Pythia 8

Philip Ilten
on behalf of the Pythia 8 Team

Massachusetts Institute of Technology

July 20, 2016

10th MC4BSM, Beijing

History

- what is PYTHIA?
  A program for the generation of high-energy physics events, i.e. for the description of collisions at high energies between elementary particles ...

- 1978: JETSET from the Lund theory group
- 1997: merged into FORTRAN based PYTHIA 6
- 2004: rewrite into C++ began
- 2007: first release of C++ based PYTHIA 8.1
- 2014: mature PYTHIA 8.2 released

- and its name?
  Apollon founded the Pythic Oracle in Delphi ... Questions were to be put to the Pythia ... The history of the Pythia program is neither as long nor as dignified as that of its eponym.

Documentation

- *An Introduction to PYTHIA 8.2*
  Comput. Phys. Commun. **191**, 159 (2015)
- *PYTHIA 6.4 Physics and Manual*
  JHEP **0605**, 026 (2006)
- any of the original research cited from the manuals

- *Pythia 8 online manual* (also provided with source)
  `http://home.thep.lu.se/~torbjorn/pythia82html/Welcome.html`
- DOXYGEN documentation (also via `help` in PYTHON)
  `http://home.thep.lu.se/~torbjorn/doxygen/annotated.html`
- `mainXY.cc` and `mainXY.py` in `examples` directory

- email the author list or the authors directly
  `http://home.thep.lu.se/~torbjorn/pythiaaux/contact.html`

Getting Started

- download and untar the PYTHIA 8 source

```
wget http://home.thep.lu.se/~torbjorn/pythia8/pythia8219.tgz
tar -xzvf pythia8219.tgz
cd pythia8219
```

- configure and compile the source (just requires C++ compiler)

```
./configure
make
```

- build and run an example

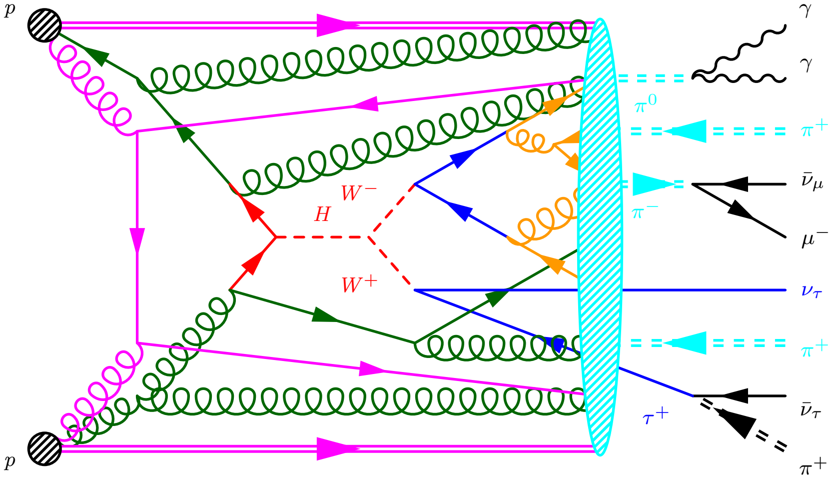
```
cd examples
make main01
./main01
```

```
// Example main program.
#include "Pythia8/Pythia.h"
void main() {
  // Initialize.
  Pythia8::Pythia pythia;
  pythia.readString("HiggsSM:all = on");
  pythia.init();
  pythia.next();
}
```

Event Anatomy

1) ***hard process*** 3) ***ISR*** 5) underlying event 7) ***particle decays***

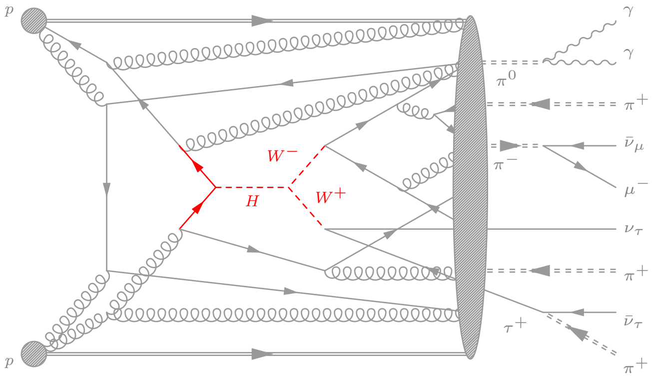
2) ***resonance decays*** 4) ***FSR*** 6) hadronization



