  pythia.readString("Beams:frameType = 4");
  if (useGzip) pythia.readString("Beams:LHEF = ttbar.lhe.gz");
  else         pythia.readString("Beams:LHEF = ttbar.lhe");
  pythia.init();
--cut--

  // Begin event loop; generate until none left in input file.
  for (int iEvent = 0; ; ++iEvent) {

    // Generate events, and check whether generation failed.
    if (!pythia.next()) {

      // If failure because reached end of file then exit event loop.
      if (pythia.info.atEndOfFile()) break;
--cut--
}
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {

  // You can always read an plain LHE file,
  // but if you ran "./configure -with-gzip" before "make"
  // then you can also read a gzipped LHE file.
#ifdef GZIPSUPPORT
  bool useGzip = true;
#else
  bool useGzip = false;
#endif
  cout « " useGzip = " « useGzip « endl;

  // Generator. We here stick with default values, but changes
  // could be inserted with readString or readFile.
  Pythia pythia;

  // Initialize Les Houches Event File run. List initialization information.
  pythia.readString("Beams:frameType = 4");
  if (useGzip) pythia.readString("Beams:LHEF = ttbar.lhe.gz");
  else         pythia.readString("Beams:LHEF = ttbar.lhe");
  pythia.init();
--cut--

  // Begin event loop; generate until none left in input file.
  for (int iEvent = 0; ; ++iEvent) {

    // Generate events, and check whether generation failed.
    if (!pythia.next()) {

      // If failure because reached end of file then exit event loop.
      if (pythia.info.atEndOfFile()) break;
--cut--
}
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {

  // You can always read an plain LHE file,
  // but if you ran "./configure -with-gzip" before "make"
  // then you can also read a gzipped LHE file.
#ifdef GZIPSUPPORT
  bool useGzip = true;
#else
  bool useGzip = false;
#endif
  cout « " useGzip = " « useGzip « endl;

  // Generator. We here stick with default values, but changes
  // could be inserted with readString or readFile.
  Pythia pythia;

  // Initialize Les Houches Event File run. List initialization information.
  pythia.readString("Beams:frameType = 4");
  if (useGzip) pythia.readString("Beams:LHEF = ttbar.lhe.gz");
  else         pythia.readString("Beams:LHEF = ttbar.lhe");
  pythia.init();
--cut--

  // Begin event loop; generate until none left in input file.
  for (int iEvent = 0; ; ++iEvent) {

    // Generate events, and check whether generation failed.
    if (!pythia.next()) {

      // If failure because reached end of file then exit event loop.
      if (pythia.info.atEndOfFile()) break;
--cut--
}
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {

  // You can always read an plain LHE file,
  // but if you ran "./configure -with-gzip" before "make"
  // then you can also read a gzipped LHE file.
#ifdef GZIPSUPPORT
  bool useGzip = true;
#else
  bool useGzip = false;
#endif
  cout « " useGzip = " « useGzip « endl;

  // Generator. We here stick with default values, but changes
  // could be inserted with readString or readFile.
  Pythia pythia;

  // Initialize Les Houches Event File run. List initialization information.
  pythia.readString("Beams:frameType = 4");
  if (useGzip) pythia.readString("Beams:LHEF = ttbar.lhe.gz");
  else         pythia.readString("Beams:LHEF = ttbar.lhe");
  pythia.init();
--cut--

  // Begin event loop; generate until none left in input file.
  for (int iEvent = 0; ; ++iEvent) {

    // Generate events, and check whether generation failed.
    if (!pythia.next()) {

      // If failure because reached end of file then exit event loop.
      if (pythia.info.atEndOfFile()) break;
--cut--
}
```

```
<LesHouchesEvents version="1.0">
<!-
File generated with PYTHIA 6.413      SM: LHEF generated from Pythia!
->
<init>
    2212   -2212  9.800000E+02  9.800000E+02     0     0     7     7     3     2
  5.220106E+00  5.384128E-01  1.000000E+00    81
  2.602564E-01  1.062492E-01  1.000000E+00    82
</init>
<event>                              SM:Masses and polarization cut off
    12    81  1.000000E+00  1.733125E+02  7.819848E-03  1.156692E-01
       2   -1    0    0  101    0  0.0000000000E+00  0.0000000000E+00  1.0838163607E+02  1.0838163607E+02  0.0
      -2   -1    0    0    0  102  0.0000000000E+00  0.0000000000E+00 -2.7976111253E+02  2.7976111253E+02  0.0
       6    2    1    2  101    0  3.3629095553E+01  8.9115695965E+00 -1.1059648961E+02  2.1241781824E+02  1.7
      -6    2    1    2    0  102 -3.3629095553E+01 -8.9115695965E+00 -6.0782986840E+01  1.7572493036E+02  1.6
      24    2    3    0    0    0 -3.0884654830E+01 -1.2140252163E+01 -4.7852784957E+00  8.6623320800E+01  7.9
       5    1    3    0  101    0  6.4513750383E+01  2.1051821759E+01 -1.0581121112E+02  1.2579449744E+02  4.8
     -24    2    4    0    0    0 -5.0940382043E+01  3.4880802250E+01 -7.5291578188E+01  1.2621743906E+02  8.0
      -5    1    4    0    0  102  1.7311286490E+01 -4.3792371846E+01  1.4508591348E+01  4.9507491299E+01  4.8
      -1    1    5    0    0  103  1.8584463332E+01  9.1657242037E+00  1.8652036768E+01  2.7881896512E+01  3.3
       2    1    5    0  103    0 -4.9469118162E+01 -2.1305976366E+01 -2.3437315264E+01  5.8741424288E+01  3.3
      13    1    7    0    0    0  9.6912588119E+00  3.9074488577E+01 -2.5560060185E+01  4.7687147069E+01  1.0
     -14    1    7    0    0    0 -6.0631640855E+01 -4.1936863270E+00 -4.9731518002E+01  7.8530291993E+01  0.0
#pdf     2   -2  1.1059350620E-01  2.8547052299E-01  1.7331247164E+02  5.5300424188E-01  3.5718362666E-01
</event>
--cut--
</LesHouchesEvents>
```

