Pythia

Patrick Robbe, IJCLab Orsay, 13/05/2026

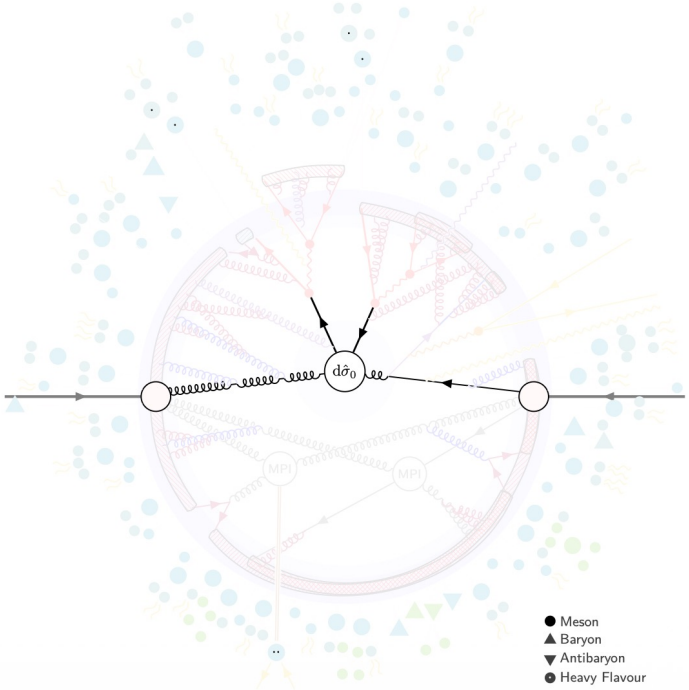
Generation of *pp* collisions

- Most of the events generated for physics analysis in LHCb are *pp* collisions, in general coming from ‘minimum bias’ interactions (=with no selection on the type of *pp* interaction)
- Pythia (Pythia 8 version, which is written in C++) is used for the majority of events
- Home page : https://www.pythia.org/
- Documentation : https://www.pythia.org/documentation/
- Can be run in docker containers:
  - docker pull hepstore/rivet - pythia
  - docker run -v $PWD:/host -it --rm hepstore/rivet – pythia
- The generation of a *pp* interaction is composed of several steps:

Composition of a Pythia event

As a function of the energy scale involved:

1. Hard Process (for ex. $t\bar{t}$)

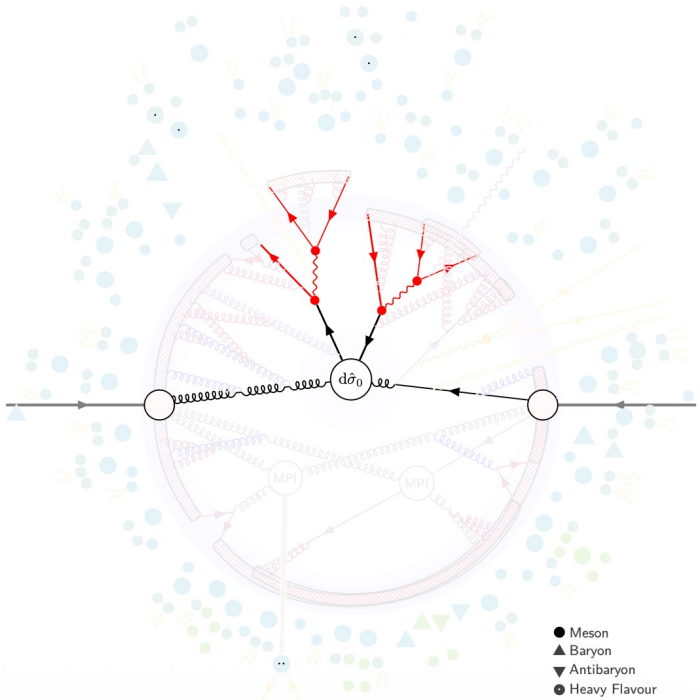


[figure by P. Skands]

Composition of a Pythia event

As a function of the energy scale involved:

1. Hard Process (for ex. $t\bar{t}$)
2. Resonance decays (Z, t, …)

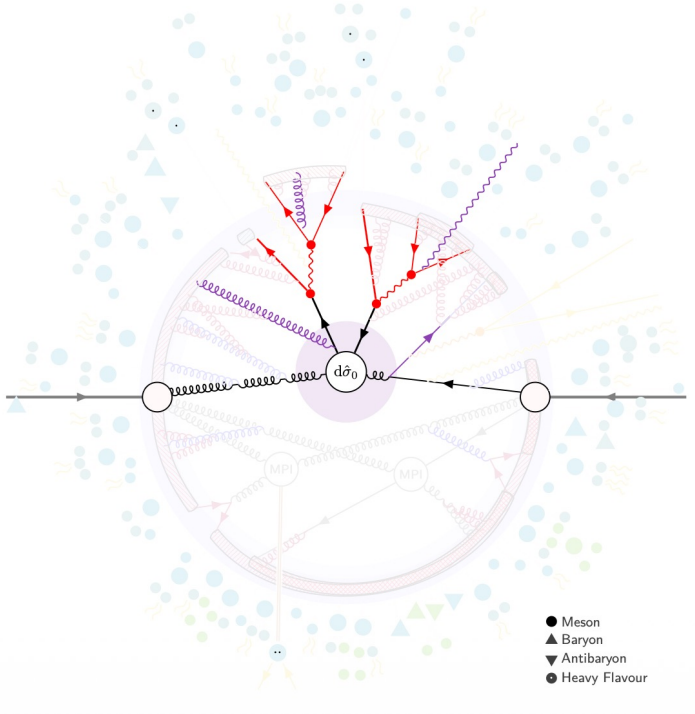


[figure by P. Skands]

Composition of a Pythia event

As a function of the energy scale involved:

1. Hard Process (for ex. $t\bar{t}$)
2. Resonance decays (Z, t, …)
3. Matching, merging and matrix element corrections

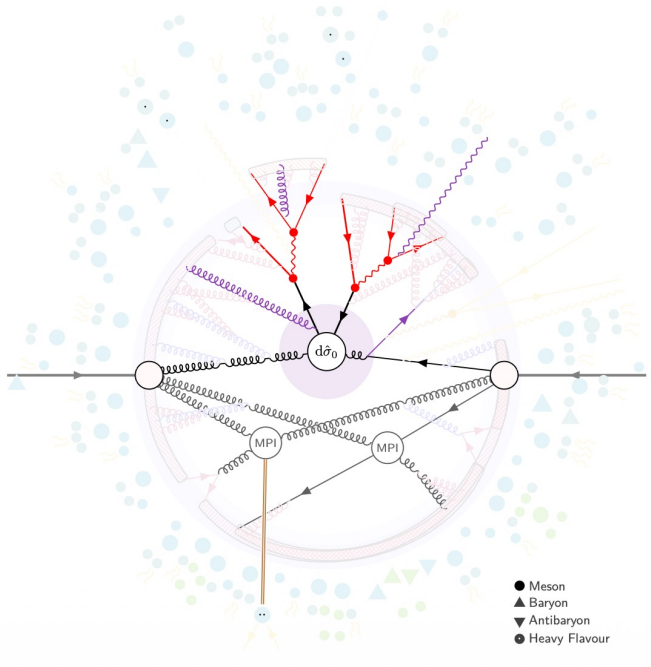


[figure by P. Skands]

Composition of a Pythia event

As a function of the energy scale involved:

1. Hard Process (for ex. $t\bar{t}$)
2. Resonance decays (Z, t, …)
3. Matching, merging and matrix element corrections
4. Multiparton interactions

