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

PYTHIA8

Bootcamp

Part 3

Stephen Mrenna
Fermilab[1]

October 26, 2017

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

Recap

End of last tutorial: downloaded
`http://home.fnal.gov/~mrenna/bootCamp.tgz`

Few tweaks:
`c++98` $\rightarrow$ `c++11` in `Makefile.inc`
Comment out `UncertaintyBands:overSampleISR/FSR` in `bootcamp.cmnd`

```
make main-bootcamp -f Makefile-bootcamp
```

```
rivet-mkhtml out∗.yoda
```

First thing for today

open `bootcamp.cmnd`

change to `Main:numberOfEvents = 10000`

save and close

```
./main-bootcamp bootcamp.cmnd
```

Listen to tutorial while events are running ($\sim$ 2 minutes)

Shower uncertainties in top pair production

Parton showers are a crucial component in a realistic event simulation.

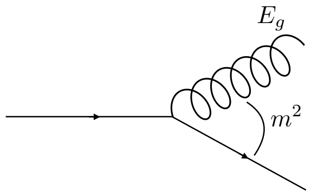
They are approximations to perturbative all-order QCD, which model the structure and evolution of jets of partons.

In this section, we will investigate some of the uncertainties of a parton shower calculation.

In the following, we will consider uncertainties that stem from truncating perturbation theory at some order.

See *“Automated Shower Variations"* in the PYTHIA online manual.

Basics of PS

$P(E_g, m^2) = \Bigg|$  $\Bigg|^2$; $m^2 = 2p_q \cdot p_g = 2E_q E_g (1 - \cos\theta_{qg})$

$$P(\lambda E_g, \lambda^2 m^2)\, d(\lambda E_g)\, d(\lambda^2 m^2) = P(E_g, m^2)\, dE_g\, dm^2 .$$

$$P(E_g, m^2)\, dE_g\, dm^2 = \frac{\alpha_s C_F}{\pi} \frac{dE_g}{E_g} \frac{dm^2}{m^2} .$$

$$z = \frac{E_g}{E_q + E_g} , \qquad 1 - \cos\theta = \frac{m^2}{2E_q E_g} \sim \theta^2$$

$$P(z, \theta^2)\, dz\, d\theta^2 \to \frac{\alpha_s C_F}{\pi} \frac{dz}{z} \frac{d\theta^2}{\theta^2} .$$

Theory Background

Parton showers resum large logarithmic enhancements in pQCD

Depend on:

- The renormalization scale in final-state and initial-state splittings;
- The finite pieces of the splitting functions used for final-state and initial-state splittings.

Different choices tell which piece of the perturbative calculation produces significant uncertainties for particular observables.

Please note that these variations do not span the real envelope of event generator all uncertainties. However, getting a feeling for the perturbative uncertainties is still a useful skill.

Shower variations: Scales

Shower variations are enabled with

```
UncertaintyBands:doVariations = on
```

in your input file `bootcamp.cmnd`.

Consider variations initial-state splittings:

```
UncertaintyBands:List = {
    scale_isr_lo isr:muRfac=0.5,
    scale_isr_hi isr:muRfac=2.0
}
```

The labels `scale_fsr_lo` etc. are arbitrary.

The first line means that PYTHIA will produce an additional event weight that contains the result of evaluating all initial-state splittings (ISR) with $\alpha_s(\frac{1}{2}\mu_F^{PS})$, the second for $\alpha_s(2\mu_F^{PS})$

Now examine your histograms and try to answer

- For which observable do you find the largest final-state shower variation? Why?
- For which observable do you find the largest initial-state shower variation? Why?
- Do the variation bands have features? Can you explain the features?

In Situ Shower Parameter Variation

- Includes renormalisation-scale and non-singular term variations
- Output = vector of alternative weights for each event
- quick estimate of uncertainties without needing separate runs
- a single sample to run through detector simulation etc.
- (hadronisation etc also only has to be carried out once).
- choose which variations you want, how large, correlated/uncorrelated

Shower is iterative selection of branchings:

$$\mathcal{R}_t \in [0,1] \;=\; \Delta(t_0, t) \;=\; \exp\left(-\int_t^{t_0} dt_1 \int dz_1 \, P(t_1, z_1)\right)$$

$P(t,z) = \dfrac{\alpha_s(t)}{2\pi}\dfrac{P(z)}{t}$ is complicated :: use veto algorithm

What is easy and what is hard?

$$r = \int_{z}^{1-z_0} \frac{1+z_p^2}{1-z_p} dz_p$$

is hard to solve for z in terms of r in a 1-to-1 way.

$$r = \int_{z}^{1-z_0} \frac{2}{1-z_p} dz_p > \int_{z}^{1-z_0} \frac{1+z_p^2}{1-z_p} dz_p$$

is easy to solve for z in terms of r.

