Tuning event generators to data

Andy Buckley
IPPP, Durham University

for the Rivet and Professor groups
(Durham U., UCL, Lund U., Berlin Humboldt U.)

ACAT08, Erice, 2008-11-04

Overview of tuning and validation

Why… bother? You mean generators aren't perfect?!

How… to make things better?

What… do we find? Can we improve on current tunes?

When… will it be perfect?!

Durham University

MC generators

All generators are based on phenomenological models: colinear parton cascades, string/cluster fragmentation, eikonal MPI, ...

The models have free parameters which are *a priori* unknown: flavour tweaks, shower cutoffs, frag function, intrinsic $k_\perp$, ...

For most experimental users, the important thing is that the MC will describe the data. MC authors / QCD experimentalists want to better understand the generator physics. Not incompatible!

All params can be tuned to best describe the data, even those like α_s which appear God-given! Of course, there are limits, and overtuning may indicate model problems.

Durham University

Validation - the importance of global comparisons

Selective tunings are insufficient: they may

- have no sensitivity to certain params
- allow unchecked distributions to become poor fits to data

Systematic global validation is essential when developing general-purpose tunings

Since event records are largely generator independent, it doesn't make sense to write separate sets of validation analyses for every generator…

Rivet is a C++ validation framework for generators, used to ensure that generators describe a wide range of data and to provide input for tunings (see later). Used by generator specialists and, tentatively, by experiments.

Durham University

The Rivet system

Rivet itself is primarily a library of tools (event shape calculators, jet algorithms, various definitions of a final state, . . .) and a growing collection of analysis routines which use them.

Also some convenience tools, such as the AGILe event generator interfaces and command line interfaces to the libraries.

Usability is a priority:

- Calculation tools ("projections") automatically cache their results
- Histograms can be auto-booked from the reference data files (included)
- User analyses can be loaded at runtime as plugins

Tracker, wiki etc.: **http://projects.hepforge.org/rivet/**

Durham University

Running Rivet

The current easiest way to run rivet is via the **`rivetgun`** command line tool: uses AGILe interfaces to pass generator params, run a variety of generators, write out histos in several formats…

```
rivetgun -g Pythia6:418 -P lep1.params \
  -p "CKIN(3)=100" -a EXAMPLE
```

In the development version, we're breaking the link with AGILe and using HepMC records written to a Unix pipe as our recommended interface.

- Rivet Python module created using SWIG
- **`rivetgun`** replaced by a **`rivet`** Python script — interface already more functional than rivetgun!
- AGILe is still useful in this model for Fortran gens.

Durham University

Rivet history

Good things come to those who wait

The system has matured a lot since ACAT 2007!

- **June 2007: 0.9 release**
  - Basic structure, hard-coded analyses, few examples
- **February 2008: 1.0 release (at last!)**
  - Dynamic runtime loading of analyses
  - More analyses
- **May 2008: 1.1 release**
  - Central projection repository: memory issues simplified/eliminated
  - Atlas interface (by James Monk)
  - Many Tevatron UE analyses added

Durham University

Writing an analysis

Projections registered with a name in the analysis constructor:

```
class MyAnalysis : public Analysis {
  MyAnalysis() {
    setBeams(PROTON, PROTON);
    ChargedFinalState cfs;
    addProjection(cfs, "CFS");
    ...
```

Apply them in the **`analyze`** method via that name:

```
void MyAnalysis::analyze(const Event& evt) {
  ...
  const FinalState& fs =
    applyProjection<FinalState>(evt, "CFS");
  ...
```

Polymorphism, name-clashes, memory management all handled automatically!

Durham University

Rivet projections

A quick selection:

- **Final states:** normal, DIS, "vetoed", charged, hadronic, unstable (for flavour studies)...
- **Event shapes:** thrust, sphericity (regularisable), Parisi C & D params, hemispheres...
- **Jets:** k_T, CDF "track jet", DØ ILC, SISCone, CDF RunII Midpoint (Durham, JADE via FastJet patch)
- **Misc:** jet shapes, primary vertex position, secondary vertices...

...and a few more. Collection mostly complete now.

Durham University

Rivet analyses

We now have too many analyses (over 30) to meaningfully list them all…

LEP: event shapes, fragmentation functions, flavour spectra

Tevatron: lots of CDF analyses, one DØ; W/Z + jets, jet shapes, jet angular correlations, W/Z $p_\perp$ spectra, underlying event analyses in min bias, QCD jets, Drell-Yan at Runs I and II.

HERA: jet energy flow, resolved photoproduction

Want more, particularly HERA multijets, B-factory hadron spectra, and RHIC pp data (good for MPI extrapolation constraint)

Durham University

Generator tuning with Professor

Parameters

We have lots of parameters:

- **PS:** t_{min}, α_s or Λ_{QCD} (really)
- **Hadronisation:** depends strongly on model

  String: string tension σ, Lund symm FF a and b params, baryon suppression, flavour params

  Cluster: constituent masses, flavour params
- **UE:** interaction form factor params (Gaussian width/p(r,h)oton radii), $p_{\perp}^{\mathrm{min}}$, colour reconnection params
- **CKKW** & friends: ME/PS matching scale

Can sometimes be tuned independently: e.g. kinematics, flavour, UE…depending on analyses

Durham University

Tuning methods

Lots of correlated parameters,
200k–10M events per run (kin. binning):
tuning is non-trivial. Too slow for serial
MCMC sampling approaches to be useful:
MC runs are "very expensive functions".

Most tunes: by eye / by grad student. Painful, uninspiring and sub-optimal

Herwig++ 2.1 default tune: brute-force random on Grid + local param grid scan

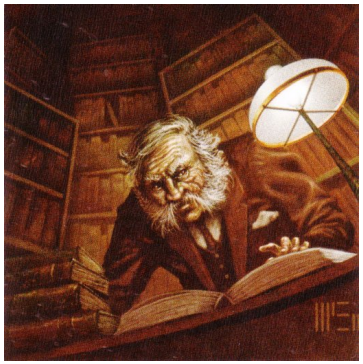
DELPHI: Hamacher et al (1995) quadratic interpolation tune. Scalable, interesting and it works ...

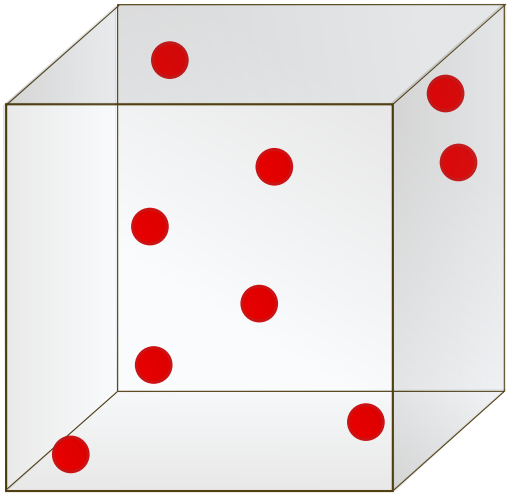
Bear in mind there is some *art* to this...

Durham University

Professor

The Professor tuning project (Durham, Lund, Dresden/Berlin) uses the DELPHI approach in a flexible Python framework:





1. Sample N random MC runs from n-param hypercube
2. For each bin b in each distribution, use the N points to fit an interpolation function using a singular value decomposition.
3. Construct overall χ^2 function and (numerically) minimise
4. Test optimised point by scanning around it in param and lin comb directions

Durham University

Singular value decomposition (SVD)

If we wanted to find the params for an exact multi-dimensional problem, we'd use a matrix inverse. We don't expect the interpolation to be exact $\rightarrow$ Moore-Penrose pseudoinverse

SVD is a deterministic method to compute the pseudoinverse

M is an $m \times n$ matrix. $\exists$ the SVD factorization

$$M = U \Sigma V^*,$$

where U is an $m \times m$ and V a $n \times n$ unitary matrix. The Σ matrix is $m \times n$ and diagonal

Related to eigenvalues — general diagonalisation for all normal matrices. Equivalent to linear (in coeffs) least squares fit

The interpolation function

Obvious interpolation function is the general 2nd order polynomial in n variables:

$$MC_b(\vec{p}) \approx f^{(b)}(\vec{p}) = \alpha_0^{(b)} + \sum_i \beta_i^{(b)} p'_i + \sum_{i \leq j} \gamma_{ij}^{(b)} p'_i p'_j$$

where shifted param vector $\vec{p}\,' \equiv \vec{p} - \vec{p}_0$, with $\vec{p}_0$ chosen as the centre of the param hypercube

Quadratic f is general-purpose and includes correlations between params. Also insensitive to choice of $\vec{p}_0$: SVD is NOT a Taylor series!

Remember that this is used to interpolate the actual MC output per bin — not the overall χ^2

Durham University

SVD in context

What matrix are we trying to invert?

Consider a 2D case in x and y:

$$\underbrace{\begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ & & & \vdots & & \end{pmatrix}}_{M \text{ (sampled param sets)}} \underbrace{\begin{pmatrix} \alpha_0 \\ \beta_x \\ \beta_y \\ \gamma_{xx} \\ \gamma_{xy} \\ \gamma_{yy} \end{pmatrix}}_{C \text{ (coeffs)}} = \underbrace{\begin{pmatrix} v_1 \\ v_2 \\ \vdots \end{pmatrix}}_{V \text{ (values)}}$$

Then $C = \tilde{\mathcal{I}}[M]\, V \quad (\tilde{\mathcal{I}} = \text{pseudoinverse operator})$

Minimum number of samples

P params require at least $N_{\min}^{(P)}$ samples:

$$N_{\min}^{(P)} = 1 + P + P(P+1)/2$$
$$= (2 + 3P + P^2)/2$$

SVD Σ-matrix is square for $N = N_{\min}^{(P)}$

$N > N_{\min}^{(P)}$ is desirable: overconstraint gives a handle on deviations from exact quadratic behaviour

Durham University

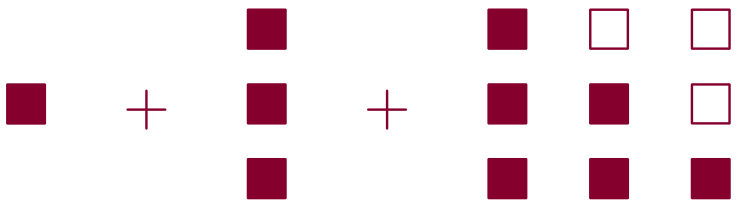
Minimum number of samples

Num params, P	Min num samples, $N_{\text{min}}^{(P)}$
1	3

Minimum number of samples

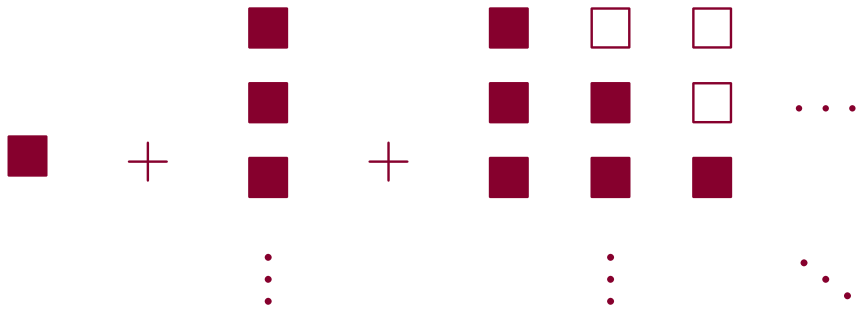
Num params, P	Min num samples, $N_{\text{min}}^{(P)}$
1	3
2	6

Minimum number of samples



Num params, P	Min num samples, $N_{\text{min}}^{(P)}$
1	3
2	6
3	10

Minimum number of samples



Num params, P	Min num samples, $N_{\text{min}}^{(P)}$
1	3
2	6
3	10
4	15
5	21
$\vdots$	$\vdots$

Goodness of fit

Using the interpolation we can predict the MC output for any set of parameters very fast. This prediction can be fitted to data, minimising the weighted χ^2:

$$\chi^2(\vec{p}) = \sum_{\text{observables}} w_{\mathcal{O}} \sum_{\text{bins}} \frac{\left(X_{\text{data}} - X_{\text{MC}}(\vec{p})\right)^2}{\sigma^2_{\text{data}} + \sigma^2_{\text{MC}}}$$

Include all the relevant data distributions in the fit! Don't assign too much *meaning* to our χ^2/N_{DoF}!

