Eur. Phys. J. C (2010) 65: 331–357
DOI 10.1140/epjc/s10052-009-1196-7

THE EUROPEAN PHYSICAL JOURNAL C

Special Article - Tools for Experiment and Theory

Systematic event generator tuning for the LHC

Andy Buckley[1,a], **Hendrik Hoeth**[2], **Heiko Lacker**[3], **Holger Schulz**[3], **Jan Eike von Seggern**[3]

[1]Institute for Particle Physics Phenomenology, Durham University, Durham, UK
[2]Department of Theoretical Physics, Lund University, Lund, Sweden
[3]Physics Department, Berlin Humboldt University, Berlin, Germany

Received: 17 July 2009 / Revised: 1 October 2009 / Published online: 20 November 2009

Abstract In this article we describe Professor, a new program for tuning model parameters of Monte Carlo event generators to experimental data by parameterising the per-bin generator response to parameter variations and numerically optimising the parameterised behaviour. Simulated experimental analysis data is obtained using the Rivet analysis toolkit. This paper presents the Professor procedure and implementation, illustrated with the application of the method to tunes of the Pythia 6 event generator to data from the LEP/SLD and Tevatron experiments. These tunes are substantial improvements on existing standard choices, and are recommended as base tunes for LHC experiments, to be themselves systematically improved upon when early LHC data is available.

1 Introduction

It is an inevitable consequence of the physics approximations in Monte Carlo event generators that there will be a number of relatively free parameters which must be adjusted if the generator is to describe experimental data. Such parameters may be found in most aspects of generator codes, from the perturbative parton cascade to the non-perturbative hadronisation models, and on the boundaries between such components. Since non-perturbative physics models are by necessity deeply phenomenological, they typically account for the majority of generator parameters: typical hadronisation models require parameters to describe e.g. the kinematic distribution of transverse momentum ($p_\perp$) in hadron fragmentation, baryon/meson ratios, strangeness and $\{\eta, \eta'\}$ suppression, and distribution of orbital angular momentum [1–4]. The result is a proliferation of parameters, of which between $\mathcal{O}(10\text{–}30)$ are of particular importance for collider physics simulations.

Apart from rough arguments about their typical scale, these parameters are freely-floating: they must be matched to experimental data for the generator to perform well. Even parameters which appear fixed by experiment, such as Λ_{QCD}, should be treated in generator tuning as having some degree of flexibility since the generator (unlike nature) can only apply them in a fixed-order scheme, albeit augmented with "resummation" of radiation in divergent (particularly soft and collinear) regions of emission phase space. It is also important that the experimental data to which parameters are tuned covers a wide range of physics, to ensure that in fitting one distribution well, others do not suffer unduly. Performing such a tune manually is slow, does not scale well, and cannot be easily adapted to incorporate new results or generator models. In addition, the results are always sub-optimal: a truly good tuning of a generator, which can highlight deficiencies in the physics model as well as provide improved simulations for experimentalists, requires a more systematic approach.

In this paper, we describe the Professor[1] tuning system, which eliminates the problems with manual and brute-force tunings by parameterising a generator's response to parameter shifts on a bin-by-bin basis, a technique introduced by TASSO and later used by ALEPH and DELPHI [5–10]. This parameterisation, unlike a brute-force method, is then amenable to numerical minimisation within a timescale short enough to make explorations of tuning criteria possible. Adding new data or generator models to the system is also relatively simple. We then apply the Professor system to optimisations of the Pythia 6 event generator against e^+e^- event shape and flavour spectrum data from LEP 1 and SLD, and to minimum bias (MB) and underlying event (UE) data from CDF. The resulting tunes (one for each of the two Pythia 6 parton shower/multiple parton interaction (MPI)

[a] e-mail: andy.buckley@cern.ch

[1]Originally derived from the construction "PROcedure For EStimating Systematic errORs", but aesthetics compel us otherwise.

models) are substantial improvements on existing tunes, and demonstrate the Professor system as an important tool for LHC event simulation both before data taking, and in response to early measurements in the new energy regime.

The Professor system is based on simulated experimental analysis data, which is conveniently provided by the Rivet analysis library [11]. As Professor and Rivet development have been closely linked, we first briefly summarise Rivet; however, Professor is not limited to tuning on data from Rivet—any source of comparable histogram data is a valid input.

2 Rivet

The Rivet library is a successor to the HERA-oriented HZTool generator analysis library [12]. Like its predecessor, Rivet is both a library of experimental analyses and of tools for calculating physical observables from an event record. It is written in object-oriented C++, and in particular emphasises the separation of analysis from generator steering: the analyses are performed on HepMC [13] event record objects with no knowledge of or ability to influence the generator behaviour. The reference data files for the $\mathcal{O}(40)$ included analyses are bundled with the package and used to synchronise the Monte Carlo simulation (MC) and reference data binnings. For efficiency, observable calculators cache their results between analyses for each event, ensuring that there are no redundant expensive computations. Standard tool libraries from inside and outside high energy physics are used where prudent; for example, almost all jet algorithms are used via the FastJet [14] package.

Unlike HZTool, the emphasis of Rivet's analysis collection is currently focused on e^+e^- and hadron collider observables, with only a few analyses ported from the HZTool collection. It is not expected that many more HERA routines will be implemented in Rivet: upgrades in HZTool to handle HepMC input will eventually make the two packages complementary for comprehensive physics studies.

For the purposes of the tuning and validation studies presented here, we used the AGILe interface library to pass parameters to generators (primarily Pythia 6, as will be seen) and to populate HepMC events from the HEPEVT common block. AGILe provides programmatic and command-line interfaces to several generators, including the Fortran Herwig 6 [2] and Pythia 6 [1] shower/hadronisation codes, optionally combined with the AlpGen multi-jet merging generator [15], the Charybdis black hole generator [16], and the Jimmy hard underlying event generator [17] for Herwig.

3 Tuning methods

While Rivet provides a system for comparing a given generator tuning to a wide range of experimental data, it has no intrinsic mechanism for improving the quality of that tune. Historically, the main methods of generator tuning have been the purely manual "by eye" method, and a brute-force scan of the parameter space.

Manual tunes Tuning any complex system by eye is evidently non-optimal, and would barely be worth mentioning were it not the most widely used method until now! Manual methods require significant insight into the algorithmic response to parameter choices for even semi-reasonable results, and are intrinsically slow since the procedure typically involves a lot of iterations of parameter choices—even with unhappily low statistics, the turn-around time of a set of runs is a day or more. The scaling is also poor: few humans can cope with manual optimisation of more than five or so parameters, guided by a similar number of comparison plots. The responsiveness to new data or models is similarly deficient, since tuning a different generator essentially involves starting from scratch and, having done a tune once, few people are enthusiastic to repeat the exercise! The prevalence of manual tunings, despite their myriad shortcomings, is a major motivator for the development of Professor.

Brute force tunes This label includes any direct approach which involves running generators very many times. Naïvely, one can think about dividing a parameter space up into a grid and then sampling on the grid line intersections. It will be readily seen that such an approach does not scale: a comprehensive scan of 5 parameters, with 10 divisions in each parameter will require 100,000 generator runs, each perhaps making 10 million events—even then, the sampling granularity will be insufficient for meaningful results. Randomly sampling the space, looking for serendipitous best-χ^2 values has more merit, but is similarly bedevilled by scaling problems and a lack of satisfying ways to either systematically improve the "best" point, or to know whether the minimum that was stumbled into is local or global.

Finally, the approach of putting a generator code into a Markov Chain Monte Carlo (MCMC) optimiser such as Minuit may be summarily dismissed. While the approaches above have the benefit of being parallelisable, MCMC is an intrinsically serial method: one must wait for the nth "function" evaluation to decide where the $(n+1)$th will be. Since generator runs take days, and even the burn-in periods of MCMC samplers may require thousands of samples, this approach is clearly unrealistic.

Parameterisation-based tunes The final approach, which has a lengthy history [5–10], is to parameterise the generator behaviour. Since the fit function itself is expected to be

complicated and not readily parameterisable, there is a layer of indirection: the polynomial is actually fitted to the generator response, MC_b, of each observable bin b to the changes in the P-element parameter vector $\boldsymbol{p} = (p_1, \ldots, p_P)$.

Having determined, via means yet undetailed, a good parameterisation of the generator response to the steering parameters for each observable bin, it remains to construct a goodness of fit (GoF) function and minimise it. The result is a predicted parameter vector, $\boldsymbol{p}_{\mathrm{tune}}$, which should (modulo checks of the technique's robustness) closely resemble the best description of the tune data that the generator can provide.

In parameterisation-based tuning, the run time is dominated by the time taken to run the generator and generate the reference data points. Assuming that sufficient CPU is available to run several hundred MC jobs in parallel, this is at most a few days; the time taken to convert this to a predicted set of best parameters is a few minutes (and can again be parallelised for different configurations as a safety check). Presuming the details elided above to be tractable, this technique offers the possibility of systematic tuning on a timescale compatible with rapid and exploratory re-tunings, ideal for responding to early LHC measurements.

Parameterisation-based optimisation is the approach taken by the Professor system. The following sections document the details of the Professor method and implementation, and tests of its robustness.

4 The professor method

To summarise, the rough formalism of systematic generator tuning is to define a goodness of fit (GoF) function between the generated and reference data, and then to minimise that function. The intrinsic problem is that the true fit function will certainly not be analytic and any iterative approach to minimisation will be doomed by the expense of evaluating the fit function at a new parameter-space point. What we require is an optimisation method designed for very computationally expensive functions whose form is not known a priori. Parameterisation-based optimisation meets these criteria by using numerical methods to mimic the behaviour of an expensive function by using inexpensive ones, and by being amenable to parallelisation in the critical stages. The details to be described in this section are: the choice of general parameterisation function, the method for fitting the general function to the specific response of a MC event generator, the goodness of fit function to be used, and the method of maximising its quality.