```
<LesHouchesEvents version="1.0">
<!-
File generated with PYTHIA 6.413      SM: LHEF generated from Pythia!
->
<init>
    2212   -2212  9.800000E+02  9.800000E+02     0     0     7     7     3     2
  5.220106E+00  5.384128E-01  1.000000E+00    81
  2.602564E-01  1.062492E-01  1.000000E+00    82
</init>
<event>                              SM:Masses and polarization cut off
    12    81  1.000000E+00  1.733125E+02  7.819848E-03  1.156692E-01
       2   -1    0    0  101    0  0.0000000000E+00  0.0000000000E+00  1.0838163607E+02  1.0838163607E+02  0.0
      -2   -1    0    0    0  102  0.0000000000E+00  0.0000000000E+00 -2.7976111253E+02  2.7976111253E+02  0.0
       6    2    1    2  101    0  3.3629095553E+01  8.9115695965E+00 -1.1059648961E+02  2.1241781824E+02  1.7
      -6    2    1    2    0  102 -3.3629095553E+01 -8.9115695965E+00 -6.0782986840E+01  1.7572493036E+02  1.6
      24    2    3    0    0    0 -3.0884654830E+01 -1.2140252163E+01 -4.7852784957E+00  8.6623320800E+01  7.9
       5    1    3    0  101    0  6.4513750383E+01  2.1051821759E+01 -1.0581121112E+02  1.2579449744E+02  4.8
     -24    2    4    0    0    0 -5.0940382043E+01  3.4880802250E+01 -7.5291578188E+01  1.2621743906E+02  8.0
      -5    1    4    0    0  102  1.7311286490E+01 -4.3792371846E+01  1.4508591348E+01  4.9507491299E+01  4.8
      -1    1    5    0    0  103  1.8584463332E+01  9.1657242037E+00  1.8652036768E+01  2.7881896512E+01  3.3
       2    1    5    0  103    0 -4.9469118162E+01 -2.1305976366E+01 -2.3437315264E+01  5.8741424288E+01  3.3
      13    1    7    0    0    0  9.6912588119E+00  3.9074488577E+01 -2.5560060185E+01  4.7687147069E+01  1.0
     -14    1    7    0    0    0 -6.0631640855E+01 -4.1936863270E+00 -4.9731518002E+01  7.8530291993E+01  0.0
#pdf     2   -2  1.1059350620E-01  2.8547052299E-01  1.7331247164E+02  5.5300424188E-01  3.5718362666E-01
</event>
--cut--
</LesHouchesEvents>
```

```
<LesHouchesEvents version="1.0">
<!-
File generated with PYTHIA 6.413      SM: LHEF generated from Pythia!
->
<init>
    2212   -2212  9.800000E+02  9.800000E+02     0     0     7     7     3     2
  5.220106E+00  5.384128E-01  1.000000E+00    81
  2.602564E-01  1.062492E-01  1.000000E+00    82
</init>
<event>                              SM:Masses and polarization cut off
    12    81  1.000000E+00  1.733125E+02  7.819848E-03  1.156692E-01
       2   -1    0    0  101    0  0.0000000000E+00  0.0000000000E+00  1.0838163607E+02  1.0838163607E+02  0.0
      -2   -1    0    0    0  102  0.0000000000E+00  0.0000000000E+00 -2.7976111253E+02  2.7976111253E+02  0.0
       6    2    1    2  101    0  3.3629095553E+01  8.9115695965E+00 -1.1059648961E+02  2.1241781824E+02  1.7
      -6    2    1    2    0  102 -3.3629095553E+01 -8.9115695965E+00 -6.0782986840E+01  1.7572493036E+02  1.6
      24    2    3    0    0    0 -3.0884654830E+01 -1.2140252163E+01 -4.7852784957E+00  8.6623320800E+01  7.9
       5    1    3    0  101    0  6.4513750383E+01  2.1051821759E+01 -1.0581121112E+02  1.2579449744E+02  4.8
     -24    2    4    0    0    0 -5.0940382043E+01  3.4880802250E+01 -7.5291578188E+01  1.2621743906E+02  8.0
      -5    1    4    0    0  102  1.7311286490E+01 -4.3792371846E+01  1.4508591348E+01  4.9507491299E+01  4.8
      -1    1    5    0    0  103  1.8584463332E+01  9.1657242037E+00  1.8652036768E+01  2.7881896512E+01  3.3
       2    1    5    0  103    0 -4.9469118162E+01 -2.1305976366E+01 -2.3437315264E+01  5.8741424288E+01  3.3
      13    1    7    0    0    0  9.6912588119E+00  3.9074488577E+01 -2.5560060185E+01  4.7687147069E+01  1.0
     -14    1    7    0    0    0 -6.0631640855E+01 -4.1936863270E+00 -4.9731518002E+01  7.8530291993E+01  0.0
#pdf     2   -2  1.1059350620E-01  2.8547052299E-01  1.7331247164E+02  5.5300424188E-01  3.5718362666E-01
</event>
--cut--
</LesHouchesEvents>
```