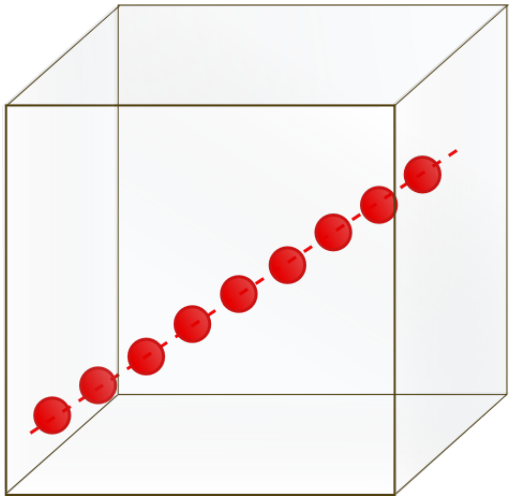
This fit only takes seconds or minutes (as compared to days or weeks for a brute force approach).

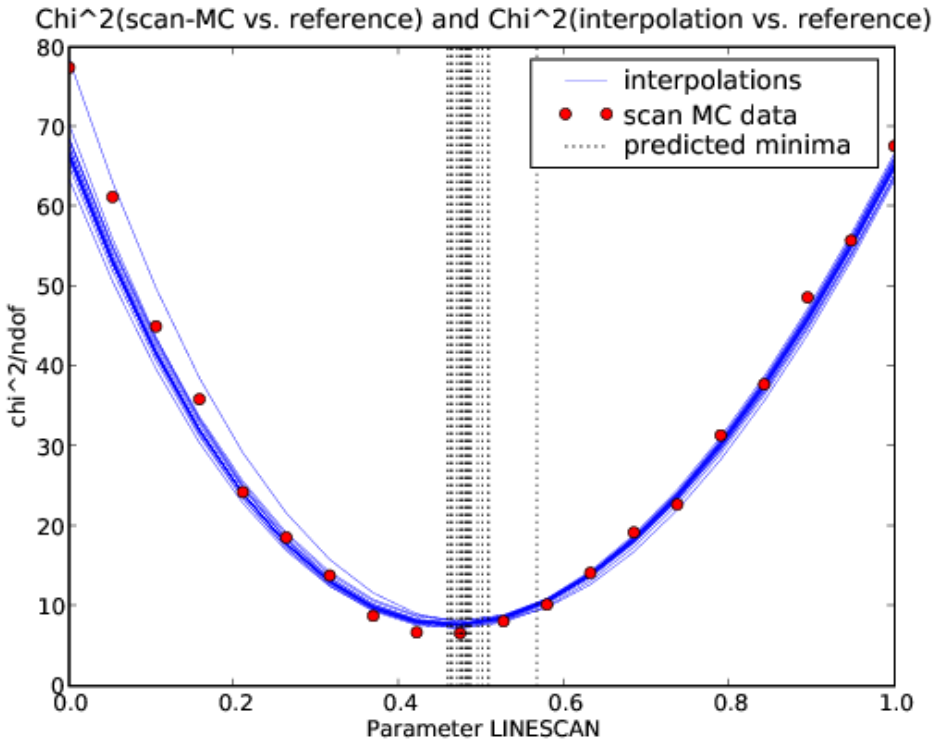
Choosing the relative weights is an art (of sorts)! Not completely objective.

Durham University

Verifying the interpolation

A line scan of χ^2/N_{DoF} for FPythia vs DELPHI event shapes through 3D param space (σ_{string}, Λ_{QCD}, Lund a) around a Professor-predicted minimum:





Comparable success for 5–10 params, different observable sets.

Tune comparisons

Pythia 6 – new tunings: LEP

Two-stage tune of Pythia 6 to LEP/SLD data:

- Flavour parameters. Tuned to identified particle multiplicities, normalized to pions.
- Fragmentation, hadronization. Tuned to event shapes, b-fragmentation measurement, multiplicities, momentum spectra.

Improvement of many identified particle multiplicities and event shapes.

NB: After tuning to LEP data even the agreement with Tevatron data has improved!

Durham University

Pythia 6 – new tunings: LEP

Flavour parameters:

	default	tuned	
PARJ(1)	0.1	0.073	di-quark suppression
PARJ(2)	0.3	0.2	strange suppression
PARJ(3)	0.4	0.94	strange di-quark suppression
PARJ(4)	0.05	0.032	spin-1 di-quark suppression
PARJ(11)	0.5	0.31	spin-1 light meson
PARJ(12)	0.6	0.4	spin-1 strange meson
PARJ(13)	0.75	0.54	spin-1 heavy meson
PARJ(25)	1.0	0.63	η suppression
PARJ(26)	0.4	0.12	η' suppression

Durham University

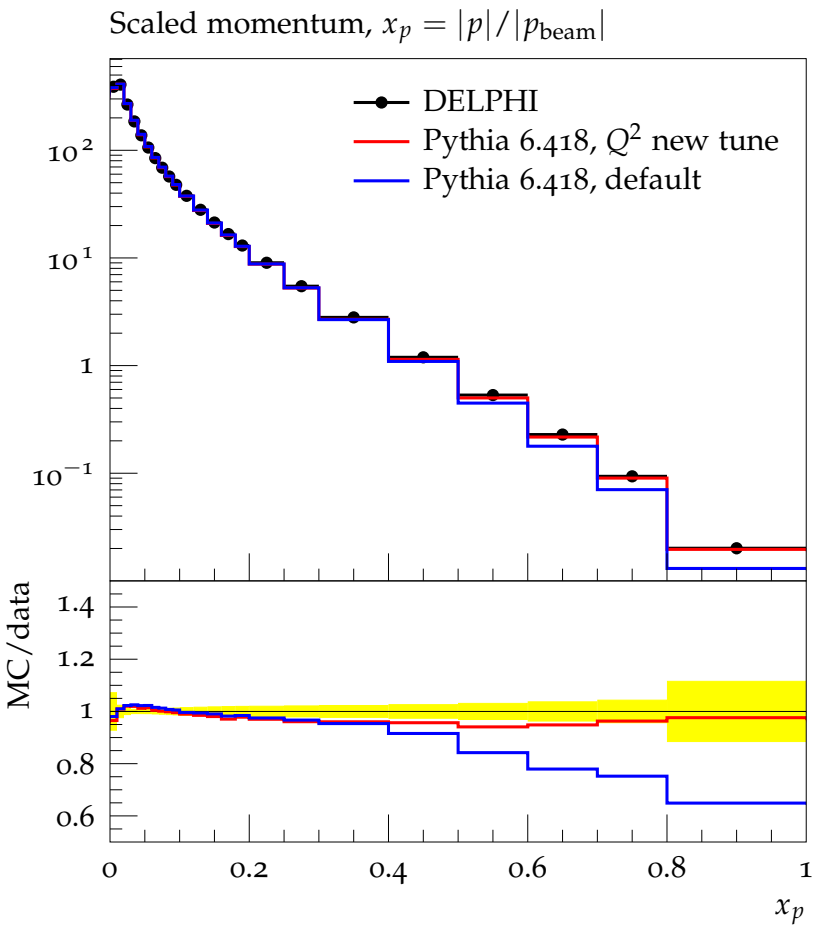
Pythia 6 – new tunings: LEP

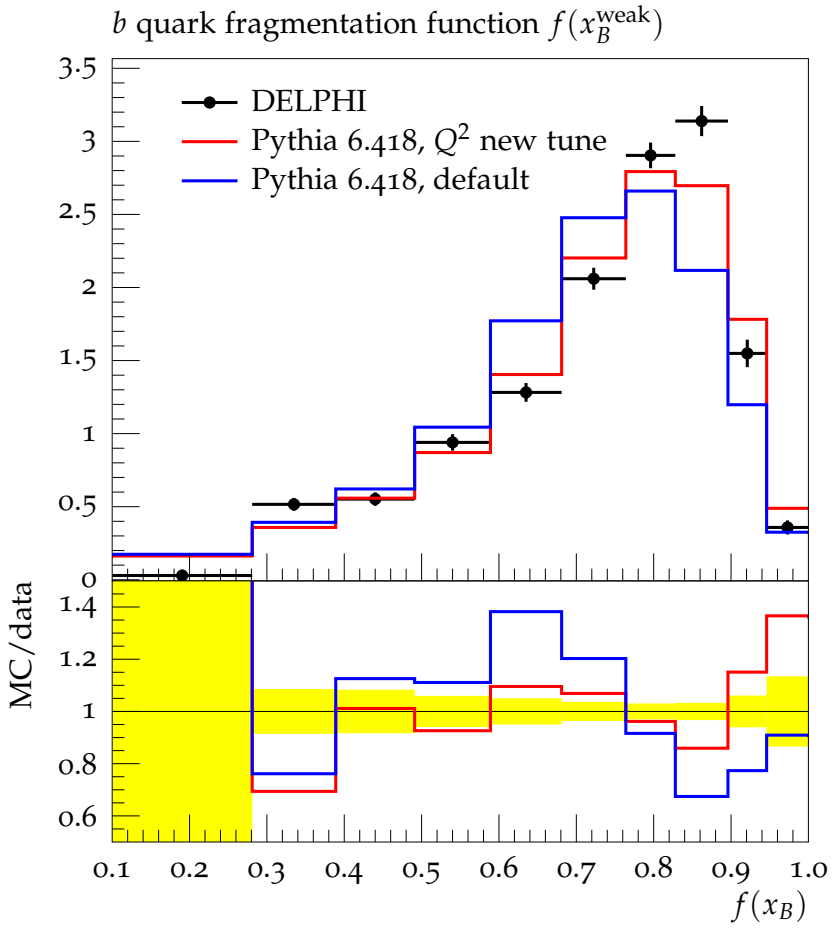
Fragmentation parameters:

	default	Q^2 shower	$p_\perp$ shower	
MSTJ(11)	4	5	5	Frag fn.
PARJ(21)	0.36	0.325	0.313	σ_q
PARJ(41)	0.3	0.5	0.49	a
PARJ(42)	0.58	0.6	1.2	b
PARJ(47)	1.0	0.67	1.0	r_b
PARJ(81)	0.29	0.29	0.257	Λ
PARJ(82)	1.0	1.65	0.8	PS cut-off

Durham University

Pythia 6 – new tunings: LEP





Pythia 6 – new tunings: LEP

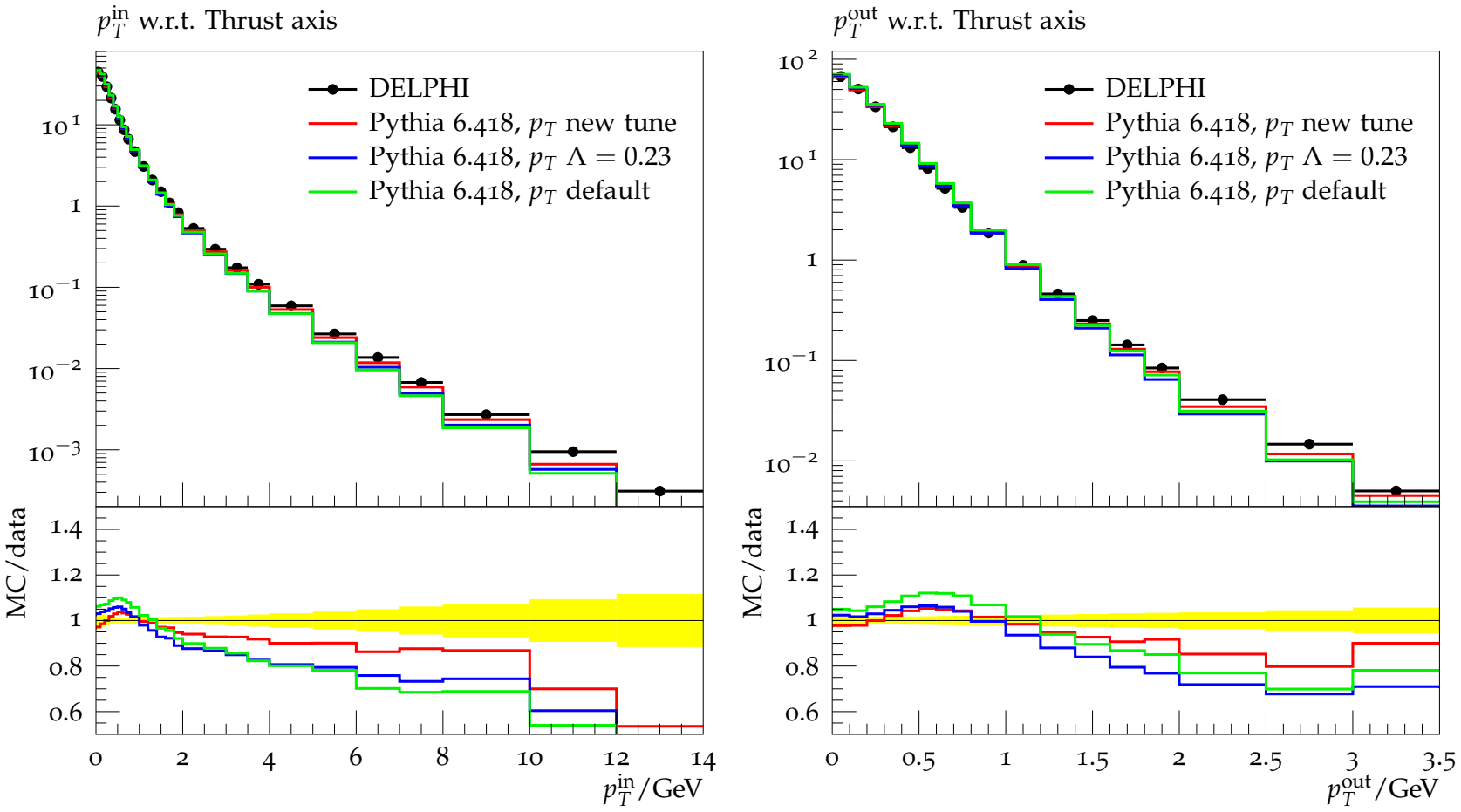
The next slides show the $p_\perp$ ordered shower. Keep in mind:

- DØ and CDF use tune S0, equivalent to the "$\Lambda = 0.23$" curve.
- ATLAS uses Arthur Moraes' tune, equivalent to the "default" curve.

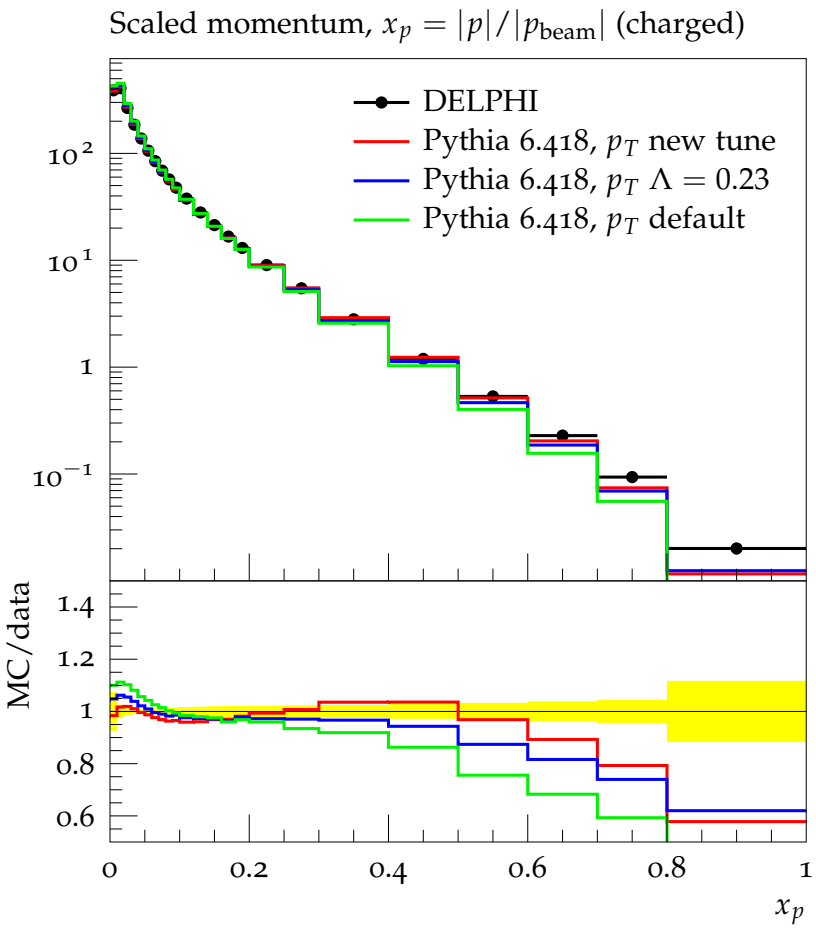
The differences you see *do* affect Tevatron/LHC distributions. Just turning on the $p_\perp$ ordered shower is *not* sufficient!