4.1 The parameterised response function

As already mentioned, the function to be parameterised is not the overall goodness of fit function between the simulation and the reference data, but the large set of observable bin values for every bin, b, in every distribution. Accordingly, the output of the first stage of Professor is a set of functions $f^{(b)}(\boldsymbol{p})$, which model the true MC response, MC_b, of each observable bin to changes in the P-element parameter vector, $\boldsymbol{p}$.

This ensemble of parameterisations is useful in two ways: first (and most importantly), it provides safety against deviations from the form of the parameterising function, since such deviations are not likely to be correlated between a majority of the bins in normal regions of parameter space. This incoherence of failure to describe the bin-wise generator response ensures that the aggregated measure of generator modelling is faithful to the true behaviour. Second, by breaking the problem down to a fine-grained level, it is possible to select particular regions of distributions as more interesting than the rest—say, the peak of the $Zp_\perp$ spectrum or the thrust distribution, which are particularly sensitive to QCD modelling.

To account for lowest-order parameter correlations, a polynomial of at least second-order is used as the basis for bin parameterisation:

$$\mathrm{MC}_b(\boldsymbol{p}) \approx f^{(b)}(\boldsymbol{p}) = \alpha_0^{(b)} + \sum_i \beta_i^{(b)} p_i' + \sum_{i \le j} \gamma_{ij}^{(b)} p_i' p_j', \quad (1)$$

where the shifted parameter vector $\boldsymbol{p}' \equiv \boldsymbol{p} - \boldsymbol{p}_0$. Here $\boldsymbol{p}_0$ is a reference point in the parameter hypercube, formally arbitrary but chosen to be the centre point of the hypercube to minimise numerical precision issues. The choice of $\boldsymbol{p}_0$ affects the coefficients, but not the shape of the fitted function; clearly $\alpha_0^{(b)} = f^{(b)}(\boldsymbol{p}_0)$, with the other coefficients expressing the deviations from that value.

The number of parameters and the order of the polynomial determine the number of coefficients to be determined. For a second order polynomial in P parameters, the number of coefficients is

$$N_2^{(P)} = 1 + P + P(P+1)/2, \quad (2)$$

since only the independent components of the matrix term $\gamma_{ij}^{(b)}$ are to be counted. For a general polynomial of order n, the number of coefficients is

$$N_n^{(P)} = 1 + \sum_{i=1}^{n} \frac{1}{i!} \prod_{j=0}^{i-1} (P + j). \quad (3)$$

How the number of parameters scales with P for 2nd and 3rd order polynomials is tabulated in Table 1.

A useful feature of using a polynomial for the fit function, other than its general-purpose robustness, is that the actual choice of $\boldsymbol{p}_0$ is irrelevant: a shift in the reference point simply redefines the $\{\alpha, \beta, \gamma\}$ coefficients, but the function remains the same. Hence we are free to choose a numerically stable value within each parameter's chosen range

Table 1 Scaling of number of polynomial coefficients $N_n^{(P)}$ with dimensionality (number of parameters) P, for polynomials of second order ($n = 2$) and third order ($n = 3$)

Num params, P	$N_2^{(P)}$ (2nd order)	$N_3^{(P)}$ (3rd order)
1	3	4
2	6	10
4	15	35
6	28	84
8	45	165
10	66	286

without loss of generality: we use the centre of the hypercube $[\boldsymbol{p}_{\min}, \boldsymbol{p}_{\max}]$, as will be defined in the next section.

4.2 Fitting the response function

Given a general polynomial, we must now determine the coefficients α, β, γ for each bin so as to best mimic the true generator behaviour. This could be done by a Monte Carlo numerical minimisation method, but there would be a danger of finding sub-optimal local minima, and automatically determining convergence is a potential source of problems. Fortunately, this problem can be cast in such a way that an efficient and deterministic method can be applied.

One deterministic way to determine the polynomial coefficients would be to run the generator at as many parameter points, N, as there are coefficients to be determined. A square $N \times N$ matrix can then be constructed, mapping the appropriate combinations of parameters on to the coefficients to be determined; a normal matrix inversion can then be used to solve the system of simultaneous equations and thus determine the coefficients. Since there is no reason for the matrix to be singular, this method will always give an "exact" fit of the polynomial to the generator behaviour. However, this suggestion fails to acknowledge the true complexity of the generator response: we have engineered the exact fit by restricting the number of samples on which our interpolation is based, and it is safe to assume that taking a larger number of samples would show deviations from what a polynomial can describe, both because of intrinsic complexity in the true response function and because of the statistical sampling error that comes from running the generator for a finite number of events. What we would like is to find a set of coefficients (for each bin) which average out these effects and are a least-squares best fit to the oversampled generator points. As it happens, there is a generalisation of matrix inversion to non-square matrices—the *pseudoinverse* [18]—with exactly this property.

As in our matrix inversion example, the set of "anchor" points for each bin are determined by randomly sampling the generator from N parameter space points in a P-dimensional parameter hypercube $[\boldsymbol{p}_{\min}, \boldsymbol{p}_{\max}]$ defined by the user. This definition requires physics input—each parameter p_i should have its upper and lower sampling limits $p_{\min,\max}$ chosen so as to encompass all reasonable values. In practice, we find that generosity in this definition is sensible, as Professor may suggest tunes which lie outside conservatively chosen ranges, forcing a repeat of the procedure. On the other hand, the parameter range should not be too large, to keep the volume of the parameter space small and to make sure that the parabolic approximation gives a good fit to the true Monte Carlo response. Each sampled point may actually consist of many (parallel) generator runs, which are then merged into a single collection of simulation histograms. The simultaneous equations solution described above is possible if the number of sampled points is the same as the number of coefficients between the P parameters, i.e. $N = N_{\min}^{(P)} = N_n^{(P)}$. The more robust pseudoinverse method applies when $N > N_{\min}^{(P)}$: we prefer to oversample by at least a factor of 2.

The numerical implementation of the pseudoinverse uses a standard singular value decomposition (SVD) [19]. First, the polynomial is cast into the form of a scalar product,

$$\mathrm{MC}_b(\boldsymbol{p}) \approx f^{(b)}(\boldsymbol{p}) = \sum_{i=1}^{N_{\min}^{(P)}} c_i^{(b)} \tilde{p}_i, \tag{4}$$

where the $c_i^{(b)}$ coefficients are the independent components of $\alpha_0^{(b)}$, $\beta_i^{(b)}$, and $\gamma_{ij}^{(b)}$ in equation (1), and $\tilde{\boldsymbol{p}}$ is an *extended parameter vector* containing all the corresponding combinations of the parameter vector components. Given sets of sampled points $\{\boldsymbol{p}\}$ and generator values $\{\mathrm{MC}^{(b)}\}$, the above implies the matrix equation,

$$\boldsymbol{v}^{(b)} = \tilde{\mathbf{P}}\boldsymbol{c}^{(b)}, \tag{5}$$

where $\boldsymbol{v}^{(b)}$ contains the generated bin values at the sample points, and the rows of $\tilde{\mathbf{P}}$ are composed of extended parameter vectors like $\tilde{\boldsymbol{p}}$.

For a two parameter case with parameters x and y, the above may be explicitly written as

$$\underbrace{\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}}_{\boldsymbol{v}\text{ (values)}} = \underbrace{\begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ & & & \vdots & & \\ 1 & x_N & y_N & x_N^2 & x_N y_N & y_N^2 \end{pmatrix}}_{\tilde{\mathbf{P}}\text{ (sampled parameter sets)}} \underbrace{\begin{pmatrix} \alpha_0 \\ \beta_x \\ \beta_y \\ \gamma_{xx} \\ \gamma_{xy} \\ \gamma_{yy} \end{pmatrix}}_{\boldsymbol{c}\text{ (coeffs)}}, \tag{6}$$

where the numerical subscripts indicate the N generator runs. Note that the columns of $\tilde{\mathbf{P}}$ include all $N_{\min}^{(2)} = 6$ combinations of parameters in the polynomial, and that $\tilde{\mathbf{P}}$ is square (i.e. minimally pseudoinvertible) when $N = N_{\min}^{(P)}$.

The $c_i^{(b)}$ can then be determined by pseudoinversion of $\tilde{\mathbf{P}}$,

$$\boldsymbol{c}^{(b)} = \tilde{\mathcal{I}}[\tilde{\mathbf{P}}]\boldsymbol{v}^{(b)}, \tag{7}$$

where $\tilde{\mathcal{I}}$ is the pseudoinverse operator.

Except for demanding more sample points than can be computed in reasonable time on the available batching facilities, the order of the polynomial has no influence on the functioning of the parameterisation. Hence the method may be extended in accuracy of the fitting function as required. In practice, a 2nd order polynomial suffices for almost every MC generator distribution studied to date, i.e. there is no correlated failure of the fitted description across a majority of bins in the vicinity of best generator behaviour.

It is worth justifying a little more our apparent obsession with polynomial parameterisations, other than their general ubiquity and robustness. The key point is that a translational free parameter like $\boldsymbol{p}_0$ is hard to express as part of the linear combination of parameters required by this inversion method; the invariance of the fitted polynomial function under shifts in the reference point is one simple way to neatly avoid this problem.

