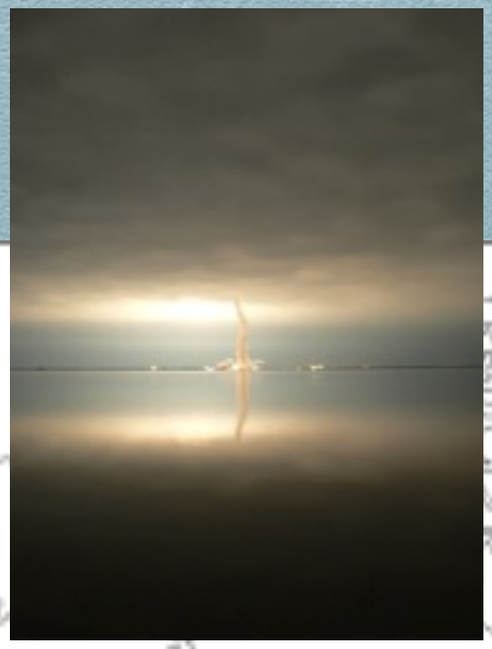
Deepak Kar, Jets@LHC Workshop, ICTS, Bangalore

Monte Carlo Generators and Rivet Tutorial

An amazing adventure...



Gainesville, FL, USA
2003-2008

Glasgow, UK
2012-2014

Dresden, Germany
2009-2011

Calcutta, India
till *2003*

Johannesburg, SA
2015 -

Analysis 101

	Step	Needed
1	Devise a strategy for for your measurement or search	1. Knowledge of theory 2. Simulated events
2	Perform the analysis	1. Analysis framework 2. Computing resources
3	Obtain/interpret the results	1. Statistical tools 2. Simulated events

Analysis 101

	Step	Needed
1	Devise a strategy for for your measurement or search	1. **Knowledge of theory** 2. **Simulated events**
2	Perform the analysis	1. **Analysis framework** 2. Computing resources
3	Obtain/interpret the results	1. Statistical tools 2. **Simulated events**

This Tutorial

- How to generate simulated events: Monte Carlo event generators
- Analysis framework: RIVET for *particle level* analysis (Sorry no ROOT today!)
- Hands on exercises
- A good time to fire up the virtual machine (hope you have it :)
- Username/password: student/2016

Monte Carlo

image collected from web

Monte Carlo

image collected from web

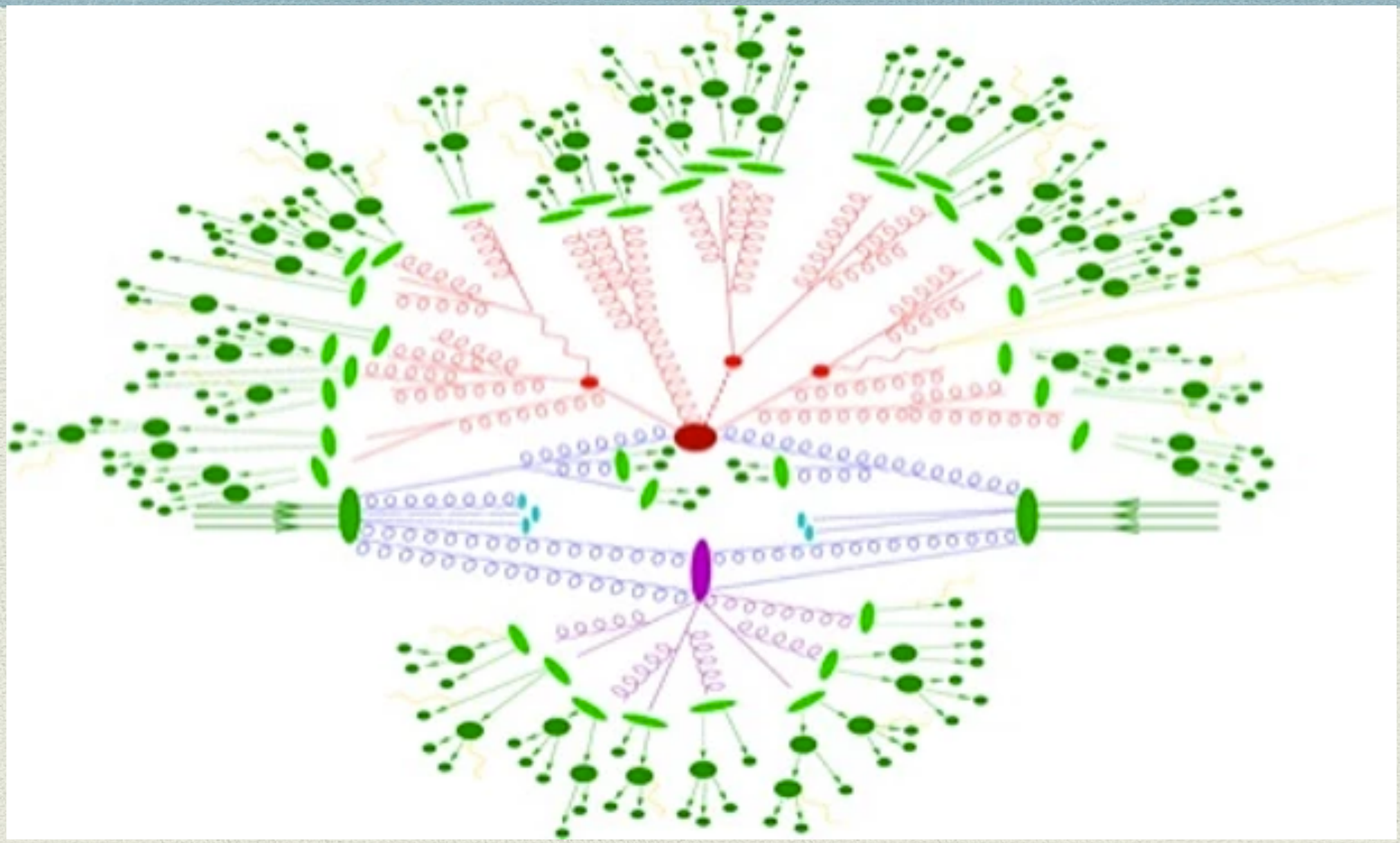
Why Probability?