```
<LesHouchesEvents version="1.0">
<!-
File generated with PYTHIA 6.413      SM: LHEF generated from Pythia!
->
<init>
    2212   -2212  9.800000E+02  9.800000E+02     0     0     7     7     3     2
  5.220106E+00  5.384128E-01  1.000000E+00    81
  2.602564E-01  1.062492E-01  1.000000E+00    82
</init>
<event>                              SM:Masses and polarization cut off
    12    81  1.000000E+00  1.733125E+02  7.819848E-03  1.156692E-01
       2   -1    0    0  101    0  0.0000000000E+00  0.0000000000E+00  1.0838163607E+02  1.0838163607E+02  0.0
      -2   -1    0    0    0  102  0.0000000000E+00  0.0000000000E+00 -2.7976111253E+02  2.7976111253E+02  0.0
       6    2    1    2  101    0  3.3629095553E+01  8.9115695965E+00 -1.1059648961E+02  2.1241781824E+02  1.7
      -6    2    1    2    0  102 -3.3629095553E+01 -8.9115695965E+00 -6.0782986840E+01  1.7572493036E+02  1.6
      24    2    3    0    0    0 -3.0884654830E+01 -1.2140252163E+01 -4.7852784957E+00  8.6623320800E+01  7.9
       5    1    3    0  101    0  6.4513750383E+01  2.1051821759E+01 -1.0581121112E+02  1.2579449744E+02  4.8
     -24    2    4    0    0    0 -5.0940382043E+01  3.4880802250E+01 -7.5291578188E+01  1.2621743906E+02  8.0
      -5    1    4    0    0  102  1.7311286490E+01 -4.3792371846E+01  1.4508591348E+01  4.9507491299E+01  4.8
      -1    1    5    0    0  103  1.8584463332E+01  9.1657242037E+00  1.8652036768E+01  2.7881896512E+01  3.3
       2    1    5    0  103    0 -4.9469118162E+01 -2.1305976366E+01 -2.3437315264E+01  5.8741424288E+01  3.3
      13    1    7    0    0    0  9.6912588119E+00  3.9074488577E+01 -2.5560060185E+01  4.7687147069E+01  1.0
     -14    1    7    0    0    0 -6.0631640855E+01 -4.1936863270E+00 -4.9731518002E+01  7.8530291993E+01  0.0
#pdf     2   -2  1.1059350620E-01  2.8547052299E-01  1.7331247164E+02  5.5300424188E-01  3.5718362666E-01
</event>
--cut--
</LesHouchesEvents>
```

```
<LesHouchesEvents version="1.0">
<!-
File generated with PYTHIA 6.413      SM: LHEF generated from Pythia!
->
<init>
    2212   -2212  9.800000E+02  9.800000E+02     0     0     7     7     3     2
  5.220106E+00  5.384128E-01  1.000000E+00    81
  2.602564E-01  1.062492E-01  1.000000E+00    82
</init>
<event>                              SM:Masses and polarization cut off
    12    81  1.000000E+00  1.733125E+02  7.819848E-03  1.156692E-01
       2   -1    0    0  101    0  0.0000000000E+00  0.0000000000E+00  1.0838163607E+02  1.0838163607E+02  0.0
      -2   -1    0    0    0  102  0.0000000000E+00  0.0000000000E+00 -2.7976111253E+02  2.7976111253E+02  0.0
       6    2    1    2  101    0  3.3629095553E+01  8.9115695965E+00 -1.1059648961E+02  2.1241781824E+02  1.7
      -6    2    1    2    0  102 -3.3629095553E+01 -8.9115695965E+00 -6.0782986840E+01  1.7572493036E+02  1.6
      24    2    3    0    0    0 -3.0884654830E+01 -1.2140252163E+01 -4.7852784957E+00  8.6623320800E+01  7.9
       5    1    3    0  101    0  6.4513750383E+01  2.1051821759E+01 -1.0581121112E+02  1.2579449744E+02  4.8
     -24    2    4    0    0    0 -5.0940382043E+01  3.4880802250E+01 -7.5291578188E+01  1.2621743906E+02  8.0
      -5    1    4    0    0  102  1.7311286490E+01 -4.3792371846E+01  1.4508591348E+01  4.9507491299E+01  4.8
      -1    1    5    0    0  103  1.8584463332E+01  9.1657242037E+00  1.8652036768E+01  2.7881896512E+01  3.3
       2    1    5    0  103    0 -4.9469118162E+01 -2.1305976366E+01 -2.3437315264E+01  5.8741424288E+01  3.3
      13    1    7    0    0    0  9.6912588119E+00  3.9074488577E+01 -2.5560060185E+01  4.7687147069E+01  1.0
     -14    1    7    0    0    0 -6.0631640855E+01 -4.1936863270E+00 -4.9731518002E+01  7.8530291993E+01  0.0
#pdf     2   -2  1.1059350620E-01  2.8547052299E-01  1.7331247164E+02  5.5300424188E-01  3.5718362666E-01
</event>
--cut--
</LesHouchesEvents>
```