4.3 Goodness of fit function

We choose a heuristic χ^2 function, but other goodness of fit (GoF) measures could certainly be used instead. Since the relative importance of various distributions in the observable set is a subjective thing—for example, given 20 event shape distributions and one charged multiplicity, it would certainly be sensible to weight up the multiplicity by a factor of at least 10 or so to maintain its relevance to the GoF measure—we include per-observable weights, $w_{\mathcal{O}}$ for each observable $\mathcal{O}$, in our χ^2 definition:

$$\chi^2(\boldsymbol{p}) = \sum_{\mathcal{O}} w_{\mathcal{O}} \sum_{b\in\mathcal{O}} \frac{(f^{(b)}(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta_b^2}, \tag{8}$$

where $\mathcal{R}_b$ is the reference value for bin b, and the error Δ_b is the total uncertainty of the reference for bin b. In practice we attempt to generate sufficient events at each sampled parameter point that the statistical MC error is much smaller than the reference error for all bins. For future tuning studies on C++ MC generators, we will include a "theory" error corresponding to the degree of disbelief ($\approx$10%) that the generator authors feel is appropriate to avoid any single observable biasing the GoF heavily (overtuning). In computing the number of degrees of freedom, the weights again enter:

$$N_{\mathrm{df}} = \sum_{\mathcal{O}} w_{\mathcal{O}} \big|\{b \in \mathcal{O}\}\big|. \tag{9}$$

It should be noted that there is unavoidable subjectivity in the choice of these weights, and a choice of equal weights is no more sensible than a choice of uniform priors in a Bayesian analysis. Physics input is necessary in both choosing the admixture of observable weights according to the criteria of the generator audience—for example, a b-physics experiment may prioritise distributions that a general-purpose detector collaboration would have little interest in—and to ensure that the end result is not overly sensitive to the choice of weights.

4.4 Maximising the total GoF

The final stage of our procedure is to minimise the parameterised χ^2 function. It is tempting to think that there is scope for an analytic global minimisation at this order of polynomial, but not enough Hessian matrix elements may be calculated to constrain all the parameters, and hence we must finally resort to a numerical minimisation. This is the numerically weakest point in the method, since the weighted quadratic sum of hundreds of polynomials is a very complex function and there is scope for getting stuck in a non-global minimum. Hence the choice of minimiser is important.

The output from the minimisation is a vector of parameter values which, if the parameterisation and minimisation stages are faithful, should be the optimal tune according to the (subjective) criterion defined by the choice of observable weights.

4.5 Final checks

On obtaining a predicted best tune point from Professor, it is prudent to check the result by running the generator again at the predicted tune: this can be done directly with Rivet. It is also useful to verify that the generator behaves in the vicinity of the predicted point as predicted by the parameterisation by scanning the generator along a line which passes through the "best" point and comparing to the Professor prediction of how the χ^2 will change. This also enables explicit comparisons of default/alternative tunes to Professor's predictions, by making the scan line intersect both points and plotting the slice of the GoF function along the line. Such a line scan can be seen in Fig. 1.

A final important point of procedure remains: so far we have spoken entirely of the procedure as a single set of runs entering the parameterisation and minimisation procedure. However, this is rather dangerous: it may be that we are picking an inappropriate set of runs, or that a subset of points are skewing the fit and minimisation away from the true behaviour. Even if this is not the case, the lack of any alternative to which we can compare means that we have little knowledge of the procedure's systematic uncertainties. Hence, we have also found it to be useful to oversample by a considerable fraction, $N \gg N_{\mathrm{min}}^{(P)}$, and then to perform the parameterisation and χ^2 minimisation for a large number of distinct

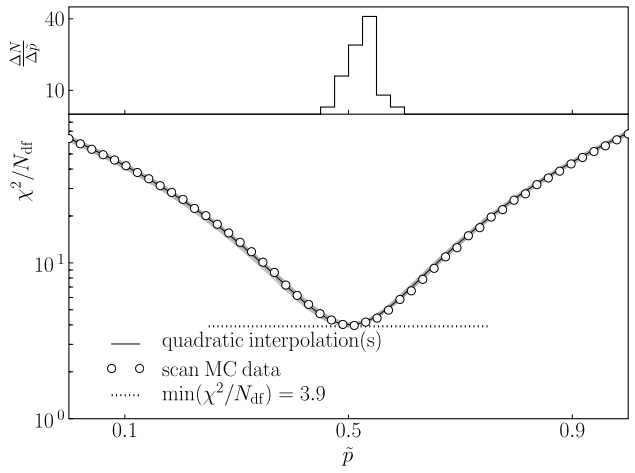


Fig. 1 The results of an example line-scan in $\chi^2/N_{\rm df}$ through nine-dimensional (flavour-) parameter space, demonstrating the agreement between Professor's predicted values (*lines*, interpolation versus data) and the true values (*white dots*, MC-scan versus data). Shown are 100 interpolations that use 299 out of 399 available runs (*grey band*) and the interpolation that uses all available runs which fits in perfectly (*black line*). The *histogram* displays the distribution of minimisation results derived from these interpolations. The *scan line* is chosen such that it pierces a hypercube that is symmetric around one of the predicted minima, such that in the linear line-scan parameter $\tilde{p}$, the predicted minimum is at $\tilde{p} = 0.5$, and the hypercube boundaries are at $\tilde{p} = \{0, 1\}$. Its volume is about 0.03% of the total volume of the parameter sampling hypercube, illustrating the futility of attacking this optimisation problem via a grid scan of the space

run-combinations, $N_{\rm tune}$, where $N_{\rm min}^{(P)} \ll N_{\rm tune} \leq N$. The set of different predicted responses and tunes from all the different $N_{\rm tune}$-run combinations provides a systematic control. It is important that the tuning run combinations need a significant degree of independence from one another if their indication of systematics is to be believed, i.e. $N_{\rm tune} \ll N$ for most of the tune run-combinations. In practice, using $N_{\rm tune} = N$ typically gives good results, but not necessarily the best possible.

5 Implementation

In this section we describe the implementation of the Professor method as a class library and a set of programs which use it. The majority of the code is written in Python, but makes use of the NumPy numerical library [20] and the SciPy scientific library [21], in which most of the core functions are implemented in CPU-efficient C code. Additionally, the SciPy "weave" functionality can be used to automatically generate C code which is compiled and cached for later use automatically at run-time, with a corresponding speed-up factor of ∼5. The Professor user interface is a set of distinct Python scripts which produce tunes based on input data, explore line scans in the parameter space, and so on. Many parts of the process are designed to be parallelisable on a batch system, and a standard templating system [22] is used to build batch scripts for this purpose.

5.1 Data generation

Having decided what parameters to tune to what observables, the random parameter points need to be sampled. This is done using the `prof-scanparams` script which only needs a file of parameter ranges and the desired number of parameter points as input. We are using Python's uniform random generator `random.uniform` to independently sample values in all dimensions of a hypercube defined by the input file. To deal with parameters whose effect is non-linear, it is recommended that the parameters themselves be invertibly transformed to a more linear form, rather than sampling the space non-uniformly: this can be easily added by modifying the script, but is not an intrinsic feature.

The default output format is a list of simple (name, value) pairs, suitable for use with the AGILe generator interfaces, but more complex templating can also be used for native use with generators, such as Herwig++, which have a more complex configuration system.

The number of runs must be at least $N_{\rm min}^{(P)}$,[2] but we typically use several times this number, so that the parameterised function is not artificially anchored to the sampled values but may float away from them to exploit the least-squares property of the pseudoinverse. This is important independently of the considerations about considering many independent run combinations—this higher-level bet hedging is less than useful if all of the available combinations are intrinsically untrustworthy.

It is worth noting that despite the scaling of $N_{\rm min}^{(P)}$, the volume of the hypercube still scales exponentially with P and that the number of samples had better keep approximate pace with this scaling, especially in wide scans where many different generator behaviour regimes may be encountered. The power law scaling of the polynomial does not obviate the responsibility to ensure that the fitting method sees a representative sample of the space to be fitted. It is also wise to ensure that the sample ranges are chosen so as to include the default tune, at least in the first phase of a tuning: hopefully this would automatically be the case, since a first set of sampling ranges should at least include all "reasonable" values of each parameter.

The job of running the generator and Rivet (and of merging the output histograms from different kinematic regions, if required) is mainly left up to the user. This is because

[2] To clarify, P here is the number of *independent* parameters to be sampled, i.e. the dimensionality of the sample space. The constrained parameters do not contribute to this parameter count.

different system configurations, the variety of batching systems, the choice of contributing Rivet analyses, etc. effectively mandate some user customisation of run scripts. Attempting to automate this process would likely lead to disastrously algorithmic tuning efforts, but we note that our choice of observables for Pythia 6, as described in Sect. 6, is as good a template as any currently available.

For most of the Professor procedure, the analysis data is defined by a directory structure containing a reference directory and a set of run directories, each of which contains histogram files from Rivet. The same structure may be conveniently used to store output from different tunes. It is also possible for analysis programs other than Rivet to provide input for Professor tuning, provided that their data format is in a format which can be used by Professor, or can be converted to such a format. The currently most-used data format is the "AIDA" XML format, as this is the main Rivet output format. When, as planned, Rivet's data format is upgraded to use the simpler "YODA" data files, Professor will also support this format. Yet more formats can also be supported in response to demand.

Loading of the data files is currently "eager", i.e. all data files are read in and stored in memory during processing. For large data sets, e.g. $\sim$1000 sampled parameter points with distributions amounting to $\sim 10^4$ bins per point, this produces a lead time of $\sim$1 minute on a typical workstation and large memory occupancy. For larger input sets, where this lead time may be less tolerable, a "lazy" loading and proactive garbage collection on unused bins will be explored.

5.2 Tuning

The main tuning stage is accessed via the `prof-tune` program. This performs the combination of parameterisation and optimisation against reference data for each of a set of MC run combinations, based on the runs found in the input directory. The run combinations can either be uniquely and randomly generated at run-time by `prof-tune`, or can be supplied via a plain text file in which each line is a whitespace separated list of run names. This latter method is most useful for parallelising the tuning for a large number of run combinations.

5.2.1 Parameterisation and fitting

Professor currently supports second- and third-order polynomials for parameterisation—as previously discussed, these are robust against origin-translation in a way necessary for the pseudoinverse method to work, and our experience is that a second-order polynomial has so far been sufficient for almost all purposes in generator tuning.

For the numerical evaluation of the pseudoinverse procedure, NumPy's implementation of the singular value decomposition is used.