- Nature is probabilistic!
- Each process (production or decay of a particle) has a certain *branching fraction*.
- Actual cross section for a process: integrate amplitude square over the phase space
- Too many degrees of freedom, so have to sample
- Generate events (list of four vectors of particles), often with weights (to account for real branching fraction)

"The predictions of the model are reasonable enough physically that we expect it may be close enough to reality to be useful in designing future experiments and to serve as a reasonable approximation to compare to data. We do not think of the model as a sound physical theory . . ."

– Richard Feynman and Rick Field, 1978

Event Generation

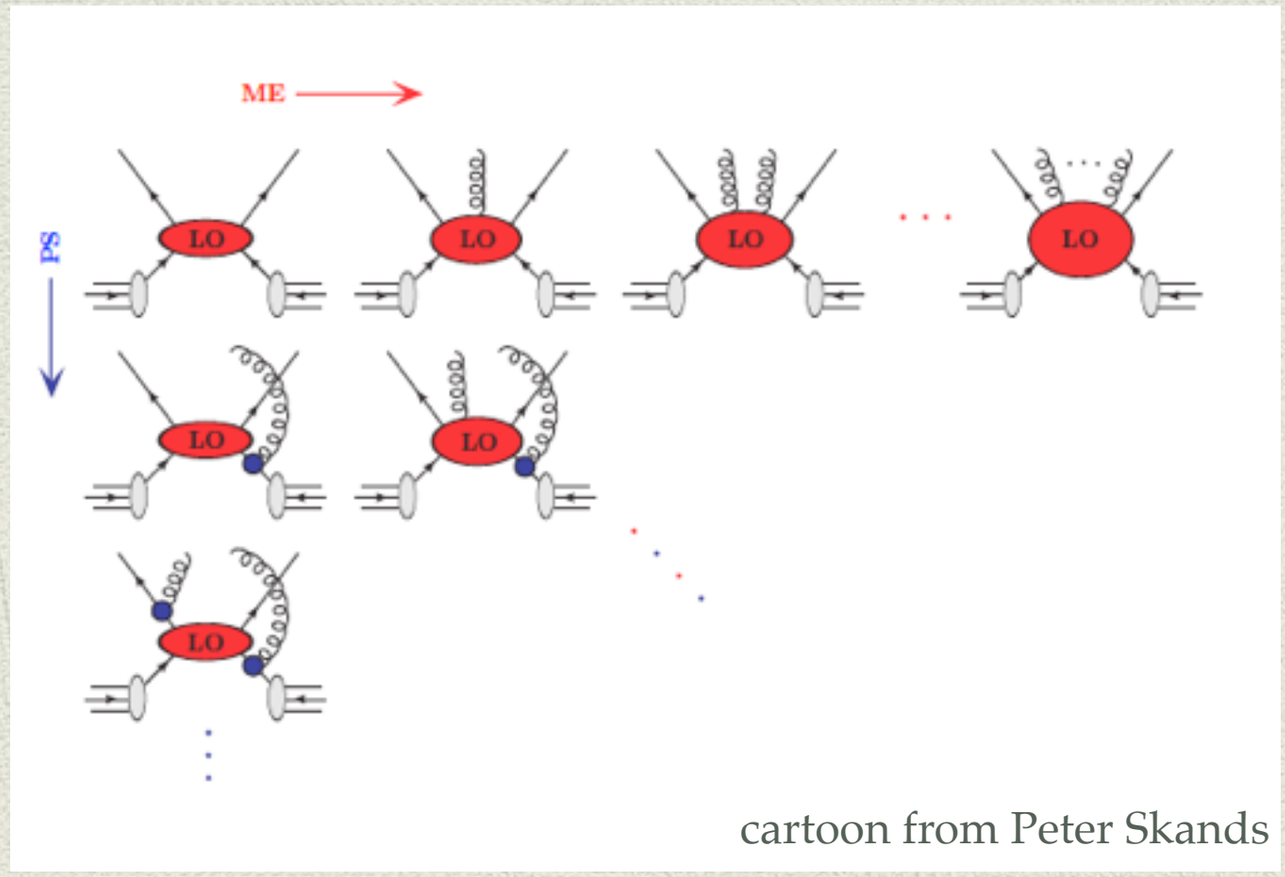


cartoon from Frank Krauss

Event Generation

Divide and conquer:

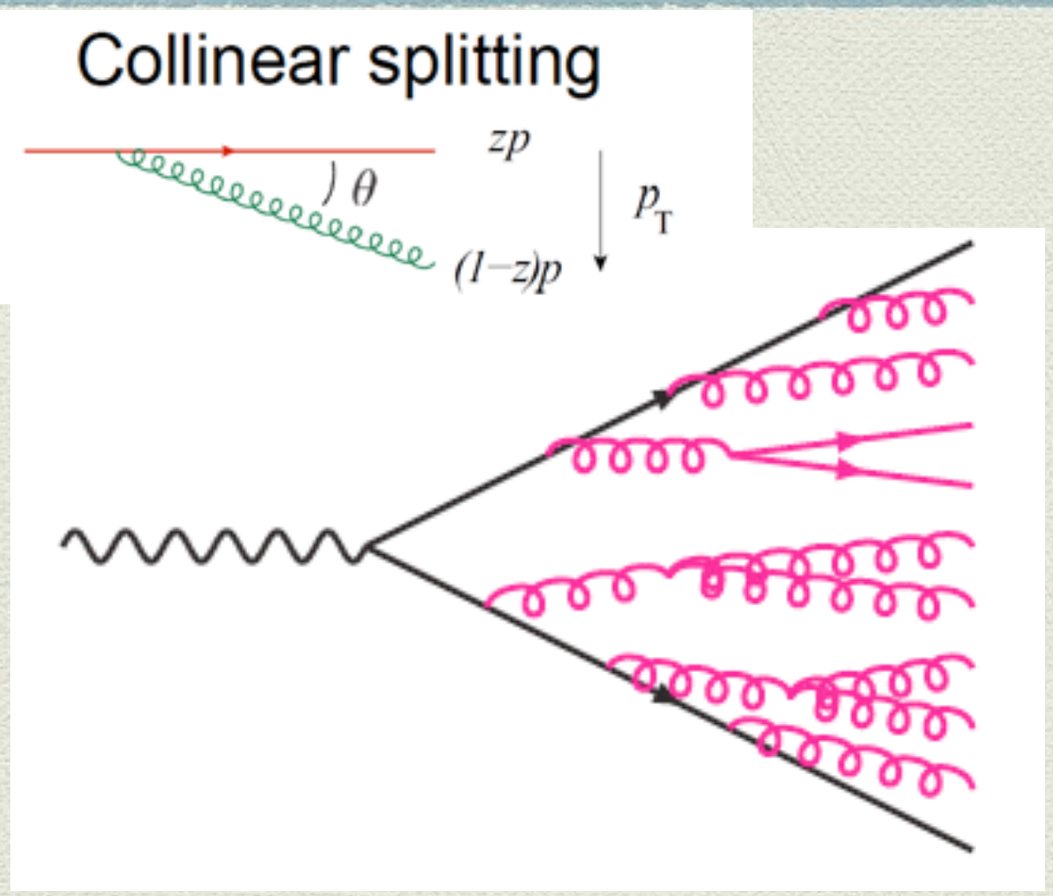
Matrix Element (ME) and Parton Shower (PS)



cartoon from Peter Skands

- Hard scatter: calculable in some order in perturbation theory (multi-leg, multi-loop)
- Parton shower: mostly non-perturbative physics, phenomenologically modeled.