```
<LesHouchesEvents version="1.0">
<!-
File generated with PYTHIA 6.413      SM: LHEF generated from Pythia!
->
<init>
    2212   -2212  9.800000E+02  9.800000E+02     0     0     7     7     3     2
  5.220106E+00  5.384128E-01  1.000000E+00    81
  2.602564E-01  1.062492E-01  1.000000E+00    82
</init>
<event>                              SM:Masses and polarization cut off
    12    81  1.000000E+00  1.733125E+02  7.819848E-03  1.156692E-01
       2   -1    0    0  101    0  0.0000000000E+00  0.0000000000E+00  1.0838163607E+02  1.0838163607E+02  0.0
      -2   -1    0    0    0  102  0.0000000000E+00  0.0000000000E+00 -2.7976111253E+02  2.7976111253E+02  0.0
       6    2    1    2  101    0  3.3629095553E+01  8.9115695965E+00 -1.1059648961E+02  2.1241781824E+02  1.7
      -6    2    1    2    0  102 -3.3629095553E+01 -8.9115695965E+00 -6.0782986840E+01  1.7572493036E+02  1.6
      24    2    3    0    0    0 -3.0884654830E+01 -1.2140252163E+01 -4.7852784957E+00  8.6623320800E+01  7.9
       5    1    3    0  101    0  6.4513750383E+01  2.1051821759E+01 -1.0581121112E+02  1.2579449744E+02  4.8
     -24    2    4    0    0    0 -5.0940382043E+01  3.4880802250E+01 -7.5291578188E+01  1.2621743906E+02  8.0
      -5    1    4    0    0  102  1.7311286490E+01 -4.3792371846E+01  1.4508591348E+01  4.9507491299E+01  4.8
      -1    1    5    0    0  103  1.8584463332E+01  9.1657242037E+00  1.8652036768E+01  2.7881896512E+01  3.3
       2    1    5    0  103    0 -4.9469118162E+01 -2.1305976366E+01 -2.3437315264E+01  5.8741424288E+01  3.3
      13    1    7    0    0    0  9.6912588119E+00  3.9074488577E+01 -2.5560060185E+01  4.7687147069E+01  1.0
     -14    1    7    0    0    0 -6.0631640855E+01 -4.1936863270E+00 -4.9731518002E+01  7.8530291993E+01  0.0
#pdf     2   -2  1.1059350620E-01  2.8547052299E-01  1.7331247164E+02  5.5300424188E-01  3.5718362666E-01
</event>
--cut--
</LesHouchesEvents>
```

More complicated (realistic) LHE Cases

This was a naively simple case

Today, even BSM is done with NLO + PS (low multiplicities) or Many Tree Level Topologies + PS

These require more complicated configurations and sometimes specialized plugins (POWHEG, Alpgen, MadGraph, AMCatNLO)

See `examples`

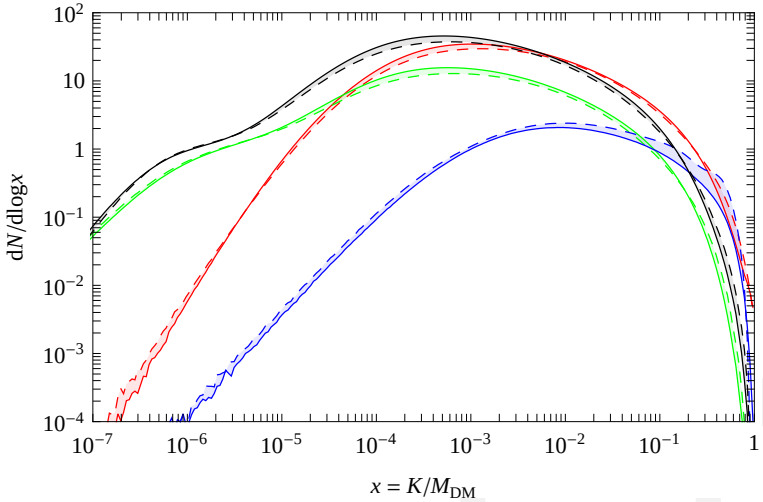
E.g., AMCatNLO requires global sharing of energy-momentum for 1st few PYTHIA FSR emissions

Dark Matter annihilation

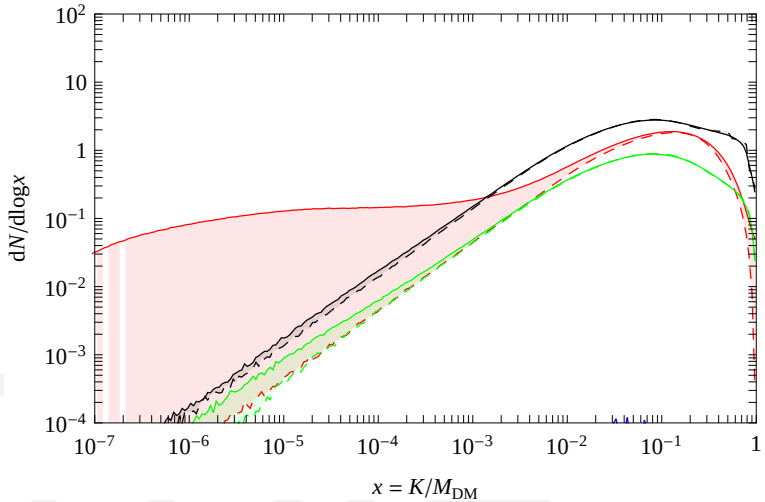
JCAP 1103 (2011) 051

Given mass and BRs, what is the observable spectrum?

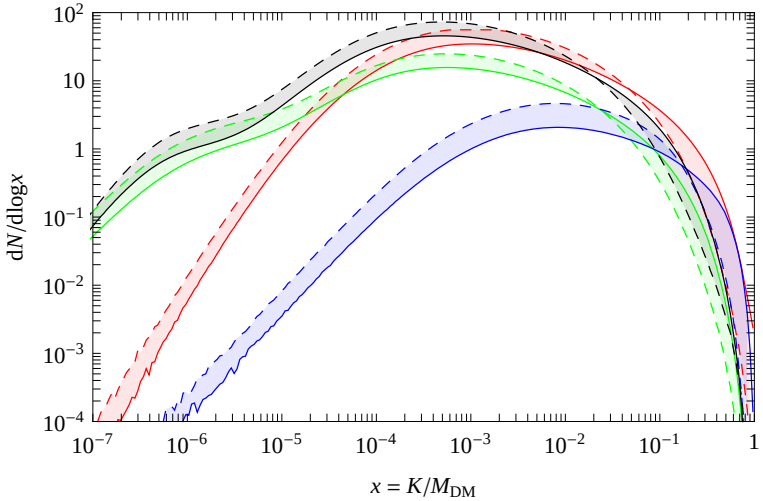
DM DM $\to q\bar{q}$ at $M_{\rm DM} = 1$ TeV



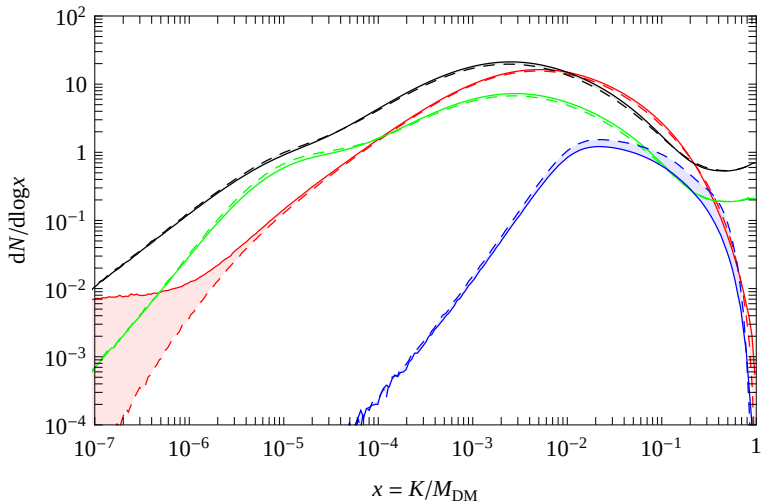
DM DM $\to \tau^+\tau^-$ at $M_{\rm DM} = 1$ TeV



DM DM $\to gg$ at $M_{\rm DM} = 1$ TeV



DM DM $\to W^+W^-$ at $M_{\rm DM} = 1$ TeV



γ
$e^{\pm}$
$\overline{p}$
ν

Pythia vs Herwig, K=Kinetic Energy

Blob production and decay

How to generate various two-body channels from astroparticle processes, e.g. neutralino annihilation or decay

"blob" of energy is created with unit cross section from the fictitious collision of two non-radiating incoming e+e-.

Decay channels of this blob can be set up

only gamma, $e^{\pm}$, p/pbar and neutrinos are stable

"single-particle gun" of `main21.cc` is another approach

Running the Blob example

Please do the following:

```
cp /opt/hep/share/Pythia8/examples/main07.cc mymain07.cc

cp /opt/hep/share/Pythia8/examples/main07.cmnd .

make mymain07

./mymain07 > mymain07.out
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

//==========================================================================
// A derived class for (e+ e- ->) GenericResonance -> various final states.

class Sigma1GenRes : public Sigma1Process {

public:

  // Constructor.
  Sigma1GenRes() {}

  // Evaluate sigmaHat(sHat): dummy unit cross section.
  virtual double sigmaHat() {return 1.;}

  // Select flavour. No colour or anticolour.
  virtual void setIdColAcol() {setId( -11, 11, 999999);
    setColAcol( 0, 0, 0, 0, 0, 0);}

  // Info on the subprocess.
  virtual string name()    const {return "GenericResonance";}
  virtual int    code()    const {return 9001;}
  virtual string inFlux()  const {return "ffbarSame";}

};
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

//==========================================================================
// A derived class for (e+ e- ->) GenericResonance -> various final states.

class Sigma1GenRes : public Sigma1Process {

public:

  // Constructor.
  Sigma1GenRes() {}

  // Evaluate sigmaHat(sHat): dummy unit cross section.
  virtual double sigmaHat() {return 1.;}

  // Select flavour. No colour or anticolour.
  virtual void setIdColAcol() {setId( -11, 11, 999999);
    setColAcol( 0, 0, 0, 0, 0, 0);}

  // Info on the subprocess.
  virtual string name()    const {return "GenericResonance";}
  virtual int    code()    const {return 9001;}
  virtual string inFlux()  const {return "ffbarSame";}

};
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

//==========================================================================
// A derived class for (e+ e- ->) GenericResonance -> various final states.

class Sigma1GenRes : public Sigma1Process {

public:

  // Constructor.
  Sigma1GenRes() {}

  // Evaluate sigmaHat(sHat): dummy unit cross section.
  virtual double sigmaHat() {return 1.;}

  // Select flavour. No colour or anticolour.
  virtual void setIdColAcol() {setId( -11, 11, 999999);
    setColAcol( 0, 0, 0, 0, 0, 0);}

  // Info on the subprocess.
  virtual string name()    const {return "GenericResonance";}
  virtual int    code()    const {return 9001;}
  virtual string inFlux()  const {return "ffbarSame";}

};
```

```
//==========================================================================

int main() {

  // Pythia generator.
  Pythia pythia;

  // A class to generate the fictitious resonance initial state.
  SigmaProcess* sigma1GenRes = new Sigma1GenRes();

  // Hand pointer to Pythia.
  pythia.setSigmaPtr( sigma1GenRes);

  // Read in the rest of the settings and data from a separate file.
  pythia.readFile("main07.cmnd");

--cut--

  // Done.
  delete sigma1GenRes;
  return 0;
}
```

```
//==========================================================================

int main() {

  // Pythia generator.
  Pythia pythia;

