



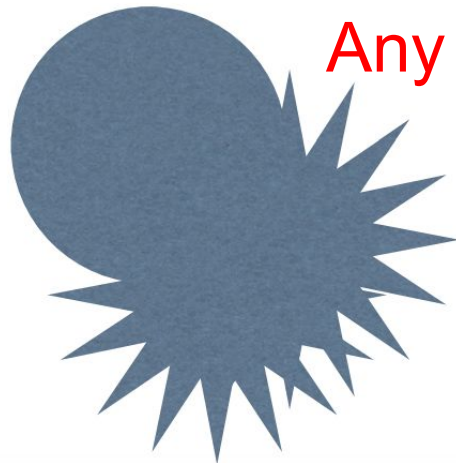
Monte Carlo Method

and Pythia8 tutorial

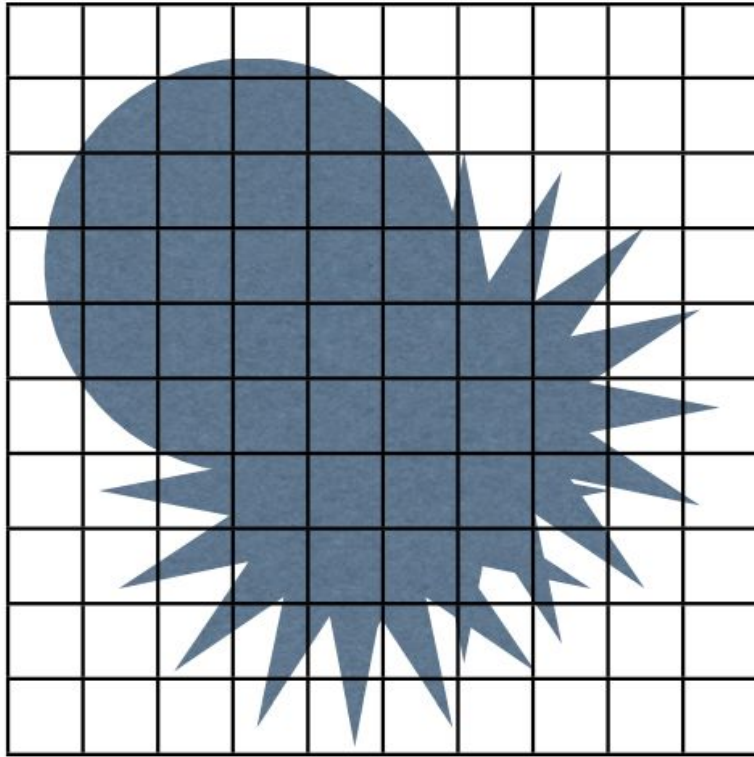
Monte Carlo Method

Monte Carlo techniques are often the only practical way to evaluate difficult integrals or to sample random variables governed by complicated probability density functions.

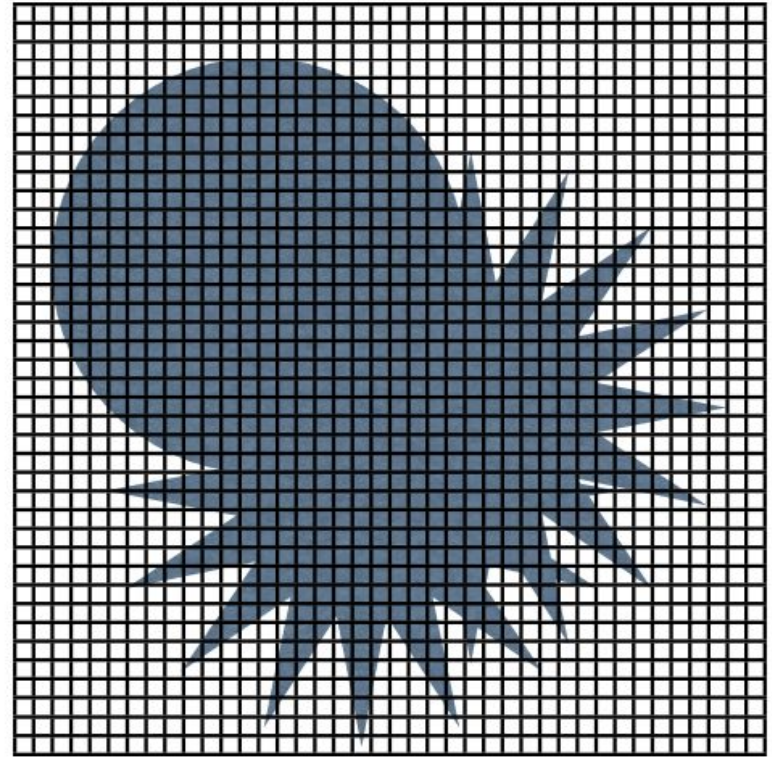
- General idea is: instead of performing long complex calculations, perform large number of experiments using random number generation and see what happens
- Problem: calculate the area of this shape



Any idea?



10x10



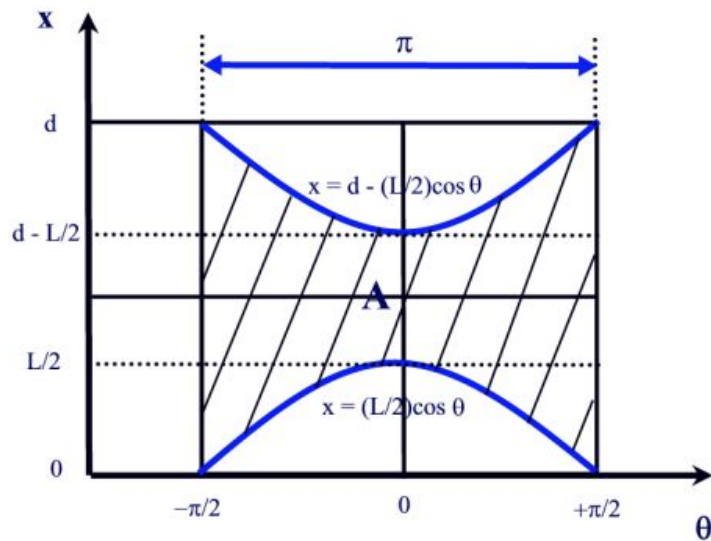
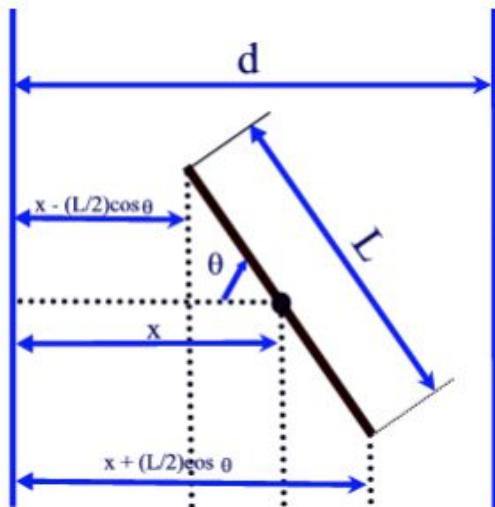
40x40

Area = (number of hits) / (number of total) x total area

Buffon's needle problem

An interesting related problem is Buffon's needle problem (1733), that can be used to design a Monte Carlo method for approximating the number π :

- Given a needle of length l dropped on a plane ruled with parallel lines d units apart, what is the probability that the needle will lie across a line upon landing?



Buffon's needle problem

Let x be the distance from the center of the needle to the closest parallel line, and θ the acute angle between the needle and one of the parallel lines. (x and θ are independent variables)

The uniform PDF of x $\begin{cases} \frac{2}{l} & : 0 \leq x \leq \frac{l}{2} \\ 0 & : elsewhere \end{cases}$ and the uniform PDF of θ $\begin{cases} \frac{2}{\pi} & : 0 \leq \theta \leq \frac{\pi}{2} \\ 0 & : elsewhere \end{cases}$

The joint PDF is: $\begin{cases} \frac{4}{l\pi} & : 0 \leq \theta \leq \frac{\pi}{2}, 0 \leq x \leq \frac{l}{2} \sin \theta \\ 0 & : elsewhere \end{cases}$ and if $x \leq \frac{l}{2} \sin \theta$ the needle

crosses one line. The probability is:
$$P = \int_{\theta=0}^{\frac{\pi}{2}} \int_{x=0}^{\frac{l}{2} \sin \theta} \frac{4}{\pi l} dx d\theta = \frac{2l}{\pi d}$$

Suppose we drop n needles and find that m of those are crossing lines, so P is approximated by the fraction M/n , then:

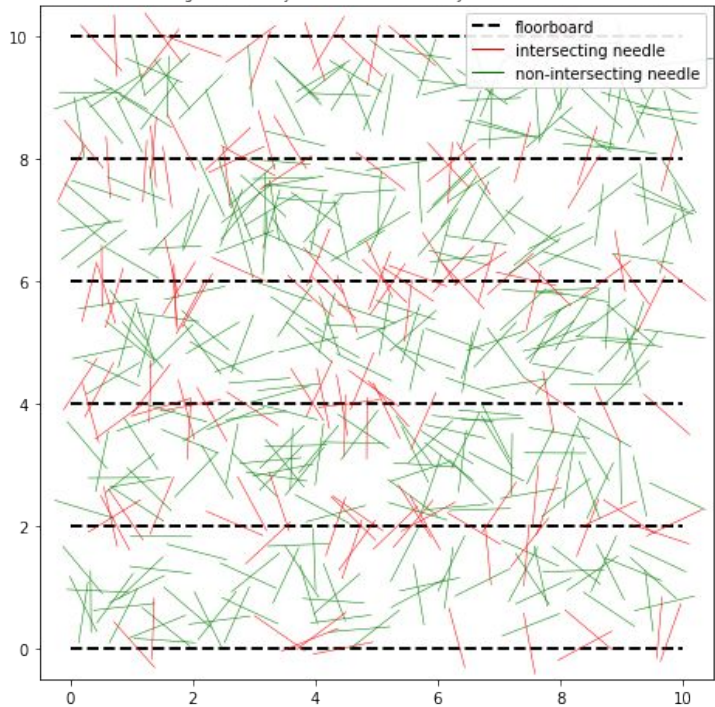
$$\pi = \frac{2ln}{dm}$$

Buffon's needles problem

https://github.com/ssilvado/MC_Buffon

Simulation of Buffon's Needle Problem
as a Monte Carlo method for approximating Pi