Oversample and reject.

Understanding the veto algorithm

$$\mathcal{P}_0(t) = \exp\left\{-\int_t^{\bar{t}} g(t')\,\mathrm{d}t'\right\} g(t)\,\underbrace{\frac{f(t)}{g(t)}}_{\mathrm{p_{acc}}} = f(t)\mathrm{e}^{-\int_t^{\bar{t}} g\,\mathrm{d}t'}$$

$$\mathcal{P}_1(t) = \int_t^{\bar{t}} \mathrm{d}t_1 \mathrm{e}^{-\int_{t_1}^{\bar{t}} g\,\mathrm{d}t'} g(t_1)\,\underbrace{\left[1-\frac{f(t_1)}{g(t_1)}\right]}_{\mathrm{p_{rej}}} \mathrm{e}^{-\int_t^{t_1} g\,\mathrm{d}t'} g(t)\,\underbrace{\frac{f(t)}{g(t)}}_{\mathrm{p_{acc}}}$$

$$\mathcal{P}_1(t) = \mathcal{P}_0(t)\int_t^{\bar{t}} \mathrm{d}t_1\,[g(t_1)-f(t_1)]$$

$$\sum_i \mathcal{P}_i(t) = \mathcal{P}_0(t)\exp\left\{\int_t^{\bar{t}}[g(t_1)-f(t_1)]\,\mathrm{d}t_1\right\} = f(t)\mathrm{e}^{-\int_t^{\bar{t}} f\,\mathrm{d}t'}$$

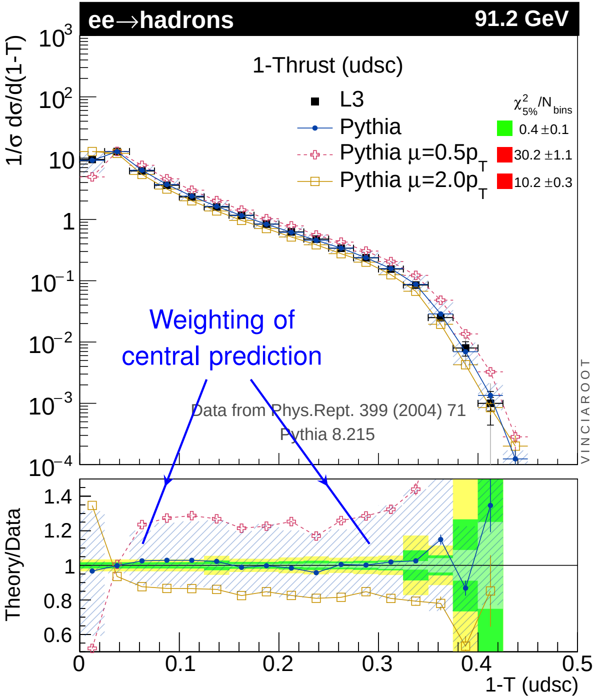
Variations on a theme

$$\underbrace{f \to f'}_{\alpha_s(\mu_R) \to \alpha_s(c\mu_R)} \quad \Rightarrow \mathrm{p}_{\mathrm{acc}}, \mathrm{p}_{\mathrm{rej}} \to \mathrm{p}'_{\mathrm{acc}}, \mathrm{p}'_{\mathrm{rej}}$$

Careful account of weights (ratios of probabilities) allows for uncertainty estimates

Similar methodology can be used to bias (say rare) emissions

Even negative weights can be handled (expected at NLL)


ee→hadrons
91.2 GeV
1-Thrust (udsc)
L3
Pythia
Pythia μ=0.5p_T
Pythia μ=2.0p_T
χ²_5%/N_bins
0.4 ±0.1
30.2 ±1.1
10.2 ±0.3
1/σ dσ/d(1-T)
Theory/Data
1-T (udsc)
Data from Phys.Rept. 399 (2004) 71
Pythia 8.215
VINCIA ROOT
Weighting of central prediction