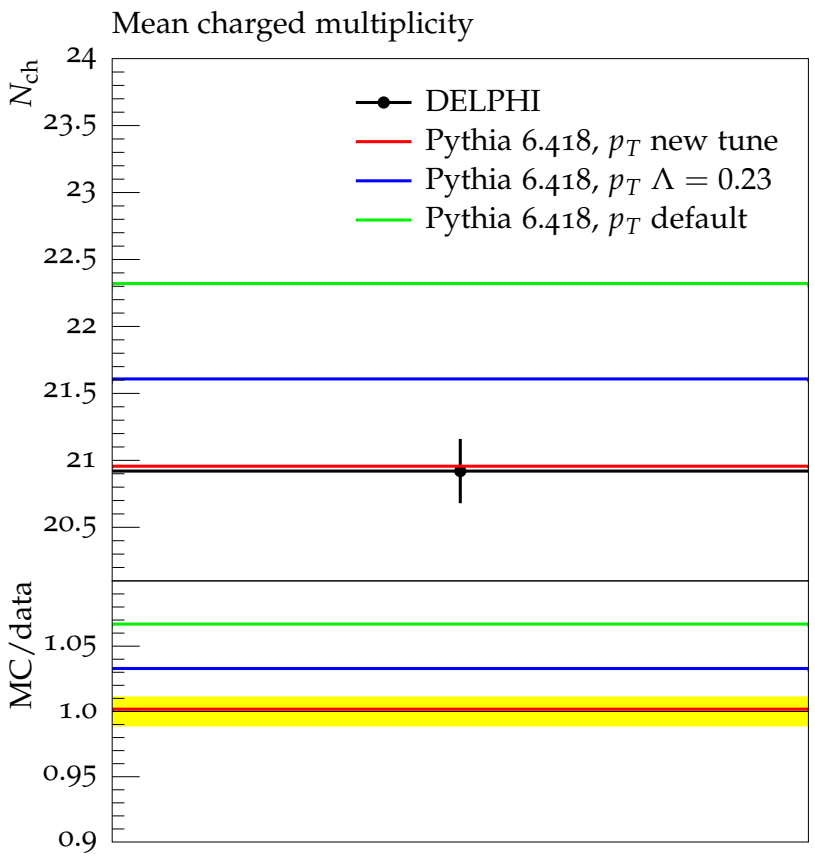
Durham University

Pythia 6 – new tunings: LEP

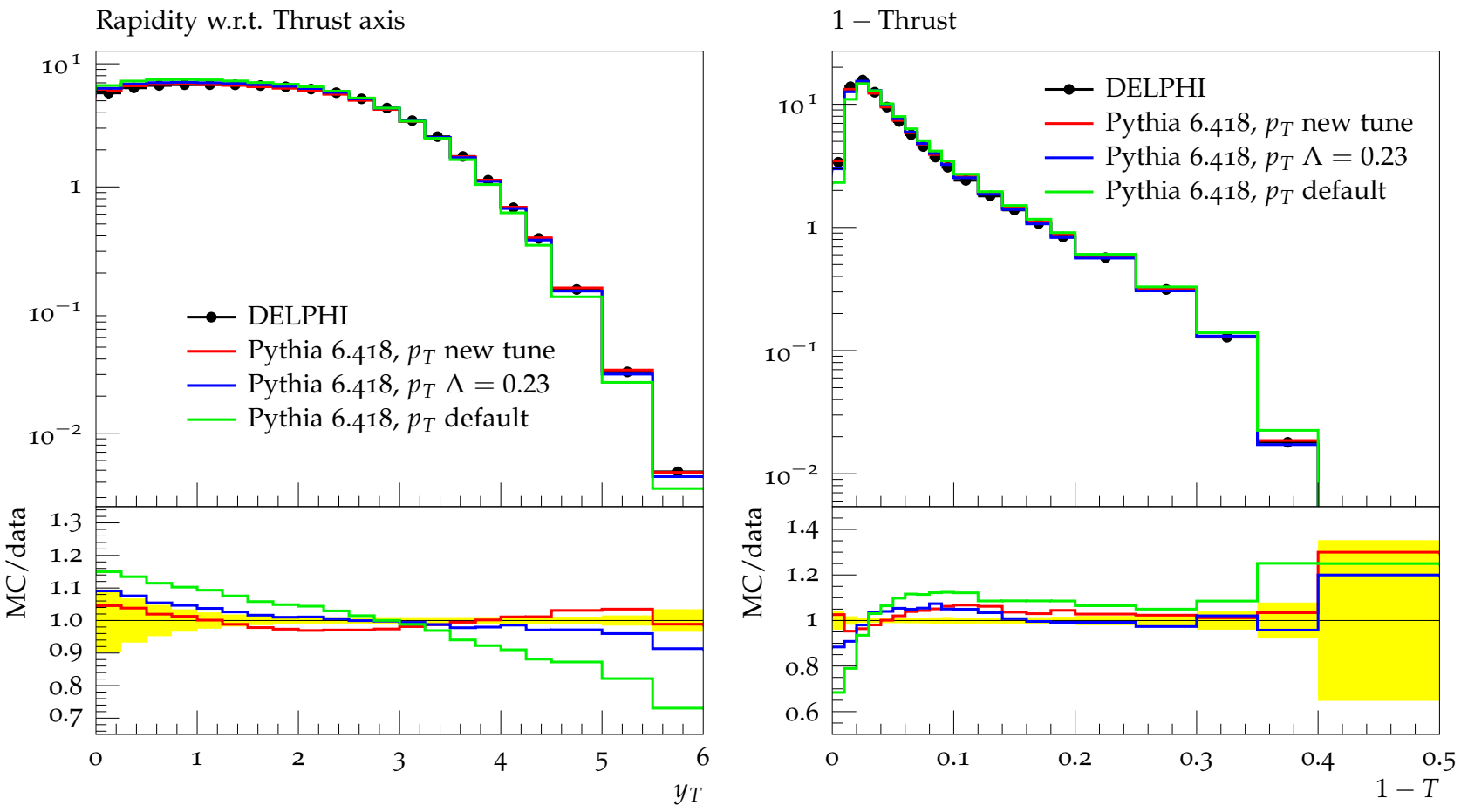


Pythia 6 – new tunings: LEP

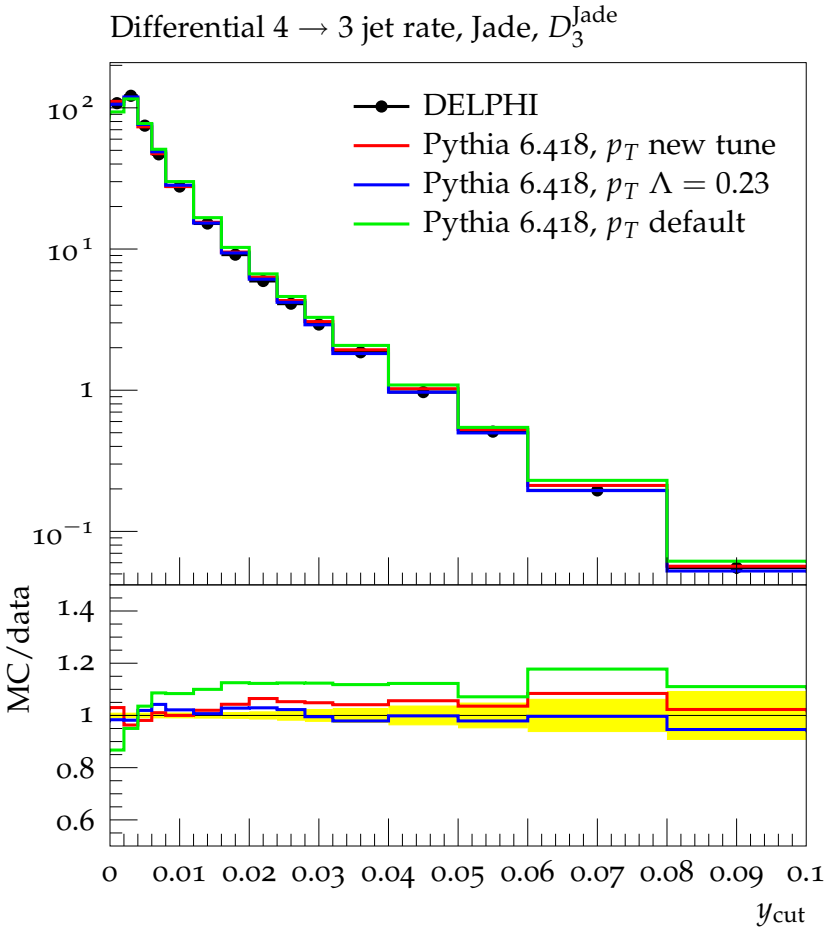


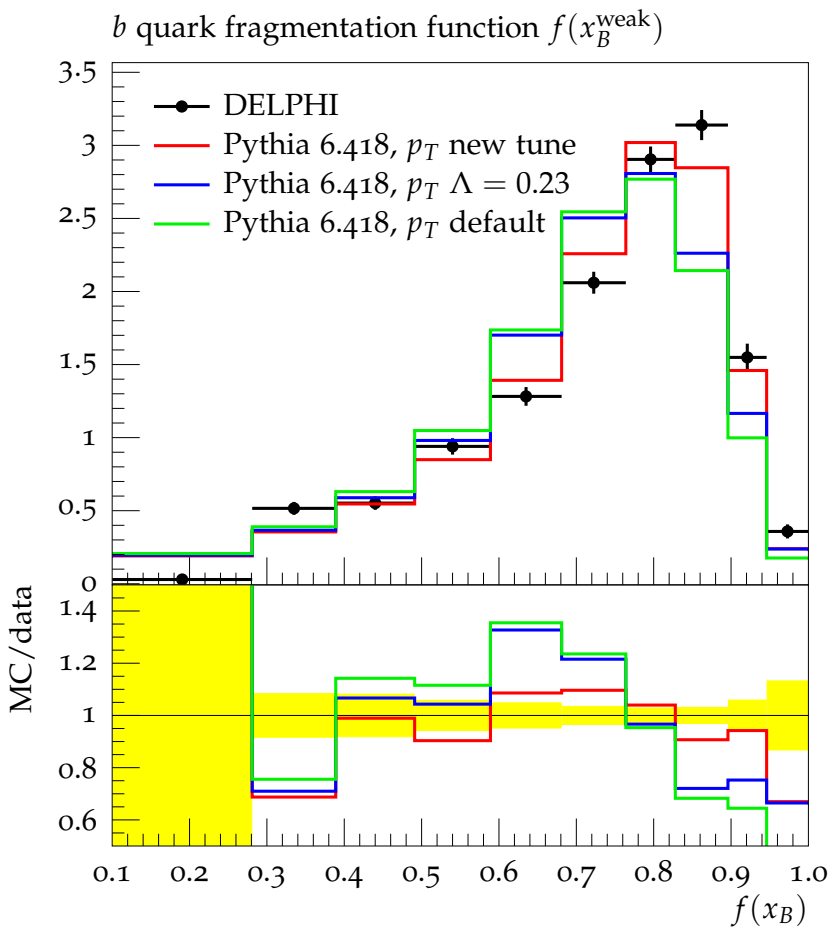


Pythia 6 – New Tunings: LEP



Pythia 6 – New Tunings: LEP





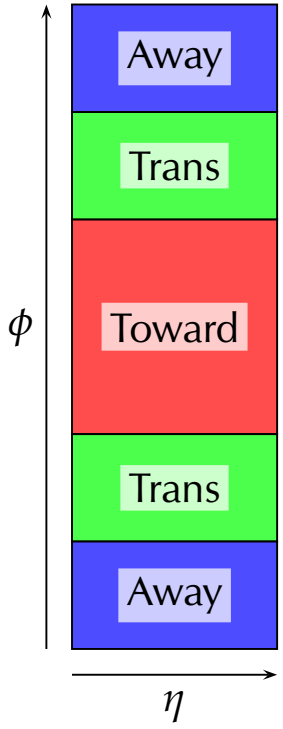
Pythia 6 – new tuning: Tevatron

We have a new UE tune for Pythia 6 with Q^2 ordered shower and old MPI model, based on the LEP tune shown on the last slides.

Using > 50 distributions from CDF and DØ:

- CDF Run-I Z $p_\perp$
- CDF Run-I jets
- CDF Run-II Drell-Yan
- CDF Run-II leading jet
- CDF Run-II $\langle p_\perp \rangle$ vs N_{ch}
- DØ Run-II jet correlations

NB: A UE tune with the $p_\perp$-ordered shower and the new MPI model is in progress.



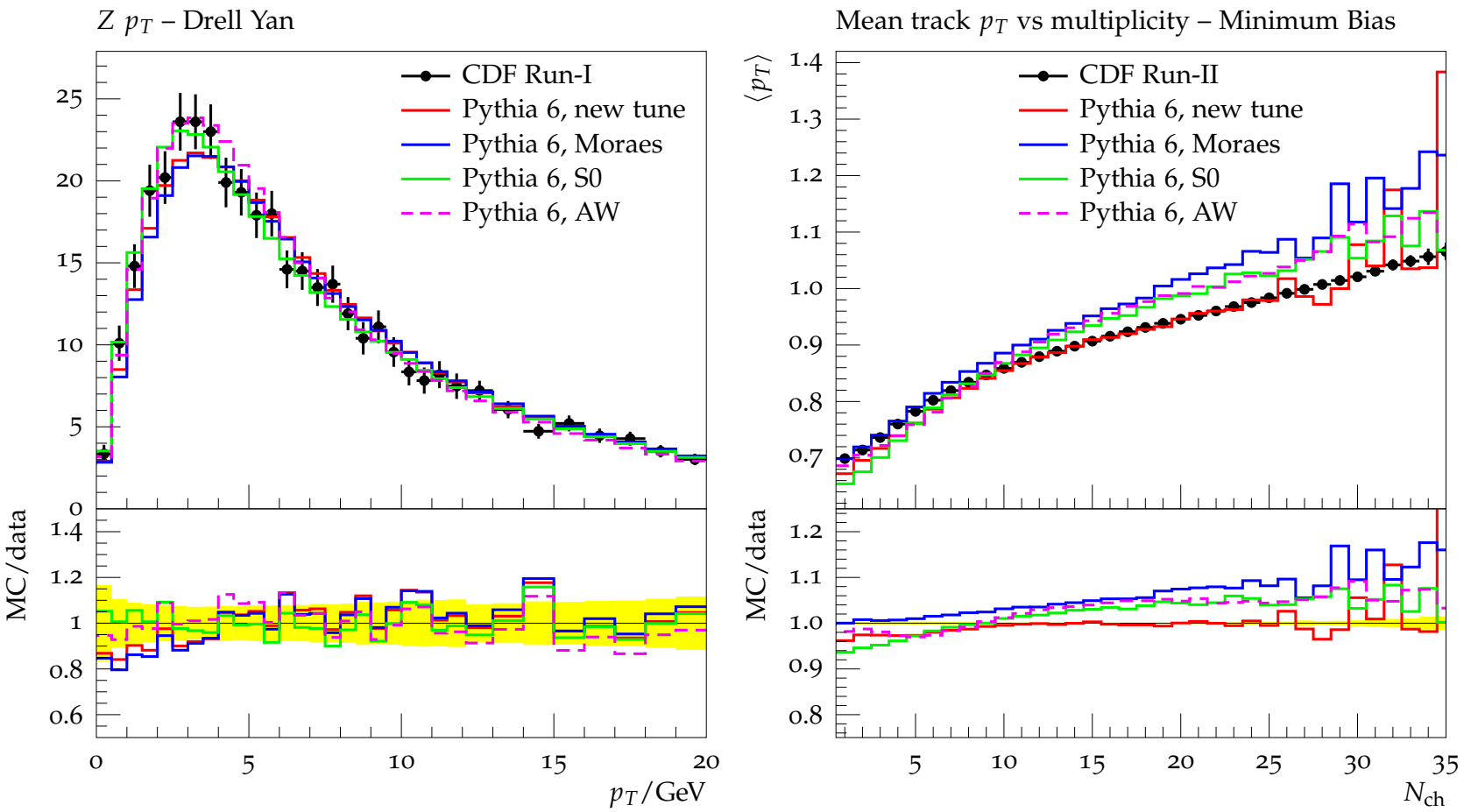
Pythia 6 – New Tuning: Tevatron

	default	tune DW	new tune	
PARP(62)	1.0	1.25	2.97	ISR cut-off
PARP(64)	1.0	0.2	0.12	ISR scale factor for α_s
PARP(67)	4.0	2.5	2.74	max. virtuality
PARP(82)	2.0	1.9	2.1	$p_\perp^0$
PARP(83)	0.5	0.5	0.84	matter distribution
PARP(84)	0.4	0.4	0.5	matter distribution
PARP(85)	0.9	1.0	0.82	colour connection
PARP(86)	0.95	1.0	0.91	colour connection
PARP(90)	0.16	0.25	0.17	$p_\perp^0$ energy evolution
PARP(91)	2.0	2.1	2.0	intrinsic $k_\perp$
PARP(93)	5.0	15.0	5.0	intrinsic $k_\perp$ cut-off

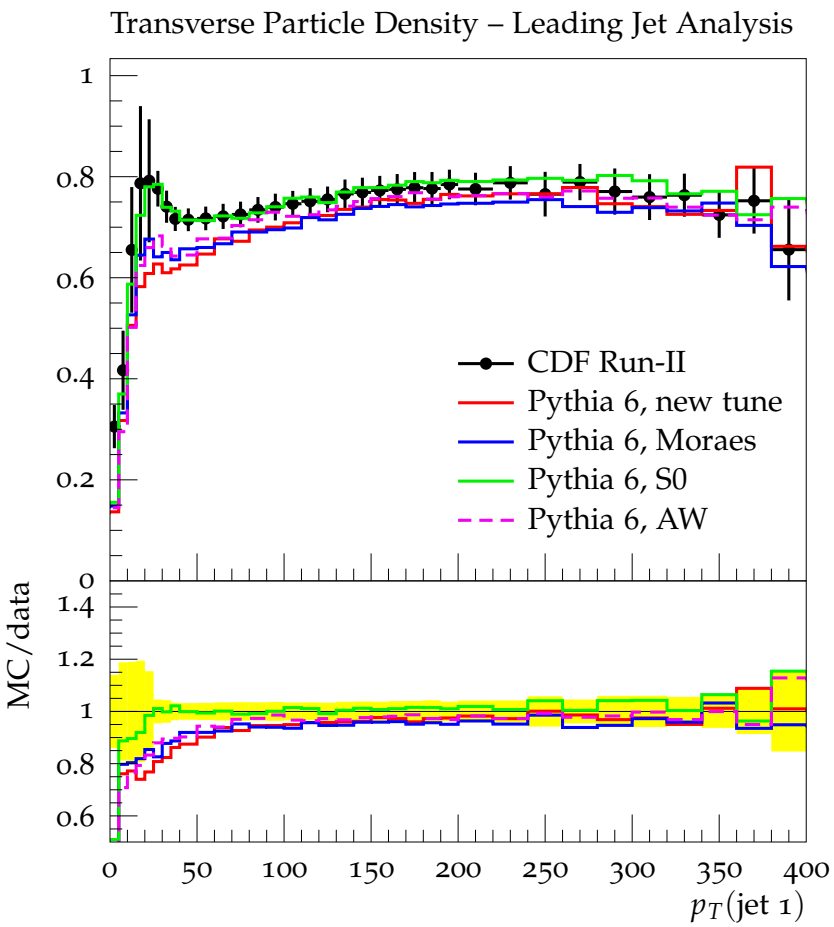
Tune uses LEP tuned parameters as shown before, Q^2 shower.