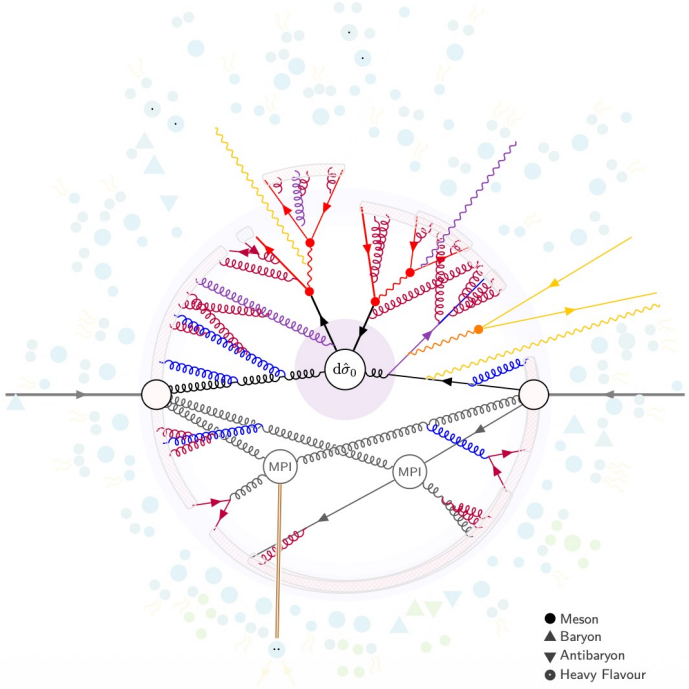


[figure by P. Skands]

Composition of a Pythia event

As a function of the energy scale involved:

1. Hard Process (for ex. $t\bar{t}$)
2. Resonance decays (Z, t, …)
3. Matching, merging and matrix element corrections
4. Multiparton interactions
5. Parton showers: ISR, FSR, QED, Weak

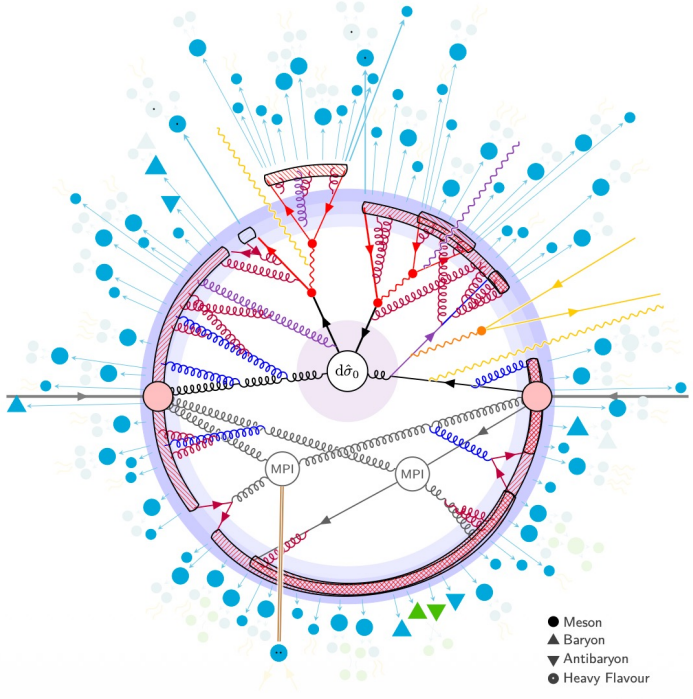


[figure by P. Skands]

Composition of a Pythia event

As a function of the energy scale involved:

1. Hard Process (for ex. $t\bar{t}$)
2. Resonance decays (Z, t, …)
3. Matching, merging and matrix element corrections
4. Multiparton interactions
5. Parton showers: ISR, FSR, QED, Weak
6. Hadronization, beam remnents

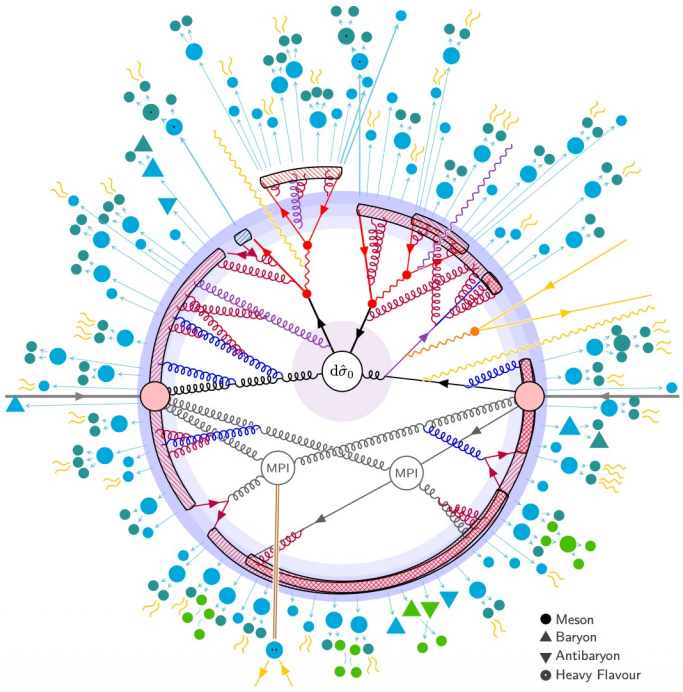


[figure by P. Skands]

Composition of a Pythia event

As a function of the energy scale involved:

1. Hard Process (for ex. $t\bar{t}$)
2. Resonance decays (Z, t, …)
3. Matching, merging and matrix element corrections
4. Multiparton interactions
5. Parton showers: ISR, FSR, QED, Weak
6. Hadronization, beam remnents
7. Decays (done almost all in EvtGen in LHCb), rescattering



[figure by P. Skands]

Hard processes

- Available internally in Pythia, used in LHCb for standard events

QCD processes
- Hard $2 \to 2$ partonic scatterings (some $2 \to 3$)
- Heavy-quark production

Electroweak processes
- Prompt photon production
- EW boson production and exchange
- Deep inelastic scattering
- Photon collisions

Onia production
- Charmonium, Bottomonium with different spin states

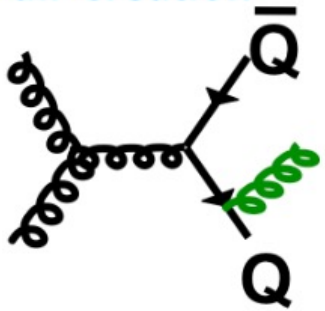
Top production
- $t\bar{t}$ pairs and single top

Higgs production
- Standard-Model Higgs, also in association with other particles
- Beyond-the-Standard-Model Higgs

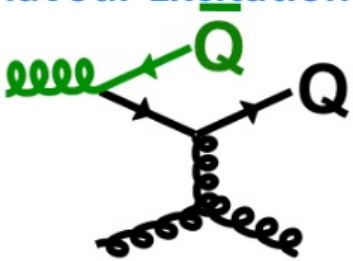
Beyond the Standard Model
- Supersymmetry
- Dark Matter
- Leptoquarks
- ...