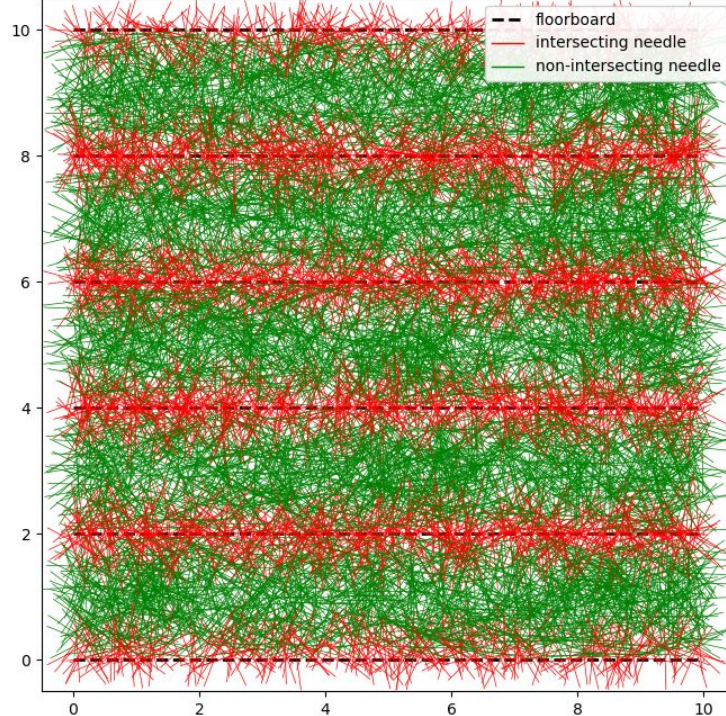
10000 needles of length 1 uniformly distributed over a 10 by 10 area with floorboards of width 2



Intersections: 149
Total Needles: 500
Approximation of pi: 3.3557046979865772
Error: 6.815398048251908%

Simulation of Buffon's Needle Problem
as a Monte Carlo method for approximating Pi

10000 needles of length 1 uniformly distributed over a 10 by 10 area with floorboards of width 2



Intersections: 3168
Total Needles: 10000
Approximation of pi: 3.1565656565656566
Error: 0.47660548730766567%

Monte Carlo in HEP

Simulation of a Collision Event

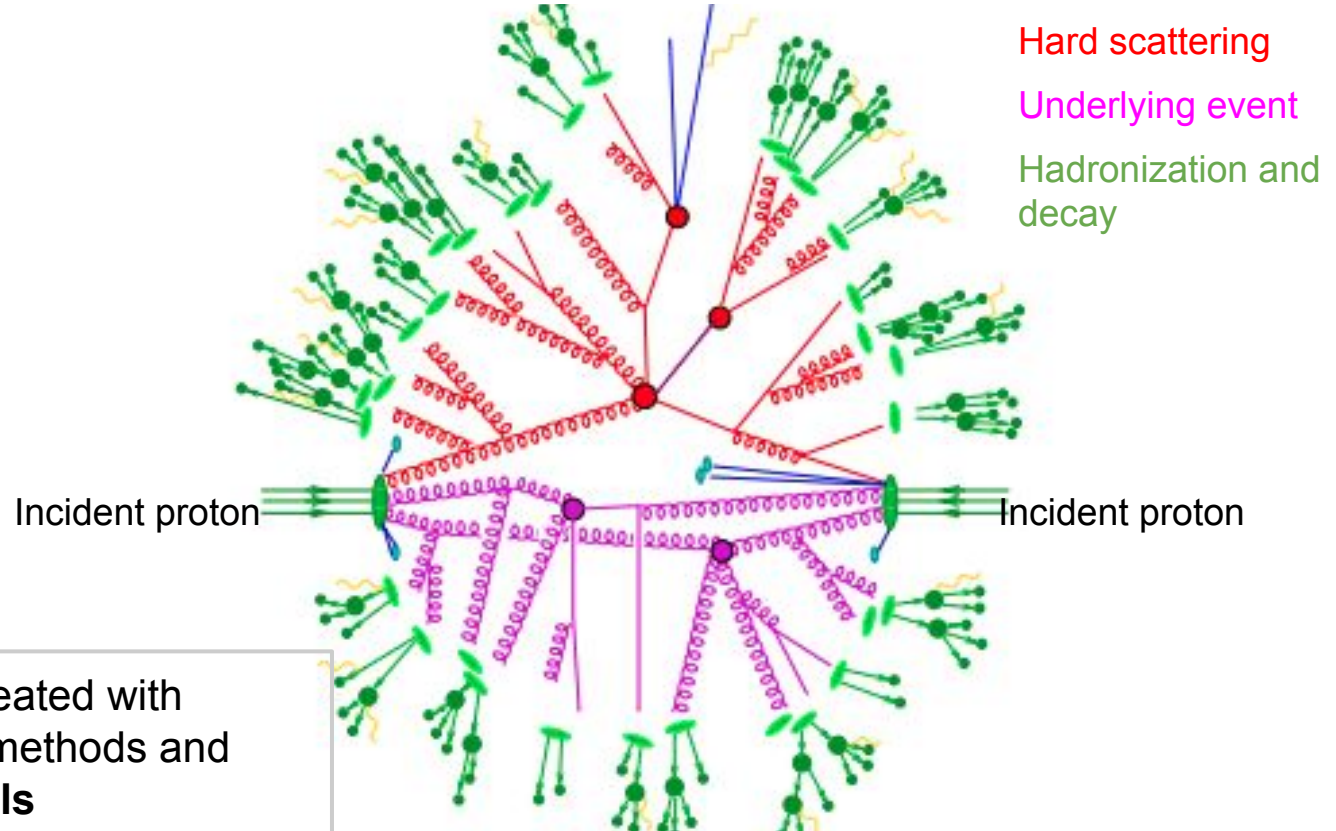
JHEP 0402 (2004) 056

Calculation:

→ **partonic**
subprocess

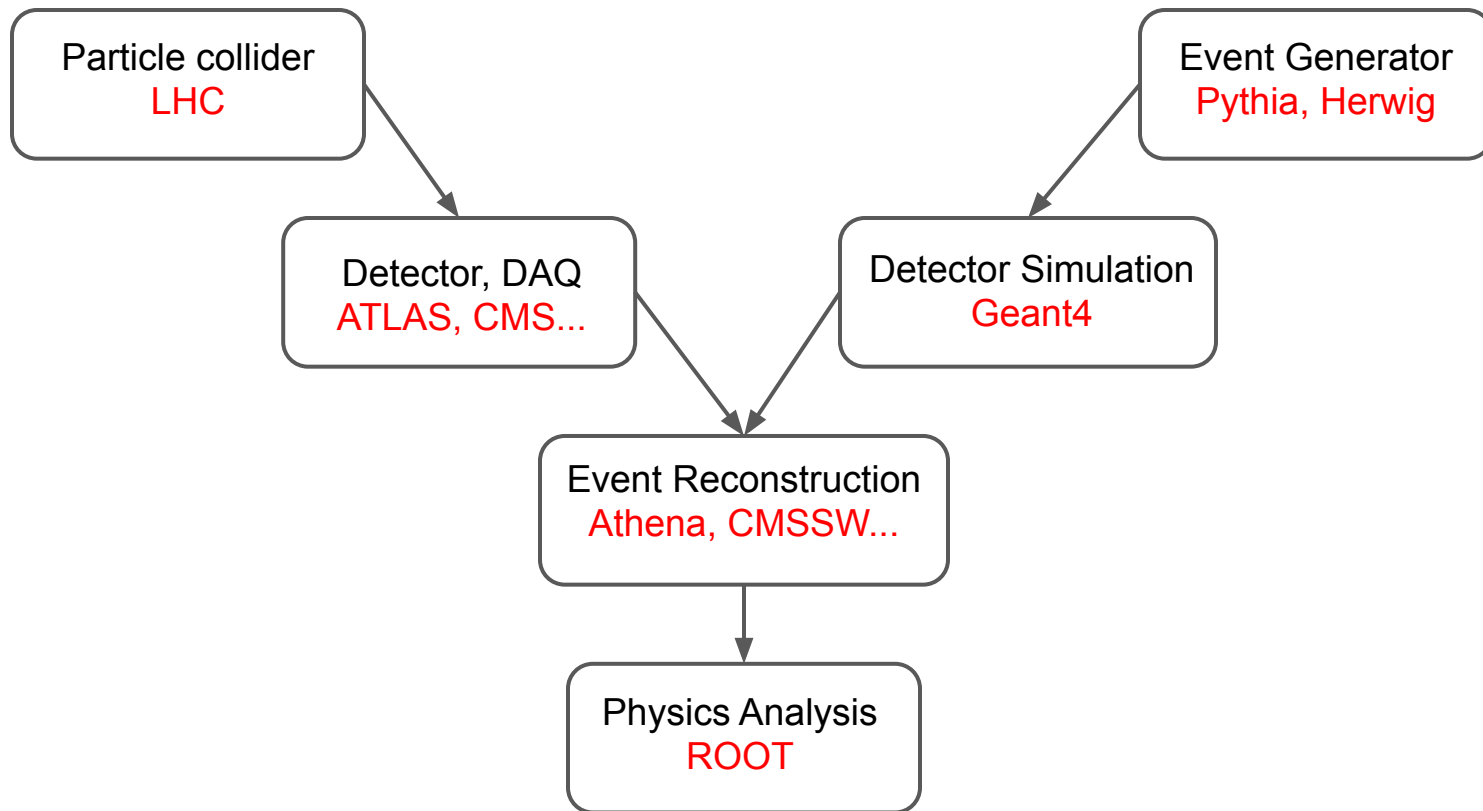
Measurement:

→ **hadrons**



Complicated process → treated with
Monte-Carlo simulation methods and
phenomenological models

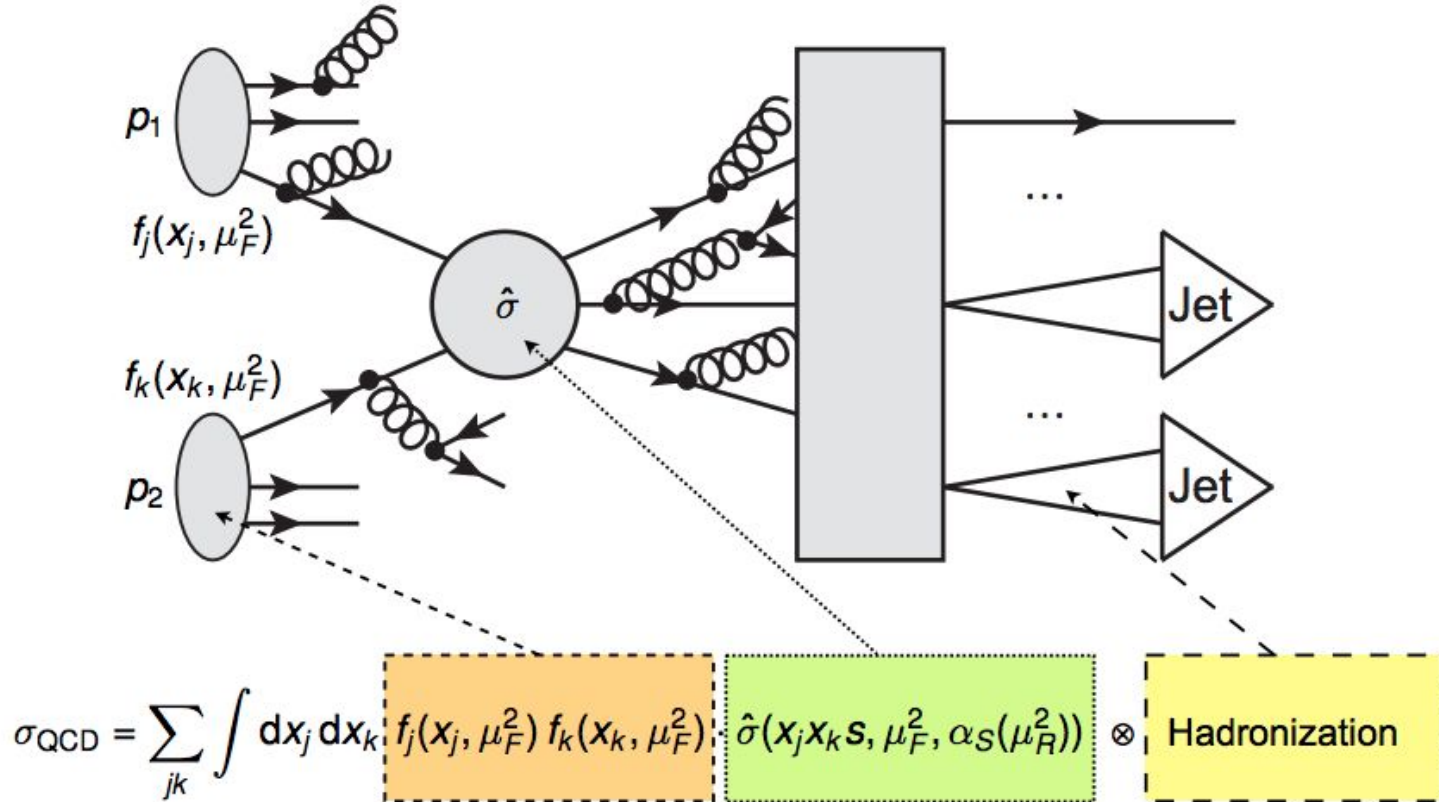
Monte Carlo in Particle Physics



Monte Carlo Event Generators

- Goal: **realistic simulation** of all relevant physics processes in a particle collision
- Problem: **complexity** of hadron-hadron collisions
 - Initial state: hadrons = compound objects, constituents (quarks and gluons) confined in hadron (running of α_s)
 - Final state: many hadrons and leptons
- Solution: **QCD factorisation**
 - **Separate treatment** of processes at low and high Q^2 .
 - High Q^2 (“hard scattering process”): **perturbation theory** in leading order or higher orders
 - Low Q^2 (“soft physics”): phenomenological **models**

QCD Factorisation Theorem



Cross section = PDFs \otimes hard process \otimes hadronization

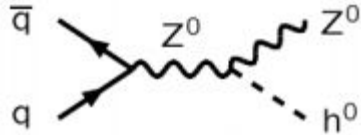
Overview of MC Generators

- Central step in any MC generator: **MC integration** of cross section of hard scattering process in fixed order perturbation theory using PDFs
- Parton-level MC generators
 - Simulation stops at level of partons (quarks and gluons)
 - No hadronisation, only events weighted with differential cross-section → no full event simulation (still useful for theoretical studies)
- Particle-level MC generators
 - Full event simulation: parton level + parton shower + hadronisation (number of MC events corresponds to theoretical expectation)
 - Provided as single comprehensive package or as combination of ME provider and parton shower MC (SMC) programme

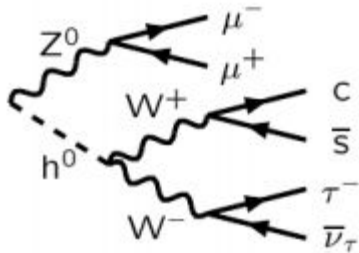
Monte Carlo Generation

Matrix Elements (ME):

1. Hard subprocess: $|M^2|$, Breit-Wigners, parton densities.

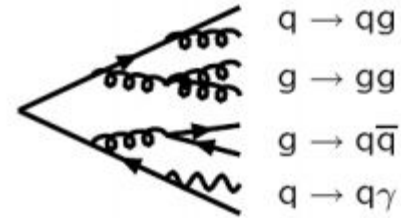


2. Resonance decays: includes correlations

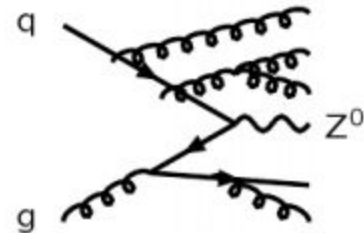


Parton Showers (PS):

3. Final-state parton showers

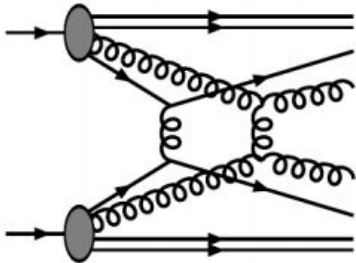


4. Initial-state parton showers

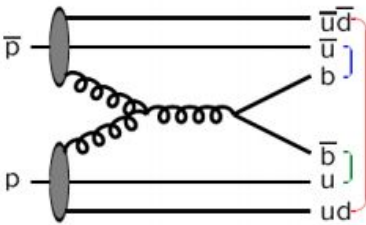


Monte Carlo Generation

5. Multiple parton-parton interactions

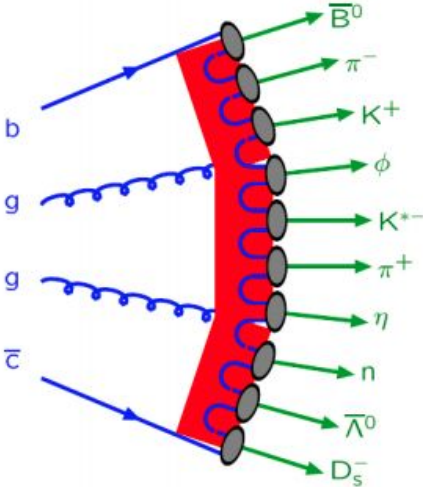


6. Beam remnants with colour connections

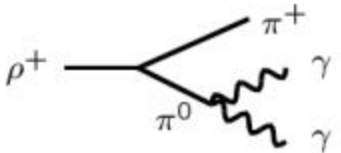


5+6 = Underlying Event

7. Hadronization



8. Ordinary decays: hadronic, τ , charm,...



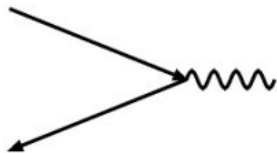
Matrix Element Calculation

Normally calculated at LO or NLO

I. Lowest order,

$\mathcal{O}(\alpha_{em})$:

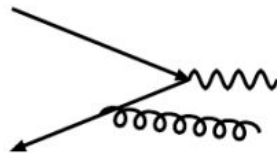
$q\bar{q} \rightarrow Z^0$



II. First-order real,

$\mathcal{O}(\alpha_{em}\alpha_s)$:

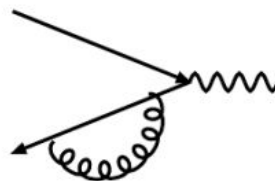
$q\bar{q} \rightarrow Z^0 g$ etc.