  // A class to generate the fictitious resonance initial state.
  SigmaProcess* sigma1GenRes = new Sigma1GenRes();

  // Hand pointer to Pythia.
  pythia.setSigmaPtr( sigma1GenRes);

  // Read in the rest of the settings and data from a separate file.
  pythia.readFile("main07.cmnd");

--cut--

  // Done.
  delete sigma1GenRes;
  return 0;
}
```

```
--cut--
! 3) Beam parameter settings. Incoming beams do not radiate.
Beams:idA = -11                    ! fictitious incoming e+
Beams:idB = 11                     ! fictitious incoming e-
PDF:lepton = off                   ! no radiation off fictitious e+e-
Beams:eCM = 500.                   ! CM energy of collision

! 4) Set up properties of the GeneralResonance and its decay channels.
! id:all = name antiName spinType chargeType colType m0 mWidth mMin mMax tau0
999999:all = GeneralResonance void 1 0 0 500. 1. 0. 0. 0.
! id:addChannel = onMode bRatio meMode product1 product2 ...
! Note: sum of branching ratios automatically rescaled to 1.
! Current channels illustrative only; insert your own decay list.
999999:addChannel = 1 0.15 101 1 -1   !  -> d dbar
999999:addChannel = 1 0.15 101 6 -6   !  -> t tbar
999999:addChannel = 1 0.15 101 15 -15 !  -> tau- tau+
999999:addChannel = 1 1.15 101 21 21  !  -> g g
999999:addChannel = 1 1.15 101 22 22  !  -> gamma gamma
999999:addChannel = 1 0.15 101 24 -24 !  -> W+ W-
999999:addChannel = 1 0.10 101 25 25  !  -> h0 h0

! 5) Tell that also long-lived should decay.
13:mayDecay   = true                ! mu+-
211:mayDecay  = true                ! pi+-
321:mayDecay  = true                ! K+-
130:mayDecay  = true                ! K0_L
2112:mayDecay = true                ! n
```

```
--cut--
! 3) Beam parameter settings. Incoming beams do not radiate.
Beams:idA = -11                     ! fictitious incoming e+
Beams:idB = 11                      ! fictitious incoming e-
PDF:lepton = off                    ! no radiation off fictitious e+e-
Beams:eCM = 500.                    ! CM energy of collision

! 4) Set up properties of the GeneralResonance and its decay channels.
! id:all = name antiName spinType chargeType colType m0 mWidth mMin mMax tau0
999999:all = GeneralResonance void 1 0 0 500. 1. 0. 0. 0.
! id:addChannel = onMode bRatio meMode product1 product2 ...
! Note: sum of branching ratios automatically rescaled to 1.
! Current channels illustrative only; insert your own decay list.
999999:addChannel = 1 0.15 101 1 -1   !  -> d dbar
999999:addChannel = 1 0.15 101 6 -6   !  -> t tbar
999999:addChannel = 1 0.15 101 15 -15 !  -> tau- tau+
999999:addChannel = 1 1.15 101 21 21  !  -> g g
999999:addChannel = 1 1.15 101 22 22  !  -> gamma gamma
999999:addChannel = 1 0.15 101 24 -24 !  -> W+ W-
999999:addChannel = 1 0.10 101 25 25  !  -> h0 h0

! 5) Tell that also long-lived should decay.
13:mayDecay   = true                ! mu+-
211:mayDecay  = true                ! pi+-
321:mayDecay  = true                ! K+-
130:mayDecay  = true                ! K0_L
2112:mayDecay = true                ! n
```

```
--cut--
! 3) Beam parameter settings. Incoming beams do not radiate.
Beams:idA = -11                     ! fictitious incoming e+
Beams:idB = 11                      ! fictitious incoming e-
PDF:lepton = off                    ! no radiation off fictitious e+e-
Beams:eCM = 500.                    ! CM energy of collision

! 4) Set up properties of the GeneralResonance and its decay channels.
! id:all = name antiName spinType chargeType colType m0 mWidth mMin mMax tau0
999999:all = GeneralResonance void 1 0 0 500. 1. 0. 0. 0.
! id:addChannel = onMode bRatio meMode product1 product2 ...
! Note: sum of branching ratios automatically rescaled to 1.
! Current channels illustrative only; insert your own decay list.
999999:addChannel = 1 0.15 101 1 -1   !  -> d dbar
999999:addChannel = 1 0.15 101 6 -6   !  -> t tbar
999999:addChannel = 1 0.15 101 15 -15 !  -> tau- tau+
999999:addChannel = 1 1.15 101 21 21  !  -> g g
999999:addChannel = 1 1.15 101 22 22  !  -> gamma gamma
999999:addChannel = 1 0.15 101 24 -24 !  -> W+ W-
999999:addChannel = 1 0.10 101 25 25  !  -> h0 h0

! 5) Tell that also long-lived should decay.
13:mayDecay   = true                ! mu+-
211:mayDecay  = true                ! pi+-
321:mayDecay  = true                ! K+-
130:mayDecay  = true                ! K0_L
2112:mayDecay = true                ! n
```

```
--cut--
! 3) Beam parameter settings. Incoming beams do not radiate.
Beams:idA = -11                     ! fictitious incoming e+
Beams:idB = 11                      ! fictitious incoming e-
PDF:lepton = off                    ! no radiation off fictitious e+e-
Beams:eCM = 500.                    ! CM energy of collision