Hard processes

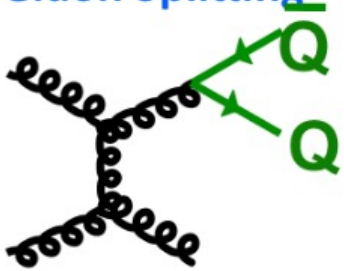
Pair Creation



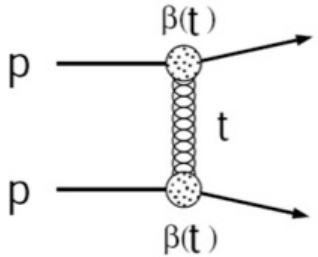
Flavour Excitation



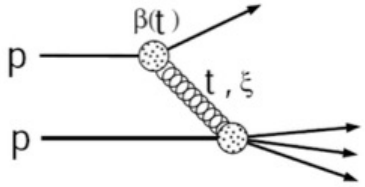
Gluon Splitting



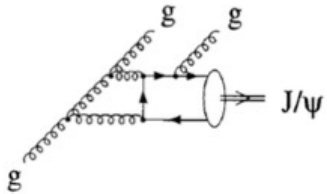
Elastic



Single diffractive

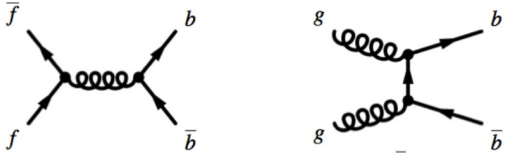


Charmonium production

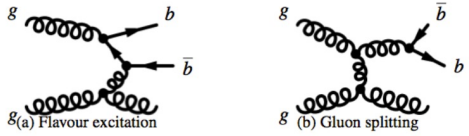


B production in Pythia

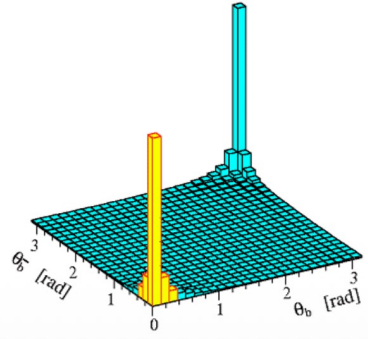
- *B* (and *D*) production processes at lowest order in α_S are the pair creation processes.



- At the LHC energy, higher order processes dominate.



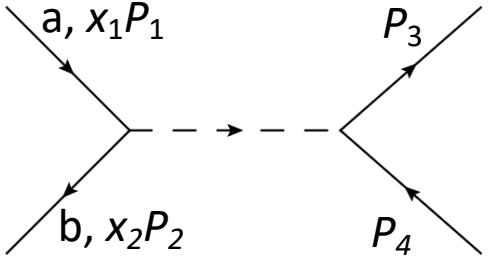
- Gluon splitting causes the $b\bar{b}$ pair to be produced forward or backward.



Hard process: 2 → 2 scattering

- Interaction of 2 protons with momentum P_1 and P_2: interaction of 2 partons a and b with momentum x_1P_1 and x_2P_2

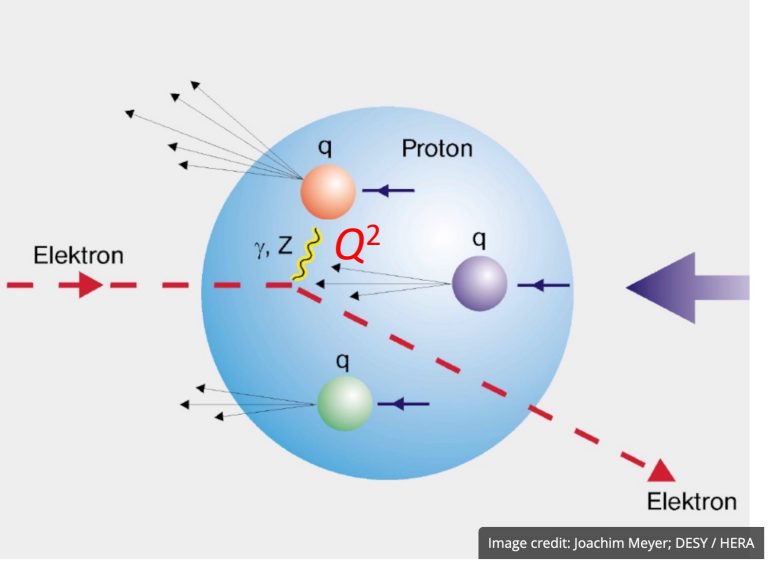
$$\sigma_{ab} = \int \frac{d\tau}{\tau} dy\, d\hat{t}\, x_1 f_a(x_1, Q^2)\, x_2 f_b(x_2, Q^2) \frac{d\hat{\sigma}(\hat{s}, \hat{t}, Q^2)}{d\hat{t}}$$



- Where:
  - a, b are quarks or gluons
  - $f_{a,b}(x_{1,2}, Q^2)$ are parton distribution functions (PDF)
  - Q^2 the factorisation scale (cut off to avoid divergences)
  - $\tau = x_1x_2$, $y = 0.5\log(\frac{x_1}{x_2})$, $\hat{s}$ and $\hat{t}$ the partonic Mandelstam variables ($s = (p_1 + p_2)^2$ and $t = (p_3 - p_1)^2$)
  - $d\hat{\sigma}(\hat{s}, \hat{t}, Q^2)$ is the process cross-section (perturbative)

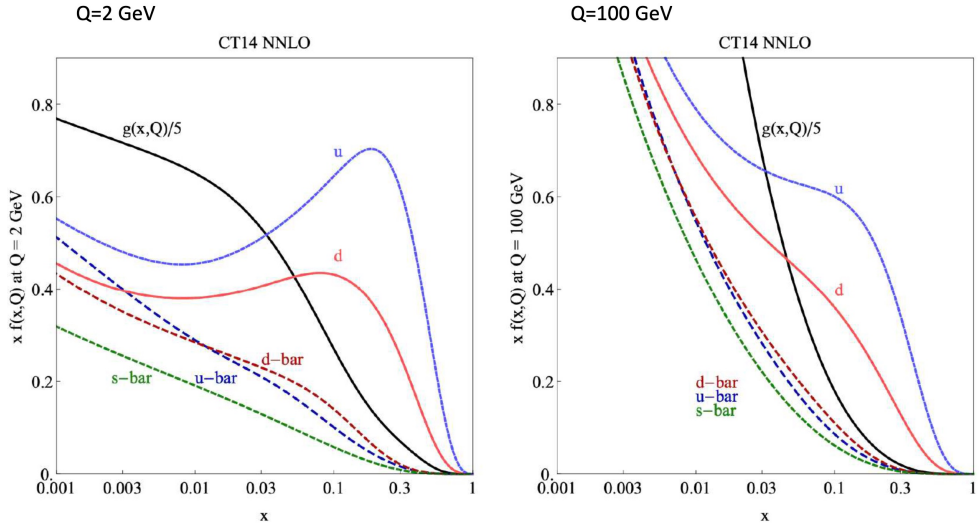
PDFs

- $f_i(x,Q^2)$ is the mean number of partons of type *i* with a fraction *x* of the proton momentum (colinear approximation).
- They depend on the scattering momentum transfer Q^2
- They are measured experimentally with Deep Inelastic Experiments at a given Q^2

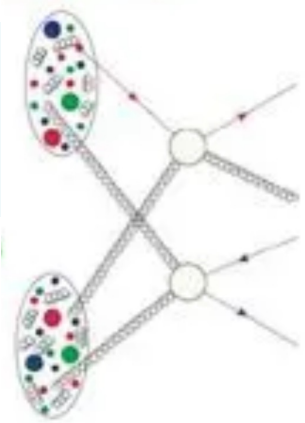


PDFs

- They can be obtained at other Q^2 values using the DGLAP Evolution Equations (Dokshitzer–Gribov–Lipatov–Altarelli–Parisi)
- PDFs used in generators are obtained from complex global QCD analysis of data including higher order (NLO, NNLO) corrections
- In LHCb, we use the CT09MCS PDF set [arXiv:0910.4183]