III. First-order virtual,

$\mathcal{O}(\alpha_{em}\alpha_s)$:

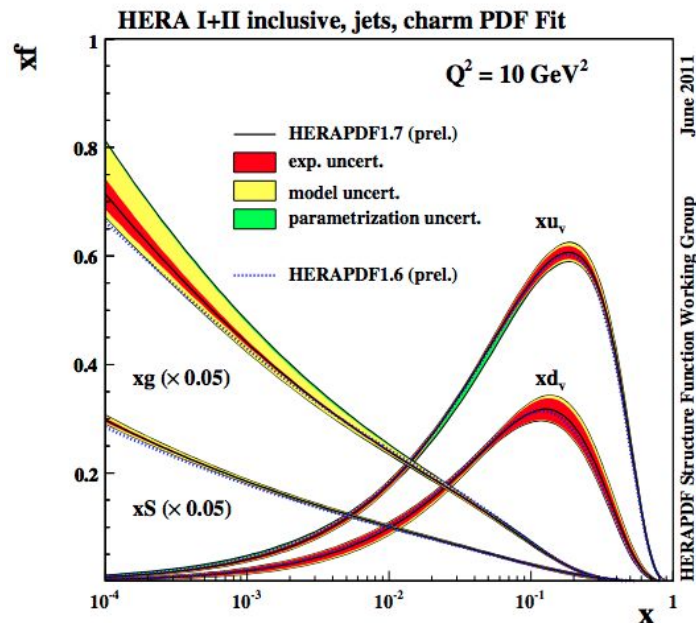
$q\bar{q} \rightarrow Z^0$ with loops



- Higher order corrections are important
 - Normalisation and shape of kinematic distributions
 - Multiplicity of objects like jets
 - Higher order correction are hard to calculate and CPU intensive
 - Several programs that will do the calculation
 - Different calculation techniques
 - Different assumptions
 - Different results
- ⇒ Theoretical modelling uncertainty

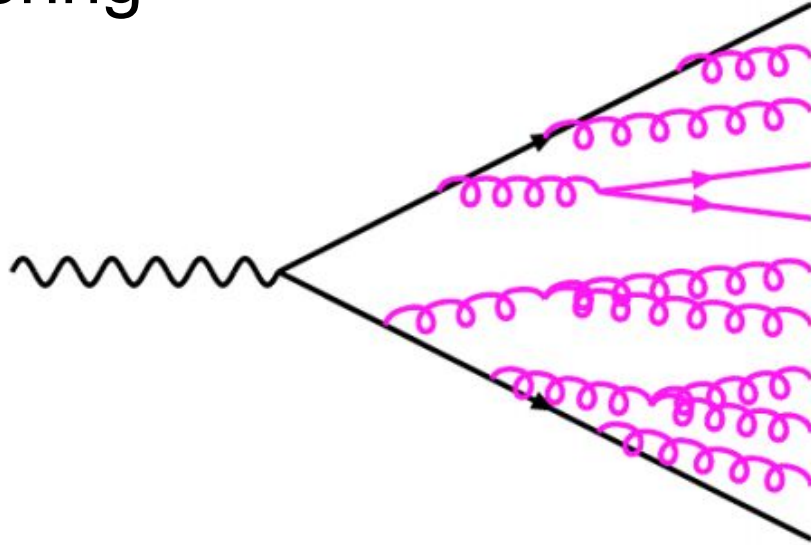
The Parton Density Function (PDF)

- Proton is not a point-like particle, it's full of partons
- Need to calculate:
 - Probability of propagator interacting with quarks/gluon
 - Needed as a function of Q^2 and x
- Various groups provide PDFs
- Parametrised differently
- LHC uses:
 - CTEQ
 - MSTW
 - NNPDF
- Get a different result using different PDFs



⇒ Theoretical modelling uncertainty

Parton Showering



- Need to go from $2 \rightarrow 2$ scattering to 100's of particles
 - A particle can decay into more particles
 - A particle can emit another particle
 - All controlled by random numbers
- Parton shower evolution is a probabilistic process
 - Occurs with unit total probability

Parton Showering

2 common approaches to parton showering

- Need to avoid divergences and infinities in calculations
 - See your QCD course for why these occur
 - Solution requires the final state partons to be ordered
- There are 2 common approaches to do this
- Pythia: $Q^2 = m^2$
 - The parton with the highest p_T is calculated first
- Herwig: $Q^2 \approx E^2(1-\cos\theta)$
 - The parton with the largest angle is calculated first

⇒ This represents a theoretical modelling uncertainty

- Both provide a good description of data but which is correct?
 - Neither is correct, but nature is unknown, we only have models
- All physics measurements need to take this into account
 - Expect to see a parton shower systematic for every result
 - Use both methods for calculation of physics result
 - Difference between results is a theoretical modelling systematic

Hadronisation

Going from partons to hadrons

- Partons are not observed directly in nature, only hadrons
- Hadronisation occurs at low energy scales
 - Perturbation theory is not valid
 - Cannot calculate this process from first principals
- Require models to simulate what happens
- 2 common approaches are used
 - `Pythia`: Lund string model
 - `Herwig`: Cluster model

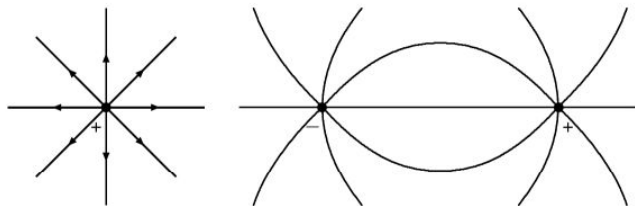
⇒ This is another theoretical modelling uncertainty

- Similar type of uncertainty as for parton showering
 - We don't know exactly how nature works
 - We have 2 reasonable models
 - Calculate physics result using each method
 - Difference is a theoretical modelling systematic

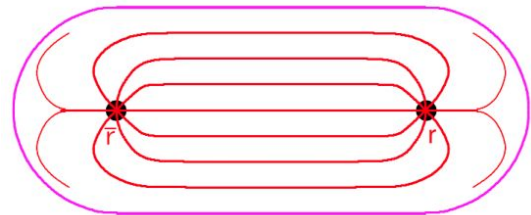
Hadronisation

The Lund string model

- In WED, field lines go all the way to infinity
- Photons do not interact with each other



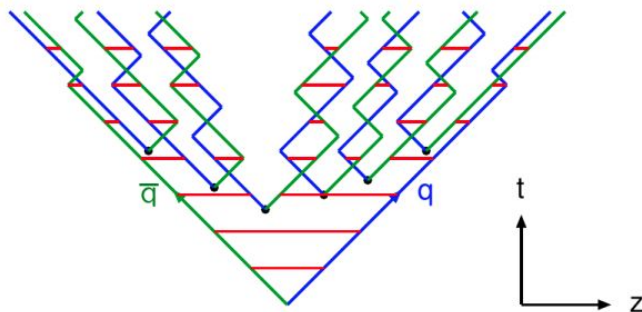
- In QCD, for large charge separation, field lines seem to be compressed into tube-like regions \Rightarrow string(s)
- Self-interaction among soft gluons in the vacuum



Hadronisation

The Lund string model

- The strings connecting the 2 partons breaks as they move apart
- Fragmentation starts in the middle and spreads out

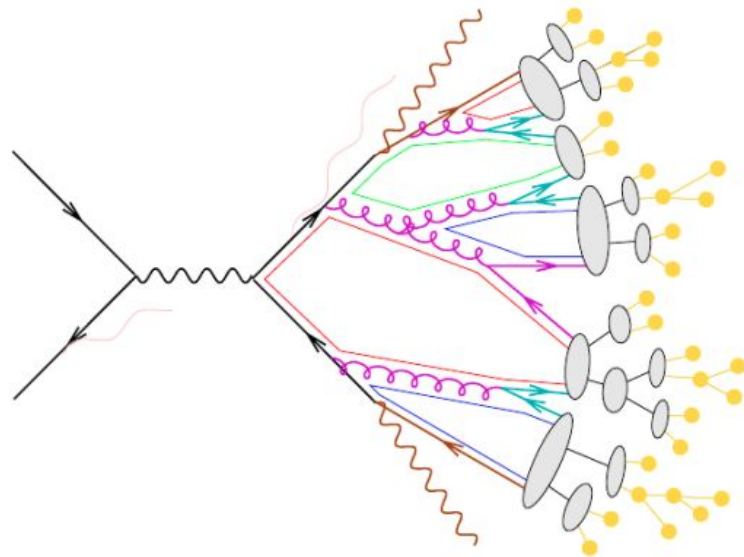


- The breakup vertices become causally disconnected
- This is governed by many internal parameters
- Implemented by the `Pythia` MC program

Hadronisation

The Cluster model

- Pre-confinement colour flow is local
- Forced g $q\bar{q}$ branchings
- Colour singlet clusters are formed
- Clusters decay isotropically to hadrons
- Relatively few internal parameters
- Implemented by the `Herwig` MC program



Pythia8 tutorial

Getting Started

- Download and untar the Pythia 8 source

```
wget http://home.thep.lu.se/~torbjorn/pythia8/pythia8244.tgz  
tar -xzvf pythia8244.tgz  
cd pythia8244.tgz
```