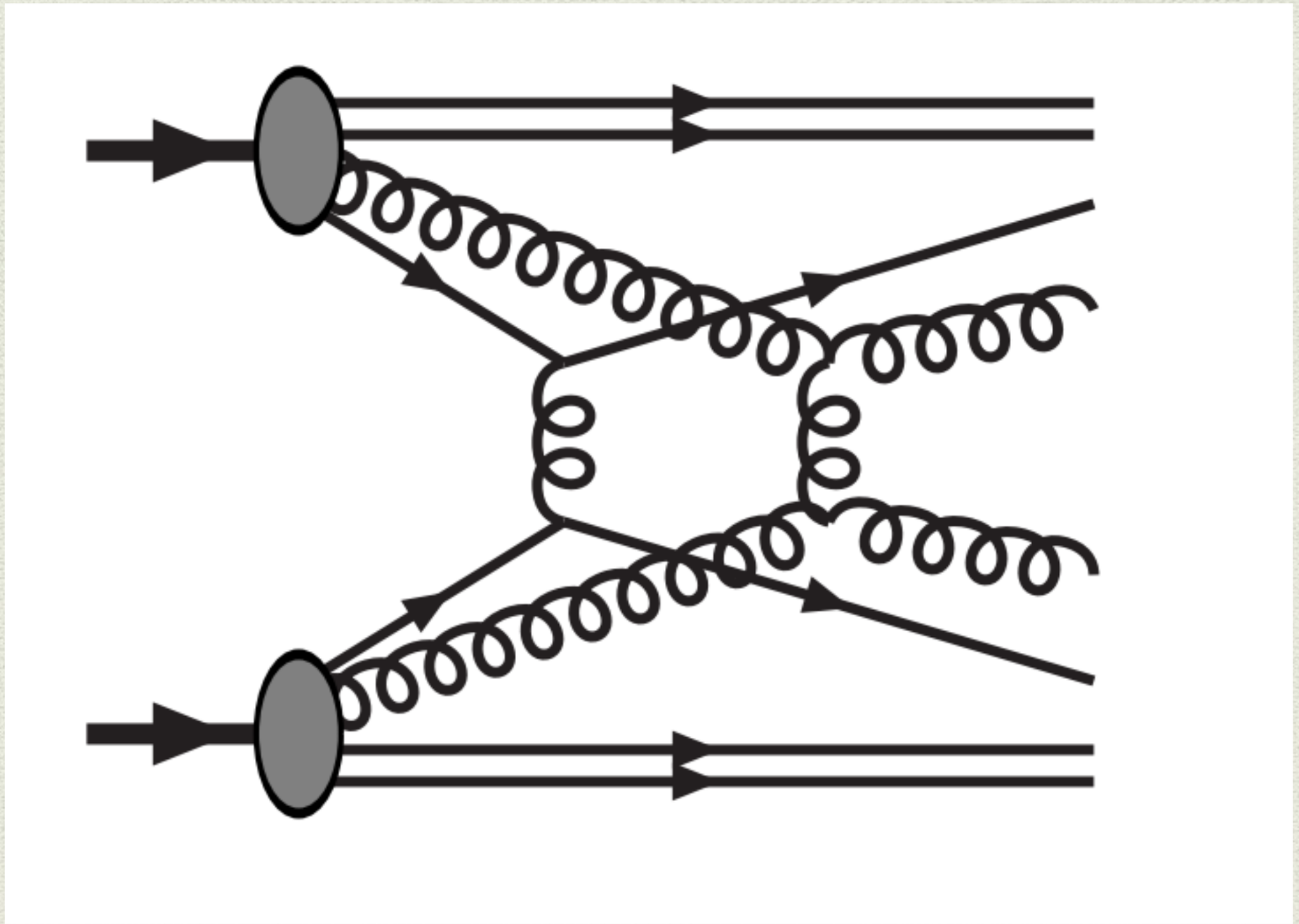
Parton Shower



Initial and Final State Radiation (I/FSR)

- Build up the complicated final state with the shower
- DGLAP equation, Sudakov form factor
- Branchings continue till all partons form colour-neutral hadrons.

