Dartmouth

TRIUMF

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

PYTHIA8

Bootcamp

Part 2

Stephen Mrenna
Fermilab[1]

October 25, 2017

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

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)

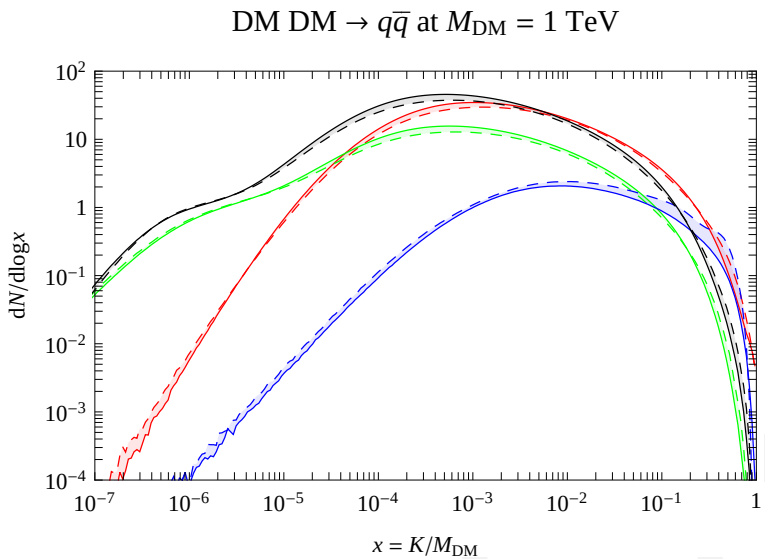
See `examples`

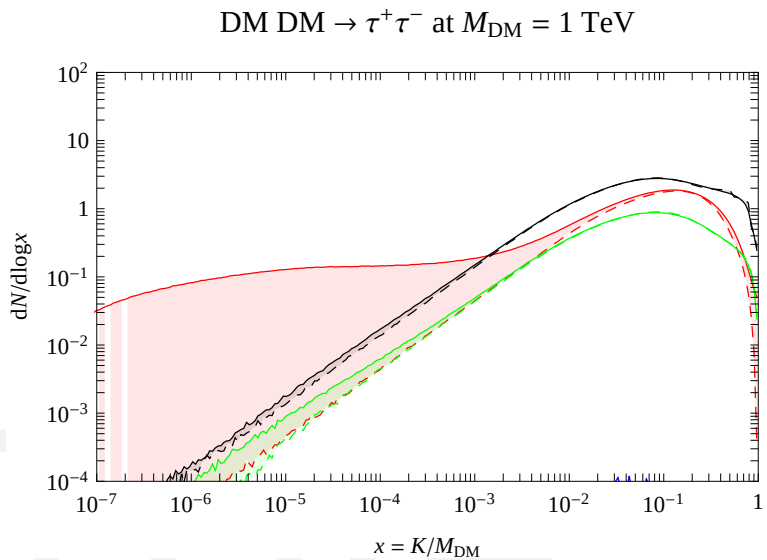
E.g., AMCatNLO requires global sharing of energy-momentum for 1st few PYTHIA FSR emissions

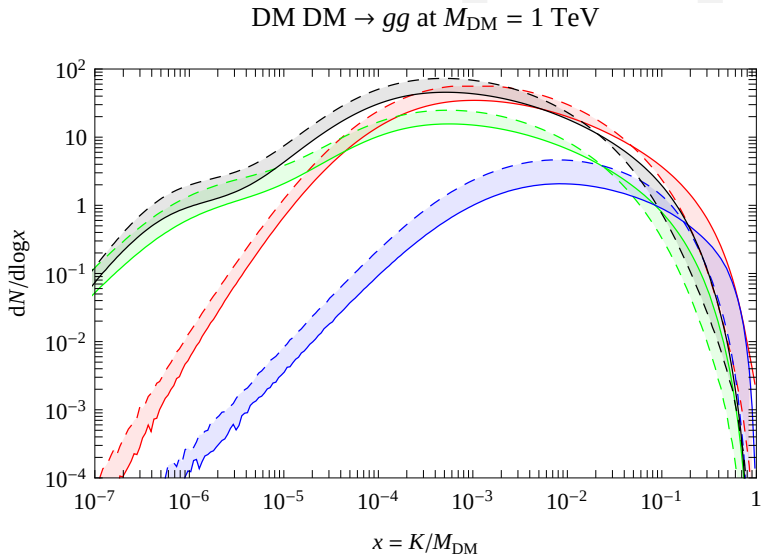
Dark Matter annihilation

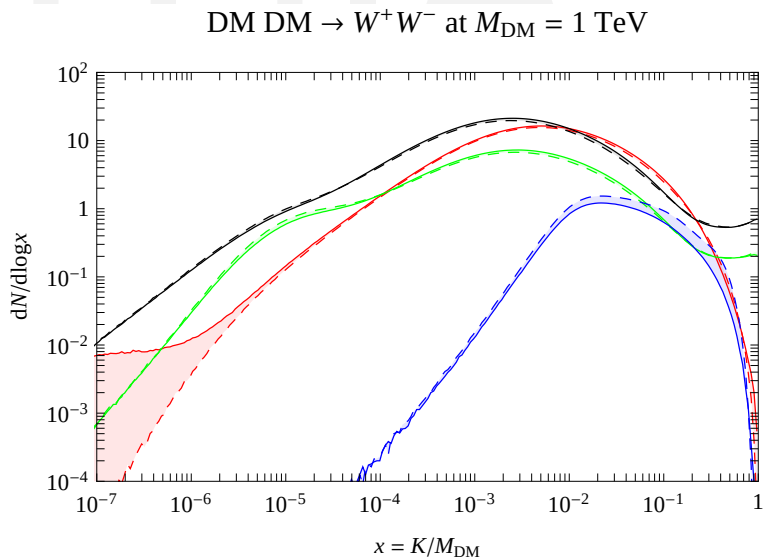
JCAP 1103 (2011) 051

Given mass and BRs, what is the observable spectrum?









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?

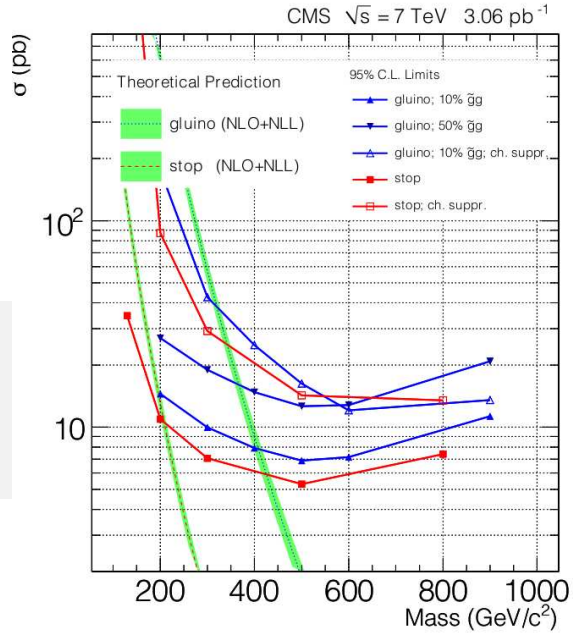
⋆ Formation of R-hadrons

$\tilde{g}q\bar{q}$	$\tilde{t}_1\bar{q}$	"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--
```