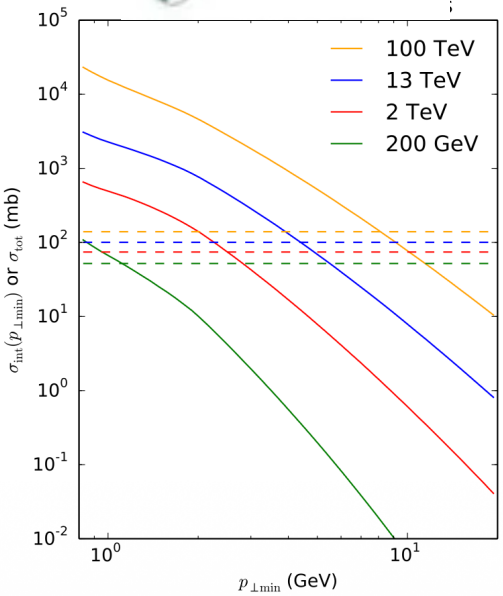
Multiple parton interactions



- Integrated cross-section for QCD 2→2 processes:

$$\sigma_{\mathrm{int}}(p_{\mathrm{T\,min}}) = \int_{p^2_{\mathrm{T\,min}}}^{s/4} \frac{\mathrm{d}\sigma}{\mathrm{d}p^2_{\mathrm{T\,min}}} \mathrm{d}p^2_{\mathrm{T\,min}}$$

- $\sigma_{\mathrm{int}} > \sigma_{\mathrm{tot}}$ for small $p_{\mathrm{T\,min}}$:
  - multiple partonic interactions per event
  - At the LHC, typically 5-6 multiple interactions per *pp* interaction:
    - p_{Tmin} ~ 2 GeV
    - $\sigma_{\mathrm{int}} / \sigma_{\mathrm{tot}}$ ~ 5



Multiple parton interactions

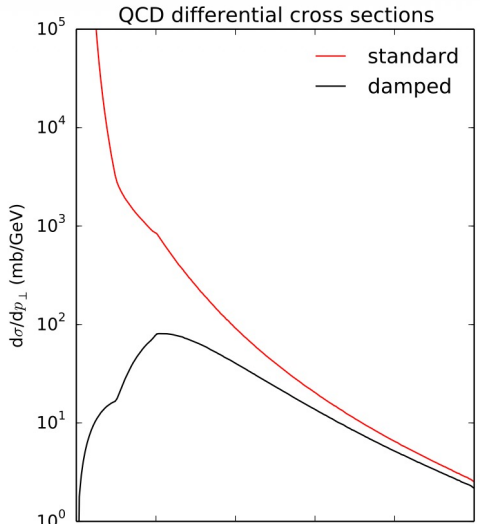
- In addition, $\mathrm{d}\sigma^{2\to2}/\mathrm{d}p_{\mathrm{T}}^2$ diverges at low p_{T}:
  - Introduce a screening parameter p_T^0

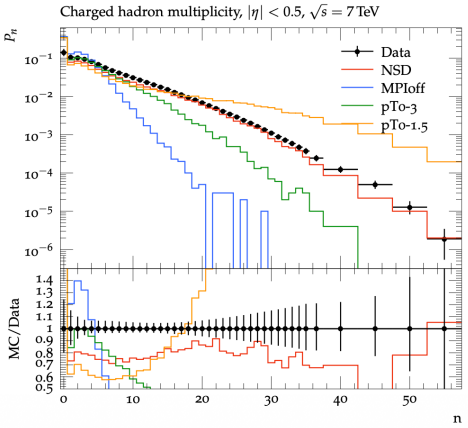
$$\frac{\mathrm{d}\sigma^{2\to2}}{\mathrm{d}p_{\mathrm{T}}^2} \propto \frac{\alpha_{\mathrm{s}}(p_{\mathrm{T}}^2)}{p_{\mathrm{T}}^4} \to \frac{\alpha_{\mathrm{s}}(p_{\mathrm{T0}}^2 + p_{\mathrm{T}}^2)}{(p_{\mathrm{T0}}^2 + p_{\mathrm{T}}^2)^2}$$

- With an energy dependent parametrization

$$p_{\mathrm{T0}}(\sqrt{s}) = p_{\mathrm{T0}}^{\mathrm{ref}}(\sqrt{s}/\sqrt{s_{\mathrm{ref}}})^{\alpha}$$

- Multiple parton interactions are very important to generate low p_{T} processes. This is the main handle used in LHCb to tune PYTHIA and reproduce the charged particle multiplicity.

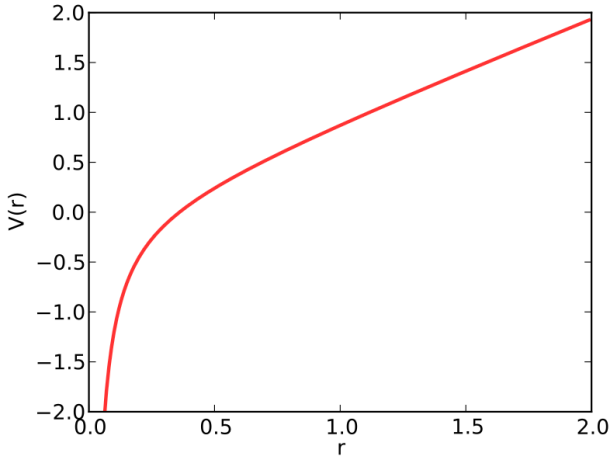


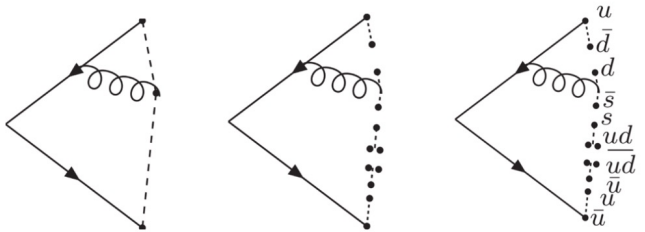


[CMS: JHEP 01 (2011) 079]

Hadronisation

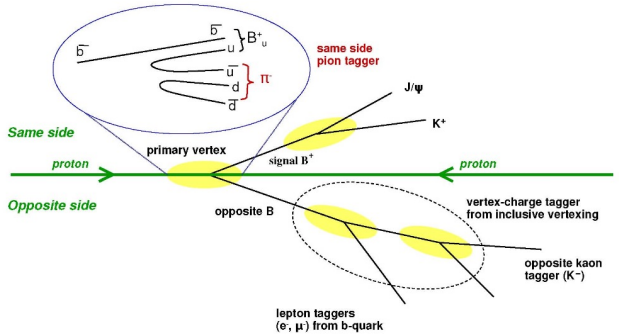
- Important parameter for LHCb also
- Based on ”Lund string model”, derived from the QCD Cornell potential $V(r) = -\frac{\frac{4}{3}\alpha_s}{r} + \kappa r$
- Partons are connected by strings which breaks creating new $q\ \bar{q}'$ pairs that give hadrons h
- Probability to create h with a momentum fraction z is $\frac{(1-z)^a}{z}\exp(-b\frac{m_{T,h}}{z})$
- The type of hadrons produced h is obtained from quark content and a parameterization of the fraction of states with different angular momentum