! 4) Set up properties of the GeneralResonance and its decay channels.
! id:all = name antiName spinType chargeType colType m0 mWidth mMin mMax tau0
999999:all = GeneralResonance void 1 0 0 500. 1. 0. 0. 0.
! id:addChannel = onMode bRatio meMode product1 product2 ...
! Note: sum of branching ratios automatically rescaled to 1.
! Current channels illustrative only; insert your own decay list.
999999:addChannel = 1 0.15 101 1 -1   !  -> d dbar
999999:addChannel = 1 0.15 101 6 -6   !  -> t tbar
999999:addChannel = 1 0.15 101 15 -15 !  -> tau- tau+
999999:addChannel = 1 1.15 101 21 21  !  -> g g
999999:addChannel = 1 1.15 101 22 22  !  -> gamma gamma
999999:addChannel = 1 0.15 101 24 -24 !  -> W+ W-
999999:addChannel = 1 0.10 101 25 25  !  -> h0 h0

! 5) Tell that also long-lived should decay.
13:mayDecay   = true                ! mu+-
211:mayDecay  = true                ! pi+-
321:mayDecay  = true                ! K+-
130:mayDecay  = true                ! K0_L
2112:mayDecay = true                ! n
```

BSM physics 2: *R*-hadrons

What if coloured (SUSY) particle like $\tilde{g}$ or $\tilde{t}_1$ is long-lived?

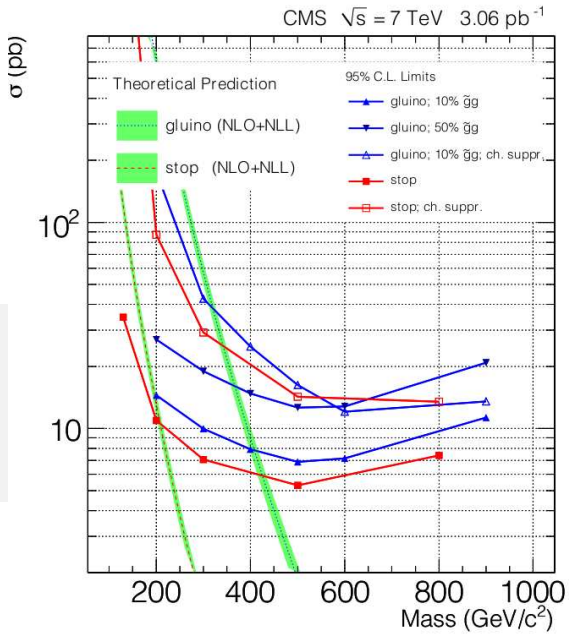
⋆ Formation of R-hadrons

$\tilde{g}q\bar{q}$	$\tilde{t}_1\bar{q}$	"mesons"
$\tilde{g}qqq$	$\tilde{t}_1qq$	"baryons"
$\tilde{g}g$		"glueballs"

⋆ Conversion between R-hadrons
by "low-energy" interactions with matter:
$\tilde{g}u\bar{d} + p \to \tilde{g}uud + \pi^+$ irreversible

⋆ Displaced vertices if finite lifetime, or else

⋆ punch-through: $\sigma \approx \sigma_{had}$ but
$\Delta E \lesssim 1$ GeV $\ll E_{kin,R}$



CMS, arXiv:1101.1645

A.C. Kraan, Eur. Phys. J. C37 (2004) 91;
M. Fairbairn et al., Phys. Rep. 438 (2007) 1

Partly event generation, partly detector simulation.
Public add-on in PYTHIA 6, now integrated part of PYTHIA 8.
Can also be applied to non-SUSY long-lived "hadrons".

Making quasi-stable R hadrons

Please do the following:

```
cp /opt/hep/share/Pythia8/examples/main28.cc mymain28.cc

cp /opt/hep/share/Pythia8/examples/sps1aNarrowStopGluino.spc

make mymain28

./mymain28 > mymain28.out
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

int main() {
--cut--
    pythia.readString("SUSY:gg2gluinogluino  = on");

  // Use hacked sps1a file, with stop (+su) and gluino made long-lived.
  // This is based on the width being less than 0.2 GeV by default.
  pythia.readString("SLHA:file = sps1aNarrowStopGluino.spc");

  // Allow R-hadron formation.
  pythia.readString("Rhadrons:allow = on");

  // If you want to do the decay separately later,
  // you need to switch off automatic decays.
  pythia.readString("RHadrons:allowDecay = off");

  // Fraction of gluinoballs.
  pythia.readString("RHadrons:probGluinoball = 0.1");

--cut--

  // Allow the R-hadrons to have secondary vertices: set c*tau in mm.
  // Note that width and lifetime can be set independently.
  // (Nonzero small widths are needed e.g. to select branching ratios.)
  pythia.readString("1000002:tau0 = 200.");
  pythia.readString("1000006:tau0 = 250.");
  pythia.readString("1000021:tau0 = 300.");
--cut--
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

int main() {
--cut--
    pythia.readString("SUSY:gg2gluinogluino  = on");

  // Use hacked sps1a file, with stop (+su) and gluino made long-lived.
  // This is based on the width being less than 0.2 GeV by default.
  pythia.readString("SLHA:file = sps1aNarrowStopGluino.spc");

  // Allow R-hadron formation.
  pythia.readString("Rhadrons:allow = on");

  // If you want to do the decay separately later,
  // you need to switch off automatic decays.
  pythia.readString("RHadrons:allowDecay = off");

  // Fraction of gluinoballs.
  pythia.readString("RHadrons:probGluinoball = 0.1");