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

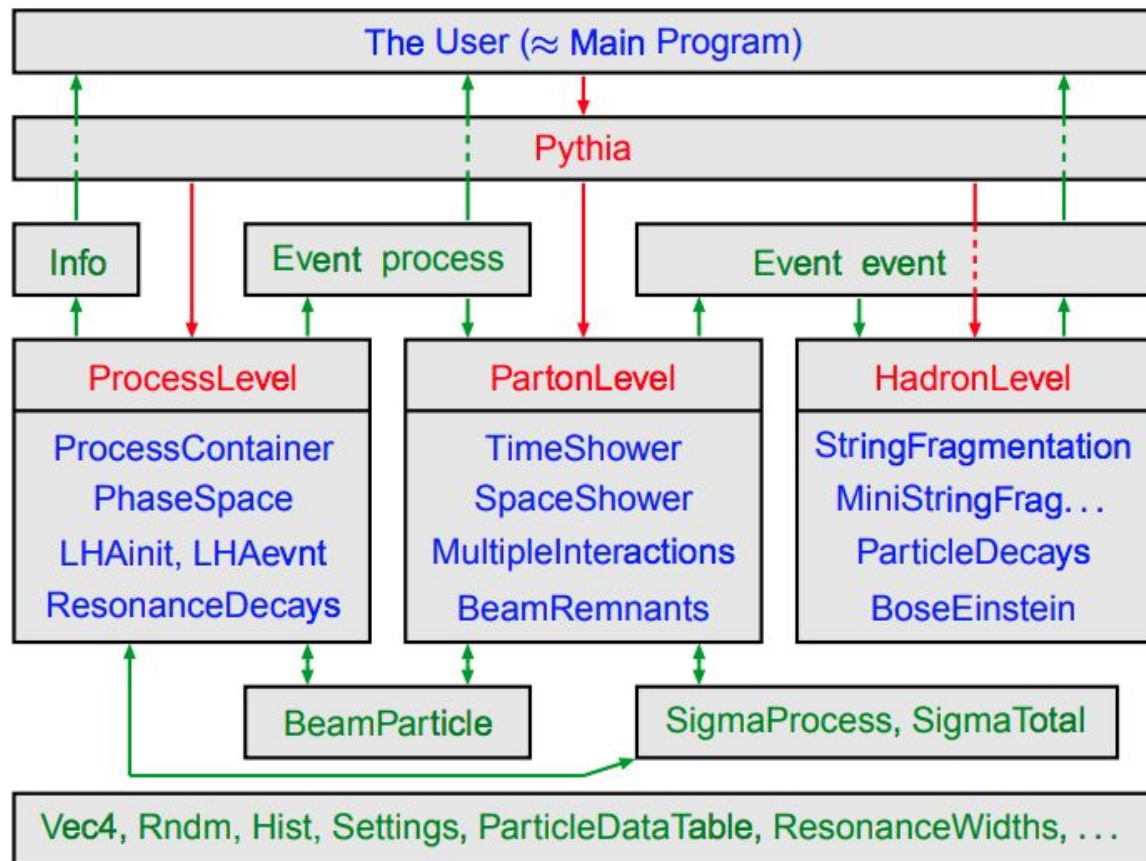
```
./configure --with-root=root-installation-directory  
make
```

- Build and run an example

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

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

Pythia8 structure



Pythia8

- General purpose Monte Carlo event generator
- Combine pQCD and models to provide link from theory (quarks, gluons) to experiment (mesons, baryons)
- Full problem “factorised” into different components
 - Hard process
 - Resonance decays
 - Parton showers
 - Underlying event
 - Hadronisation
 - Hadron decays
- Different parts may be handled by other external programs (e.g. Tauola)
- Outputs exclusive hadronic events
 - Analyse (e.g. FastJet)
 - Pass to detector simulator (e.g. GEANT)
 - ...

Physics Overview - Beams and hard processes

- Beams

- Incoming beams: hadron pair, a photon pair, a lepton and a hadron or a photon and a hadron
- Built-in PDFs sets:
 - GRV 94L, CTEQ5L
 - MSTW2008 (LO and NLO), MRST LO
 - CTEQ6L, CTEQ6L1, CTEQ6.6, CT09MC1, CT09MC2, CT09MCS
- Easy to link to LHAPDF for many more

- Hard processes

- Built-in library of many leading-order processes
- SM: almost all $2 \rightarrow 1$ and $2 \rightarrow 2$, some $2 \rightarrow 3$
- BSM: a bit of everything
- External input through Les Houches Accord (LHA) and Les Houches Event Files (LHEF)

Example of a main program: examples/main01.cc

```
// File: main01.cc. The charged multiplicity distribution at the LHC.
#include "Pythia.h"
using namespace Pythia8;
int main() {
    // Generator. Process selection. LHC initialization. Histogram.
    Pythia pythia;
    pythia.readString("HardQCD:all = on");
    pythia.readString("PhaseSpace:pTHatMin = 20.");
    pythia.init( 2212, 2212, 14000.);
    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;
        if (iEvent < 1) {pythia.info.list(); pythia.event.list();}
        // 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.statistics();
    cout << mult;
    return 0;
}
```

Reading header files = "tools you need"

Starting calculation

Make "object" of class called Pythia

Reading parameters to set up. You can also read it from file:
pythia.readFile(fileName);

Initialization

Event generation

Pythia member function

readString

next(): generate next event

event[i]: array of particles of an events

size(): size of the events

isFinal(): final state or not

isCharged()

stat(): print out statistics

Program Structure

- Proper header file must be included:

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

- Create a generator object: `Pythia pythia;`

- Pythia's settings and particle data

```
pythia.readString(string); // for changing a single variable  
pythia.readFile(fileName); // for changing a set of variables, one per  
line in the input file
```

- Initialize all aspects of the subsequent generation: `pythia.init();`
- Generate the next event: `pythia.next();`
- Run statistics: `pythia.stat();`

Compile and Execute: examples/main01.C

```
cd examples
make main01
./main01 >& test_main01
less test_main01
```

```
*----- PYTHIA Process Initialization -----*
|
| We collide p+ with p+ at a CM energy of 8.000e+03 GeV
|
|-----|
| Subprocess                               Code | Estimated
|                                           | max (mb)
|-----|
| g g -> g g                               111 | 1.403e+00
| g g -> q qbar (uds)                      112 | 1.817e-02
| q g -> q g                               113 | 1.010e+00
| q q(bar)' -> q q(bar)'                   114 | 1.100e-01
| q qbar -> g g                            115 | 8.378e-04
| q qbar -> q' qbar' (uds)                 116 | 3.698e-04
| g g -> c cbar                            121 | 5.988e-03
| q qbar -> c cbar                         122 | 1.225e-04
| g g -> b bbar                            123 | 5.400e-03
| q qbar -> b bbar                         124 | 1.160e-04
|-----|
*----- End PYTHIA Process Initialization -----*

*----- PYTHIA Multiparton Interactions Initialization -----*
|
|          sigmaNonDiffractive =    51.78 mb
|
| pT0 = 2.35 gives sigmaInteraction = 257.10 mb: accepted
|
*----- End PYTHIA Multiparton Interactions Initialization -----*

*----- PYTHIA Flag + Mode + Parm + Word + FVec + MVec + PVec + WVec Settings (changes only) -----*
|
| Name                                     | Now | Default | Min | Max |
|-----|-----|-----|-----|-----|
| Beams:eCM                               | 8000.000 | 14000.000 | 10.00000 |
| HardQCD:all                             | on | off |
| PhaseSpace:pThatMin                     | 20.00000 | 0.0 | 0.0 |
|-----|-----|-----|-----|
*----- End PYTHIA Flag + Mode + Parm + Word + FVec + MVec + PVec + WVec Settings -----*
```

Process code:

```
pythia.info.code()
```

Number of Multi Partonic Interaction:

```
pythia.info.nMPI()
```

Event Information: examples/main01.C

```
----- PYTHIA Event Listing (hard process) -----  
  
no      id  name          status  mothers  daughters  colours  p_x    p_y    p_z    e      m  
0       90  (system)     -11     0  0      0  0  0  0  0.000  0.000  0.000  8000.000  8000.000  
1       2212 (p+)     -12     0  0      3  0  0  0  0.000  0.000  4000.000  4000.000  0.938  
2       2212 (p+)     -12     0  0      4  0  0  0  0.000  0.000 -4000.000  4000.000  0.938  
3        21  (g)       -21     1  0      5  6 101 102  0.000  0.000  148.263  148.263  0.000  
4        21  (g)       -21     2  0      5  6 103 101  0.000  0.000  -2.904   2.904  0.000  
5        21  g         23     3  4      0  0 103 104  18.969  6.722  54.243   57.856  0.000  
6        21  g         23     3  4      0  0 104 102 -18.969 -6.722  91.115   93.311  0.000  
Charge sum: 0.000      Momentum sum: 0.000  0.000  145.358  151.167  41.503  
  
----- End PYTHIA Event Listing -----
```

The `event.list()` listing provides the main properties of each particle, by column:

- The index number of the particle
- PDG particle identity code (method `id()`)
- Particle name (method `name()`)
- Status (method `status()`)
- Mothers and daughters (method `mother1()`, `mother2()`, `daughter1()` and `daughter2()`)
- The colour flow of the process (methods `col()` and `acol()`)
- The components of the momentum 4-vector (p_x , p_y , p_z , E), in units of GeV with $c=1$ (methods `px()`, `py()`, `pz()` and `e()`)
- The mass (method `m()`)

Final Cross Section: examples/main01.C

```
*----- PYTHIA Event and Cross Section Statistics -----*
```

Subprocess	Code	Number of events			sigma +- delta (estimated) (mb)	
		Tried	Selected	Accepted		
g g -> g g	111	414	65	65	1.977e-01	1.482e-02
g g -> q qbar (uds)	112	8	2	2	2.626e-03	8.808e-04
q g -> q g	113	277	26	26	1.216e-01	1.158e-02
q q(bar)' -> q q(bar)'	114	33	6	6	2.056e-02	4.284e-03
q qbar -> g g	115	0	0	0	0.000e+00	0.000e+00
q qbar -> q' qbar' (uds)	116	0	0	0	0.000e+00	0.000e+00
g g -> c cbar	121	1	1	1	3.914e-03	3.914e-03
q qbar -> c cbar	122	0	0	0	0.000e+00	0.000e+00
g g -> b bbar	123	0	0	0	0.000e+00	0.000e+00
q qbar -> b bbar	124	0	0	0	0.000e+00	0.000e+00
sum		733	100	100	3.464e-01	1.970e-02

```
*----- End PYTHIA Event and Cross Section Statistics -----*
```