5.2.2 GoF optimisation

To have an intuitive way of excluding single bins from the GoF calculation, the implemented χ^2 function differs slightly from (8) in that weights are not applied on a per-observable but on a per-bin scope. The N_{df} definition is changed accordingly. By this, single bins can be left out of the χ^2 calculation by setting their respective weights to zero. We use this to veto bins with zero error as this usually indicates that these bins were not filled during the data generation. After applying the weights and vetoes all bins with zero weight are filtered out. From the resulting bins the N_{df} is computed and a χ^2 function $\chi^2(\boldsymbol{p})$ is constructed, which is passed to the minimiser.

The optimisation of the heuristic χ^2/N_{df} function is implemented using minimisers from SciPy and also PyMinuit [23], a Python interface to the CERN Minuit package [24]. As Minuit uses a Markov chain method, which copes with high dimensional problems better than the SciPy Nelder-Mead simplex minimiser, and also offers error estimates and covariance calculations, it is the preferred and default choice. Professor is also able to apply limits to each parameter in minimisation, to exclude unphysical results. The limits used in such cases should not just be the sampling limits, unless those were determined by physical restrictions, since a minimisation falling outside the sample limits is actually a useful result which should not be obscured.

By default the starting point for the minimiser is the centre of the parameter space defined by our parameter sampling ranges. It is also possible to specify a manual or random starting point. Minuit evaluates the parameter uncertainties by calculating those parameter points at which the χ^2/N_{df} value exceeds that of the minimum by 1: for a truly χ^2-distributed test statistic this should correspond to a "1σ" 68% confidence limit estimate.

A successful minimisation will write out its details to file, including the optimal parameters and their correlations, a file of parameterised histograms for each of the observables included in the fit (based on the parameterisation at the tune point), and information about the number of parameters, optimal GoF value(s), etc. These can then be studied and plotted as described in the next section.

5.2.3 Tuning output and visualisation

The result of the tuning stage, in the form of the `prof-tune` program, is a file of tune points, including their GoF scores. If the tuning has been parallelised, there will be several such files, which can be merged together if desired. The tunes can be visualised either textually or graphically.

Graphical visualisation is particularly useful, and comes in several different forms:

GoF vs. parameter value Each tuning parameter produces a plot of GoF vs. parameter value, with parameter sampling boundaries indicated. Run combinations of different size are represented in different colours, and points which lie outside *any* of the sampling boundaries are indicated by lighter shades of their point colour: this makes it easy to see how predicted tunes fit into the high-dimensional sample space without having to perform feats of mental gymnastics. Clearly, if a cluster of points falls outside one or more sampling boundaries, their GoF values are less than trustworthy and a re-run of the generator sampling, with expanded boundaries is recommended.

GoF vs. weight combination If several combinations of weights have been tested, they can be graphically displayed side-by-side to verify that the tune is robust against reasonable shifts of GoF definition.

Correlation display For each minimisation result we also store the covariance matrix between parameters and values, as calculated by the minimiser. The Professor-system provides the user with the possibility to calculate the coefficients of correlation ρ_{ij} for each pair of parameters (i, j) from the (symmetric and real) P-dimensional covariance matrix $\mathcal{C}$:

$$\rho_{ij} = \frac{\mathcal{C}_{ij}}{\sqrt{\mathcal{C}_{ii}\mathcal{C}_{jj}}}. \tag{10}$$

The resulting parameter-parameter correlations can be displayed as colour-map plots or as tables such as in Table 9.

Sensitivities It is desirable to tune to those observables most sensitive to parameter changes. Clearly, if a parameter has no effect on an observable at all the minimiser is very likely to yield useless results or it might even fail to converge. The Professor package offers the calculation of a bins sensitivity to changes of the values of the parameters in question for tuning based on the parameterisation:

$$S_i^{(b)}(\boldsymbol{p}) \approx \frac{f^{(b)}(\boldsymbol{p}+\boldsymbol{\varepsilon}) - f^{(b)}(\boldsymbol{p})}{f^{(b)}(\boldsymbol{p})} \cdot \frac{\boldsymbol{p}}{\boldsymbol{p}+\boldsymbol{\varepsilon}}, \tag{11}$$

where the $\boldsymbol{\varepsilon}_i$ are conveniently set to one percent of the interval $[\boldsymbol{p}_i^{\min}, \boldsymbol{p}_i^{\max}]$.

A side-by-side comparison of an observables sensitivity to all parameters included in the parameterisation is available as colour-map plots. These plots may be used to identify and remove those observables that have little or no impact at all. They could also be seen as an a posteriori justification for the choice of observables included in a tuning.

Interactive parameterisation explorer The script `prof-I` can be used to interactively explore the effect of shifts in parameter space on the shapes of observables. All parameter values can be adjusted via sliders in a graphical user interface. The resulting shapes of the observables, calculated from the parameterisations, are updated in real time as the sliders are moved, and static data or MC histograms can also be shown for comparison. As the goodness of fit is also displayed, `prof-I` can be also used as an aid in manual tunings.

Histogram prediction `prof-tune` also helpfully produces a directory of histogram files, one for each tuning, which makes it possible to see how each distribution is predicted to behave at that point without running the generator and incurring the usual (typically multi-day) delay. This is particularly useful when choosing how to weight distributions to achieve the desired quality of fits—a subjective prioritising of particular physics which cannot be avoided and usually requires some iteration.

These visualisations of tunes are extremely useful, not least for iterating the choice of sample boundaries in the early stages of a tune. At this point, the sampling boundaries must be wide enough to include the predicted tunes—if the prediction consistently falls outside the boundaries, it is probably indicative of a problem with the generator physics model—but also narrow enough that the sampling is representative of the space. A too-wide initial scan may be too coarse to yield stable results. Care should also be taken when tightening boundaries for secondary tune stages, since in the case of strong parameter–parameter correlations the optimum may unexpectedly appear once again outside the boundaries due to the improved description of the correlations. The main rule is that there *are* no truly reliable rules, and some iteration will certainly be required.

Once the boundaries have been chosen for a final stage of tuning, if there are observed cases where the polynomial does not sufficiently describe the generator, it is worth trying the third-order polynomial.

5.3 Validation

Before parameterising real MC-generated data, the parameterisation algorithm was tested for robustness against the distribution of the anchor points and its behaviour when dealing with data which cannot fit exactly to the parameterising polynomial: such troublesome data was simulated by smeared sampling from polynomials of various orders. This round of validation also sought to verify that the GoF calculated from the parameterised bins resembles that obtained directly from the MC-generated data, and is described briefly in this section.

5.3.1 Robustness of the parameterisation algorithm

The Professor GoF function can be influenced by several observable weight combinations, $w_{\mathcal{O}}$, and also by the number

of runs in each random run combination used for the parameterisation. This offers possibilities to check the predicted minima for systematic errors due to improper parameterisation or overtuning to a specific set of observable weights.

In addition, the minimisation results obtained from quadratic interpolations were compared to those obtained from cubic interpolations. We did not find a significant difference between the predictions, although the cubic interpolation describes the generator response better in regions that are far away from the minimum.

The basic functioning of the polynomial parameterisation was tested with input data generated with a second-order polynomial with random coefficients—the *known* coefficients were compared to those of the resulting parameterisations. The robustness of parameterising error-smeared and data from non-second-order distributions was also tested. Example input data were generated using second- to fourth-order polynomials, especially polynomials of the form

$$f(\boldsymbol{p}) = (\boldsymbol{p} - \boldsymbol{m}_1)^2(\boldsymbol{p} - \boldsymbol{m}_2)^2 + \boldsymbol{a} \cdot \boldsymbol{p}, \tag{12}$$

and were smeared using an Gaussian error. Then, the *un*smeared original polynomial and the parameterisation were evaluated at 10,000 randomly located points and a simple χ^2/N_{df} and "pull" statistics were calculated as GoF measures, where the pulls were calculated as follows:

$$p = \sum_{i=1}^{10{,}000} \frac{f_{\mathrm{unsmeared}}(\boldsymbol{x}_i) - f_{\mathrm{param}}(\boldsymbol{x}_i)}{\sigma}, \tag{13}$$

with the $\boldsymbol{x}_i$ being the test points, $f_{\mathrm{unsmeared}}$ and f_{param} the unsmeared polynomial and parameterisation respectively, and σ the width of the Gaussian error distribution. A Gaussian distribution was then fitted to the pull histogram. This procedure was followed for several different dimensions of parameter space P and different numbers of sample points $N = N_{\mathrm{min}}^{(P)}, N_{\mathrm{min}}^{(P)} + 2, \ldots$.

Using the minimal $N_{\mathrm{min}}^{(P)}$ sample points resulted in wide variation of fit quality, with observed χ^2/N_{df} varying across several orders of magnitude, and broad—in the low dimensional case even biased—pull distributions. Using additional sample points reduced all this unwanted behaviour, e.g. in the case of a 7-dimensional parameter space and data generated from a fourth-order polynomial, the average width of pull distribution fell from 7.9 for $N_{\mathrm{min}}^{(7)}$ samples to 3.2 for $N_{\mathrm{min}}^{(7)} + 6$ samples, and the corresponding ranges of observed χ^2/N_{df} fell from $\mathcal{O}(10\text{–}10^3)$ to $\mathcal{O}(1\text{–}10)$. Consequently, we discourage use of parameterisations based on the minimal number of sample points: the evidence is simply that they do not provide reliable descriptions of the parameter space. Examples of pull distributions are shown in Fig. 2.