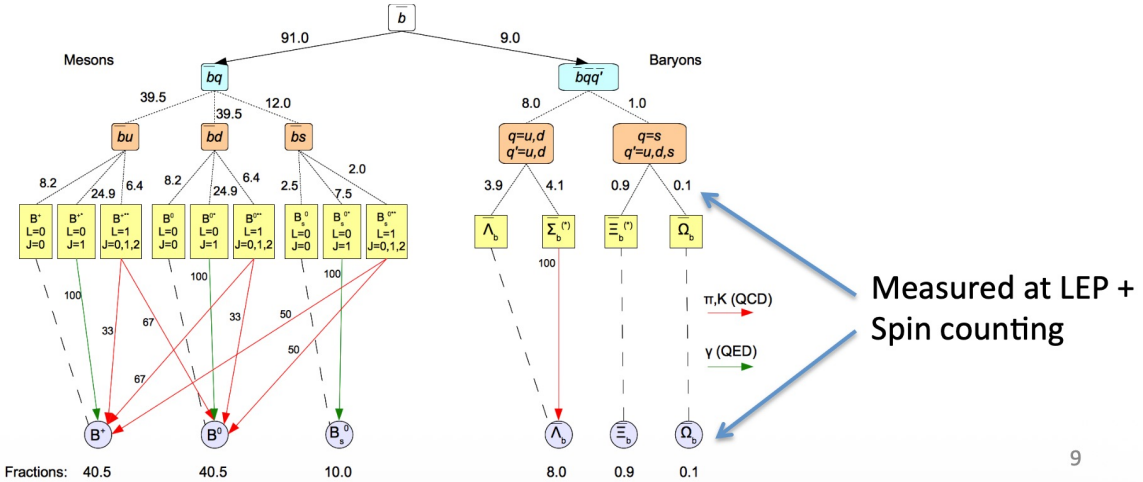




Excited *B* states tuning in Pythia

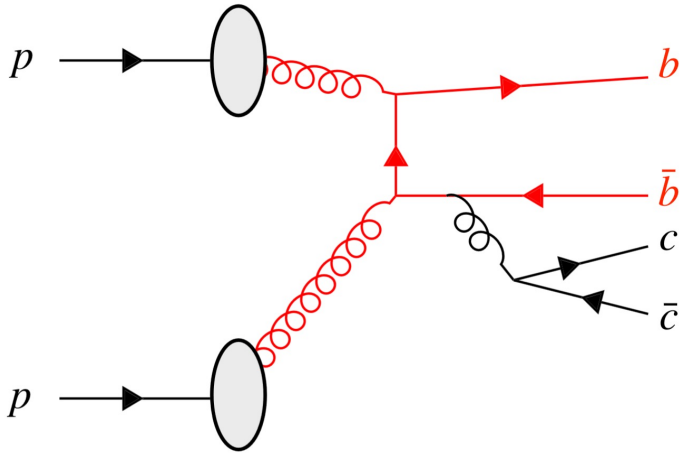
- *B* flavour tagging in hadron colliders is based on the properties of the other *B* decay in the event, but also on the fragmentation characteristics of the signal *B*.



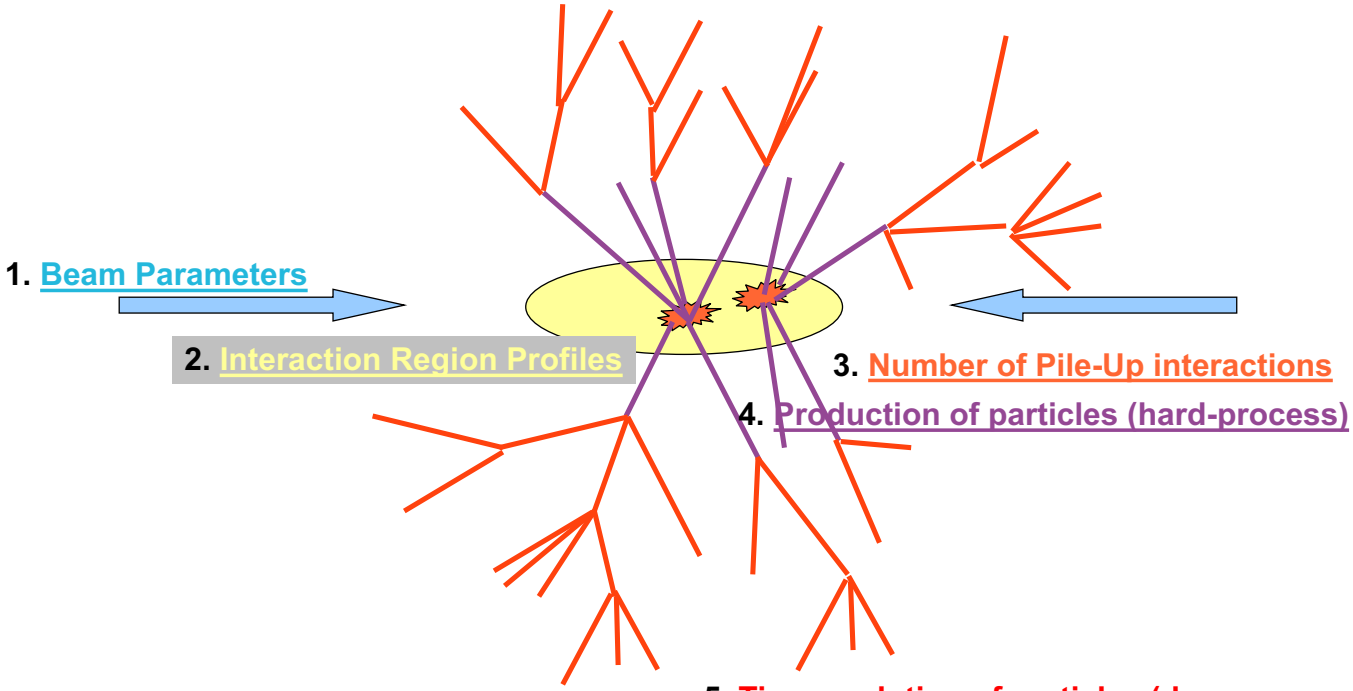


Special generators

- The production of hadrons with multiple heavy flavor (B_c, Ξ_{bc}, …) is very rare from the processes activated in LHCb for Minimum Bias: it requires creation of a second pair of high mass quarks which is unprobable.
- Dedicated generators exist which are interfaces with Pythia: these generators generate the hard process, Pythia takes care of the rest (hadronization, FSR, …)
- For example, BcVegPy for the simulation of B_c^+ [C-H Chang, C. Driouchi, P. Eerola, Z-G Wu, CPC159 (2004) 192]



Putting everything together



External Libraries

- The most important actions are performed by external libraries, developped outside LHCb: `PYTHIA`, `EvtGen`, ...
- Gauss is organizing the sequence of actions needed to generate events, calling these external libraries at the right moment, through interfaces.
- The interfaces to the external generators are generic: generators can be exchanged easily only via configurables, for example to use `SHERPA` or `HERWIG` instead of `PYTHIA`.

Minimum Bias (MB) Generation

- Most simple generation case is generation of minimum-bias (all what is produced by *pp* collisions) events.
- Sequence logic is:
  - 1. Generate *proton* beam momentum
  - 2. Determine number *N* of pile-up interactions
  - 3. Determine space positions of the interactions (PV)
  - 4. Generate *N pp* collisions
  - 5. Decay all produced particles

MB Generation Sequence

- In Gauss, this is implemented in one single `Gaudi` algorithm, **Generation**, which calls several `Gaudi` tools.

Generation

1. Determine number *N* of pile-up interactions
2. Produce *N pp* collisions:
    a. Determine beam momenta
    b. Generate *pp* collision
    c. Decay all particles produced
    d. Determine spatial position of primary vertex

(position of *pp* collision)

MB Generation Sequence

- Each step is realized by a `Gaudi` tool.
- Each tool has:
  - A generic abstract interface: names of generic functions that the tool has to provide
  - Concrete implementations: various ways of implementing the requested functionalities.
- The **Generation** algorithm and the abstract interfaces are defined in the package `Gen/Generators`.