Additionally, you can change the scales for FSR and check the impact of varying finite pieces of the splitting functions. For this, use

```
UncertaintyBands:List = {
    finitePieces_fsr_lo fsr:cNS=-2.0,
    finitePieces_fsr_hi fsr:cNS=2.0,
    finitePieces_isr_lo isr:cNS=-2.0,
    finitePieces_isr_hi isr:cNS=2.0
}
```

You will receive four additional weights that you can use for histogramming.
Please do this on your own if you choose.

Shower variations: Bonus questions

If you still have time and you are willing to generate a large number of events, it might be interesting to check how your uncertainty estimate changes if you assume different correlations between the variations. For now, we have treated the variations as independent. If you e.g. use the settings

```
UncertaintyBands:List = {
    scale_shower_lo fsr:muRfac=0.5 isr:muRfac=0.5,
    scale_shower_hi fsr:muRfac=2.0 isr:muRfac=2.0
}
```

instead, then PYTHIA will provide only two additional weights. The first of these weight is a combination of “down" variations, while the second gives a combination “up" variations. How does the envelope of these weights differ from your previous results? Would you expect significant differences?

CKKW-L Merging

Here we will experiment with the CKKW-L scheme

The process $\mathrm{W}^{+} + \leq 2$ jets will be taken as an example.

It uses the LHE files

`w+_production_lhc_0.lhe` for $\mathrm{W}^{+} + 0$ partons
`w+_production_lhc_1.lhe` for $\mathrm{W}^{+} + 1$ parton
`w+_production_lhc_2.lhe` for $\mathrm{W}^{+} + 2$ partons

in the `examples` directory to produce a result that simultaneously describes $\mathrm{W}^{+} + 0, 1, 2$ jet observables with leading-order matrix elements, while also including arbitrarily many shower emissions.

Running the CKKW-L example

Please do the following:

```
cp /opt/hep/share/Pythia8/examples/main80.cc mymain80.cc

cp /opt/hep/share/Pythia8/examples/main80.cmnd .

cp /opt/hep/share/Pythia8/examples/w+_production*.lhe .

make mymain80

./mymain80 > mymain80.out
```

More is better

(look at output file)

(look at LHE files)

Download more events from the PYTHIA webpage within VM

Move files to working directory and edit `main80.cmnd`

Run while listening to the tutorial

Theory background

Say we want to study a one-jet observable, e.g. the transverse momentum of the jet *j* in events with *exactly* one jet.

Want to take “hard” jets from the $\mathrm{pp} \rightarrow \mathrm{W}j$ matrix element (ME), while “soft” jets should be modelled by parton-shower (PS) emissions off the $\mathrm{pp} \rightarrow \mathrm{W}$ states

To smoothly merge these two samples, we have to know in which measure “hard” is defined, and which value of this measure separates the hard and soft regions.

`main80.cmnd`, these definitions are

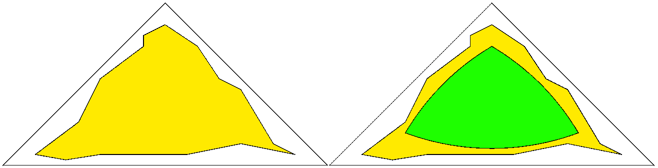
```
Merging:doKTMerging = on
Merging:ktType      = 2
Merging:TMS         = 30.
```

This fixes what we mean when we talk about “hard” and “soft” jets:

Hard jets:	min$\{$any relative $k_\perp$ between sets of partons$\} > t_{\text{MS}}$
Soft jets:	min$\{$any relative $k_\perp$ between sets of partons$\} < t_{\text{MS}}$

We need to remove phase space regions with min$\{$any $k_\perp\} < t_{\text{MS}}$ from the W+1-parton matrix element calculation. Otherwise, there would be an overlap between the “soft jet” and “hard jet” samples.

This requirement means that the merging-scale definition should be implemented as a *cut in the matrix element generator*.



Triangle depicts the whole phase space, with soft or collinear divergences located on the edges

Yellow area symbolises the phase-space region used for the generation of the LHEF events

Green area represents the phase space after PYTHIA 8 has enforced the merging-scale cut on the input events.