Finally, the influence of the distribution of the sample points in the parameter hypercube on the parameterisation

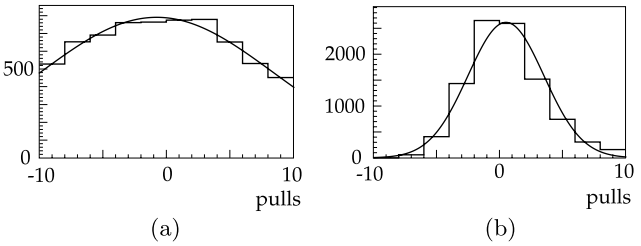


Fig. 2 Example pull distributions: Parameterisations of data generated with a smeared fourth-order polynomial (see (12)) in 7 dimensions are compared to the unsmeared polynomial. The parameterisations have been created (**a**) using the minimal number of anchor points $N_{\mathrm{min}}^{(7)} = 37$ and (**b**) using $N_{\mathrm{min}}^{(7)} + 6 = 43$ anchor points. One can clearly observe that the pull distribution narrows when using additional sample points

quality was tested. We performed 5000 parameterisations based on error-smeared data. χ^2/N_{df} values were computed as above, along with several different measures of the anchor point distributions

- Average and minimal Cartesian distance
- Average and minimal distance of the projections on the parameter axis

These were studied as 2D histograms. Analysing the low-dimensional cases revealed a dependence of the GoF on the averaged distances for infrequent anchor point samples, which again could be eliminated by oversampling. The more relevant high-dimensional cases did not show this dependence; however oversampling narrowed the range of observed χ^2/N_{df} by several orders of magnitude. The parameter space dimensionality for these latter tests ranged from $P = 1$ to 10, and the number of anchor points from $N = N_{\mathrm{min}}^{(P)}$ to $N_{\mathrm{min}}^{(P)} + 10$.

5.3.2 *Tune verification*

As mentioned, it is useful to visualise Professor tunes along lines in parameter space, particularly lines which intersect both the predicted tune and an alternative or default configuration. Professor provides a program, `prof-scanchi2`, to perform this scan for saved parameterisations and/or to generate generator/Rivet batch scripts from supplied templates. This enables verification that the GoF really behaves as parameterised and that the chosen point really is close to a GoF extremum.

To reduce the risk that a minimum returned by the numerical minimisation is a local minimum, `prof-tune` can perform several minimisations with different starting points.

A tighter tune, either Professor or grid-scan based, could be performed based on the correspondence between the true and parameterised GoF in the tune region, although in practice the deviations are small enough that we have not risked overtuning by attempting to do so.

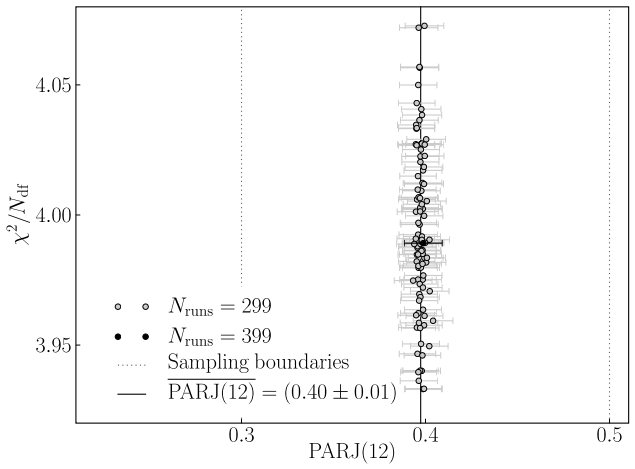


Fig. 3 Two dimensional $\chi^2/N_{\rm df}$ versus parameter value plot showing minimisation results derived from 100 interpolations that use 299 out of 399 available runs and one result that uses all available runs for the probability that a strange meson has spin 1. The observable-weight combination used can be found in Table 6. The parameter errors are estimated by the minimiser by going up one unit in χ^2

5.3.3 Tuning stability

The Professor system offers two different ways to get an estimate of how well a predicted minimum is defined.

We can benefit from oversampling the parameter space with respect to $N_{\rm min}^{(P)}$ in such a way that numerous run combinations[3] may be chosen for different parameterisations simply by omitting a fraction of all the available Monte Carlo runs. In order to reduce the correlation between run combinations we usually choose this fraction to be about one-third. This is clearly a compromise between the quality of the parameterisation and the degree of correlation introduced by choosing several run combinations.

The outcome of all minimisations can be displayed parameter-wise such as in Fig. 3. We observe that the minimisation result derived from all available Monte Carlo runs always lies within the distribution of $\chi^2/N_{\rm df}$ from random run combination tunes, illustrating that certain interpolations fit the data better than others but that using all the information available always gives a good average description.

Instead of varying the parameterisation it is also possible to influence the GoF function. This can be done by independently applying a weight to each observable included in the tuning. This more-or-less subjective approach is justified by two facts. Firstly, we certainly do not expect the generator's response function to be a simple polynomial, and secondly we know that most models should not be expected to be capable of describing all observables: for example, all generators fail to describe the $p_\perp^{\rm out}$ distribution in e^+e^- data.

[3]We usually do about 100 minimisations. The run combinations are chosen in a randomised procedure, and it is explicitly checked that there are no duplicates.

5.4 Performance and comments

The focus in testing and commissioning the Professor system has until recently been focused on Pythia 6 tunes against LEP, SLD, and Tevatron data. Here we were able to interpolate and minimise up to 10 parameters at a time for roughly 100 distributions, but beyond this the minimisation time became large and we were less satisfied with the resulting minima. The latter effect probably represents the fact that $N_{\rm min}^{(P)}$ only specifies the minimum number of samples needed to algebraically constrain a curve in P-space, but tells us nothing about the number of points needed to adequately represent the space—this depends on more complex things like the rate of change of the function, the extent to which it oscillates, and the degree of correlation between parameters: in general it will rise much faster than the algebraic constraint.

We advise that a maximum of 10 parameters be observed whenever possible, and less than that is advisable. Since generators usually have many more parameters than this, some approximate factorisations into semi-independent groups must be found. As usual, this requires appreciation of the generator physics, and ideally input from the generator authors.

6 Complete tuning of Pythia 6

For the first production tuning we chose the Pythia 6 event generator, as this is a well-known generator which has been tuned before and which we expected to behave well. Additionally, the tuning of Pythia 6 was arguably more pressing than that of any other generator as it is used for the majority of LHC experiment MC simulation, and the newer parton shower/MPI model had never before been tuned in detail. All results in this paper are based on the version 6.418.

Our multi-stage approach to tuning was to fix the flavour composition and the fragmentation parameters to the precision data from LEP and SLD before continuing with the parameters related to hadron collisions, for which we use data from the Tevatron. We do not include data from HERA e^-p collisions in these tunes, mainly because of the lack of availability of such data in Rivet, but also because the distinct initial state places demands on model universality with which we do not wish to obscure the current study.

6.1 Parameter factorisation strategy

In Pythia, the parameters for flavour composition decouple well from the non-flavour hadronisation parameters such as the Lund string parameters a, b, σ_q, and from the shower parameters ($\alpha_{\rm s}$, cut-off). Parameters related to the underlying

event and multiple parton interactions are decoupled from the flavour and fragmentation parameters. In order to keep the number of simultaneously tuned parameters small, we decided to follow a three-stage strategy. In the first step the flavour parameters were optimised, keeping almost everything else at its default values (including using the virtuality-ordered shower). In the second step the non-flavour hadronisation and shower parameters were tuned—using the optimised flavour parameters obtained in the first step. The final step was tuning the underlying event and multiple parton interaction parameters to data from the two Tevatron experiments CDF and DØ.

6.2 Flavour parameter optimisation

The observables used in the flavour tune were hadron multiplicities and their ratios with respect to the π^+ multiplicity measured at LEP 1 and SLD [25], as well as the b-quark fragmentation function measured by the DELPHI collaboration [26], and flavour-specific mean charged multiplicities as measured by the OPAL collaboration [27]. For this first production we chose to use the Lund-Bowler fragmentation function for b-quarks (invoked in Pythia 6 by setting MSTJ(11) = 5) with a fixed value of $r_b = 0.8$ (PARJ(47)), as first tests during the validation phase of the Professor framework showed that this setting yields a better agreement with data than the default common Lund-Bowler parameters for c and b quarks.

For the tuning we generated 500,000 events at each of 180 parameter points. The tuned parameters are the basic flavour parameters like diquark suppression, strange suppression, or spin-1 meson rates. All parameters are listed in Table 2 together with the tuning results and a χ^2/N_{df} line-scan plot comparing the default with tuned parameter sets in Fig. 4. The physics observables and their weights for the tuning are listed in Table 6.

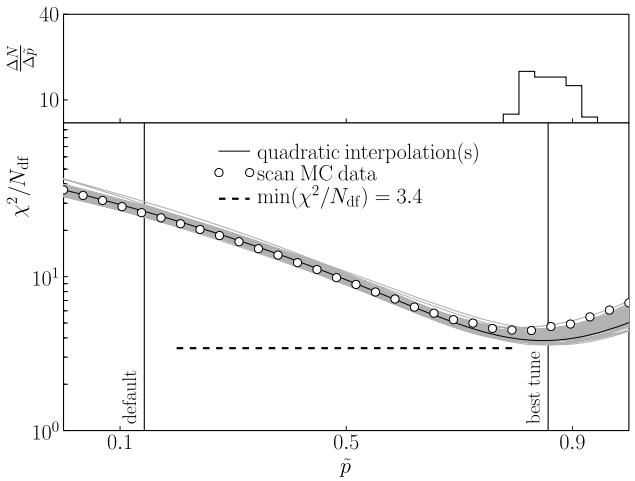


Fig. 4 Pythia 6 (Q^2 shower) χ^2/N_{df} variation along a line in the 9D flavour-hyperspace, linearly parameterised by $\tilde{p} \in [0, 1]$. The *line* shown runs between the default and the Professor flavour tuning. The *white dots* are the true generator χ^2/N_{df} values, and the *grey lines* an ensemble of parameterisations from the Professor procedure that use about two-thirds of the available MC-runs. The *black line* represents the interpolation calculated from all available runs. The *histogram* displays the distribution of minimisation results (if projected on the scan-line) derived from these interpolations

Since the virtuality-ordered shower was used for tuning the flavour parameters, we tested our results also with the $p_\perp$-ordered shower in order to check if a separate tuning was necessary. Turning on the $p_\perp$-ordered shower and setting $\Lambda_{\mathrm{QCD}} = 0.23$ (the recommended setting before our tuning effort) we obtained virtually the same multiplicity ratios as with the virtuality-ordered shower. This confirms the decoupling of the flavour and the fragmentation parameters and no re-tuning of the flavour parameters with the $p_\perp$-ordered shower is needed.