Internal Hard Processes

1) *hard process* 3) *ISR* 5) underlying event 7) *particle decays*

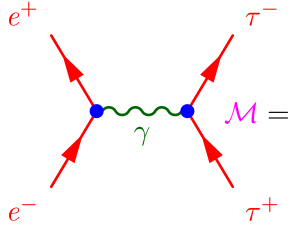
2) *resonance decays* 4) *FSR* 6) hadronization



Introduction

$$\mathcal{L}_{\text{QED}} = i\psi^\dagger\gamma_\mu\partial^\mu\psi - m\psi^\dagger\psi - iQg_e A_\mu\psi^\dagger\gamma^\mu\psi - \left(\partial^\mu A^\nu - \partial^\nu A^\mu\right)\left(\partial_\mu A_\nu - \partial_\nu A_\mu\right)$$

- build matrix element from diagrams



$$\mathcal{M} = \bar{v}_{e^+}\, iQg_e\gamma^\mu\, u_{e^-}\, \frac{-ig_{\mu\nu}}{q^2}\, \bar{u}_{\tau^-}\, iQg_e\gamma^\nu\, v_{\tau^+}$$

- integrate over phase-space for partonic cross-section

$$\hat{\sigma} = \int \left(\frac{1}{8\pi}\right)^2 \frac{\left\langle\, |\mathcal{M}|^2 \,\right\rangle}{\left(E_{e^-} + E_{e^+}\right)} \frac{\left|\vec{p}_{\mu^-}\right|}{\left|\vec{p}_{e^-}\right|}\, \mathrm{d}\Omega$$

- convolute with PDFs for full cross-section

$$\sigma_{a_1 a_2 \to B} = \int\int x_{a_1}\left(x_{p_1}, Q^2, p_1\right) x_{a_2}\left(x_{p_2}, Q^2, p_2\right) \sigma_{p_1 p_2 \to B}\, \mathrm{d}x_{p_1}\, \mathrm{d}x_{p_2}$$

SUSY

[arXiv:1109.5852]

- MSSM and nMSSM implementations of SUSY production
  - all MSSM cross-sections validated
  - $q\bar{q} \to \tilde{\chi}^0\tilde{\chi}^0$, $q\bar{q} \to \tilde{\chi}^\pm\tilde{\chi}^0$, $q\bar{q} \to \tilde{\chi}^+\tilde{\chi}^-$
  - $q\bar{g} \to \tilde{\chi}^0\tilde{q}$, $q\bar{g} \to \tilde{\chi}^\pm\tilde{q}$
  - $gg \to \tilde{g}\tilde{g}$, $q\bar{q} \to \tilde{g}\tilde{g}$
  - $qg \to \tilde{q}\tilde{g}$
  - $gg \to \tilde{q}\bar{\tilde{q}}$, $q\bar{q} \to \tilde{q}\bar{\tilde{q}}$, $qq \to \tilde{q}\tilde{q}$
  - $qq \to \bar{\tilde{q}}$
- $\tilde{q}\bar{\tilde{q}}$ and $\tilde{q}\tilde{q}$ processes include EW contributions
  - `qq2squarksquark:onlyQCD` and `qq2squarkantisquark:onlyQCD`
- possible to turn on all SUSY production and select requested final state(s)
  - `SUSY:idA`, `SUSY:idB`, `SUSY:idVecA`, `Susy:idVecB`

SUSY

[arXiv:1109.5852]