Initialization and Generation commands

- Standard in beginning:

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

- Initialization by one of different forms:
 - `pythia.init(idA, idB, eA, eB)` along $\pm z$ axis
 - `pythia.init(idA, idB, eCM)` in c.m. frame
 - `pythia.init("filename")` for Les Houches Event Files
 - `pythia.init()` takes above kinds of input from "cards"
 - `pythia.init(LHAinit*, LHAevent*)` for Les Houches Accord returns `false` if failed normally user setup mistake!
- Generation of next event by: `pythia.next()` with no arguments, but value `false` if failed
- At the end of the generation loop: `pythia.statistics()` provides summary information

Settings and Particle Data

- Can read in settings and particle data changes by:

`pythia.readString("command")`

`pythia.readFile("filename")` with one command per line in file

- Settings come in 4 types:
 - **Flags** on/off switches, `bool` (on=yes=ok=true=1, off=no=false=0)
 - **Modes** enumerated options, `int`
 - **Parms** (short for parameters) continuum of values, `double`
 - **Words** characters (no blanks), `string` and command is of form `task:property = values`, e.g.:
 - `PartonLevel:ISR = off` no initial-state radiation
 - `SigmaProcess:alphaSOrder = 0` freeze α_s
 - `TimeShower:pTmin = 1.0` cut off final-state radiation at 1 GeV
- To access particle data, instead command should be a form `id:property = value` or `id:channel:property = value`, e.g.:
 - `3122:mayDecay = no` do not allow Λ^0 decay
 - `215:3:products = 211 111 111` to let $a_2^+ \rightarrow \pi^+ \pi^0 \pi^0$

Setup: example of W' and so on

- **Beam energy:** `pythia.readString("Beams:eCM = 8000.");`
- **Process:** `pythia.readString("NewGaugeBoson:ffbar2Wprime = on");`
- **Particle mass, decay mode:**
 - `pythia.readString("4:m0 = 2000.");`
 - `pythia.readString("34:onMode = off");` switch off decay
 - `pythia.readString("34:onIfAny = 24");` then is final state contain pdg code 24 (namely, W)
- **Energy of hard interaction, minimum pT (important to reduce the number of events to be generated)**
 - `pythia.readString("PhaseSpace:mHatMin = 1300.");`
 - `pythia.readString("haseSpace:pTHatMin = 500.");`

Advanced Setup

- Choice of PDF:

- `pythia.readString("PDF:pSet = 7");` CTEQ 6L NLO

- Using LHAPDF (PDF package):

```
using LHAPDF
```

```
pythia.readString("PDF:pSet = LHAPDF6:CT10");
```

```
pythia.readString("Random:setSeed = on");
```

- External input (specify numbers when you run the code)

- If you do not specify anything, code produce same events. If you want to make huge number of events, you have to restart your code with different random number seed:

```
string iseed = argv[1]
```

```
string dummy = "Random:seed = "+iseed; // This makes longer
```

```
"string"
```

```
pythia.readString(dummy);
```

- Final state radiation: `pythia.readString("Tune::pp = 11");` // using Tunes

Example of a “cards” file

```
! This file contains commands to be read in for a Pythia8 run.
! Lines not beginning with a letter or digit are comments.

! 1) Settings that could be used in a main program, if desired.
Main:idBeamA = 2212          ! first beam, p = 2212, pbar = -2212
Main:idBeamB = 2212          ! second beam, p = 2212, pbar = -2212
Main:eCM = 14000.           ! CM energy of collision
Main:numberOfEvents = 1000   ! number of events to generate
Main:numberToList = 2        ! number of events to print
Main:timesToShow = 20        ! show how far along run is
Main:showChangedSettings = on ! print changed flags/modes/parameters
Main:showAllSettings = off   ! print all flags/modes/parameters

! 2) Settings for the hard-process generation.
HiggsSM:gg2H = on           ! Higgs production by gluon-gluon fusion
25:m0 = 123.5                ! Higgs mass
25:onMode = off              ! switch off all Higgs decay channels
25:onIfMatch = 22 22         ! switch back on Higgs -> gamma gamma
SigmaProcess:alphaSvalue = 0.12 ! alpha_s(m_Z) in matrix elements

! 3) Settings for the subsequent event generation process.
SpaceShower:alphaSvalue = 0.13 ! alpha_s(m_Z) in initial-state radiation
MultipleInteractions:pT0Ref = 3.0 ! pT_0 regularization at reference energy
#PartonLevel:MI = off        ! no multiple interactions
#PartonLevel:ISR = off       ! no initial-state radiation
#PartonLevel:FSR = off       ! no final-state radiation
#HadronLevel:Hadronize = off ! no hadronization
```

More on Settings

- Settings are stored in 4 separated maps (flags/modes/parms/words)
- For each setting, need to store:
 - **name**: of form `task:property`, e.g. `TimeShower:pTmin`
 - **default value**
 - **current value**
 - **allowed range**: minimum/maximum on/off (not for flags)
- Useful commands:
 - `pythia.settings.listAll()` : complete list
 - `pythia.settings.listChanged()` : only changed ones

```
*----- PYTHIA Flag + Mode + Parm + Word Settings (changes only) -----*
```

Name	Now	Default	Min	Max
HardQCD:all	on	off		
Main:eCM	14000.000	1960.000	10.00000	
Main:numberToList	1	2	0	
Main:showChangedParticleData	on	off		
Main:timesToShow	20	50	0	
MultipleInteractions:pTmin	3.00000	0.20000	0.10000	10.00000
PhaseSpace:pTHatMin	50.00000	0.0	0.0	
PromptPhoton:all	on	off		
SpaceShower:pT0Ref	2.00000	2.20000	0.50000	10.00000

```
*----- End PYTHIA Flag + Mode + Parm + Word Settings -----*
```

Save to .hepmc files

main41.cc

```
#include "Pythia8/Pythia.h"  
#include "Pythia8Plugins/HepMC2.h"
```

Need new function written in HepMC2.h

```
using namespace Pythia8;
```

```
int main() {
```

```
    // Interface for conversion from Pythia8::Event to HepMC event.  
    HepMC::Pythia8ToHepMC ToHepMC;
```

Tool to convert pythia events to HepMC

```
    // Specify file where HepMC events will be stored.  
    HepMC::IO_GenEvent ascii_io("hepmcout41.dat", std::ios::out);
```

Name of the output file

```
    // 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.  
    for (int iEvent = 0; iEvent < 100; ++iEvent) {  
        if (!pythia.next()) continue;
```

Memory of HepMC event

```
        // 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 );
```

Copy and convert Pythia events to HepMC

```
        // Construct new empty HepMC event and fill it.  
        // Units will be as chosen for HepMC build; but can be changed  
        // by arguments, e.g. GenEvt( HepMC::Units::GEV, HepMC::Units::MM)  
        HepMC::GenEvent* hepmcevt = new HepMC::GenEvent();  
        ToHepMC.fill_next_event( pythia, hepmcevt );
```

Write to hepmcout41.dat file

```
        // Write the HepMC event to file. Done with it.  
        ascii_io << hepmcevt;  
        delete hepmcevt;
```

Kill the memory

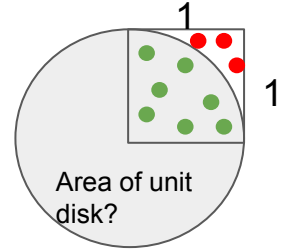
```
    // End of event loop. Statistics. Histogram.  
}
```

Pythia8 and C++

- Pythia8 and HepMC are “class”
- Pythia8 can generate events and it also contain another class
- You can use objects in the different class by writing
 - `Pythia8::name`
 - `hepMC::name`
- HepMC objects are defined in HepMC2.h
- If you use `namespace Pythia8;` allows you to omit `Pythia8::`
- Object defined with `*` is an address; without `*`, it is the contents at the address.
- Member can be accessed by `object->member_name` for the object defined by the address, and if it is not the address, access by `object.member_name`
- If you create by method `new`, then `delete` to avoid memory leak