In Table 12, we compare several measured mean hadron multiplicities in e^+e^- collisions at 91 GeV to Pythia predictions with default settings and with our tune. In particular, the strange sector is significantly improved, although there is a slight degradation for charm and bottom mesons.

6.3 Fragmentation optimisation

Based on the new flavour parameter settings, the non-flavour hadronisation and shower parameters were tuned separately for the virtuality-ordered and for the $p_\perp$-ordered shower. The observables used in this step of the tuning were event shape variables, momentum spectra, and the mean charged multiplicity measured by the DELPHI collaboration [10], momentum spectra and flavour-specific mean charged multiplicities measured by the OPAL collaboration [27], and the b-quark fragmentation function measured by the DELPHI collaboration [26]. All observables and their weights are listed in Table 7.

We tuned the same set of parameters for both shower types (Table 3). To turn on the $p_\perp$-ordered shower, MSTJ(41) was set to 12. In the case of the virtuality-ordered shower, this parameter stayed at its default value. For both tunes, we generated 1 million events at each of 100 parameter points.

During the tuning of the $p_\perp$-ordered shower it transpired that the fit prefers uncomfortably low values of the shower cut-off PARJ(82). Since this value needs to be at least $2 \cdot \Lambda_{\mathrm{QCD}}$, and preferably higher, it was manually fixed to 0.8 to keep the parameters in a physically meaningful regime. Then the fit was repeated with the remaining five parameters.

The second issue we encountered with the $p_\perp$-ordered shower was that the polynomial parameterisation $f^{(b)}$ for the mean charged multiplicity differed from the real Monte Carlo response by about 0.2 particles. This discrepancy was accounted for during the χ^2 minimisation, so that the final result does not suffer from a bias in this observable.

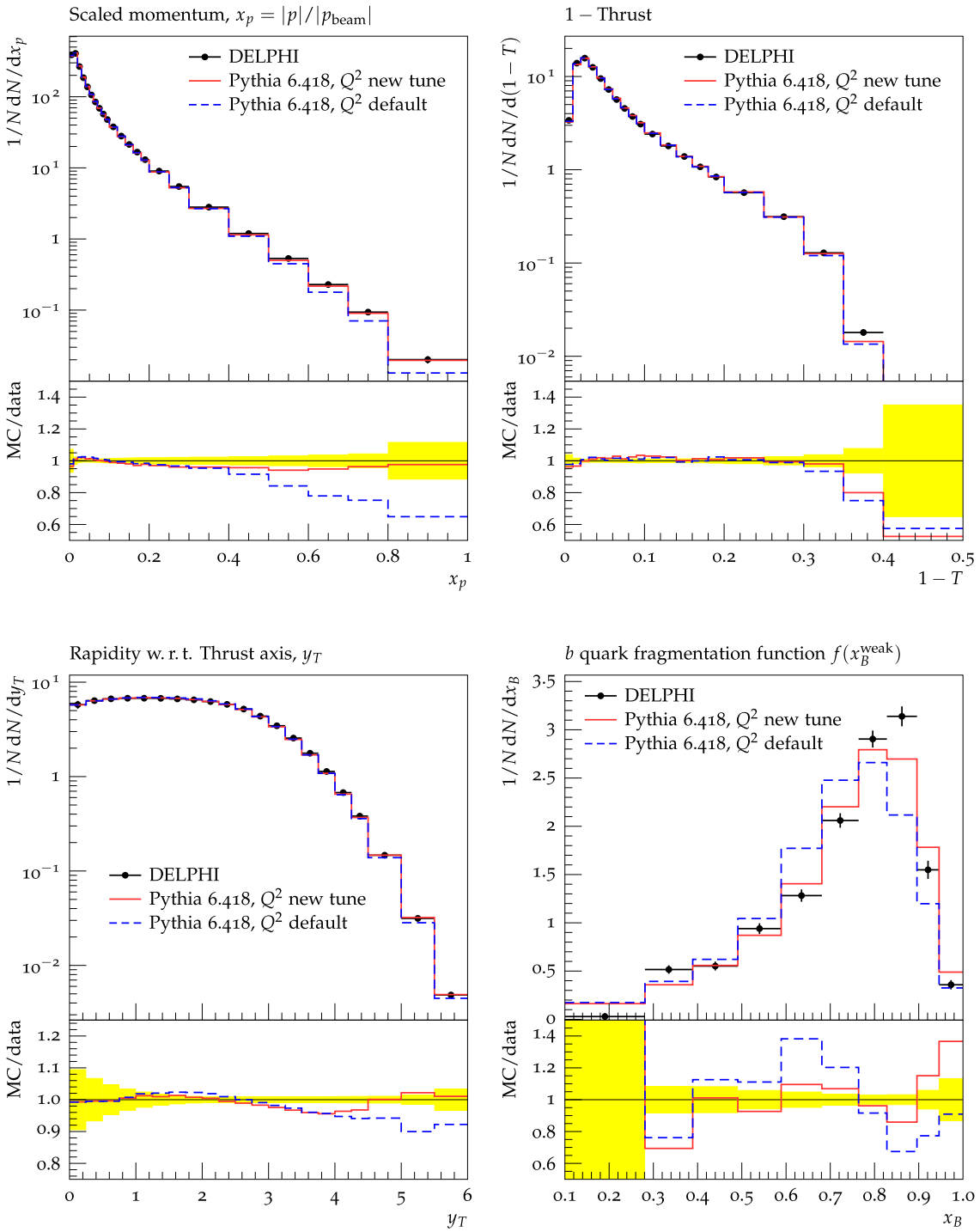


Fig. 5 Some example distributions for e^+e^- collisions using the virtuality-ordered shower. The *solid line* shows the new tune, the *dashed line* is the default. Even though the virtuality-ordered shower is well-tested and Pythia has been tuned several times, especially by the LEP collaborations, there is still room for improvement in the default settings. Note the different scale in the ratio plot of the rapidity distribution. The data in these plots has been published by DELPHI [10, 26]

Figure 5 shows some comparison plots between the Pythia default and our new tune of the virtuality-ordered shower. Even though this shower has been around for many years, and Pythia has been tuned before in this mode, there was still clearly room for improvement in the default settings.

Figure 6 shows comparisons of the $p_\perp$-ordered shower. This shower is a new option in Pythia and has not been tuned

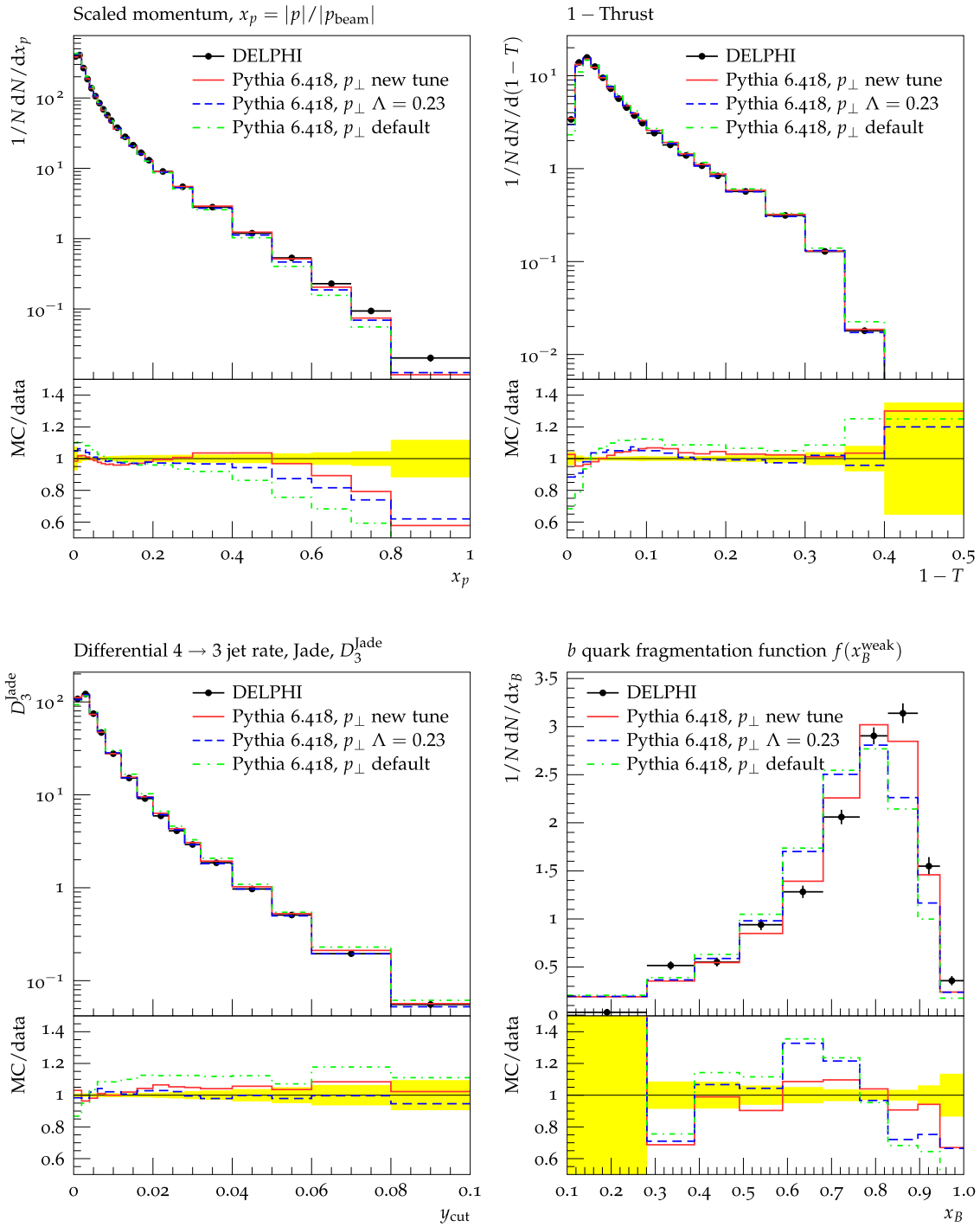


Fig. 6 Some example distributions for e^+e^- collisions using the $p_\perp$-ordered shower. The *solid line* shows the new tune, the *dashed line* is the old recommendation for using the $p_\perp$-ordered shower (i.e. changing Λ_{QCD} to 0.23), and the *dashed–dotted line* is produced by switching on the $p_\perp$-ordered shower leaving everything else at its default (the choice made for the ATLAS tune). The data has been published by DELPHI [10, 26]

systematically before. Nevertheless, the Pythia manual recommends to set Λ_{QCD} to 0.23. The ATLAS collaboration in their 2008 production tune chose to leave this parameter as set for the Q^2 shower, so for a full breadth of comparison our plots show our new tune, the default with $\Lambda_{\text{QCD}} = 0.23$, and the settings used by ATLAS [28].