- super-CKM basis used to describe the mass eigenstates
  - $R^u : (\tilde{u}_L, \tilde{c}_L, \tilde{t}_L, \tilde{u}_R, \tilde{c}_R, \tilde{t}_R) \to (\tilde{u}_1, \tilde{u}_2, \tilde{u}_3, \tilde{u}_4, \tilde{u}_5, \tilde{u}_6)$
  - $R^d : (\tilde{d}_L, \tilde{s}_L, \tilde{b}_L, \tilde{d}_R, \tilde{s}_R, \tilde{b}_R) \to (\tilde{d}_1, \tilde{d}_2, \tilde{d}_3, \tilde{d}_4, \tilde{d}_5, \tilde{d}_6)$
  - $\mathcal{N} : (i\tilde{B}, -i\tilde{W}_3, H_1, H_2) \to (\tilde{\chi}_1^0, \tilde{\chi}_2^0, \tilde{\chi}_3^0, \tilde{\chi}_4^0)$
  - $\mathcal{U} : (iW^+, H^+) \to (\tilde{\chi}_1^+, \tilde{\chi}_2^+)$
  - $\mathcal{V} : (iW^-, H^-) \to (\tilde{\chi}_1^-, \tilde{\chi}_2^-)$
- couplings and masses must be provided with an SLHA spectrum
  - from LHEF ⟨`slha`⟩ blocks: `SLHA:readFrom = 1`
  - directly from SLHA file: `SLHA:file = spectrum.slha`
  - either SLHA 1 or 2 allowed, but SLHA 2 preferred
  - fine-grained options on which parameters taken from SLHA

Higgs

- SM Higgs production
  - $f\bar{f} \to H$, $gg \to H$, $\gamma\gamma \to H$
  - $f\bar{f} \to HZ$, $f\bar{f} \to HW^{\pm}$
  - $f\bar{f} \to Hf\bar{f}$, $q\bar{q} \to Ht\bar{t}/b\bar{b}$, $gg \to Ht\bar{t}/b\bar{b}$
  - $qg \to Hq$, $gg \to Hg$, $q\bar{q} \to Hg$
  - partial widths can be scaled to NLO: `HiggsSM:NLOWidths`
- BSM Higgs production
  - generic two Higgs doublet model, H_u and H_d
  - five Higgs bosons defined: h^0/H_1^0(`H1`), H^0/H_2^0(`H2`), A^0/H_3^0(`A3`), $H^{\pm}$
  - $f\bar{f} \to H^{\pm}$, $bg \to H^{\pm}t$
  - $f\bar{f} \to A^0h^0$, $f\bar{f} \to A^0H^0$, $f\bar{f} \to H^{\pm}h^0$, $f\bar{f} \to H^{\pm}H^0$, $f\bar{f} \to H^+H^-$
- standard MSSM $\mathcal{CP}$-even/$\mathcal{CP}$-odd and mass ordering not required
  - couplings can be read from SLHA spectrum
  - can also be set via individual coupling parameters: `HiggsH1:coup2d`, `H1:coup2u`, ...

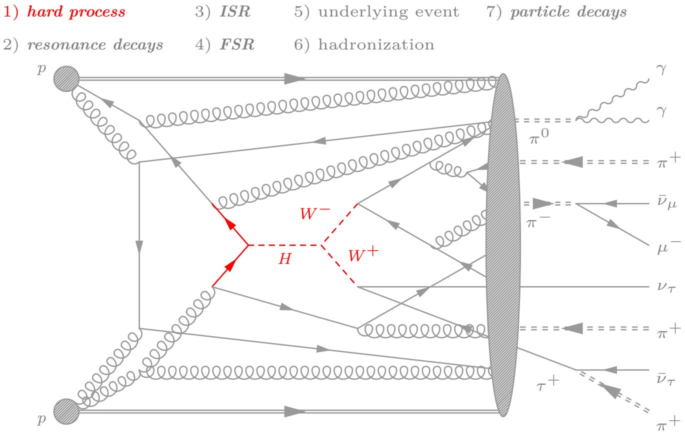
More BSM

- new gauge boson production
  - $f\bar{f} \to Z'$, $f\bar{f} \to W'^{\pm}$
  - fully flexible $\gamma/Z/Z'$ interference: Zprime:gmZmode
  - non-universal couplings allowed: Zprime:vd, Zprime:ad, ...
- left-right symmetries
  - includes a $SU(2)_R$ group at a larger scale from the SM $SU(2)_L$
  - based on the model of Nucl. Phys. B **487**, 27 (1997)
  - includes Z_R, $W_R^{\pm}$, and $H_L^{++/--}$ production
- leptoquarks
  - simple scalar leptoquark model with arbitrary quark/lepton flavor
  - $q\ell \to LQ$, $qg \to LQ\,\ell$, $gg \to LQ\,\overline{LQ}$, $q\bar{q} \to LQ\,\overline{LQ}$
  - default ue^- LQ-numbers can be changed: 42:0:products = Q L
  - required to decay *before* fragmentation, cannot be stable
  - cross-section and width modified by k-factor: LeptoQuark:kCoup