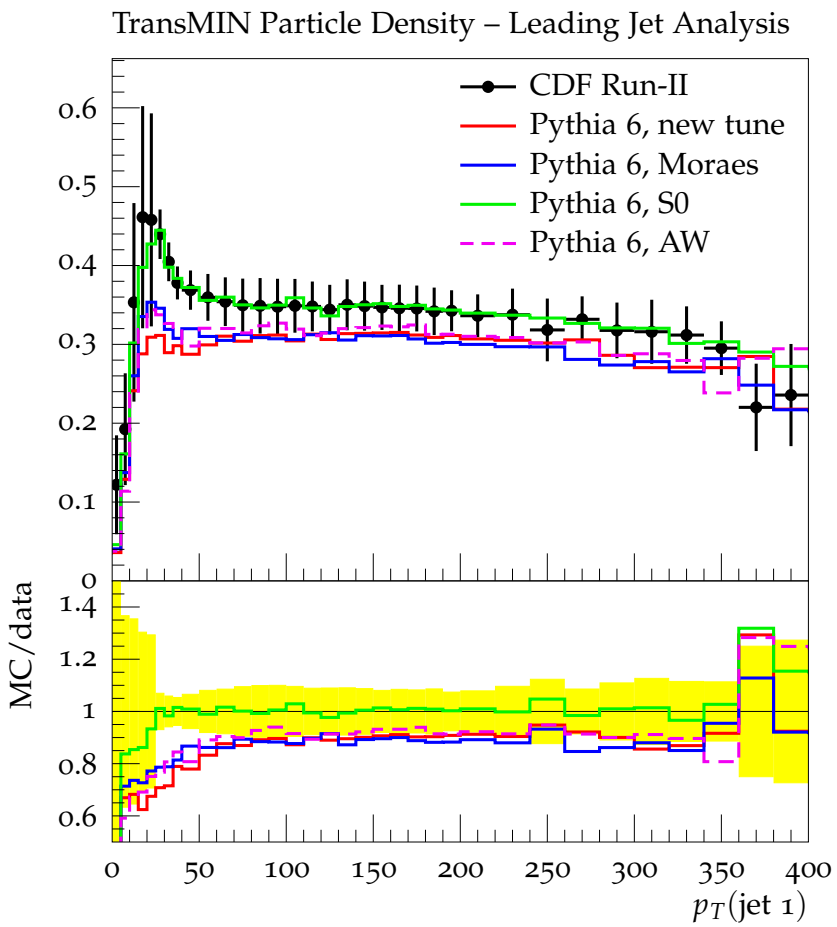
Durham University

Pythia 6 – Tevatron Comparisons

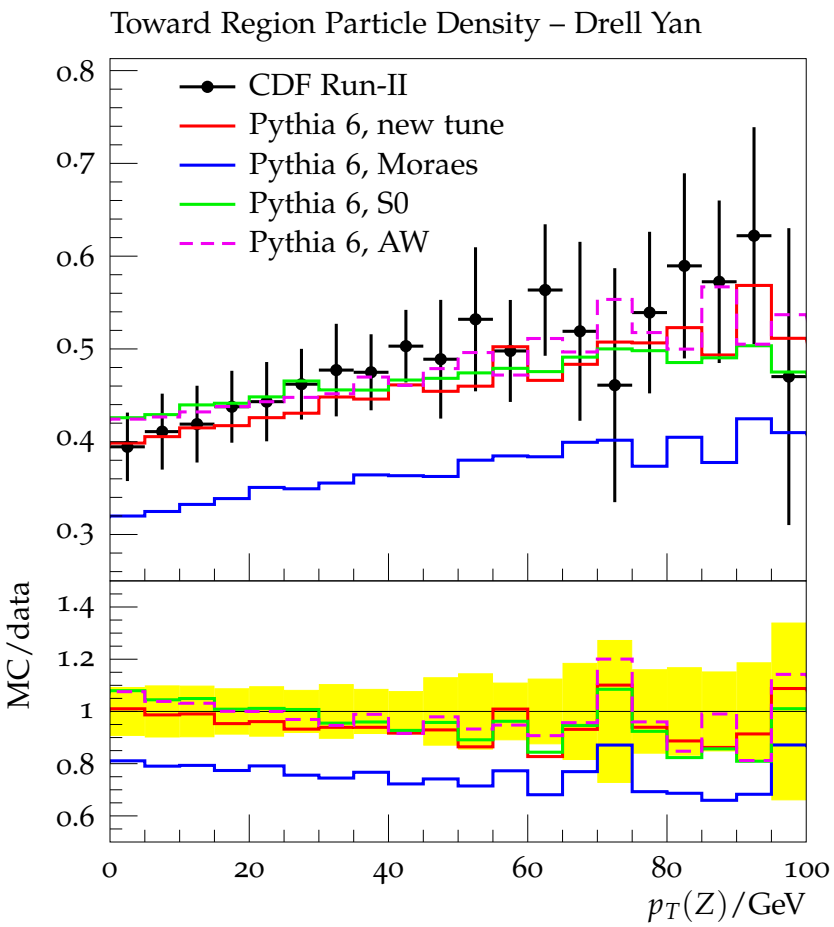


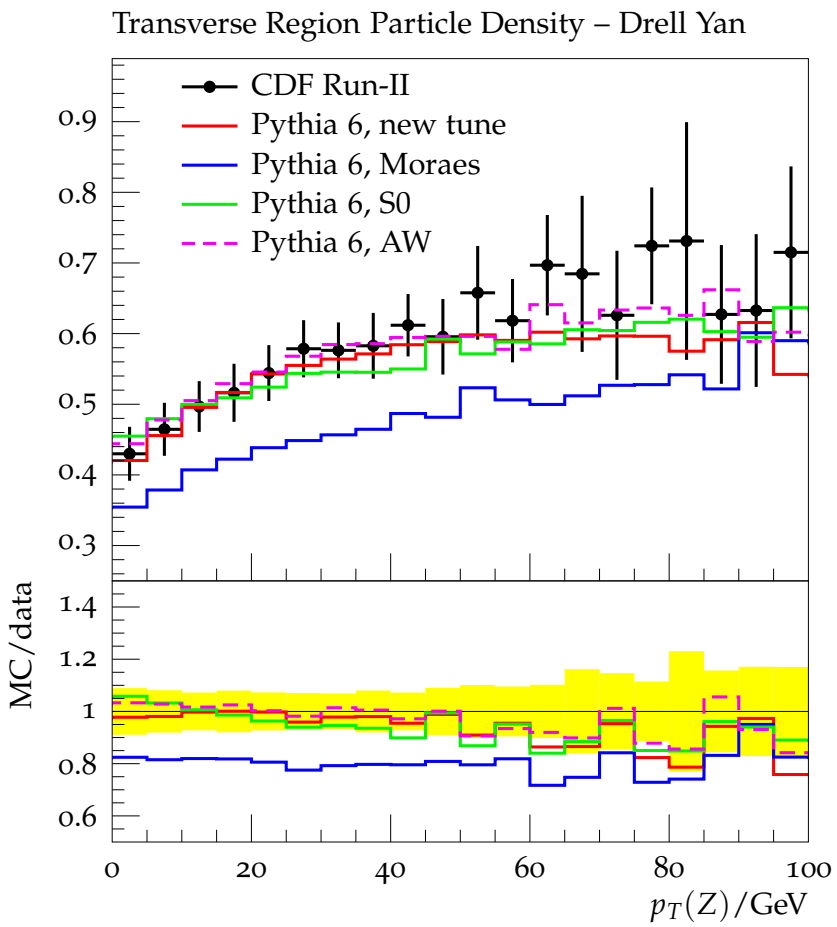
Pythia 6 – Tevatron Comparisons





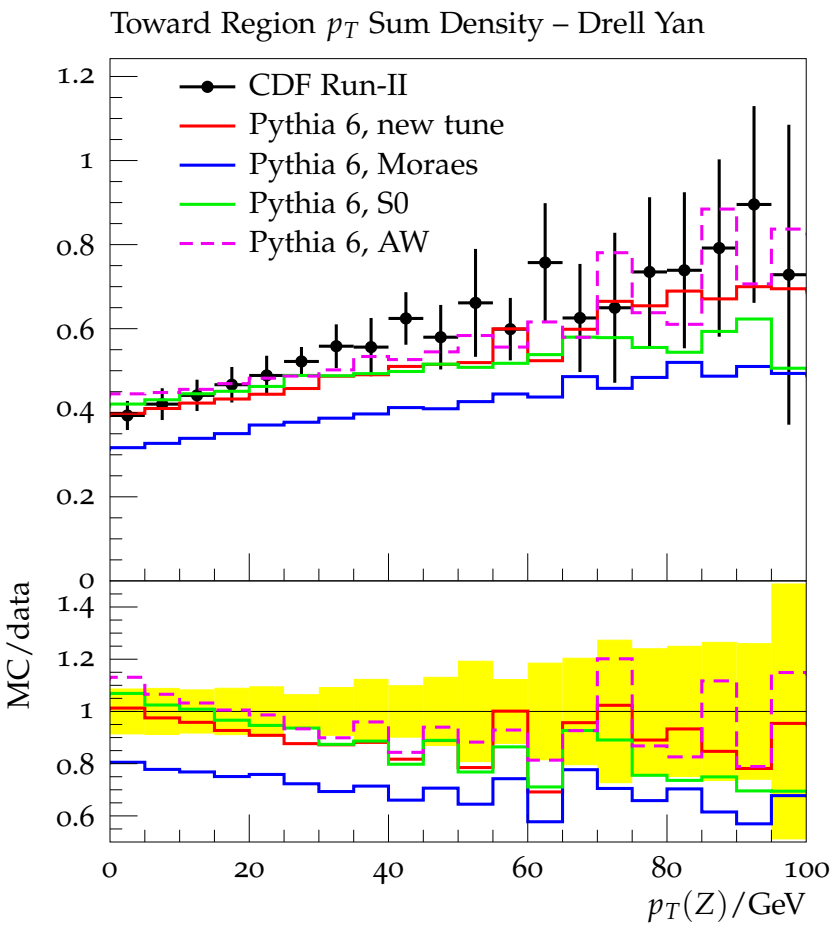
Pythia 6 – Tevatron Comparisons

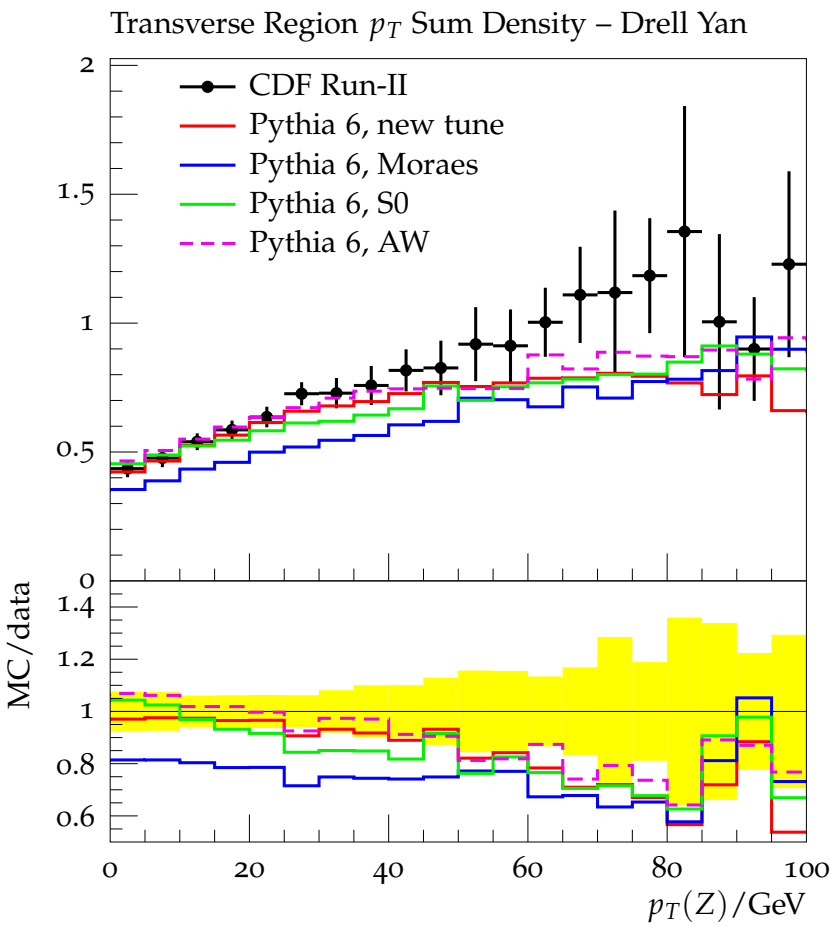




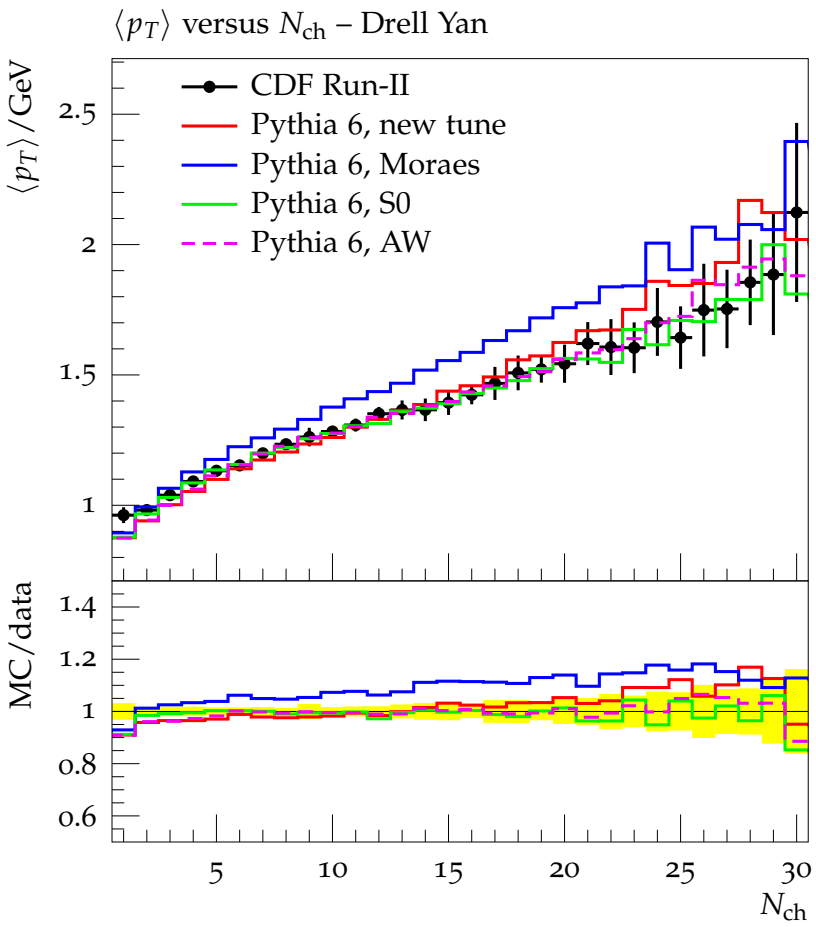
Durham University

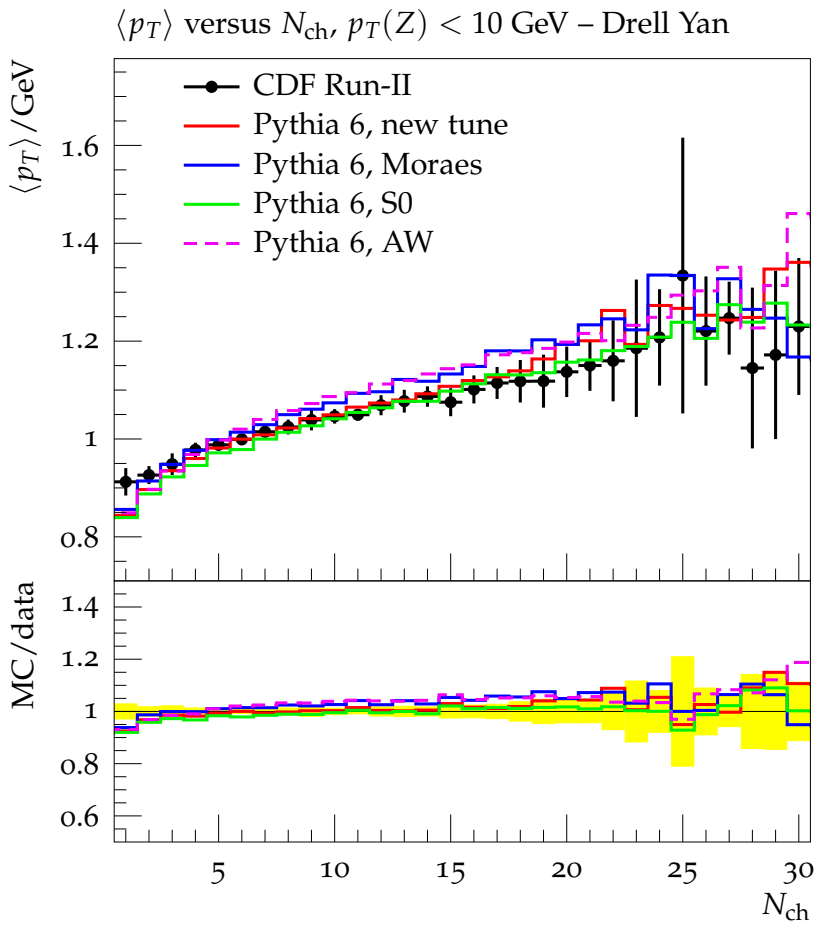
Pythia 6 – Tevatron Comparisons





Pythia 6 – Tevatron Comparisons





Durham University

Comparisons