6.4 Underlying event and multiple parton interactions

For the third step we tuned the parameters relevant to the underlying event, again both for the virtuality-ordered shower

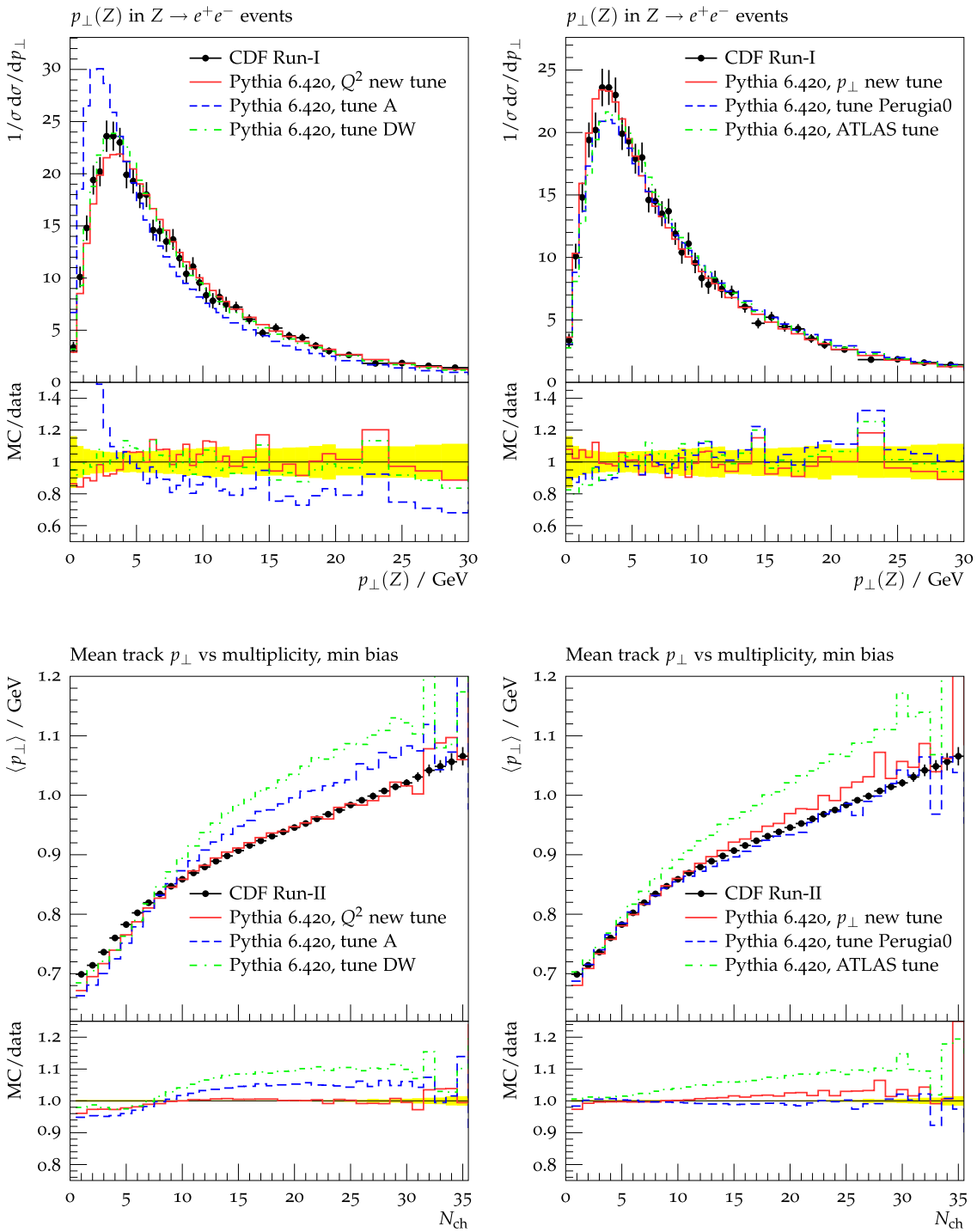


Fig. 7 The *upper plots* show the $Zp_\perp$ distribution as measured by CDF [29] compared to different tunes of the virtuality-ordered shower with the old MPI model (*left*) and the $p_\perp$-ordered shower with the interleaved MPI model (*right*). Except for tune A, all tunes describe this observable; the fixed version of tune A, called AW, is basically identical to DW. The *lower plots* show the average track $p_\perp$ as a function of the charged multiplicity in minimum bias events [32]. This observable is quite sensitive to colour reconnection. Only the recent tunes hit the data here (except for ATLAS)

and the old MPI model, and for the $p_\perp$-ordered shower with the interleaved MPI model. This was based on various Drell-Yan, jet physics, and minimum bias measurements performed by CDF and DØ in Run-I and Run-II [29–35]. Table 8 lists all observables and their corresponding weights used in the tuning.

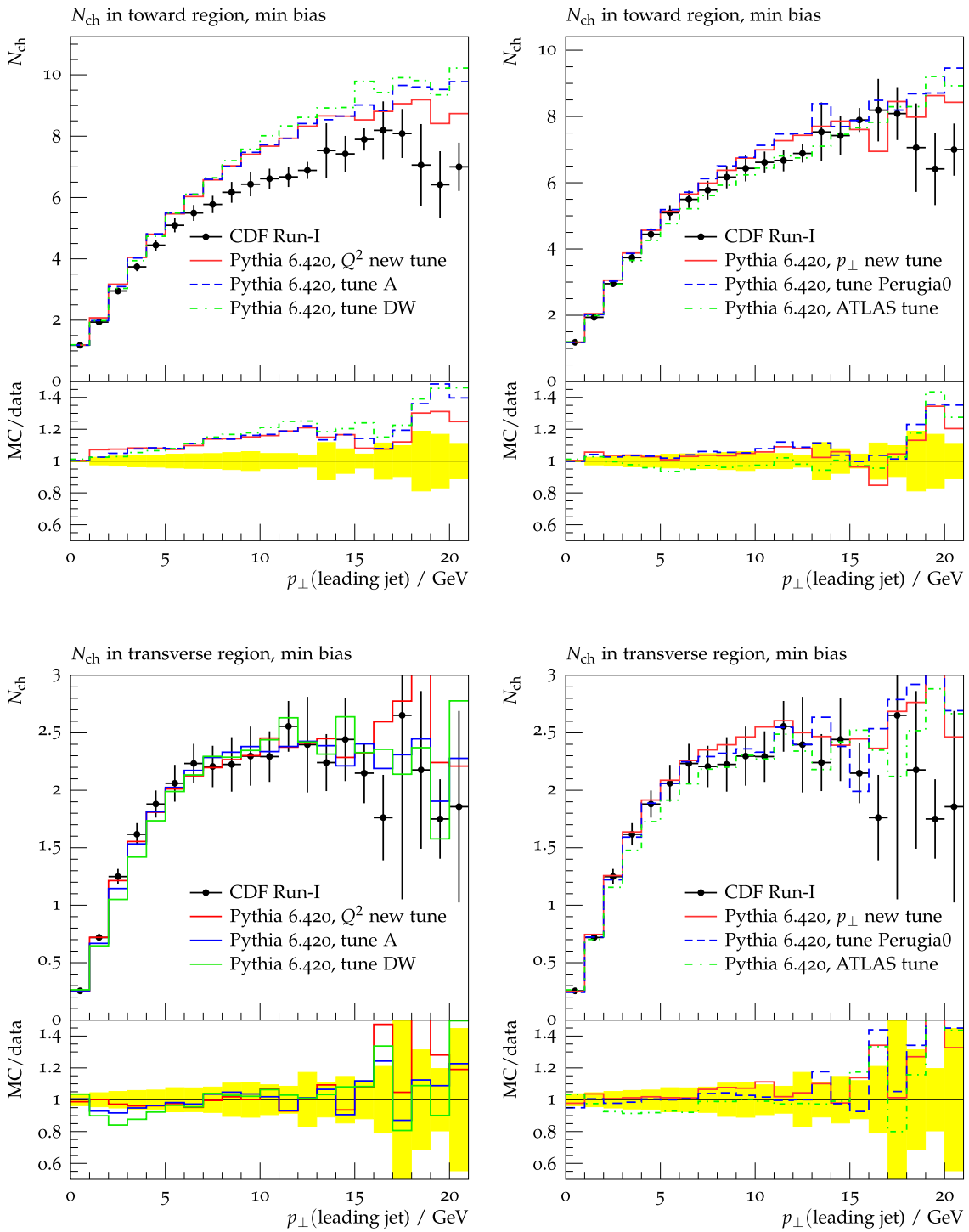


Fig. 8 These plots show the average charged multiplicity in the toward and transverse regions as function of the leading jet $p_\perp$ in minimum bias events [30]. The *left side* shows tunes of the virtuality-ordered shower with the old MPI model, while on the *right side* the $p_\perp$-ordered shower with the interleaved MPI model is used. The old model is known to be a bit too "jetty" in the toward region, which can be seen in the first plot. Other than this, all tunes are very similar