MB Generation Sequence

Generation

1. Determine number *N* of pile-up interactions
2. Produce *N pp* collisions:
    a. Determine beam momenta
    b. Generate *pp* collision
    c. Decay all particles produced
    d. Determine spatial position of primary vertex (position of *pp* collision)

Abstract Interface

Pile-Up Tool: IPileUpTool

Sample Generation Tool: ISampleGenerationTool

Beam Tool: IBeamTool

Production Tool: IProductionTool

Decay Tool: IDecayTool

Vertex Smearing Tool: IVertexSmearingTool

MB Generation Sequence

Generation

1. Determine number *N* of pile-up interactions → **Pile-Up Tool:** IPileUpTool
2. Produce *N pp* collisions: → **Sample Generation Tool:** ISampleGenerationTool
   a. Determine beam momenta → **Beam Tool:** IBeamTool
   b. Generate *pp* collision → **Production Tool:** IProductionTool
   c. Decay all particles produced → **Decay Tool:** IDecayTool
   d. Determine spatial position of primary vertex (position of *pp* collision) → **Vertex Smearing Tool:** IVertexSmearingTool

Abstract Interface

Pile-Up Tools (1)

- Determines the number of interactions *N* per event.
- Available implementations:
  - FixedLuminosity (default), *N* follows a Poisson law of mean value ν, with $N \neq 0$.

$$\nu = \frac{\mathcal{L}\,\sigma_{tot}}{f}$$

  - NB: σ_{tot} is the total cross-section. For data taking, the relevant cross-section is $\sigma_{visible}$, the cross-section for having at least one track in the VELO. The mean value of the number of visible interactions is $\mu = 0.699\nu$.

```
# Ex: 2025 options
from Configurables import Gauss
# Set the instantaneous luminosity (L)
Gauss().Luminosity = 0.835*(10**30)/(SystemOfUnits.cm2*SystemOfUnits.s)
# Set the crossing frequency (f)
Gauss().CrossingRate = 11.245*SystemOfUnits.kilohertz
# Set the total cross-section (sigma_tot)
Gauss().TotalCrossSection = 102.5*SystemOfUnits.millibarn
```

Pile-Up Tools (2)

- FixedNInteractions: *N* is constant.

- FixedLuminosityForRareProcess: used for generation of rare events . (*N*-1) follows a Poisson distribution with mean value ν, where .

$$\nu = \frac{\mathcal{L}\sigma_{tot}}{f}$$

The parameters are set exactly the same way than for FixedLuminosity.

- For rare processes, the probability to find the process in an event with 2 interactions is 2x larger than in an event with 1 interaction, 3x in an event with 3 interactions, …

$$\mathcal{P}(N_{\text{rare}} = k) \propto k\frac{\nu^k e^{-\nu}}{k!} \propto \frac{\nu^{k-1} e^{-\nu}}{(k-1)!}$$

MB Generation Sequence

Generation

1. Determine number *N* of pile-up interactions → **Pile-Up Tool:** IPileUpTool
2. Produce *N pp* collisions: → **Sample Generation Tool:** ISampleGenerationTool
   a. Determine beam momenta → **Beam Tool:** IBeamTool
   b. Generate *pp* collision → **Production Tool:** IProductionTool
   c. Decay all particles produced → **Decay Tool:** IDecayTool
   d. Determine spatial position of primary vertex (position of *pp* collision) → **Vertex Smearing Tool:** IVertexSmearingTool

Abstract Interface

Sample Generation Tool

- Implements how the generated sample is obtained.
- In LHCb, 4 different types of event samples can be produced (see later):
  - **MinimumBias**: all events without any requirements
  - **Inclusive**: keep only minimum bias events with a given particle type (or a list of particle types)
  - **Signal** (can be **SignalPlain**, **SignalRepeatedHadronization** or **SignalForcedFragmentation**): keep only minimum bias events with a given particle type and force this particle to decay to a given decay mode
  - **Special**: use special generators or settings to produce a very rare process (not in minimum bias)

```
from Gauss.Configuration import *
# Change configuration of Generation algorithm
myGen = Generation("Generation")
# Use MinimumBias generation method
myGen.SampleGenerationTool = "MinimumBias"
```

MB Generation Sequence

Generation

1. Determine number N of pile-up interactions
2. Produce N pp collisions:
   a. Determine beam momenta
   b. Generate pp collision
   c. Decay all particles produced
   d. Determine spatial position of primary vertex (position of pp collision)

Abstract Interface

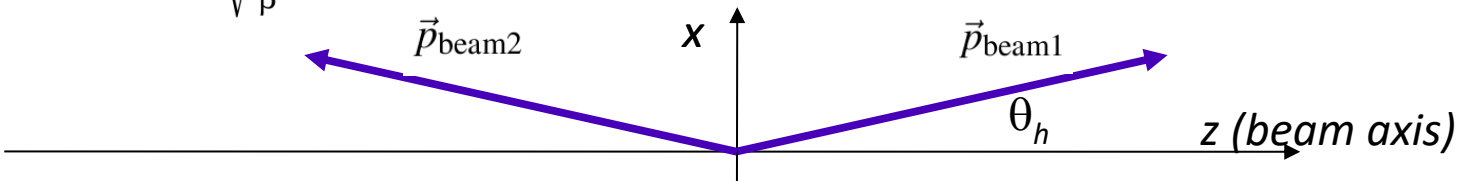
Pile-Up Tool:	IPileUpTool
Sample Generation Tool:	ISampleGenerationTool
Beam Tool:	IBeamTool
Production Tool:	IProductionTool
Decay Tool:	IDecayTool
Vertex Smearing Tool:	IVertexSmearingTool

Beam Tool

- Generates the beam particle (*p*) momenta for each collision.
- Available implementations:
  - **CollidingBeams**: generates 2 colliding beams with a crossing angle which is smeared by a Gaussian distribution.

$$\vec{p}_{\text{beam1}} = \begin{pmatrix} p \sin\theta_h \\ p \sin\theta_v \\ p \end{pmatrix} \qquad \vec{p}_{\text{beam2}} = \begin{pmatrix} p \sin\theta_h \\ p \sin\theta_v \\ -p \end{pmatrix}$$

  - where θ_h and θ_v follow Gaussian distributions with a defined mean value and $\sigma = \sqrt{\frac{\varepsilon}{\beta^*}}$



Top view

Beam Tool

- **FixedTarget**: generates one single beam:

$$\vec{p}_{\text{beam1}} = \begin{pmatrix} p \sin\theta_h \\ p \sin\theta_v \\ p \end{pmatrix} \qquad \vec{p}_{\text{beam2}} = \vec{0}$$

```
# Values for 2025
from Gauss.Configuration import *
# Set the beam momentum (p)
Gauss().BeamMomentum = 6.8*SystemOfUnits.TeV
Gauss().B2Momentum = -6.8*SystemOfUnits.TeV
# Set the half crossing angle (vertical and horizontal planes)
Gauss().BeamHCrossingAngle = +0.139*SystemOfUnits.mrad  # internal
Gauss().BeamVCrossingAngle = +0.200*SystemOfUnits.mrad  # external
# Set the emittance (epsilon)
Gauss().BeamEmittance = 0.0038*SystemOfUnits.mm
# Set the beta*
Gauss().BeamBetaStar = 2.0*SystemOfUnits.m
```