Fortran Herwig + Jimmy:

- Shown untuned — Atlas tune is better than this.
- Not able to produce min-bias.

Herwig++:

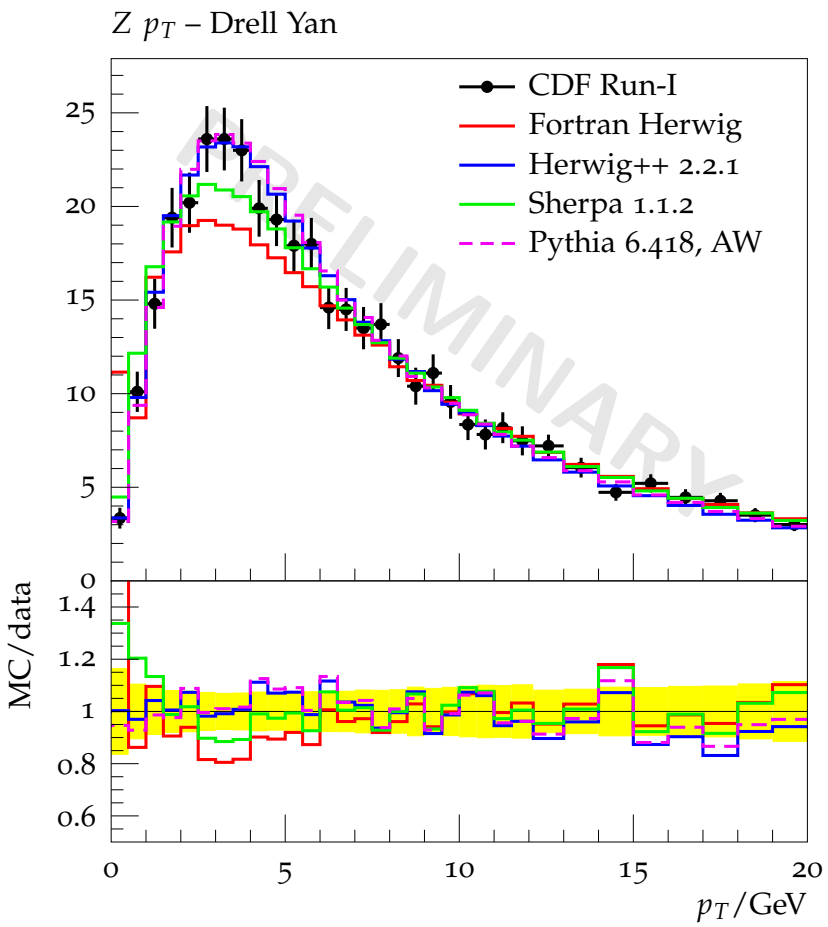
- Stable version 2.2.1, using multiple scattering UE model. Brute-force tuned.
- Not able to produce min-bias in 2.2.1, but 2.3 will include it.

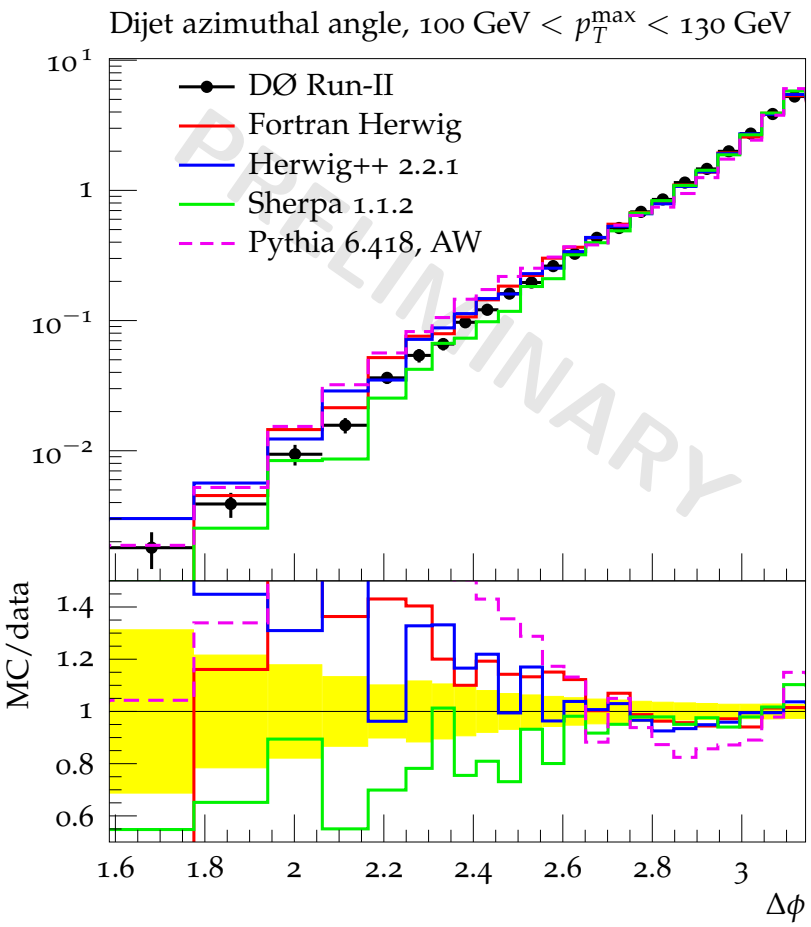
Sherpa:

- Version 1.1.2 shown: QCD plots are $p\overline{p} \rightarrow \leq 3$ jets, DY plots are $\rightarrow e^+e^-(+ \leq 2$ jets) from ME.
- Cluster hadronization untuned; eikonal MPI $\rightarrow$ no min bias.