Even More BSM

- compositeness
  - composite fermions can result in excited sharp resonances
  - $2 \to 1$ processes via gauge boson interactions: $dg \to d^*, \ldots$
  - $2 \to 2$ processes via contact interactions: $qq \to d^* q, \ldots$
  - decays include matrix element corrections
  - only gauge boson interaction decays implemented: $d^* \to \gamma d$
- hidden valleys
  - based on the work of JHEP **1104**, 091 (2011)
  - hidden unbroken $SU(N)$ symmetry: `HiddenValley:Ngauge`
  - hidden sector mirrors SM fermions: d_v, e_v^-, $\ldots$
  - $gg \to q_v \bar{q}_v$, $q\bar{q} \to g \to q_v \bar{q}_v$, $f\bar{f} \to \gamma^*/Z \to f_v \bar{f}_v$
  - f_v can radiate g, γ, and γ_v/g_v
- extra dimensions
  - Randall-Sundrum resonances based on Phys. Lett. B **503**, 341 (2001) (G^*) and JHEP **1201**, 018 (2012) (Kaluza-Klein g_{KK})
  - γ_{KK} and Z_{KK} excited electroweak resonances and unparticle production

External Hard Processes



1) hard process
2) resonance decays
3) ISR
4) FSR
5) underlying event
6) hadronization
7) particle decays
p
p
W^-
W^+
H
γ
γ
π^0
π^+
$\bar{\nu}_\mu$
π^-
μ^-
ν_τ
π^+
τ^+
$\bar{\nu}_\tau$
π^+

Les Houches Accord

- read in Les Houches Event Format files with versions 1, 2, or 3
  - set beam input to LHEF: `Beams:frameType = 4`
  - provide the LHEF name: `Beams:LHEF = events.lhe`
  - optionally provide separate header: `Beams:LHEFheader = header.lhe`
  - full examples provided in `main25.cc`, `main31.cc`, `main32.cc`, `main37.cc`, `main38.cc`, and `main43.cc`
- create an `LHAup` derived class to pass LHA information to Pythia
  - set beam input to `LHAup`: `Beams:frameType = 5`
  - pass `LHAup` pointer to Pythia instance

```
pythia.setLHAupPtr(LHAupPtr);
```

  - `LHAupFortran` reads `HEPRUP` and `HEPEUP` Fortran common blocks

Semi-Internal Process

- create a `SigmaProcess` derived class to pass to PYTHIA
  - `Sigma1Process`, `Sigma2Process`, and `Sigma3Process` for $2 \to 1$, 2, and 3

```
pythia.setSigmaPtr(SigmaProcessPtr);
```

  - example in `main22.cc`
- `double SigmaProcess::sigmaHat()` calculates the cross-section
  - $2 \to 1$: return $\hat{\sigma}(\hat{s})$
  - $2 \to 2$: return $\mathrm{d}\hat{\sigma}/\mathrm{d}\hat{t}$
  - $2 \to 3$: return $|\mathcal{M}|^2$ with normalization $\hat{\sigma} = \int |\mathcal{M}|^2 /(2\hat{s})\,\mathrm{d}\Phi$
- `string SigmaProcess::inFlux()` defines the incoming partons
  - `gg`, `qg`, `fgm`, `ggm`, `gmgm`
  - `qq`, `qqbar`, `qqbarSame`, `ff`, `ffbarSame`, `ffbarChg`

PowhegBox

- PowhegBox matrix elements, see http://powhegbox.mib.infn.it, can be passed via `LHAup` pointer
  - PowhegBox binaries require special compilation flags

```
sed -i "s/F77= gfortran/F77= gfortran -rdynamic -fPIE -fPIC -pie/g→
    " Makefile
```

  - configure Pythia with `--with-powheg-bin=/powheg/bin`
- `PowhegProcsIn` handles loading the `LHAup` pointer
  - full example given in `main33.cc`

```
Pythia pythia;
PowhegProcs procs(&pythia, "hvq");
// Read POWHEGBOX configuration from file.
procs.read
// Or read from strings passed.
procs.readString("ih1 1");
pythia.readString("Beams:frameType = 5");
procs.init();
pythia.init();
```