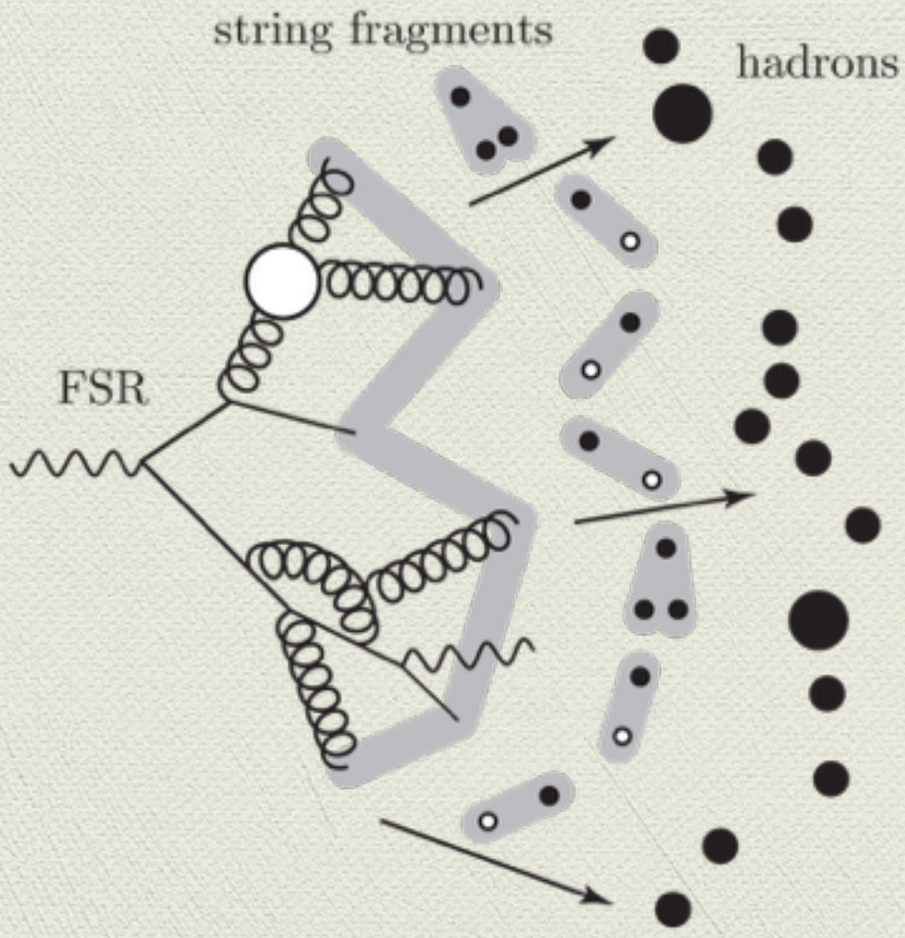
Double/Multiple Parton Interaction

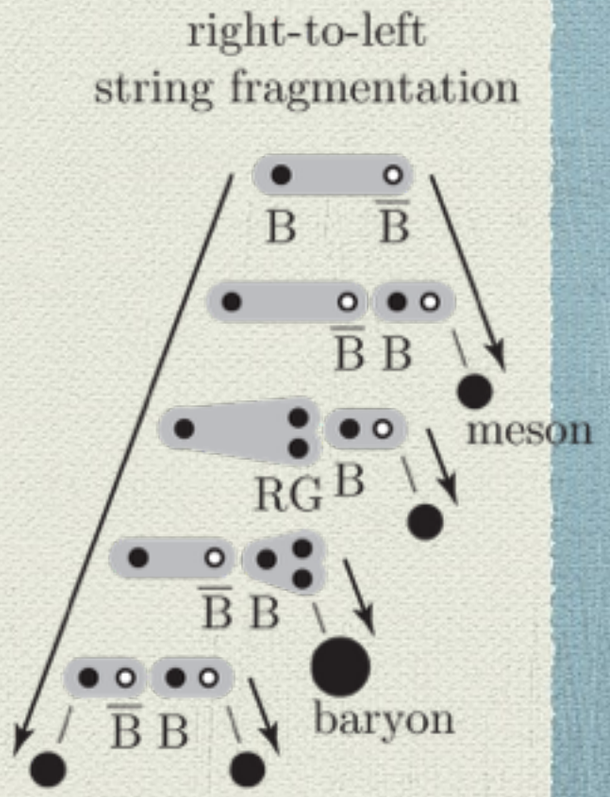


Hadronisation / Fragmentation

Often used interchangeably

- Hadronisation: formation of hadrons from partons (cluster or string model)
- Fragmentation: decay of hadrons



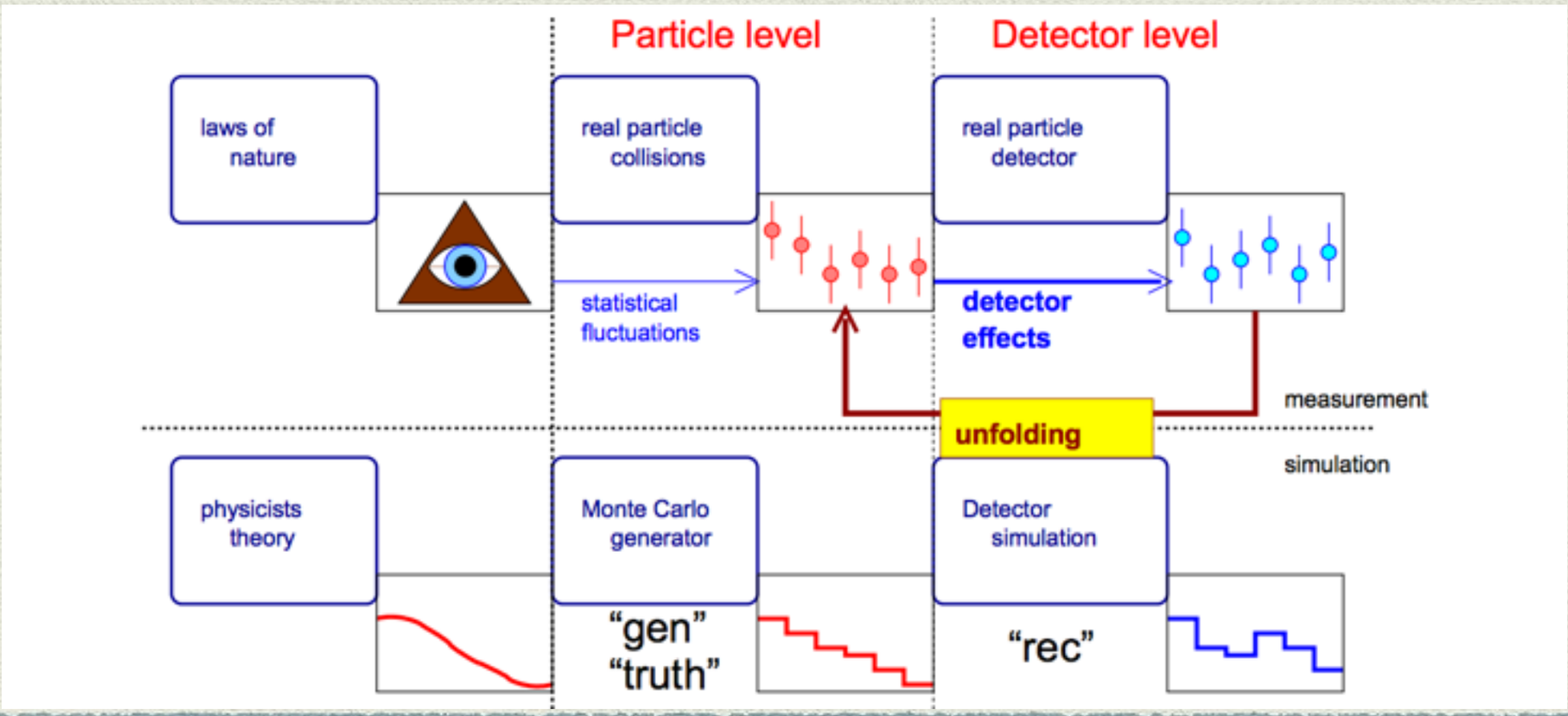


Tunes

- Free parameters controlling different aspects of PS
- Often highly correlated or anti-correlated
- Change them simultaneously to get the best description of data

What you see is not what you get!