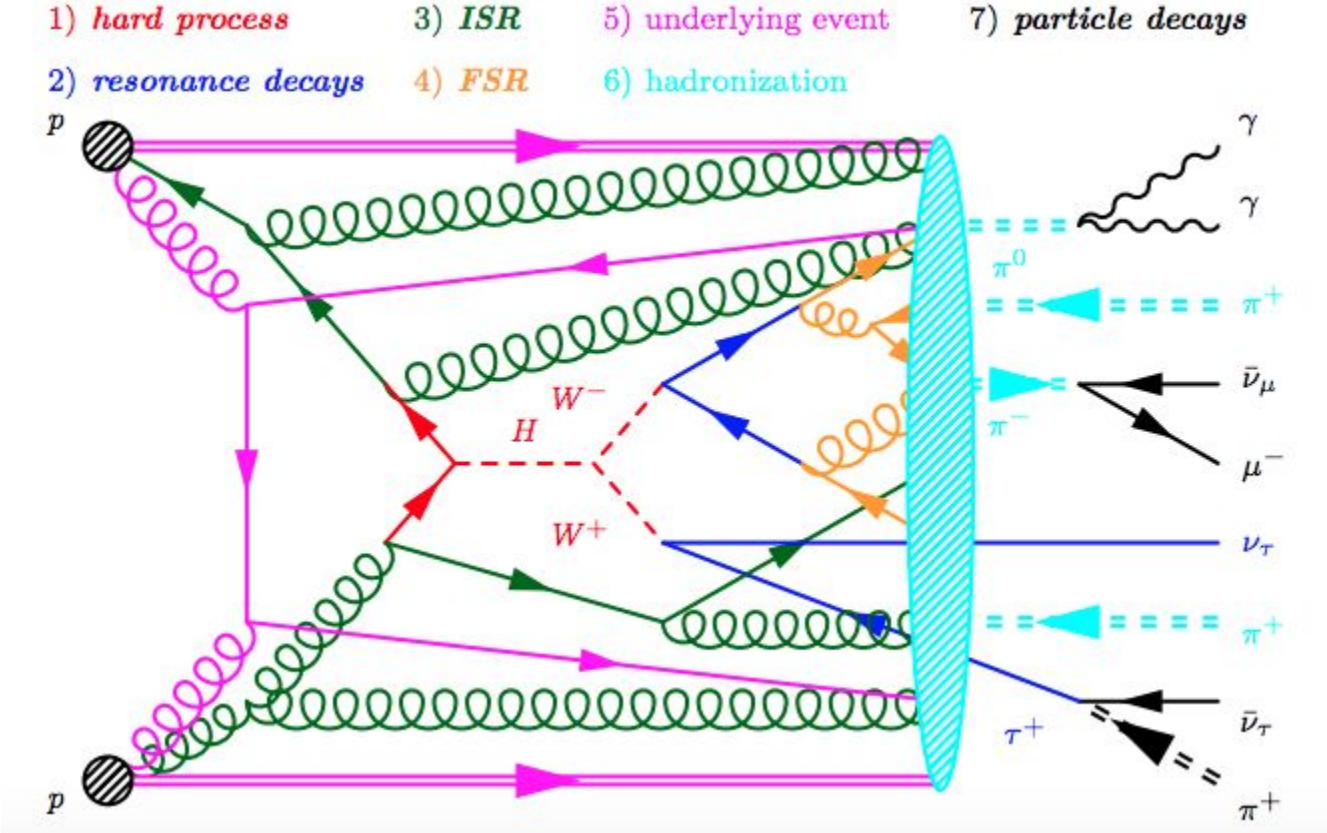
Exercises

1. Write a code that estimate the area of a unit disk using the hit-or-miss Monte Carlo method. We know the radius of the unit disk is 1 thus the unit circle is inscribed within a square of length 2. Tip: generate samples within this square and count the number of points falling within the disk. To test whether the point is inside (hit) or outside (miss) the disk, we simply need to measure the distance of the sample from the origin (the center of the unit disk) and check whether this distance is smaller (or equal) than the disk radius (which is equal to 1 for a unit disk).
2. Choose your process of new physics, change the particle mass, see how cross section reduces with mass



Backup Slides

Event Anatomy



Internal Hard Processes

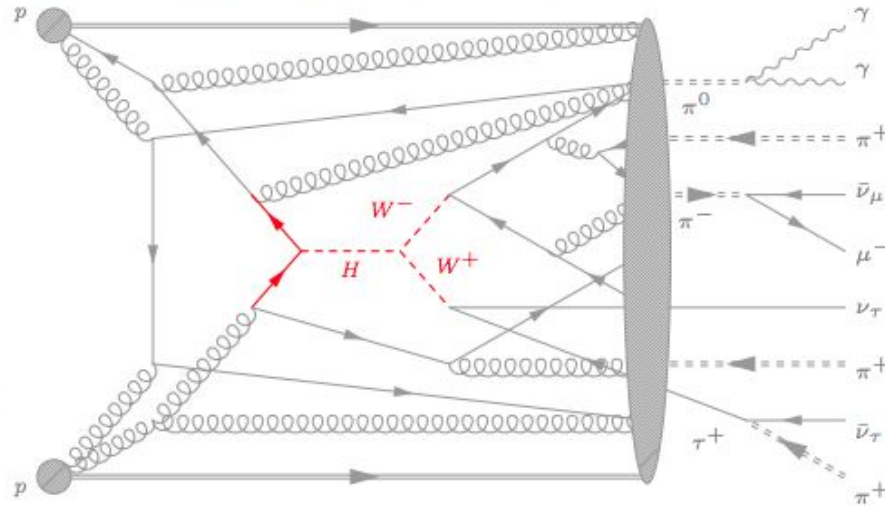
Internal Hard Processes

Process Selection

- QCD
- Electroweak
- Onia
- Top
- Fourth Generation
- Higgs
- SUSY
- New Gauge Bosons
- Left-Right Symmetry
- Leptoquark
- Compositeness
- Hidden Valleys
- Extra Dimensions

- A Second Hard Process
- Phase Space Cuts
- Couplings and Scales
- Standard-Model Parameters
- Total Cross Sections
- Resonance Decays
- Timelike Showers
- Spacelike Showers
- Automated Shower Variations
- Weak Showers
- Multiparton Interactions
- Beam Remnants
- Colour Reconnection
- Diffraction
- Fragmentation
- Flavour Selection
- Particle Decays
- R-hadrons

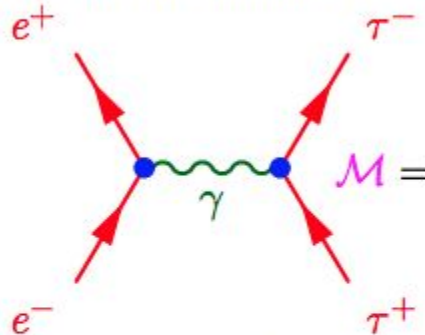
- 1) *hard process*
- 2) *resonance decays*
- 3) *ISR*
- 4) *FSR*
- 5) *underlying event*
- 6) *hadronization*
- 7) *particle decays*



Introduction

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

- 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

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

- convolute with PDFs for full cross-section

$$\sigma_{a_1 a_2 \rightarrow B} = \int \int x_{a_1}(x_{p_1}, Q^2, p_1) x_{a_2}(x_{p_2}, Q^2, p_2) \sigma_{p_1 p_2 \rightarrow B} dx_{p_1} dx_{p_2}$$

SUSY

- MSSM and nMSSM implementations of SUSY production
 - all MSSM cross-sections validated
 - $q\bar{q} \rightarrow \tilde{\chi}^0\tilde{\chi}^0$, $q\bar{q} \rightarrow \tilde{\chi}^\pm\tilde{\chi}^0$, $q\bar{q} \rightarrow \tilde{\chi}^+\tilde{\chi}^-$
 - $q\bar{g} \rightarrow \tilde{\chi}^0\tilde{q}$, $q\bar{g} \rightarrow \tilde{\chi}^\pm\tilde{q}$
 - $gg \rightarrow \tilde{g}\tilde{g}$, $q\bar{q} \rightarrow \tilde{g}\tilde{g}$
 - $qg \rightarrow \tilde{q}\tilde{g}$
 - $gg \rightarrow \tilde{q}\tilde{q}$, $q\bar{q} \rightarrow \tilde{q}\tilde{q}$, $qq \rightarrow \tilde{q}\tilde{q}$
 - $qq \rightarrow \tilde{q}$
- $\tilde{q}\tilde{q}$ 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

- super-CKM basis used to describe the mass eigenstates
 - $R^u : (\tilde{u}_L, \tilde{c}_L, \tilde{t}_L, \tilde{u}_R, \tilde{c}_R, \tilde{t}_R) \rightarrow (\tilde{u}_1, \tilde{u}_2, \tilde{u}_3, \tilde{u}_4, \tilde{u}_5, \tilde{u}_6)$
 - $R^d : (\tilde{d}_L, \tilde{s}_L, \tilde{b}_L, \tilde{d}_R, \tilde{s}_R, \tilde{b}_R) \rightarrow (\tilde{d}_1, \tilde{d}_2, \tilde{d}_3, \tilde{d}_4, \tilde{d}_5, \tilde{d}_6)$
 - $\mathcal{N} : (i\tilde{B}, -i\tilde{W}_3, H_1, H_2) \rightarrow (\tilde{\chi}_1^0, \tilde{\chi}_2^0, \tilde{\chi}_3^0, \tilde{\chi}_4^0)$
 - $\mathcal{U} : (iW^+, H^+) \rightarrow (\tilde{\chi}_1^+, \tilde{\chi}_2^+)$
 - $\mathcal{V} : (iW^-, H^-) \rightarrow (\tilde{\chi}_1^-, \tilde{\chi}_2^-)$
- 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
 - $f\bar{f} \rightarrow H$, $gg \rightarrow H$, $\gamma\gamma \rightarrow H$
 - $f\bar{f} \rightarrow HZ$, $f\bar{f} \rightarrow HW^\pm$
 - $f\bar{f} \rightarrow Hf\bar{f}$, $q\bar{q} \rightarrow Ht\bar{t}/b\bar{b}$, $gg \rightarrow Ht\bar{t}/b\bar{b}$
 - $qg \rightarrow Hq$, $gg \rightarrow Hg$, $q\bar{q} \rightarrow Hg$
 - 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
 - $f\bar{f} \rightarrow H^\pm$, $bg \rightarrow H^\pm t$
 - $f\bar{f} \rightarrow A^0 h^0$, $f\bar{f} \rightarrow A^0 H^0$, $f\bar{f} \rightarrow H^\pm h^0$, $f\bar{f} \rightarrow H^\pm H^0$, $f\bar{f} \rightarrow H^+ H^-$
- 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
 - $f\bar{f} \rightarrow Z', f\bar{f} \rightarrow W'^{\pm}$
 - 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
 - $q\ell \rightarrow LQ$, $q\bar{q} \rightarrow LQ\ell$, $gq \rightarrow LQ\bar{L}\bar{Q}$, $q\bar{q} \rightarrow LQ\bar{L}\bar{Q}$
 - 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
 - $2 \rightarrow 1$ processes via gauge boson interactions: $dg \rightarrow d^*, \dots$
 - $2 \rightarrow 2$ processes via contact interactions: $qq \rightarrow d^*q, \dots$
 - decays include matrix element corrections
 - only gauge boson interaction decays implemented: $d^* \rightarrow \gamma d$
- hidden valleys
 - based on the work of **JHEP 1104, 091 (2011)**
 - hidden unbroken $SU(N)$ symmetry: **HiddenValley:Ngaug**
 - hidden sector mirrors SM fermions: d_v, e_v^-, \dots
 - $gg \rightarrow q_v \bar{q}_v, q \bar{q} \rightarrow g \rightarrow q_v \bar{q}_v, f \bar{f} \rightarrow \gamma^*/Z \rightarrow f_v \bar{f}_v$
 - f_v can radiate $g, \gamma,$ 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