- shower matching parameters should be set: `POWHEG:*`

MadGraph

- two options for using MadGraph with Pythia
  - output LHEF and pass to Pythia
  - create `SigmaProcess` class and pass to Pythia
- first option automated through `LHAupMadgraph` class
  - can run with either `MG5` or `aMC@NLO` output
  - configure Pythia with `--with-gzip`
  - creates a GRID-pack structure for fast additional runs
  - full example given in `main34.cc`

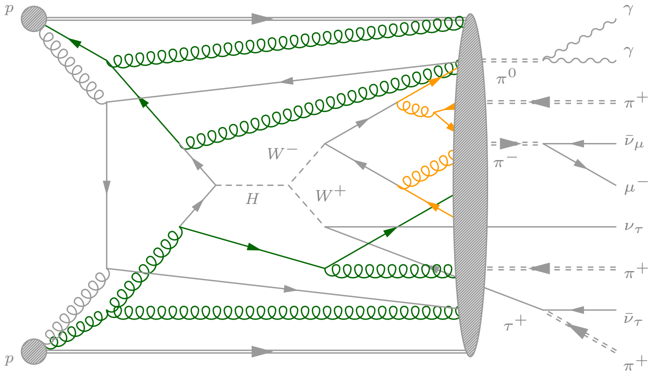
```
LHAupMadgraph madgraph(&pythia, true, "madgraphrun", "mg5_aMC");
madgraph.readString("generate p p > mu+ mu-");
madgraph.readString(" set ebeam1 6500");
madgraph.readString(" set ebeam2 6500");
madgraph.readString(" set mmll 80");
pythia.readString("Random:setSeed = on");
pythia.readString("Random:seed = 1");
pythia.setLHAupPtr(&madgraph);
pythia.init();
```

- remember to set up matching/merging correctly

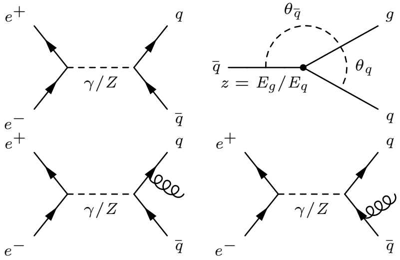
Showers

1) ***hard process*** 3) ***ISR*** 5) underlying event 7) ***particle decays***

2) ***resonance decays*** 4) ***FSR*** 6) hadronization



Introduction



- diverges for three scenarios
  - $z \to 0$ (soft)
  - $\theta \to 0$ (collinear to q)
  - $\theta \to \pi$ (collinear to $\bar{q}$)

- factorize collinear divergences as independent emissions

$$\mathrm{d}\sigma_{e^+e^-\to q\bar{q}g} \approx \sigma_{e^+e^-\to q\bar{q}} \sum_i \left(\left(\frac{\mathrm{d}\theta^2_{p_i}}{\theta^2_{p_i}} \right) \left(\frac{\alpha_s}{2\pi} \right) \left(\frac{N_c^2 - 1}{2N_c} \right) \left(\frac{1 + (1-z)^2}{z} \right) \mathrm{d}z \right)$$

- generalize for all processes with splitting functions $\mathcal{P}_{b_j b_i}$

$$\mathrm{d}\sigma_{A\to Bb_j} \approx \sigma_{A\to B} \sum_i \left(\left(\frac{\mathrm{d}\theta^2_{b_i}}{\theta^2_{b_i}} \right) \mathcal{P}_{b_j b_i}(z, \alpha_s)\, \mathrm{d}z \right)$$

- parton b_i emits parton b_j

Internal Showers

- timelike shower (final state radiation) is fully interleaved p_{T}-ordered

$$\Delta_{ij}(q_1^2, q_2^2) = \exp\left(-\int_{q_2^2}^{q_1^2} \frac{1}{q^2} \int_{Q_0^2/q^2}^{1-Q_0^2/q^2} \mathcal{P}_{ji}(z, \alpha_s)\, \mathrm{d}z\, \mathrm{d}q^2 \right)$$