- What we measure is at detector level
- What MC generators give you is at generator/particle level (or so called truth-level)
- Unfold the measured data back to particle level/apply detector simulation on particle level simulation to bring it to detector level.



Pythia8

- ... is a leading order PS generator.
- One of the most widely used for many years.
- Relatively easy to install (along with its *friends*: HepMC, and LHAPDF6) and run, online user manual: http://home.thep.lu.se/~torbjorn/pythia82html/Welcome.html
- Run via various *mainXX* programs.

Generating events with Pythia8

- Start the terminal (green box, bottom left corner)
- Go to: tutorial/mc/higgs/pythia
- We will use main42, a generic main program. Copy it:
  cp/opt/hep/share/Pythia8/examples/main42.* .
- Compile: make main42, should result in a main42 executable in the directory.
- Input (which process to generate, how many events, collision energy, ...) are specified via a *runcard (cmnd files)*, we will use main42.cmnd
- Generate 5000 Z-boson events, decaying to muon pairs.

Modifying the Runcard

- Generate at 13 TeV (will be relevant later)

```
! 5a) Pick processes and kinematics cuts.
WeakSingleBoson:ffbar2gmZ = on    ! inclusive Z production
23:onMode = off                   ! turn all decays off
23:onIfAny = 13                   ! decay only to muons
23:mMin = 60.                     ! minimum mass of the Z


! 6) Other settings. Can be expanded as desired.
! Note: may overwrite some of the values above, so watch out.

Tune:ee = 7
Tune:pp = 14                       ! use Monash Tune

ParticleDecays:limitTau0 = on      ! set long-lived particle stable ...
ParticleDecays:tau0Max = 10        ! ... if c*tau0 > 10 mm
```

- ./main42 main42.cmnd out.hepmc

So then ...

Analyze the events

- ROOT is used extensively by the experiments
- But unless you are an experimentalist, it is probably too intimidating for you
- Many times, you just want to quickly look at simulated events...

RIVET

based somewhat on Andy Buckley's LH13 tutorial

- A generator agonistic analysis system for generators (no direct data analysis!) in C++ (now in C++11)
- Physics plots from generator output (in HepMC format)
- Compare MC predictions with *built-in* actual (unfolded) data analyses from different experiments
- Everything defined in terms of stable final state objects
- Details: https://rivet.hepforge.org/

Important!

Analyses intended to be based on physical objects:

- Final state hadrons
- Jets (FastJet)
- Muons, Electrons (dressed)
- Bosons reconstructed from particles (rather than taken from event record)

Version 2.5 contains $\sim$ 350 Analyses (195 LHC)

- Monte Carlo validation and tuning, data preservation
- Lots of code examples to get inspired

Trying out RIVET

- ... it is setup for you!
- rivet --help
- rivet --list-analyses (ATLAS_ , MC_)

Running a Data Analysis

- Since we are looking at Z-boson events, lets try
  rivet -a ATLAS_2015_CONF_2015_041_MU out.hepmc
- Output: Rivet.yoda
- Look inside the yoda file
- Plot with rivet-mkhtml Rivet.yoda
  :“legend label” (--mc-errs)
- View plots by firefox plots/index.html

Writing an Analysis

- The analyses named MC_ are pure MC based analysis, no reference data to compare with.

- Useful for testing generator predictions.
- Make a template: rivet-mkanalysis MC_MyAna
- Find the MC_MyAna.cc file in the directory (also MC_MyAna.info and MC_MyAna.plot)
- Look inside the cc file!

Writing an Analysis

- Analyses are classes and inherit from **`Rivet::Analysis`**
- Usual init/execute/finalize-type event loop structure (certainly familiar from experimental frameworks)
- Weird *projection* things in **`init`** and **`analyze`**
- *Mostly* normal-looking everything else

Walkthrough

```
// Basic include stuff
#include "Rivet/Analysis.hh"
#include "Rivet/Projections/FinalState.hh"
#include "Rivet/Projections/ChargedFinalState.hh"
#include "Rivet/Projections/FastJets.hh"
#include "Rivet/Projections/VetoedFinalState.hh"
#include "Rivet/Projections/ZFinder.hh"
#include "Rivet/Math/Vector4.hh"
```

Walkthrough

```
//Usual Rivet template

namespace Rivet {

  using namespace Cuts;


class MC_MyAna : public Analysis {
public:

  /// Constructor
  MC_MyAna()
    : Analysis("MC_MyAna")
  {    }


public:
```

Projections

- Observable calculators - from an event, *project* out the *physical* observables.
- Already defined in the framework
- Registered with a name in *init*
- Applied to the current event in *analyze*
- Avoids unnecessary repetition in the code

Walkthrough

```
// Objects to be used in the analysis, example of projections

      FinalState fs;
      Cut cuts = etaIn(-2.5,2.5) & (pT >= 10*GeV);


      // Z-boson
      ZFinder zfinder(fs, cuts, PID::MUON, 66.0*GeV, 116.0*GeV, 0.1, ZFinder::CLUSTERNODECAY);
      addProjection(zfinder, "ZFinder");

    // Define veto FS in order to prevent Z-decay products entering the jet algorithm
    VetoedFinalState had_fs;
    had_fs.addVetoOnThisFinalState(getProjection<ZFinder>("ZFinder"));
    FastJets jets(had_fs, FastJets::ANTIKT, 0.4);
    jets.useInvisibles(true);
    addProjection(jets, "jets");



  // Declare the histograms. Add more as needed, keeping in mind the type

      //Z plots
      _hZpt        = bookHisto1D("Zpt",100,0,500);

      //Z and jet plots
      _hjets         = bookHisto1D("Jets", 10, 0, 10);
      _hleadjetpt    = bookHisto1D("leadjetpt",100,0,500);
      _hdphi12       = bookHisto1D("dphi12",157,0,3.14);
```

Some Details

- ChargedFinalState
- NeutralFinalState
- UnstableFinalState
- IdentifiedFinalState
- VetoedFinalState
- DISFinalState
- VisibleFinalState
- HadronicFinalState

Some Other Details

`Particle` and `Jet` both have a `momentum()` method which returns a `FourMomentum`.