External Hard Processes

Les Houches Accord

SUSY Les Houches Accord
HepMC Interface
ProMC Files

Semi-Internal Processes

Semi-Internal Resonances

MadGraph5 Processes

AlpGen Event Interface
Matching and Merging
-- **POWHEG Merging**
-- aMC@NLO Matching
-- CKKW-L Merging
-- Jet Matching
-- UMEPS Merging
-- NLO Merging

User Hooks

Hadron-Level Standalone

External Decays

Beam Shape

Parton Distributions

Jet Finders

Random Numbers

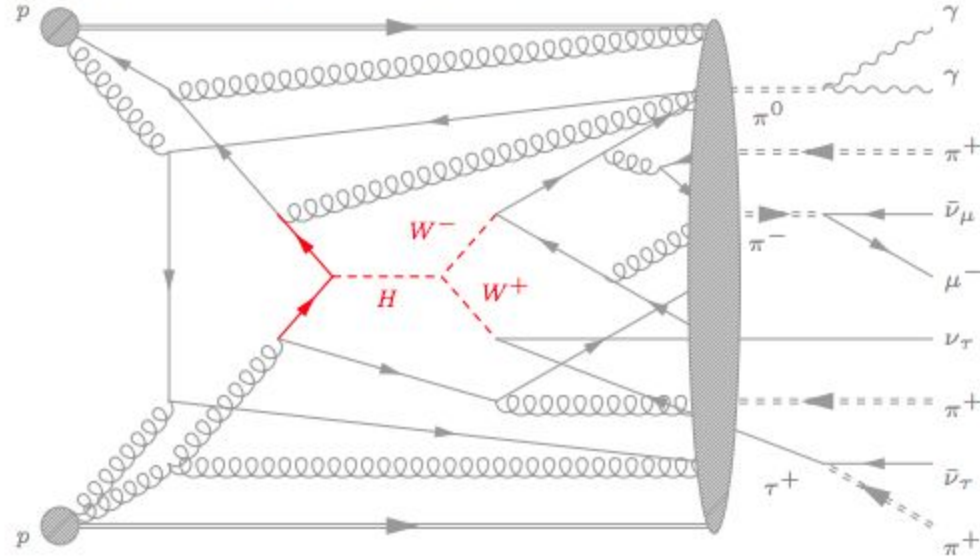
Implement New Showers

RIVET usage

ROOT usage

A Python Interface

- 1) *hard process*
- 2) *resonance decays*
- 3) *ISR*
- 4) *FSR*
- 5) *underlying event*
- 6) *hadronization*
- 7) *particle decays*



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 Processes

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

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

- example in `main22.cc`
- `double SigmaProcess::sigmaHat()` calculates the cross-section
 - $2 \rightarrow 1$: return $\hat{\sigma}(\hat{s})$
 - $2 \rightarrow 2$: return $d\hat{\sigma}/d\hat{t}$
 - $2 \rightarrow 3$: return $|\mathcal{M}|^2$ with normalization $\hat{\sigma} = \int |\mathcal{M}|^2 / (2\hat{s}) d\Phi$
- `string SigmaProcess::inFlux()` defines the incoming partons
 - `gg`, `qg`, `fgm`, `ggm`, `gm gm`
 - `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 mll 80");
pythia.readString("Random:setSeed = on");
pythia.readString("Random:seed = 1");
pythia.setLHAupPtr(&madgraph);
pythia.init();
```

- remember to set up matching/merging correctly

Showers

Showers

Process Selection

- QCD
- Electroweak
- Onia
- Top
- Fourth Generation
- Higgs
- SUSY
- New Gauge Bosons
- Left-Right Symmetry
- Leptoquark
- Compositeness
- Hidden Valleys
- Extra Dimensions

A Second Hard Process

- Phase Space Cuts
- Couplings and Scales
- Standard-Model Parameters
- Total Cross Sections
- Resonance Decays

Timelike Showers

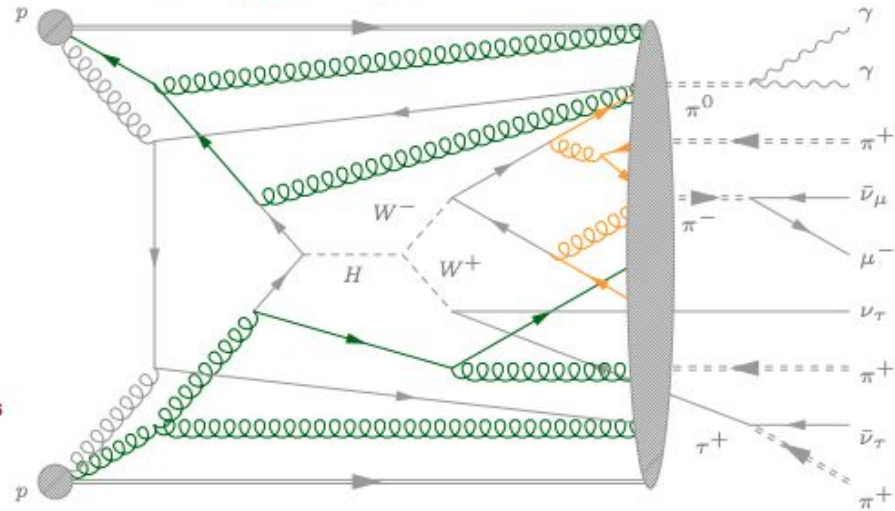
Spacelike Showers

Automated Shower Variations

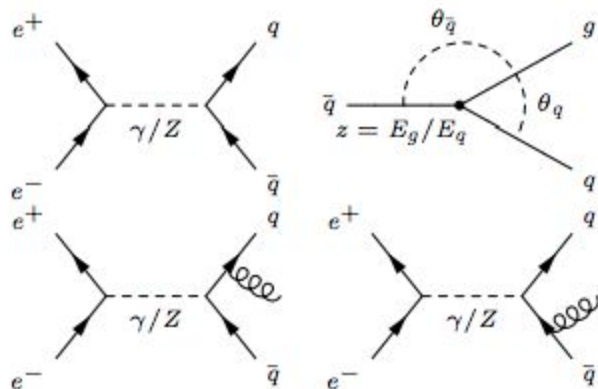
Weak Showers

- Multiparton Interactions
- Beam Remnants
- Colour Reconnection
- Diffraction
- Fragmentation
- Flavour Selection
- Particle Decays
- R-hadrons

- 1) *hard process*
- 2) *resonance decays*
- 3) *ISR*
- 4) *FSR*
- 5) *underlying event*
- 6) *hadronization*
- 7) *particle decays*



Introduction



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

- factorize collinear divergences as independent emissions

$$d\sigma_{e^+e^- \rightarrow q\bar{q}g} \approx \sigma_{e^+e^- \rightarrow q\bar{q}} \sum_i \left(\left(\frac{d\theta_{p_i}^2}{\theta_{p_i}^2} \right) \left(\frac{\alpha_s}{2\pi} \right) \left(\frac{N_c^2 - 1}{2N_c} \right) \left(\frac{1 + (1-z)^2}{z} \right) dz \right)$$

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

$$d\sigma_{A \rightarrow B b_j} \approx \sigma_{A \rightarrow B} \sum_i \left(\left(\frac{d\theta_{b_i}^2}{\theta_{b_i}^2} \right) \mathcal{P}_{b_j b_i}(z, \alpha_s) dz \right)$$

- parton b_i emits parton b_j

Internal Showers

- timelike shower (final state radiation) is fully interleaved p_T -ordered

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

- spacelike showers (initial state radiation) is fully interleaved p_T -ordered

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

- 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

- 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

- 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](https://arxiv.org/abs/1605.06142)
 - dipole-antenna shower plugin to PYTHIA
 - provide $2 \rightarrow 3$ shower kernels
 - captures both collinear dynamics and soft singularities
 - available from <http://vincia.hepforge.org>

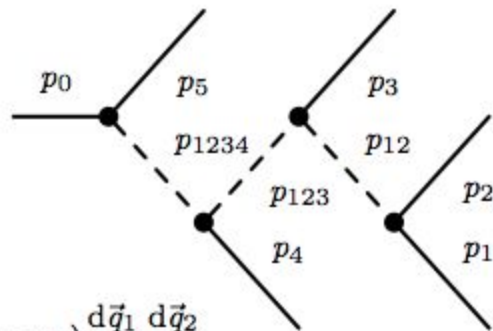
Matching and Merging

- 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

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
 - $\tilde{q} \rightarrow \tilde{\chi}, \tilde{q}W/Z, qq, lq$
 - $\tilde{g} \rightarrow \tilde{q}q$
 - $\tilde{\chi} \rightarrow \tilde{\chi}W/Z, \tilde{q}q, \tilde{\ell}\ell$
 - $\tilde{\chi}^0 \rightarrow qq\bar{q}$
 - $\tilde{\ell} \rightarrow \tilde{l}\tilde{\chi}, \tilde{\ell}W/Z$
- 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
 - $\omega, \phi \rightarrow \pi^+ \pi^- \pi^0$
 - $V \rightarrow PS PS$ with V from $PS \rightarrow PS V$ or $PS \rightarrow \gamma V$
 - Dalitz decay $X \rightarrow Y \ell^+ \ell^-$
 - double Dalitz decay $X \rightarrow \ell^+ \ell^- \ell^+ \ell^-$
 - weak decays
 - $B \rightarrow \gamma X$
- 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
 - $1 \rightarrow n$ 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

- 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!