Green area has to be fully contained inside the yellow one, i.e. the cut in the ME generator has to be more inclusive than the t_{MS}-cut. For optimal efficiency, the yellow and green areas should be identical (MG5).

```
// Specify merging parameters.
Merging:doKTMerging       = on   ! switch on kT merging
Merging:ktType            = 2    ! Durham kT with pseudorapidity
Merging:TMS               = 30.  ! merging scale, here in kT
Merging:Process           = pp>e+ve  ! process in MadGraph notation
Merging:nJetMax           = 2    ! maximal number of additional jets

--cut--
// Subruns are used to select which LHE file to read when.
// Subrun info should always go at the end of the input file.
// Here Main:subrun = iMerge of pythia.readFile(...) in main80.cc.
Beams:frameType           = 4   ! use LHEF input for incoming beams
!---------------------------------------------
Main:subrun               = 0   ! input for first subrun, W + 0 jets
Beams:LHEF                = w+_production_lhc_0.lhe
!---------------------------------------------
Main:subrun               = 1   ! input for second subrun, W + 1 jet
Beams:LHEF                = w+_production_lhc_1.lhe
!---------------------------------------------
Main:subrun               = 2   ! input for third subrun, W + 2 jets
Beams:LHEF                = w+_production_lhc_2.lhe
!---------------------------------------------
```

```
// Specify merging parameters.
Merging:doKTMerging          = on    ! switch on kT merging
Merging:ktType               = 2     ! Durham kT with pseudorapidity
Merging:TMS                  = 30.   ! merging scale, here in kT
Merging:Process              = pp>e+ve  ! process in MadGraph notation
Merging:nJetMax              = 2     ! maximal number of additional jets

--cut--
// Subruns are used to select which LHE file to read when.
// Subrun info should always go at the end of the input file.
// Here Main:subrun = iMerge of pythia.readFile(...) in main80.cc.
Beams:frameType              = 4    ! use LHEF input for incoming beams
!---------------------------------------------
Main:subrun                  = 0    ! input for first subrun, W + 0 jets
Beams:LHEF                   = w+_production_lhc_0.lhe
!---------------------------------------------
Main:subrun                  = 1    ! input for second subrun, W + 1 jet
Beams:LHEF                   = w+_production_lhc_1.lhe
!---------------------------------------------
Main:subrun                  = 2    ! input for third subrun, W + 2 jets
Beams:LHEF                   = w+_production_lhc_2.lhe
!---------------------------------------------
```

```
// Specify merging parameters.
Merging:doKTMerging         = on    ! switch on kT merging
Merging:ktType              = 2     ! Durham kT with pseudorapidity
Merging:TMS                 = 30.   ! merging scale, here in kT
Merging:Process             = pp>e+ve  ! process in MadGraph notation
Merging:nJetMax             = 2     ! maximal number of additional jets

--cut--
// Subruns are used to select which LHE file to read when.
// Subrun info should always go at the end of the input file.
// Here Main:subrun = iMerge of pythia.readFile(...) in main80.cc.
Beams:frameType             = 4    ! use LHEF input for incoming beams
!---------------------------------------------
Main:subrun                 = 0    ! input for first subrun, W + 0 jets
Beams:LHEF                  = w+_production_lhc_0.lhe
!---------------------------------------------
Main:subrun                 = 1    ! input for second subrun, W + 1 jet
Beams:LHEF                  = w+_production_lhc_1.lhe
!---------------------------------------------
Main:subrun                 = 2    ! input for third subrun, W + 2 jets
Beams:LHEF                  = w+_production_lhc_2.lhe
!---------------------------------------------
```

The Event Loop

Due to the difference in phase-space coverage, a fair fraction of all input events are rejected.

Those events that survive come with a weight

```
      double weight = pythia.info.mergingWeight();
```

which contains Sudakov factors (to remove the double counting between samples of different multiplicity), α_s ratios (to incorporate the α_s running not available in matrix element generators), and ratios of parton distributions (to include variable factorization scales).

This weight *must* be used when filling histogram bins, as is e.g. done by

```
      pTWnow.fill( pTW, weight);
```

for the $p_\perp$ of the W boson. The sum of weights also goes into the calculation of the total generated cross section.

THE END