Some `FourMomentum` methods: `eta()`, `pT()`, `phi()`, `rapidity()`, `E()`, `px()` etc., `mass()`. Hopefully intuitive!

Histogramming

- Declare at *init* by bookHisto1D or bookProfile1D (usual name, binning)
- Can be *autobooked* from reference data!
- Usual fill method in *analyze*
- scale or normalize in *finalize*
- Declare the pointers

Plot File (for later)

```
BEGIN PLOT /ATLAS_2015_I1343107/d18-x01-y01
XLabel=$E^{\rm{miss}}_T$ [GeV]
YLabel=Events
XMin=150
END PLOT
```

Walkthrough

```
//First: look at the Z

 const ZFinder& zfinder = applyProjection<ZFinder>(event, "ZFinder");
 if (zfinder.constituents().size()!=2) vetoEvent;
 if (zfinder.bosons().size() != 1) vetoEvent;

 FourMomentum z = zfinder.bosons()[0].momentum();
 FourMomentum l1 = zfinder.constituents()[0].momentum();
 FourMomentum l2 = zfinder.constituents()[1].momentum();


 // plot Z pT, Z mass, Z eta
 double Zpt    = zfinder.bosons()[0].momentum().pT()/GeV;

cout << "Zpt" << Zpt << endl;

 _hZpt->fill(Zpt, weight);
```

Walkthrough

```
//Then jets

Jets jets;

foreach(const Jet& jet, applyProjection<FastJets>(event, "jets").jetsByPt(20*GeV)) {
  FourMomentum jmom = jet.momentum();
  if (jmom.absrap() < 4.4 && deltaR(l1, jmom) > 0.5  && deltaR(l2, jmom) > 0.5) {
  jets.push_back(jet);
  }
 }

 double jet_mult = jets.size();

//cout << " Number of jets: " << jet_mult << endl;
_hjets->fill(jet_mult, weight);

 if(jet_mult >0){

 double pt_jetlead   = jets[0].pT()/GeV;

 _hleadjetpt -> fill(pt_jetlead, weight);

//double phi12 = Zphi - phi_jetlead;
 //for (; std::fabs(phi12) > M_PI; phi12 += (phi12 > 0. ? -2.*M_PI : 2.*M_PI) );
 //hdphi12 -> fill(phi12, weight);
```

Walkthrough

```
// Finalize: this would contain normalizing the histograms.
   void finalize() {
   normalize(_hZpt);
   normalize(_hjets);
   normalize(_hleadjetpt);
   normalize(_hdphi12);
   }

private:
        Histo1DPtr _hZpt, _hjets, _hleadjetpt, _hdphi12;
 };


 // The hook for the plugin system
 DECLARE_RIVET_PLUGIN(MC_MyAna);
```

Walkthrough

```
// Finalize: this would contain normalizing the histograms.
   void finalize() {
   normalize(_hZpt);
   normalize(_hjets);
   normalize(_hleadjetpt);
   normalize(_hdphi12);
   }  _h->scale(crossSection()/femtobarn*20.3/sumOfWeights());

private:
        Histo1DPtr _hZpt, _hjets, _hleadjetpt, _hdphi12;
 };


 // The hook for the plugin system
 DECLARE_RIVET_PLUGIN(MC_MyAna);
```

MC_MyAna

- Compile by: rivet-buildplugin RivetMC_MyAna.so MC_MyAna.cc
- export RIVET_ANALYSIS_PATH=$PWD (or use —pwd switch)
- Run 5000 events, like before
- Now add some plots: Z mass, Z eta
- Get the Z and leading jet phi, uncomment the phi difference plot
- Recompile
- Run again, make sure your new plots are filled!

FIFO

- HepMC files tend to become unmanageably large (5000 events ~ 1 GB)
- Often times, we need millions of events
- We use fifo (file in, file out), which is like a pipe. One event enters, gets processed, only then the second event is generated ...
- Look at Run.sh file (we will *run* that later)

Fifo Script

```
#Simple script to run pythia8 and rivet together via a fifo

export RIVET_ANALYSIS_PATH=$PWD  # Rivet needs to know where the analysis is
export RIVET_REF_PATH=$PWD
rm -rf my.hepmc                  # just a protection
mkfifo my.hepmc                  # create the fifo file

rm *.log                         # make sure log files are new

./main42 main42.cmnd my.hepmc > pythia.log &     # run Pythia, output goes to the pipe, always good to have a log file

rivet -a MC_MYANA  my.hepmc &> rivet.log           # run Rivet over, input comes from the pipe

rm my.hepmc
```

aMC@NLO

- ... is a NLO ME generator.
- Used where we need better accuracy than LO
- Need interfacing with a PS generator for a *physical final state*.
- Same framework as LO multileg Madgraph generator.
- Details: http://amcatnlo.web.cern.ch/amcatnlo/