--cut--

  // Allow the R-hadrons to have secondary vertices: set c*tau in mm.
  // Note that width and lifetime can be set independently.
  // (Nonzero small widths are needed e.g. to select branching ratios.)
  pythia.readString("1000002:tau0 = 200.");
  pythia.readString("1000006:tau0 = 250.");
  pythia.readString("1000021:tau0 = 300.");
--cut--
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

int main() {
--cut--
    pythia.readString("SUSY:gg2gluinogluino  = on");

  // Use hacked sps1a file, with stop (+su) and gluino made long-lived.
  // This is based on the width being less than 0.2 GeV by default.
  pythia.readString("SLHA:file = sps1aNarrowStopGluino.spc");

  // Allow R-hadron formation.
  pythia.readString("Rhadrons:allow = on");

  // If you want to do the decay separately later,
  // you need to switch off automatic decays.
  pythia.readString("RHadrons:allowDecay = off");

  // Fraction of gluinoballs.
  pythia.readString("RHadrons:probGluinoball = 0.1");

--cut--

  // Allow the R-hadrons to have secondary vertices: set c*tau in mm.
  // Note that width and lifetime can be set independently.
  // (Nonzero small widths are needed e.g. to select branching ratios.)
  pythia.readString("1000002:tau0 = 200.");
  pythia.readString("1000006:tau0 = 250.");
  pythia.readString("1000021:tau0 = 300.");
--cut--
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

int main() {
--cut--
    pythia.readString("SUSY:gg2gluinogluino  = on");

  // Use hacked sps1a file, with stop (+su) and gluino made long-lived.
  // This is based on the width being less than 0.2 GeV by default.
  pythia.readString("SLHA:file = sps1aNarrowStopGluino.spc");

  // Allow R-hadron formation.
  pythia.readString("Rhadrons:allow = on");

  // If you want to do the decay separately later,
  // you need to switch off automatic decays.
  pythia.readString("RHadrons:allowDecay = off");

  // Fraction of gluinoballs.
  pythia.readString("RHadrons:probGluinoball = 0.1");

--cut--

  // Allow the R-hadrons to have secondary vertices: set c*tau in mm.
  // Note that width and lifetime can be set independently.
  // (Nonzero small widths are needed e.g. to select branching ratios.)
  pythia.readString("1000002:tau0 = 200.");
  pythia.readString("1000006:tau0 = 250.");
  pythia.readString("1000021:tau0 = 300.");
--cut--
```

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;

int main() {
--cut--
    pythia.readString("SUSY:gg2gluinogluino  = on");

  // Use hacked sps1a file, with stop (+su) and gluino made long-lived.
  // This is based on the width being less than 0.2 GeV by default.
  pythia.readString("SLHA:file = sps1aNarrowStopGluino.spc");

  // Allow R-hadron formation.
  pythia.readString("Rhadrons:allow = on");

  // If you want to do the decay separately later,
  // you need to switch off automatic decays.
  pythia.readString("RHadrons:allowDecay = off");

  // Fraction of gluinoballs.
  pythia.readString("RHadrons:probGluinoball = 0.1");

--cut--

  // Allow the R-hadrons to have secondary vertices: set c*tau in mm.
  // Note that width and lifetime can be set independently.
  // (Nonzero small widths are needed e.g. to select branching ratios.)
  pythia.readString("1000002:tau0 = 200.");
  pythia.readString("1000006:tau0 = 250.");
  pythia.readString("1000021:tau0 = 300.");
--cut--
```

```
--cut--

  // Begin event loop.
  int iAbort = 0;
  for (int iEvent = 0; iEvent < nEvent; ++iEvent) {

    // Generate events. Quit if failure.
    if (!pythia.next()) {

--cut--

    // If you have set R-hadrons stable above,
    // you can still force them to decay at this stage.
    pythia.forceRHadronDecays();
    if (iEvent < nList) pythia.event.list(true);

  // End of event loop.
  }
--cut--
```

```
--cut--

  // Begin event loop.
  int iAbort = 0;
  for (int iEvent = 0; iEvent < nEvent; ++iEvent) {

    // Generate events. Quit if failure.
    if (!pythia.next()) {

--cut--

    // If you have set R-hadrons stable above,
    // you can still force them to decay at this stage.
    pythia.forceRHadronDecays();
    if (iEvent < nList) pythia.event.list(true);

  // End of event loop.
  }
--cut--
```

```
--cut--
##       The SUSY decays have calculated using SDECAY 1.1a         *
##                                                                 *
##*****************************************************************
#
--cut--
#
BLOCK MASS  # Mass Spectrum
# PDG code           mass       particle
--cut--
   2000015     2.06867805E+02   # ~tau_2
   1000016     1.84708464E+02   # ~nu_tauL
   1000021     6.07713704E+02   # ~g
   1000022     9.66880686E+01   # ~chi_10
   1000023     1.81088157E+02   # ~chi_20
--cut--
#
#         PDG            Width
DECAY   1000021     0.00001E+00   # gluino decays
#          BR         NDA      ID1       ID2
     2.08454202E-02    2     1000001        -1   # BR(~g -> ~d_L  db)
     2.08454202E-02    2    -1000001         1   # BR(~g -> ~d_L* d )
     5.07075274E-02    2     2000001        -1   # BR(~g -> ~d_R  db)
     5.07075274E-02    2    -2000001         1   # BR(~g -> ~d_R* d )
--cut--
     4.80642793E-02    2     1000006        -6   # BR(~g -> ~t_1  tb)
     4.80642793E-02    2    -1000006         6   # BR(~g -> ~t_1* t )
     0.00000000E+00    2     2000006        -6   # BR(~g -> ~t_2  tb)
     0.00000000E+00    2    -2000006         6   # BR(~g -> ~t_2* t )
--cut--
```