MB Generation Sequence

Generation

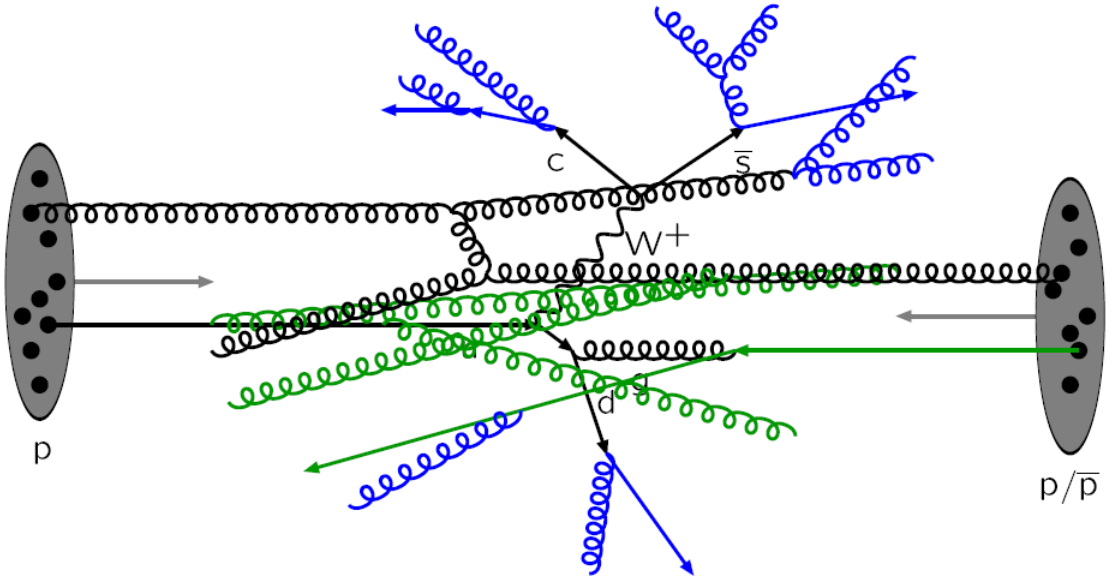
1. Determine number *N* of pile-up interactions
2. Produce *N pp* collisions:
   a. Determine beam momenta
   b. Generate *pp* collision
   c. Decay all particles produced
   d. Determine spatial position of primary vertex (position of *pp* collision)

Abstract Interface

Pile-Up Tool:	IPileUpTool
Sample Generation Tool:	ISampleGenerationTool
Beam Tool:	IBeamTool
Production Tool:	IProductionTool
Decay Tool:	IDecayTool
Vertex Smearing Tool:	IVertexSmearingTool

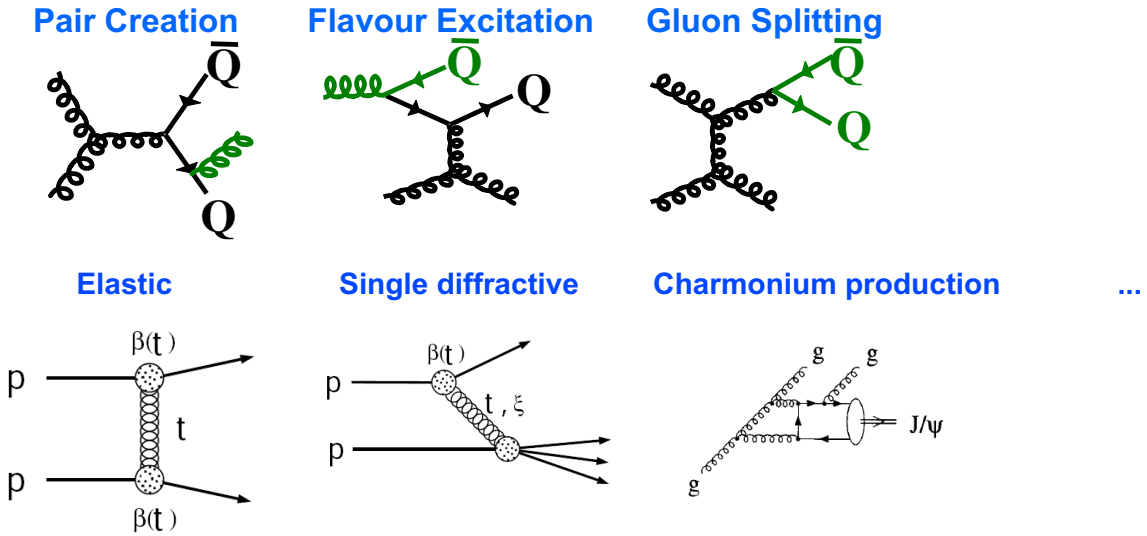
Production Tool

- It is used to generate the *pp* collisions (hard process, hadronization, ...)



Production Tool (Pythia 8)

- Default options constitute « LHCb tuning », which is done to extrapolate at higher energies charged track multiplicities seen at the UA1 experiment.
- The activated physics processes are the dominant ones for LHC energies. They define LHCb « minimum bias » out of which all major samples are generated.



MB Generation Sequence

Generation

1. Determine number *N* of pile-up interactions
2. Produce *N pp* collisions:
   - a. Determine beam momenta
   - b. Generate *pp* collision
   - c. Decay all particles produced
   - d. Determine spatial position of primary vertex (position of *pp* collision)

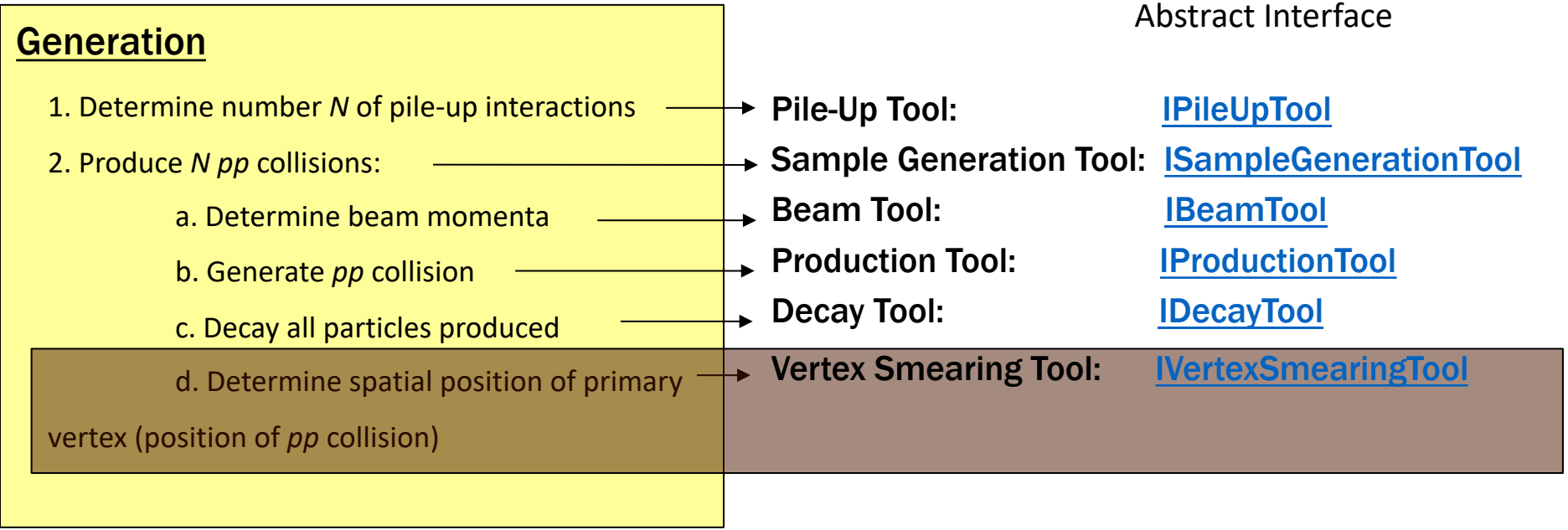
Abstract Interface

Pile-Up Tool:	IPileUpTool
Sample Generation Tool:	ISampleGenerationTool
Beam Tool:	IBeamTool
Production Tool:	IProductionTool
Decay Tool:	IDecayTool
Vertex Smearing Tool:	IVertexSmearingTool