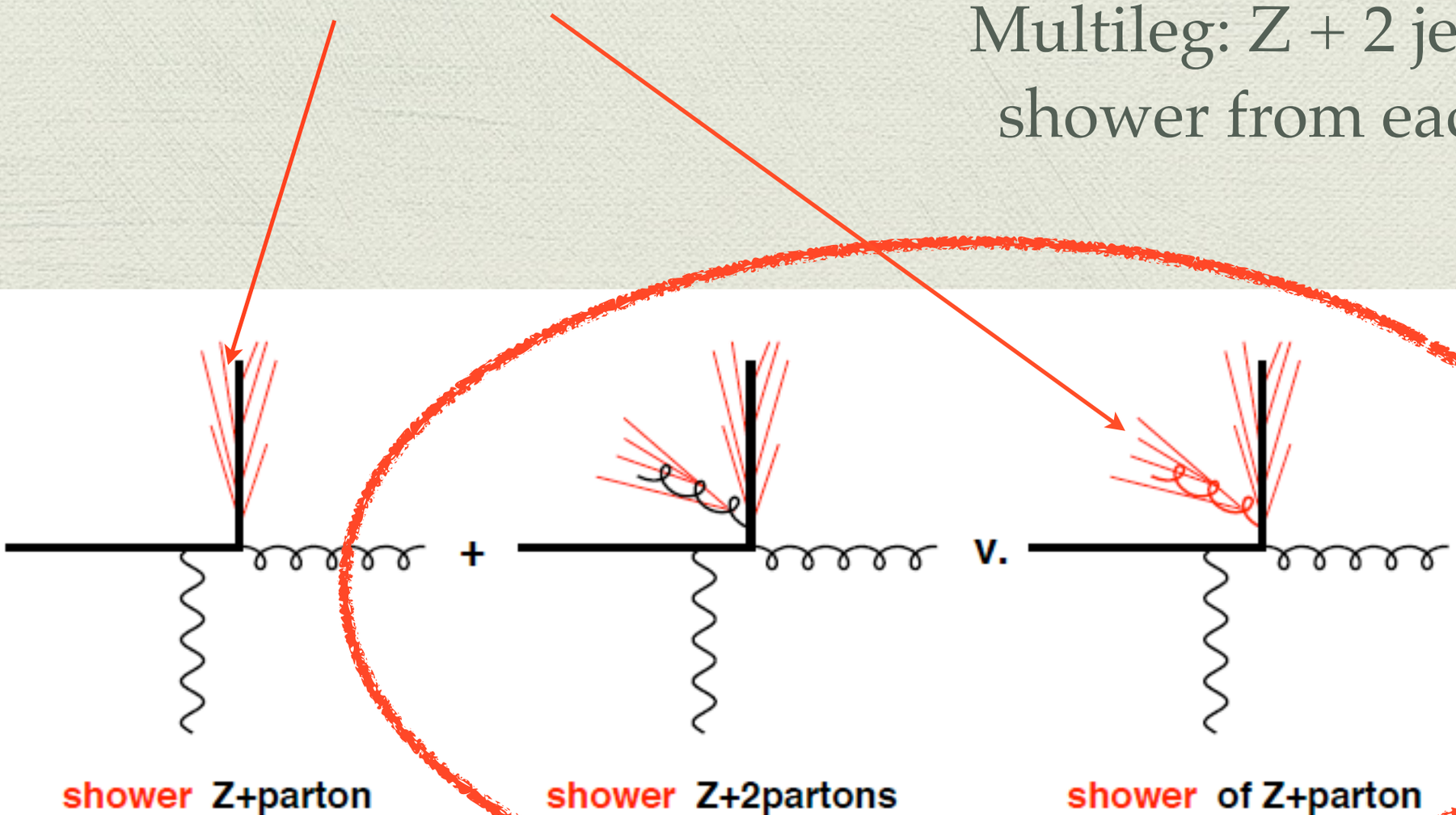
Generating events with aMC@NLO

- Go to tutorial/mc/higgs/MG5_aMC_v2_5_2
- ./bin/mg5_aMC
- generate p p > z [QCD] at 13 TeV
- output run1
- launch run1
- Fix the options (change shower to PYTHIA8 in run_card, number of events, center of mass energy)
- Output in run1/Events/run01
- *Submit a run to generate 20,000 events, as we move on.*

Matching / Merging

PS: Z + 1 jet + shower

Multileg: Z + 2 jets, then shower from each legs



Double Counting!

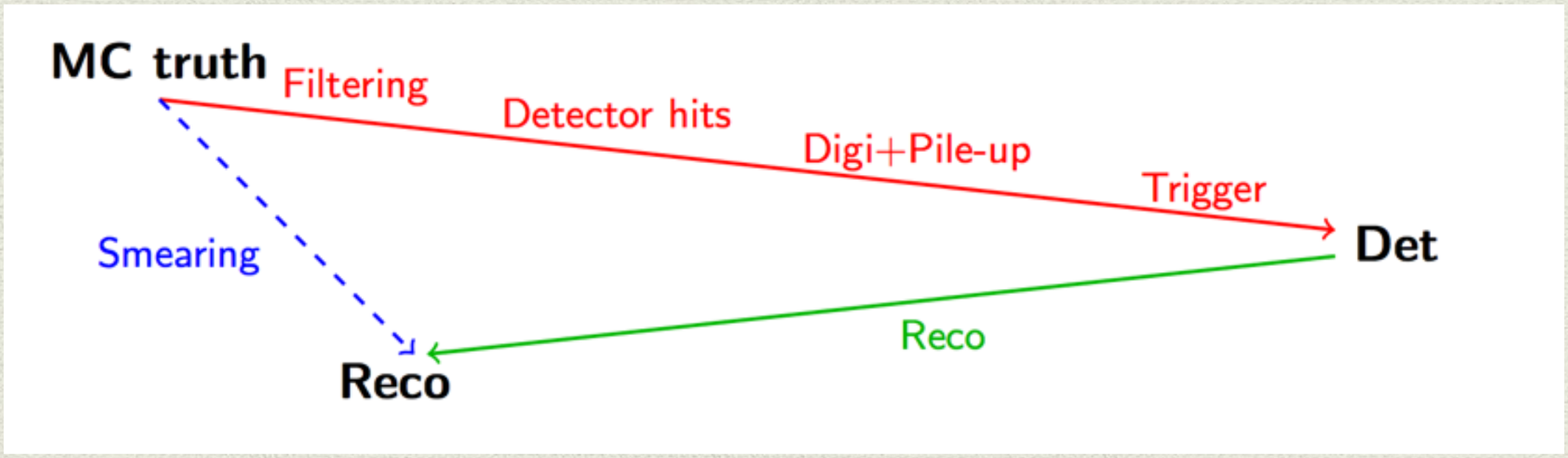
Final Step!

- Lets compare the Pythia8 output to aMC@NLO output
- Generate 20,000 Pythia Z-boson events at 13 TeV, run it via Rivet MC_MyAna analysis, rename the Rivet.yoda file (it will otherwise be overwritten!)
- Move the events.lhe.gz file from aMC@NLO run directory to here, unzip, modify the main42.cmnd to run over the lhe and do the same as above
- Compare by: rivet-mkhtml pythia8.yoda: “Pythia8” amc.yoda: “aMC@NLO”

Congratulations!

New Feature

- For searches, no unfolded data
- Approximate detector response/efficiencies can be made available
- Smearing of final state objects implemented (from v2.5.0)



Thats' it.