Durham University

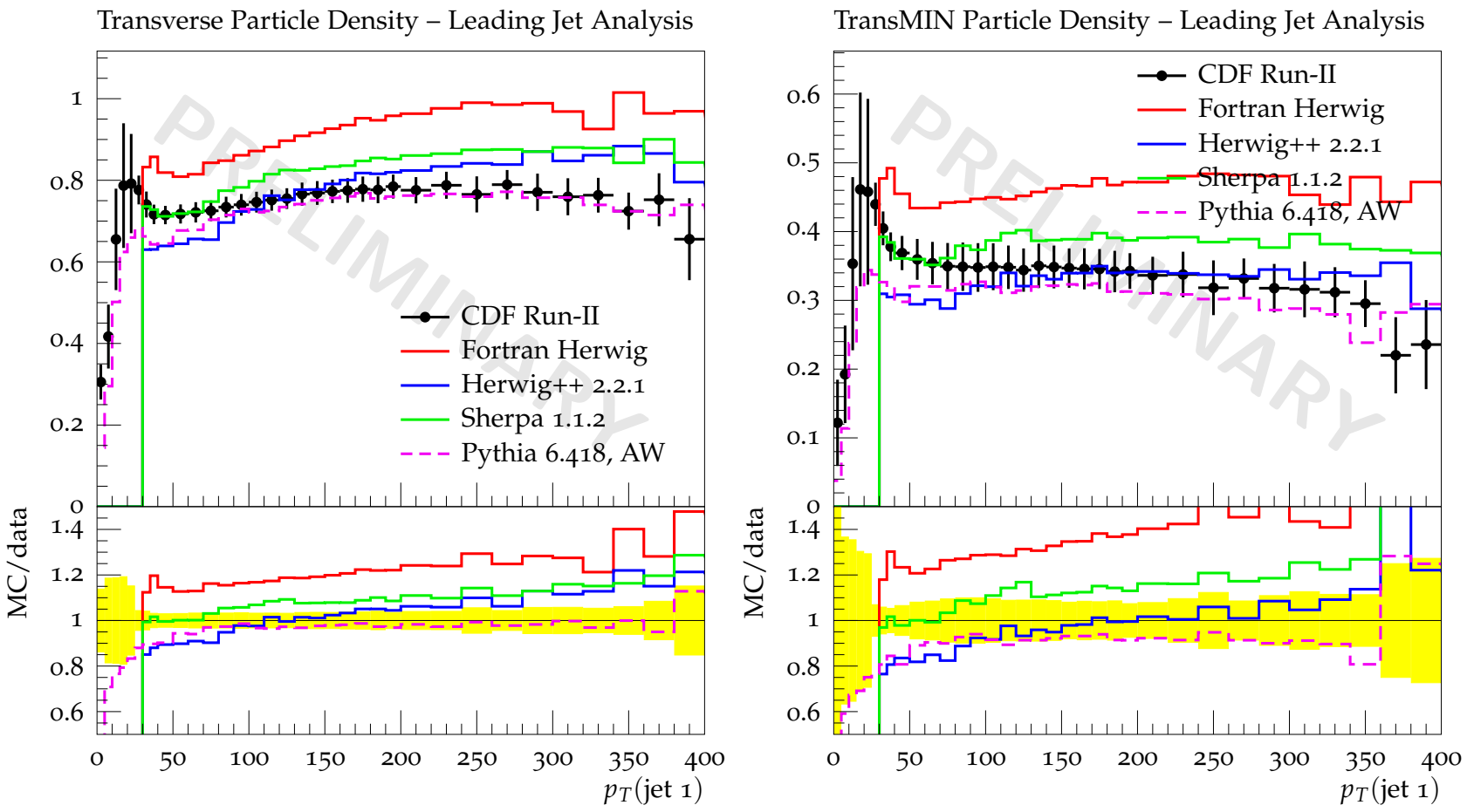
Herwig, Sherpa – Tevatron Comparisons



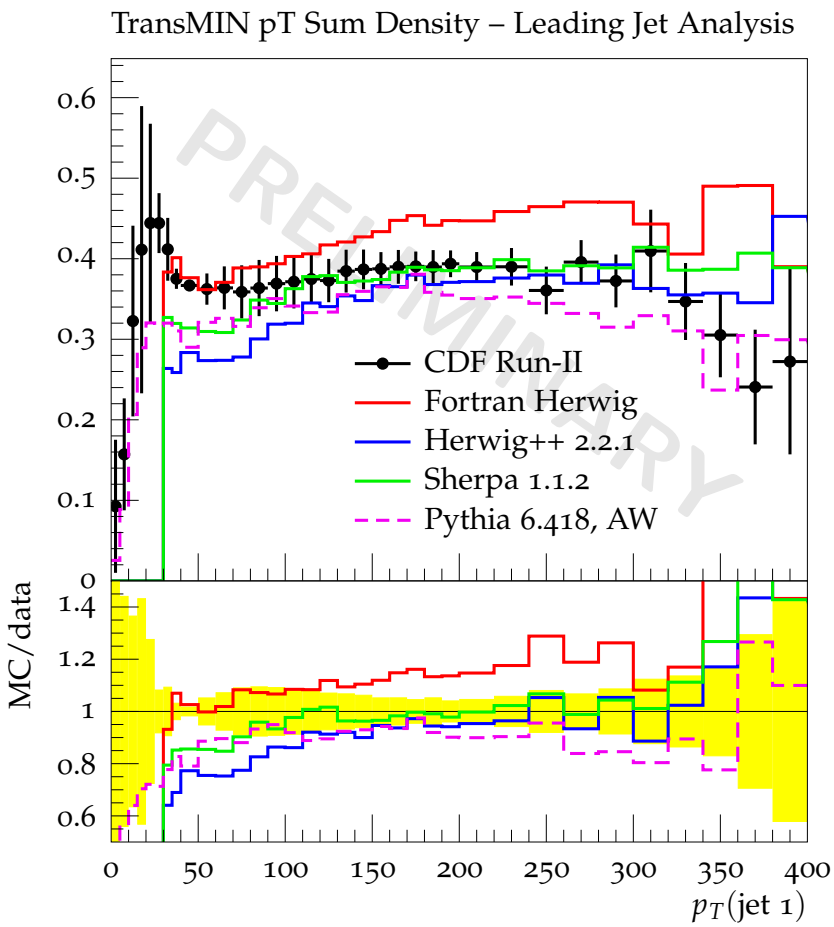


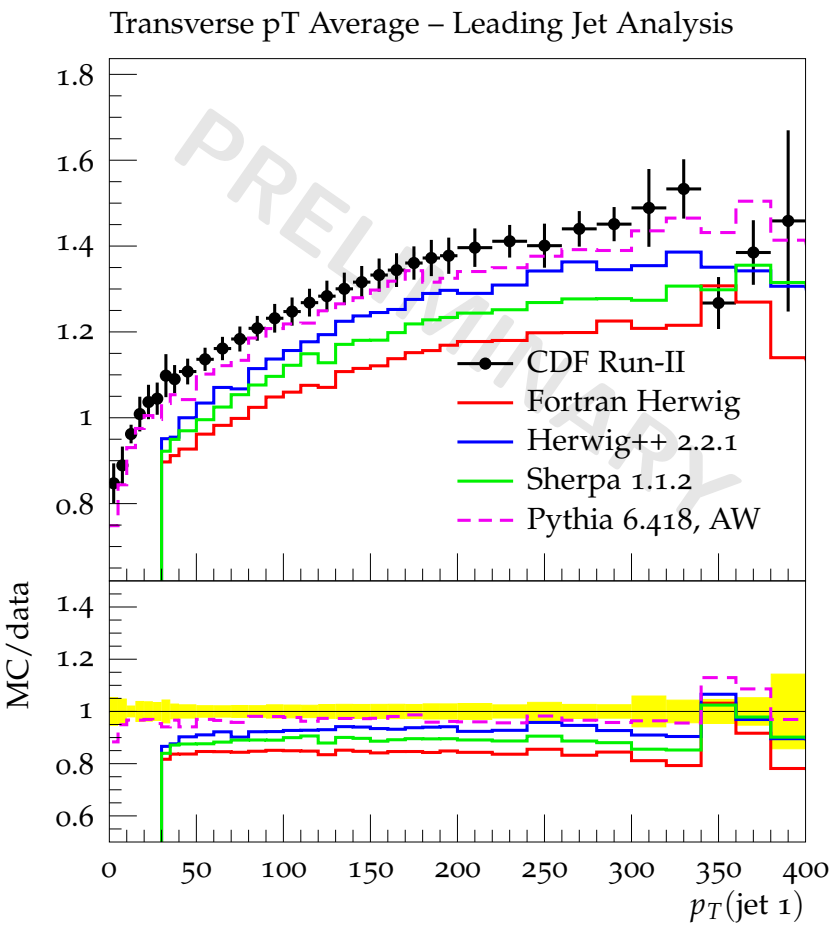
Durham University

Herwig, Sherpa – Tevatron Comparisons



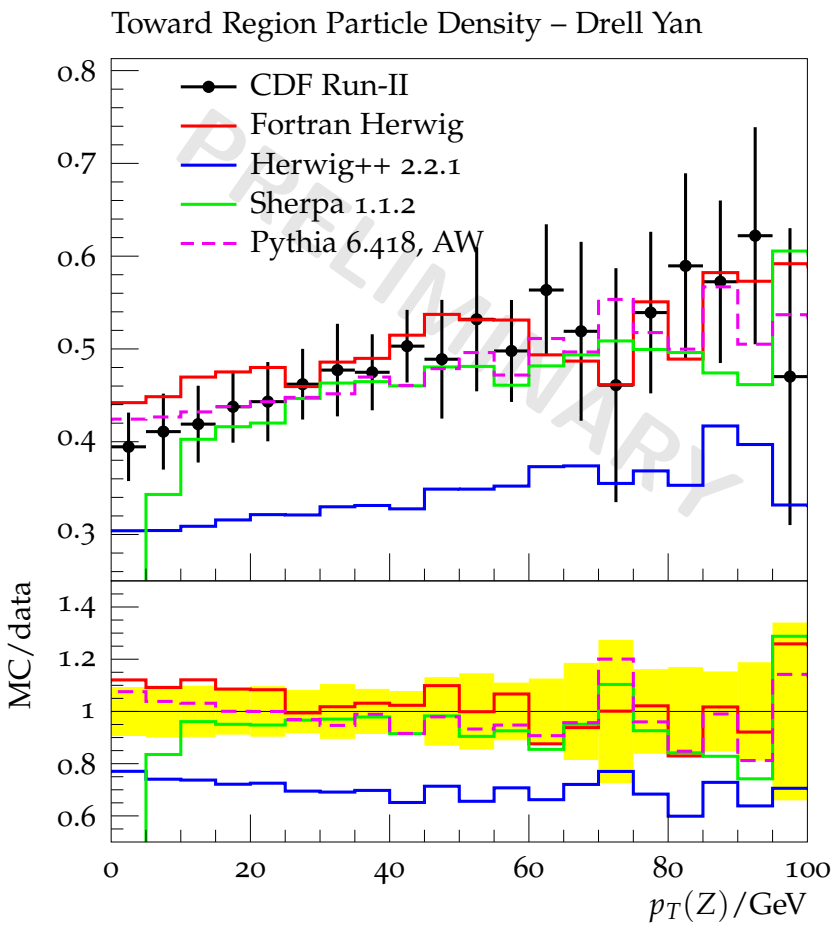
Herwig, Sherpa – Tevatron Comparisons

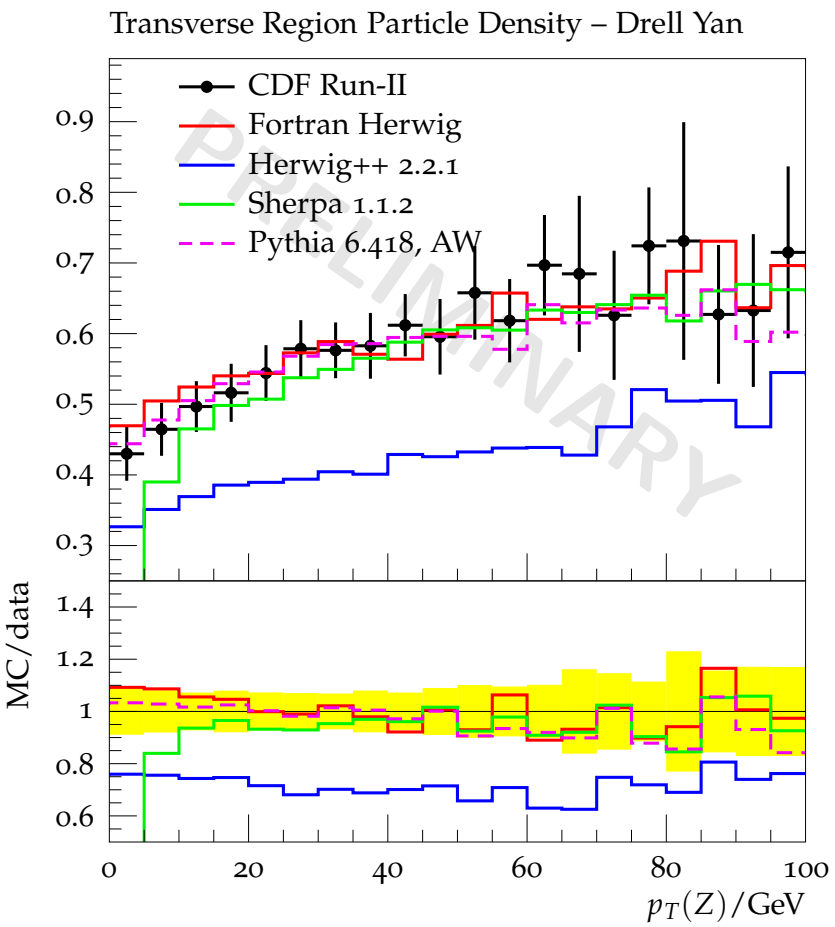




Durham University

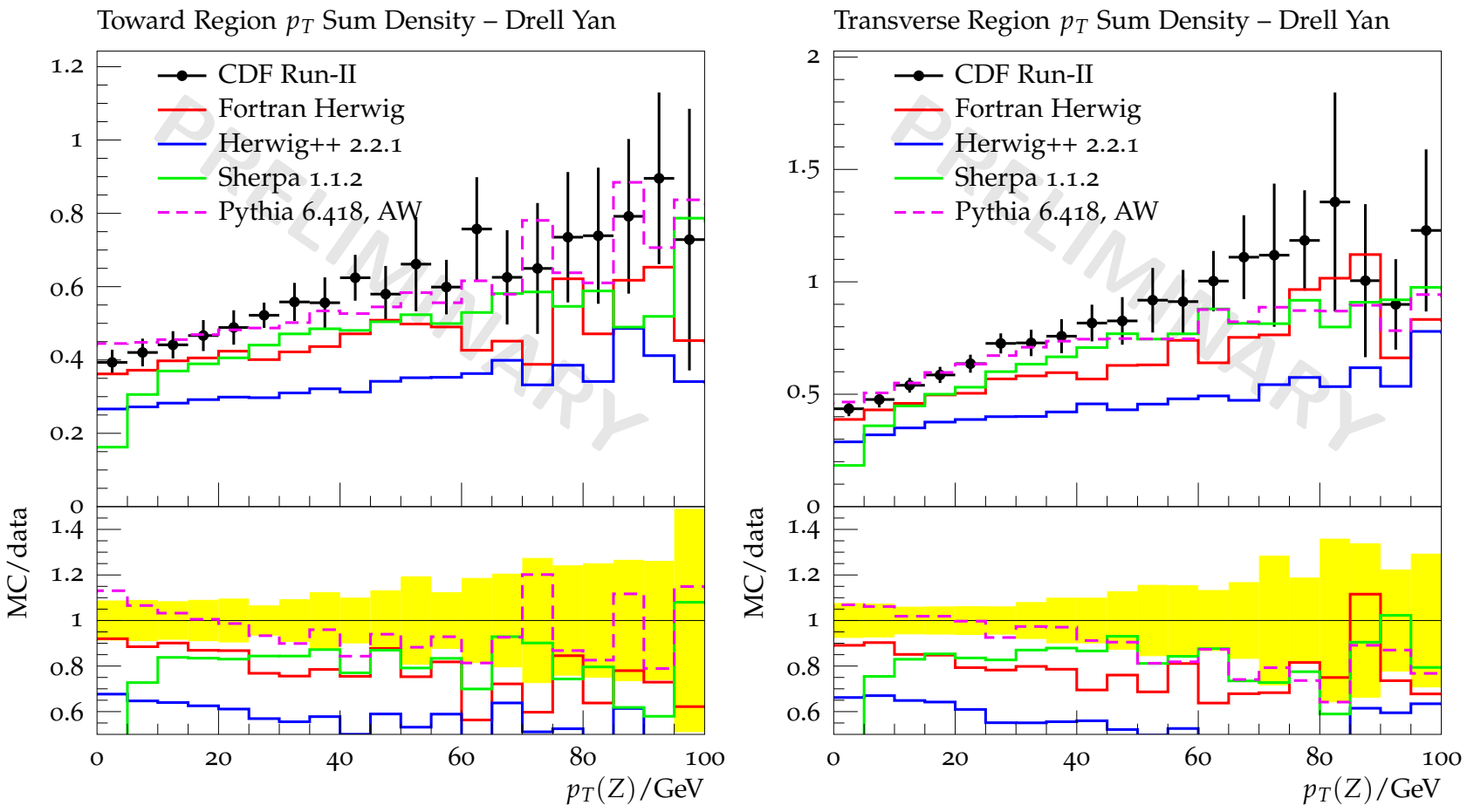
Herwig, Sherpa – Tevatron Comparisons



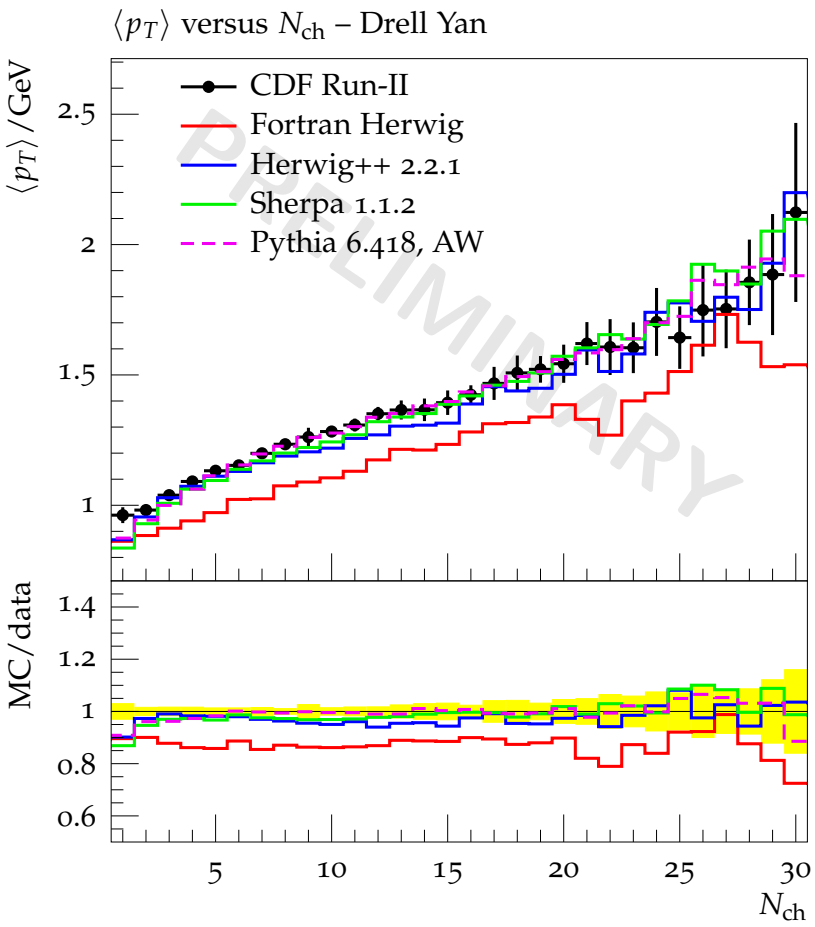


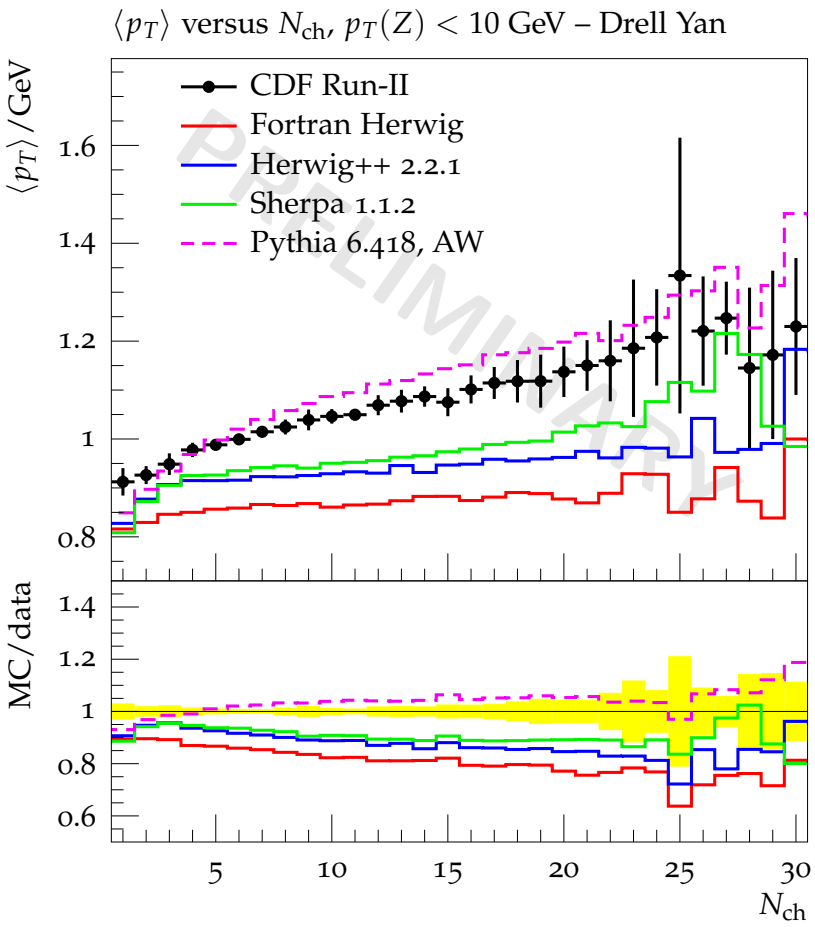
Durham University

Herwig, Sherpa – Tevatron Comparisons



Herwig, Sherpa – Tevatron Comparisons



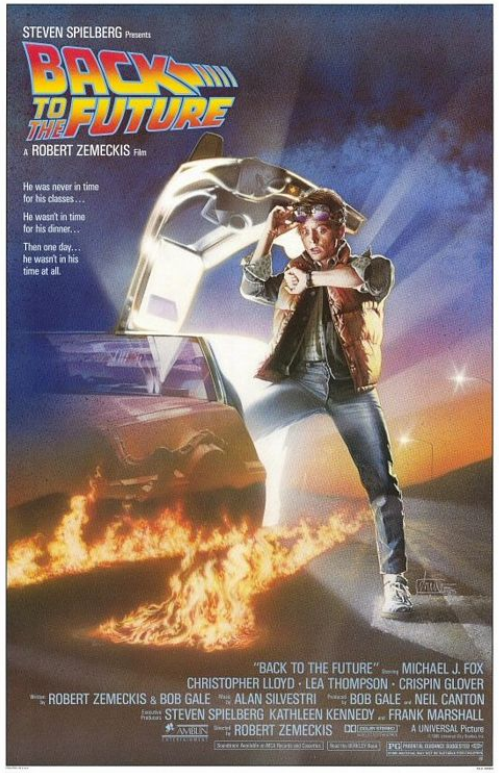


Durham University

Conclusions

Future plans

- Rivet is a mature framework with a simple, efficient, and very flexible usage model. 1.2 release before Xmas, with new histogramming classes. Framework is then 95% stable.
- Work is underway on a Rivet-based validation system, to regression-check both generators and Rivet between versions. Also useful for comparative studies and as a high-level Grid/batch interface to run Rivet jobs.
- Plan to provide a Web archive of standard tune/generator comparisons (as seen here), primarily for LHC experiment reference (cf. JetWeb).



Durham University

Professor future

Main objective right now is to consolidate what we've got: publish the implementation details of the system with robustness and systematics tests (skew, testing 3rd order polynomial in edge cases), plus the LEP and UE tunes.

Next step is to tune the Pythia 6 $p_\perp$-ordered shower and interleaved shower/MPI for Tevatron t-mass analysis systematics, and for LHC UE extrapolations. Using RHIC data to lock down the cross-section energy evolution.

After that: C++ generators, working towards standardising LHC experiment tunes. **Don't expect Professor to be a "push button" system, but we expect collaborative work with expts when framework is fully mature.**

Durham University

Rivet/Professor summary

Rivet is in use by experiments, MCnet, and generator authors for generator tuning and validation

Professor uses Rivet analyses to sample generatr parameter spaces, interpolate the results, and predict optimal tunes. This still requires quite a bit of human intervention, and always will: what it provide is a systematic way for improvements to be explored.

Some major successes and interesting results recently: papers on frameworks and tunes will be ready soon. New version 1.2 of Rivet by the end of the year will be the last major infrastructure upgrade.

New tunes of LHC expt main generators in next year, particularly for constraining RHIC → TVT → LHC underlying event extrapolations.

Durham University