Decay Tool

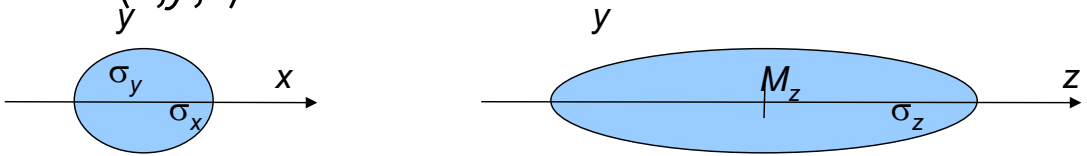
- It is used to decay all particles, and to generate correct time dependance (CP violation, mixing), correct angular correlations, etc...
- Default implementation: EvtGen

MB Generation Sequence



Vertex Smearing Tool

- Generates profiles of luminous region (position and size).
- Available implementations:
  - **BeamSpotSmearVertex**: The primary vertex position follows Gaussian distributions in (*x*,*y*,*z*) and has a fixed time *t*.




```
# 2025 settings
from Configurable import Gauss
# Set Interaction Size
Gauss().BunchRMS= 60.67*SystemOfUnits.mm
# Set Interaction Center
Gauss().InteractionPosition= [ 0.480*SystemOfUnits.mm, 0.070*SystemOfUnits.mm,
                               0.500*SystemOfUnits.cm ]
```

$$\sigma_z = \mathrm{BunchRMS}/\sqrt{2}$$

$$\sigma_x = \sigma_y = \sqrt{\epsilon\beta^*}$$

Vertex Smearing Tool

- **FlatZSmearVertex**: The primary vertex position follows Gaussian distributions in (*x*,*y*), a flat distribution in *z* and has a fixed time *t*.

```
from Gauss.Configuration import *
# Set Beam Size
Gauss().BeamSize = [ 0.045*SystemOfUnits.mm, 0.045*SystemOfUnits.mm ]
Generation("Generation").VertexSmearingTool = "FlatZSmearVertex"
```

Inclusive Generation Sequence

- Important samples are inclusive samples: $b\bar{b}$ or $c\bar{c}$ inclusive samples.
- They are obtained from minimum bias generation, but requiring that each event contains at least one particle of a given type (*B* hadron, *D* hadron, ...)
- To obtain more interesting samples, a cut is also performed at generator level to keep only useful events.
- The generation method is changed:

```
from Gauss.Configuration import *
# Change generation method
myGen = Generation("Generation")
myGen.SampleGenerationTool = "Inclusive"
# Gives list of PID of particles to produce (b/bbar)
myGen.Inclusive.InclusivePIDList = [521, -521, 511, -511, 531,
  -531, 541, -541, 5122, -5122, 5222, -5222, 5212, -5212, 5112,
  -5112, 5312, -5312, 5322, -5322, 5332, -5332, 5132, -5132,
  5232, -5232]
```

Inclusive Generation Sequence

Generation

1. Determine number *N* of pile-up interactions → **Pile-Up Tool:** IPileUpTool
2. Produce *N pp* collisions: → **Sample Generation Tool:** ISampleGenerationTool
    a. Determine beam momenta → **Beam Tool:** IBeamTool
    b. Generate *pp* collision → **Production Tool:** IProductionTool
    c. Apply cut at generator level → **Cut Tool:** IGenCutTool
    d. Decay all particles produced → **Decay Tool:** IDecayTool
    e. Determine spatial position of primary vertex (position of *pp* collision) → **Vertex Smearing Tool:** IVertexSmearingTool

Abstract Interface

Cut tool (1)

- Accept or reject an event based on generator level quantities.
- Available implementations:
  - **LHCbAcceptance**: cut on signal direction: $0 \leq \theta_{signal} \leq 400$ mrad

```
from Gauss.Configuration import *
from Configurables import Inclusive
# Set cut tool
myGen = Generation("Generation")
myGen.SampleGenerationTool = "Inclusive"
myGen.addTool( Inclusive() )
myGen.Inclusive.CutTool = "LHCbAcceptance"
```

  - **DaughtersInLHCb**: cut on direction of decay products of signal particle:
    - 10 mrad $\leq \theta_{charged} \leq$ 400 mrad, 5 mrad $\leq \theta_{neutral} \leq$ 400 mrad
    - No cut on Λ and K_s^0 daughters, and on neutrinos.
    - Only cut on γ if they come from π^0 or η.

Cut tool (2)

- Variations of **DaughtersInLHCb**:
  - **DaughtersInLHCbAndFromB**: signal particle is coming from a b-hadron decay,
  - **ListOfDaughtersInLHCb**: only particles of given types are required to be in the acceptance of LHCb,
  - **SelectedDaughtersInLHCb**: only particles coming from the decay of given particles are required to be in the acceptance of LHCb,
  - **BcDaughtersInLHCb**, **UpsilonDaughtersInLHCb**: needed for signal obtained through special generation methods.

Signal Generation

- To generate signal sample, an extra step is added to the generation of inclusive: the presence of a given particle (B^0, B^+, ...) is required, and its decay is forced to a signal decay mode.
- To speed up generation process for signal B, **SignalRepeatedHadronization** method exist: when a b quark is found, the event is re-hadronized until the B of interest is found. (For example, b hadronizes to B_s^0 with 10% probability). However it can bias the momentum distribution of the signal hadrons, so in general the use xof the **SignalPlain** method is preferred.

```
from Gauss.Configuration import *
from Configurables import SignalPlain
# Change generation method
myGen = Generation("Generation")
myGen.addTool( SignalPlain() )
myGen.SampleGenerationTool = "SignalPlain"
# Gives list of PID of particles to Bs0/Bs0bar
myGen.SignalPlain.SignalPIDList = [ 531, -531]
```

Signal Generation

- The decay of the signal is forced to a given decay mode (if 2 are present in the same event, only one is forced).
- To do this, EvtGen aliases are defined. They are copies of particles (have the same properties) but their decay mode can be redefined (in a EvtGen user decay file) without affecting the decay mode of the « normal » particle.
- Aliases names are `<Name of the particle>sig:`
  - `B0sig, anti-B0sig`
  - `B+sig, B-sig`
  - `D_s0sig, anti-D_s0sig`
  - `...`

Signal Generation

- *B* are always produced in pairs
- For example for a B^+ signal decay:
  - since the signal decays have usually very small branching fractions, the probability to find a signal decay in an event with $B^+ + B^-$ is almost twice to find it in an event with $B^+ + B_x$*(not* B^-*)* or $B^- + B_x$*(not* B^+*)*, more exactly it is

$$\frac{P(B^+ + B^-)}{P(B^+ + B_x \text{ or } B^- + B_x)} = 2 - Br(\text{signal})$$

  - So events with only one B^+ or only one B^- have to be rejected randomly with this probability to generate the correct proportion of B^+/B_x in the signal sample

Redecay

- Full simulation of signal events can take a long computing time
- But only a small fraction of the entire event is related to the signal (~5 particles out of ~100 particles coming from the ***rest of the event*** or pile-up interactions)
- Algorithm to save CPU time is to re-use the rest of the event several times, only changing the signal decay.
- Sequence:
    1. Generate full Monte Carlo event, including signal decay
    2. Remove signal particle and its decay products from the full event
    3. Simulate fully through detector/Geant4 the rest of the event
    4. Generate a signal particle and its decay, using the same momentum and production vertex than the original one
    5. Simulate it through detector/Geant4
    6. Merge the signal simulation with the rest of the event
    7. Repeat 4. to 6. N times (typically 100 times)
- This allows us to speed up the simulation time for signal decays by a factor 10 to 20. But this introduces also correlations between events, since they are generated using the same momentum and position