Epilogue

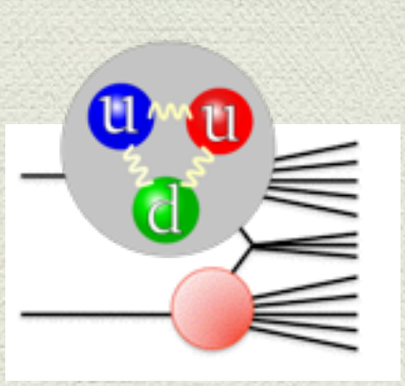
- MCNET (http://www.montecarlonet.org/) organises schools, and short-term (all expense paid) studentships for Ph.D students. Do contact me if interested.
- NRF bursary available for masters/doctoral study in our group at Wits, this year, or next year. Again, just ask :)

Backups

Getting Started

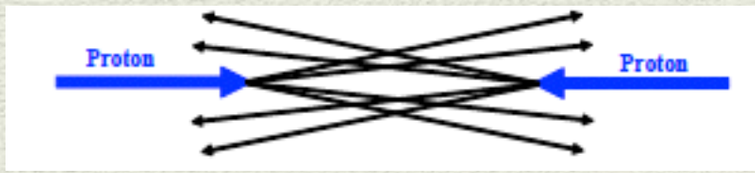
- Super convenient bootstrap script to install (Rivet and all its dependencies) at rivet.hepforge.org
- Source codes of existing analyses serve as useful examples
- Helping the community by adding your analysis to the official library

Hadron-hadron Collision

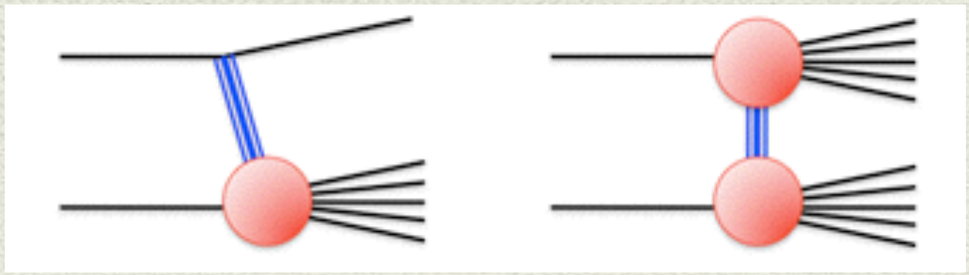


results in

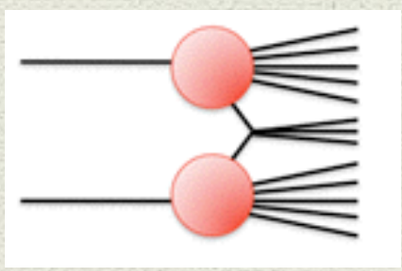
no (or elastic) collision!



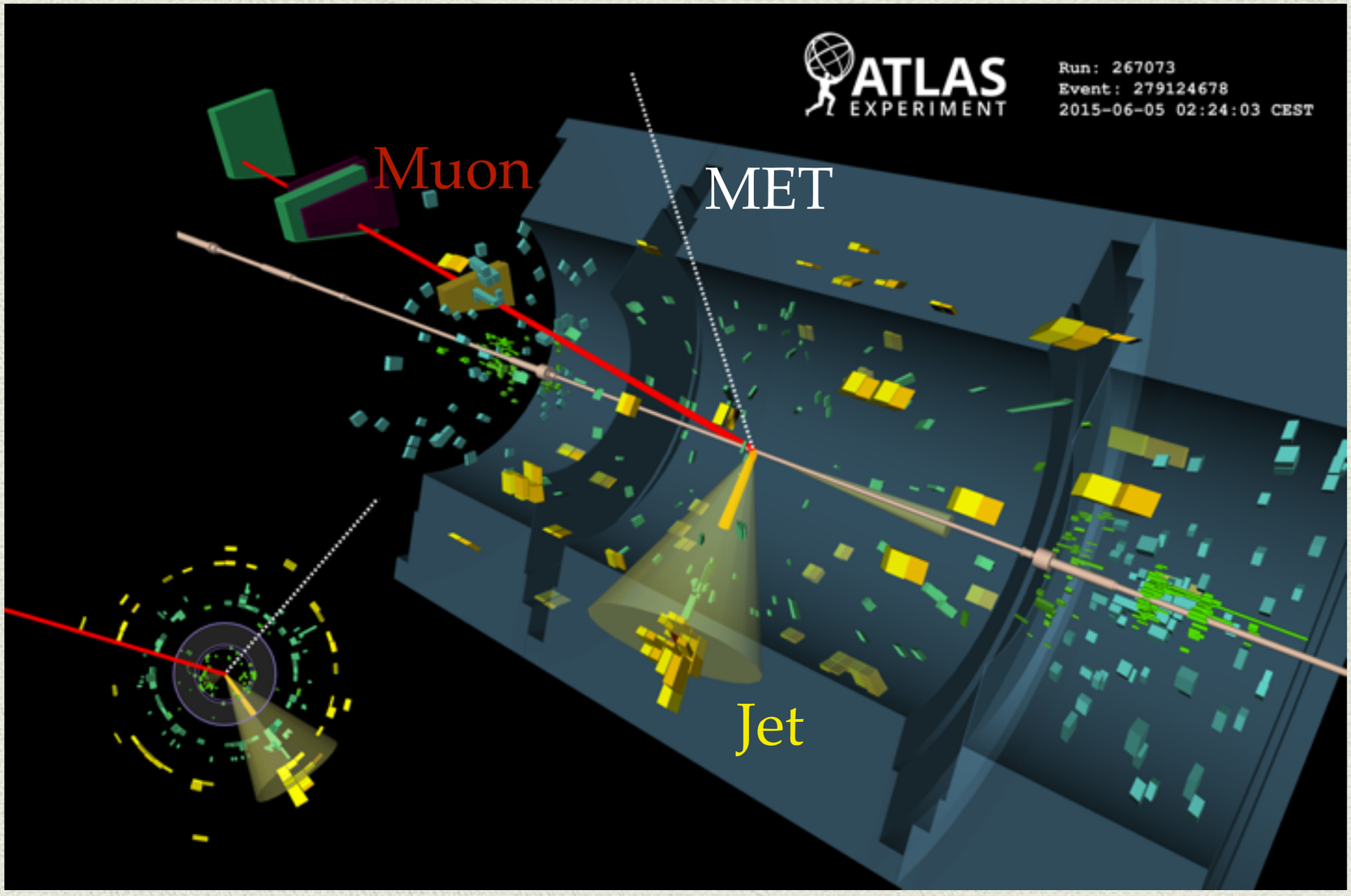
diffractive interaction



interesting stuff!



Objects we see (and dont see)



Also: photons, electrons, charged particles (mostly pions) tracks...

Co-ordinates