- spacelike showers (initial state radiation) is fully interleaved p_{T}-ordered

$$\Delta_{ij}(q_1^2, q_2^2, x) = \exp\left(-\int_{q_2^2}^{q_1^2} \frac{1}{q^2} \int_{Q_0^2/q^2}^{1-Q_0^2/q^2} \mathcal{P}_{ij}(z, \alpha_s) \left(\frac{x}{zx}\right) \left(\frac{f(x/z, q^2, j)}{f(x, q^2, i)}\right) \mathrm{d}z\, \mathrm{d}q^2 \right)$$

  - high Q^2 and small x to small q^2 and large x
- define cut-off Q_0: `TimeShower:pTmaxMatch`, `SpaceShower:pTmaxMatch`
  - `1` - *wimpy*: factorization scale
  - `2` - *power*: half the dipole mass

Automatic Shower Variations

[arXiv:1605.08352]

- available only for QCD showers
- variations on renormalization scale (multiplicative) and non-singular terms (additive)
- variations turned on with: `UncertaintyBands:doVariations`
- specified by: `UncertaintyBands:List = {name fsr:muRfac=0.5 isr:muRfac=0.5, ...}`
- keywords for variable terms
    - `fsr:muRfac`, `isr:muRfac`: renormalization scale factor
    - `fsr:cNS`, `isr:cNS`: additive non-singular term
    - `fsr:G2GG:muRfac`, `fsr:Q2QG:muRfac`, `fsr:G2QQ:muRfac`, `fsr:X2XG:muRfac`: finer grain control
- accessed via `Pythia::info.weight(i)`

Weak Showers

[JHEP **1404**, 115 (2014)]

- weak showers are available, with a few caveats
  - Bloch-Nordsieck violations from $W^{\pm}$ flavor changing are not accounted for
  - γ^*/Z interference is not handled: low masses use γ^* and high masses are Z
- activated via: `TimeShower:weakShower` and `SpaceShower:weakShower`
- specify the allowed splittings: `TimeShower:weakShowerMode` and `TimeShower:weakShowerMode`
  - `0` $W^{\pm}$ and Z
  - `1` only $W^{\pm}$
  - `1` only Z

External Showers

- DiRe (dipole resummation)
  - Eur. Phys. J. C **75**, no. 9, 461 (2015)
  - careful treatment of collinear enhancements
  - very modular and extensible
  - implemented both as Pythia plugin and within Sherpa
  - available from `https://direforpythia.hepforge.org/`
- Vincia (Virtual Numerical Collider with Interleaved Antennae)
  - arXiv:1605.06142
  - dipole-antenna shower plugin to Pythia
  - provide $2 \to 3$ shower kernels
  - captures both collinear dynamics and soft singularities
  - available from `http://vincia.hepforge.org`

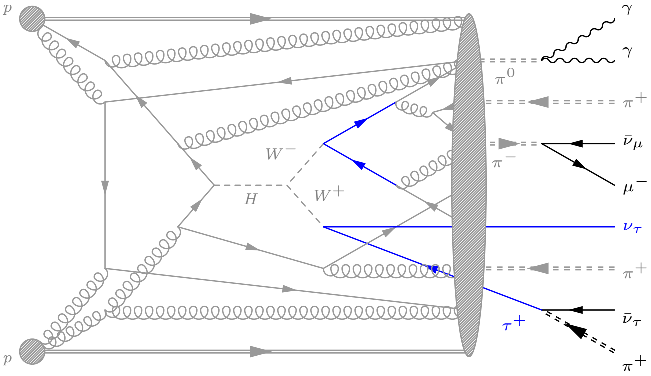
Matching and Merging

- MLM (`main89.cc`)
  - calculate Sudakov factor on all lines
  - shower, reject emission using factor
- CKKW-L
  - perform shower and cluster jets
  - match jets to partons, reject if $N_p \neq N_{\text{jets}}$
- POWHEG (`main31.cc`)
  - pick largest p_T emission from NLO normalized $\mathcal{M}$
  - evolve shower downwards to p_T scale
- UMEPS
  - unitarized matrix element and parton shower merging
  - tree-level n-leg merging without inclusive cross-section modification
- UNLOPS
  - unitarized next-to-leading-order parton shower
  - n-leg merging at NLO but more generalized
- FXFX
  - R. Frederix and S. Frixione
  - merging and matching of aMC@NLO

Decays

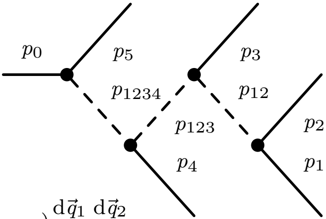
1) *hard process* 3) ***ISR*** 5) underlying event **7) *particle decays***

2) *resonance decays* 4) ***FSR*** 6) hadronization



Introduction

- m-generator
  - mass generator
  - two-body decays through intermediate masses



$$d\Phi_2(q_0, q_1, q_2) = \left(\frac{1}{(2\pi)^2 2^2}\right) \delta(q_0 - q_1 - q_2) \frac{d\vec{q}_1}{E_1} \frac{d\vec{q}_2}{E_2}$$

$$d\Phi_3(q_0, q_1, q_2, q_3) = \left(\frac{2}{\pi}\right) d\Phi_2(q_0, q_{12}, q_3) m_{12}\, dm_{12}\, d\Phi_2(q_{12}, q_1, q_2)$$

- re-weight by $\mathcal{M}$ for the phase-space point
- difference between *resonance* and *particle* purely technical
  - *resonances*: states with a typical lifetime shorter than the hadronization scale
  - *particles*: states with a lifetime comparable to or longer than the hadronization scale

Resonance Decays

- any state with $m_0 > 20$ GeV is resonance by default
  - light SUSY particles also treated as resonances
- resonance branching fractions modify the relevant cross-section
- SUSY resonance decays implemented with weighting
  - $\tilde{q} \rightarrow \tilde{\chi}$, $\tilde{q}W/Z$, qq, lq
  - $\tilde{g} \rightarrow \tilde{q}q$
  - $\tilde{\chi} \rightarrow \tilde{\chi}W/Z$, $\tilde{q}q$, $\tilde{\ell}\ell$
  - $\tilde{\chi}^0 \rightarrow qqq$
  - $\tilde{\ell} \rightarrow \tilde{l}\tilde{\chi}$, $\tilde{\ell}W/Z$
- long-lived $\tilde{g}$, $\tilde{b}$, and $\tilde{t}$ can be allowed to hadronize: `RHadrons:allow`
- unknown resonances decayed with flat phase-space decays
  - partial width can be forced, or allowed to run with various schemes

Particle Decays

- most particle decays are flat phase-space with some exceptions
  - $\omega, \phi \to \pi^+\pi^-\pi^0$
  - $V \to PS\,PS$ with V from $PS \to PS\,V$ or $PS \to \gamma V$
  - Dalitz decay $X \to Y\ell^+\ell^-$
  - double Dalitz decay $X \to \ell^+\ell^-\ell^+\ell^-$
  - weak decays
  - $B \to \gamma X$
- τ decays are not flat phase-space and can handle spin effects
  - spin effects calculated internally
  - correlations handled for W, Z, $W^{'}$, $Z^{'}$, H, h^0, H^0, A^0, and $H^{\pm}$
  - lepton-flavor violating gauge boson decays allowed
  - $\mathcal{CP}$-mixing of the extended Higgs can be specified
  - external spin information can override internal calculation

External Decays

- create a `ResonanceWidths` derived class to pass to PYTHIA
  - does not re-weight the decay
  - `void ResonanceWidth::calcWidth` calculates total width
  - full example given in `main22.cc`
- create a `DecayHandler` derived class to pass to PYTHIA
  - $1 \to n$ decays specified via `bool DecayHandler::decay`
  - chains specified via `bool DecayHandler::decayChain`
  - full example given in `main17.cc`
- external decays via EVTGEN, http://evtgen.warwick.ac.uk
  - plugin class `EvtGenDecays` applies decays to PYTHIA event record
  - allows forced decays and provides event weight
  - full example given in `main48.cc`

Outlook

Outlook

- PYTHIA designed to be simple and easy-to-use, yet flexible
- large selection of internal hard processes for fast use
  - external hard process from LHEF input, `LHAup` pointers, or `SigmaProcess`
  - dedicated plugins for POWHEGBOX and MADGRAPH
- robust shower algorithms
  - Hidden Valley showers and weak shower available
  - alternative DIRE and VINCIA shower plugins
  - exhaustive collection of matching and merging schemes
- spin correlated tau decays and resonance SUSY decays
- not mentioned today
  - multi-parton interaction framework
  - Lund string fragmentation model for hadronization
- new PYTHON interface!
- questions on anything? please ask!