The new MPI model differs significantly from the old one, hence we had to tune different sets of parameters for these two cases. For the virtuality-ordered shower and old MPI model we took Rick Field's tune DW [36] as guideline. In the case of the new model we consulted Peter Skands, author of the new MPI model, and used a setup similar to his tune SØ [37, 38] as starting point. All switches and parameters for the UE/MPI tune, and our results, are listed in

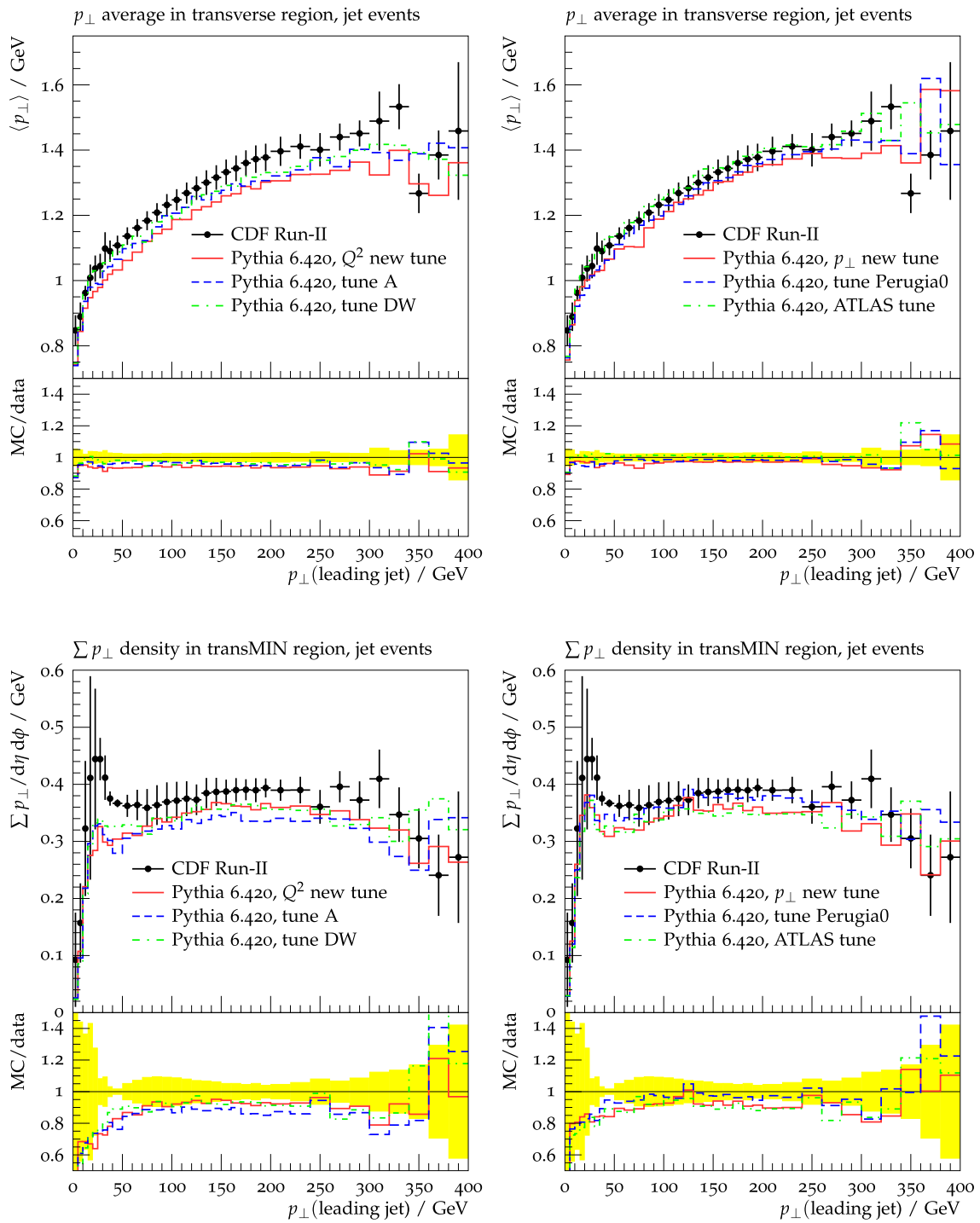


Fig. 9 These plots show the average track $p_\perp$ in the transverse region (*top*) and the $\sum p_\perp$ density in the transMIN region (*bottom*) in leading jet events [34]. The new model (on the *right*) seems to have a slight advantage over the virtuality-ordered shower with the old MPI model shown on the *left*, both in the turn-on hump and in overall activity

Tables 4 and 5. Correlation coefficients for the fragmentation and $p_\perp$-ordered UE parameters are listed in Tables 10 and 11.

One of the main differences we observed between the models is their behaviour in Drell-Yan physics. The old model had difficulties describing the $Zp_\perp$ spectrum [29] and we had to assign a high weight to that observable in order to force the Monte Carlo to get the peak region of the distribution right (note that this is the only observable to which we assigned different weights for the tunes of the old and

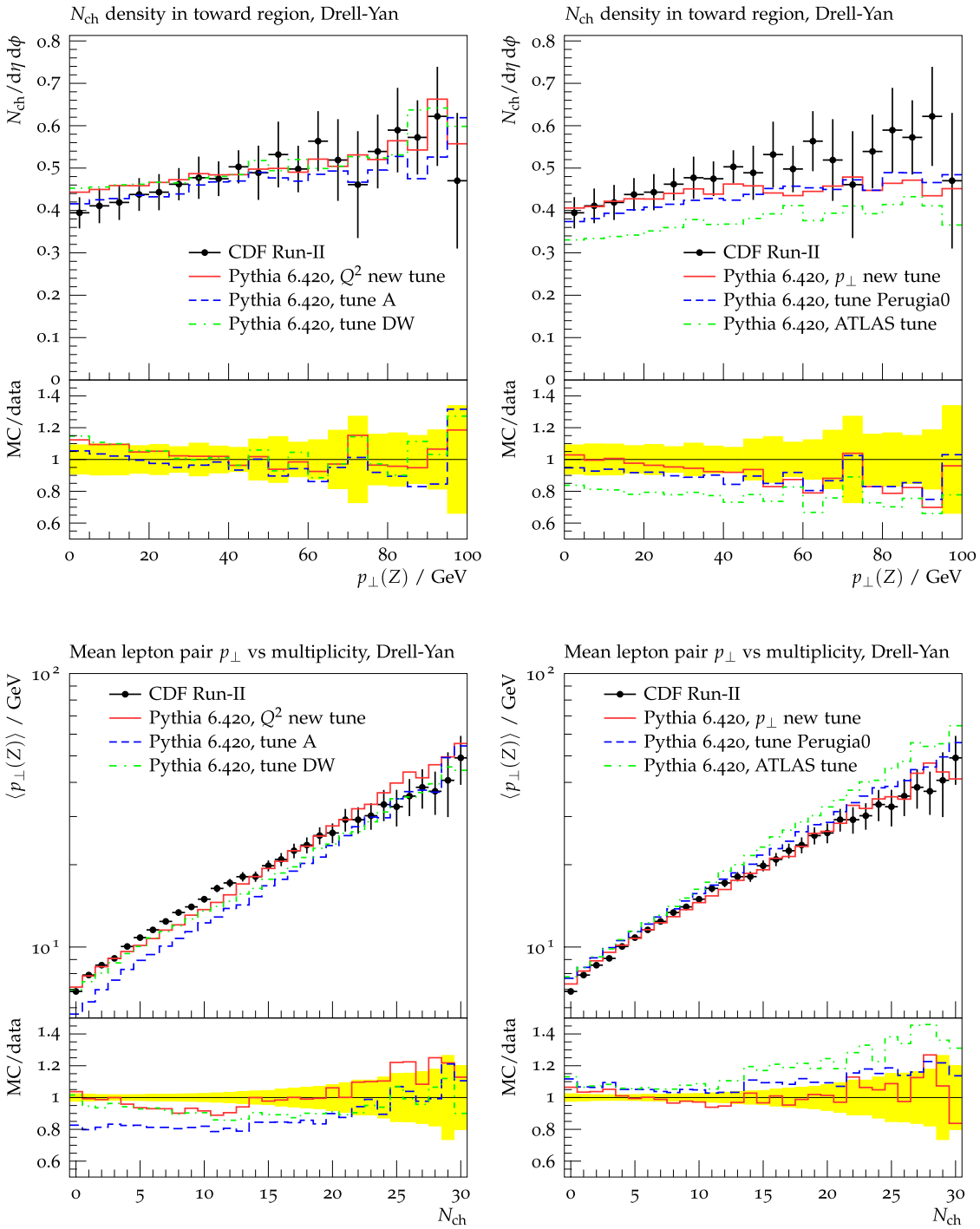


Fig. 10 In Drell-Yan [33] the new MPI model consistently produces less underlying event activity than for the old model (*top plots*). This underestimation is particularly pronounced for the ATLAS tune. Nevertheless, most of the recent tunes are able to describe the multiplicity dependence of the $Zp_\perp$ (*bottom plots*)

the new MPI model). The new model on the other hand gets the $Zp_\perp$ correct almost out of the box, but underestimates the underlying event activity in Drell-Yan events as measured in [33]. The same behaviour can be observed in Peter Skands' tunes [39]. We are currently investigating this issue together with the generator authors.

Another (albeit smaller) difference shows in the hump of the turn-on in many of the UE distributions in jet physics. This hump is described by the new model, but mostly missing in the old model. Although the origin of this hump is thought to be understood as an ambiguity in defining the event direction for events with only little and soft activity,

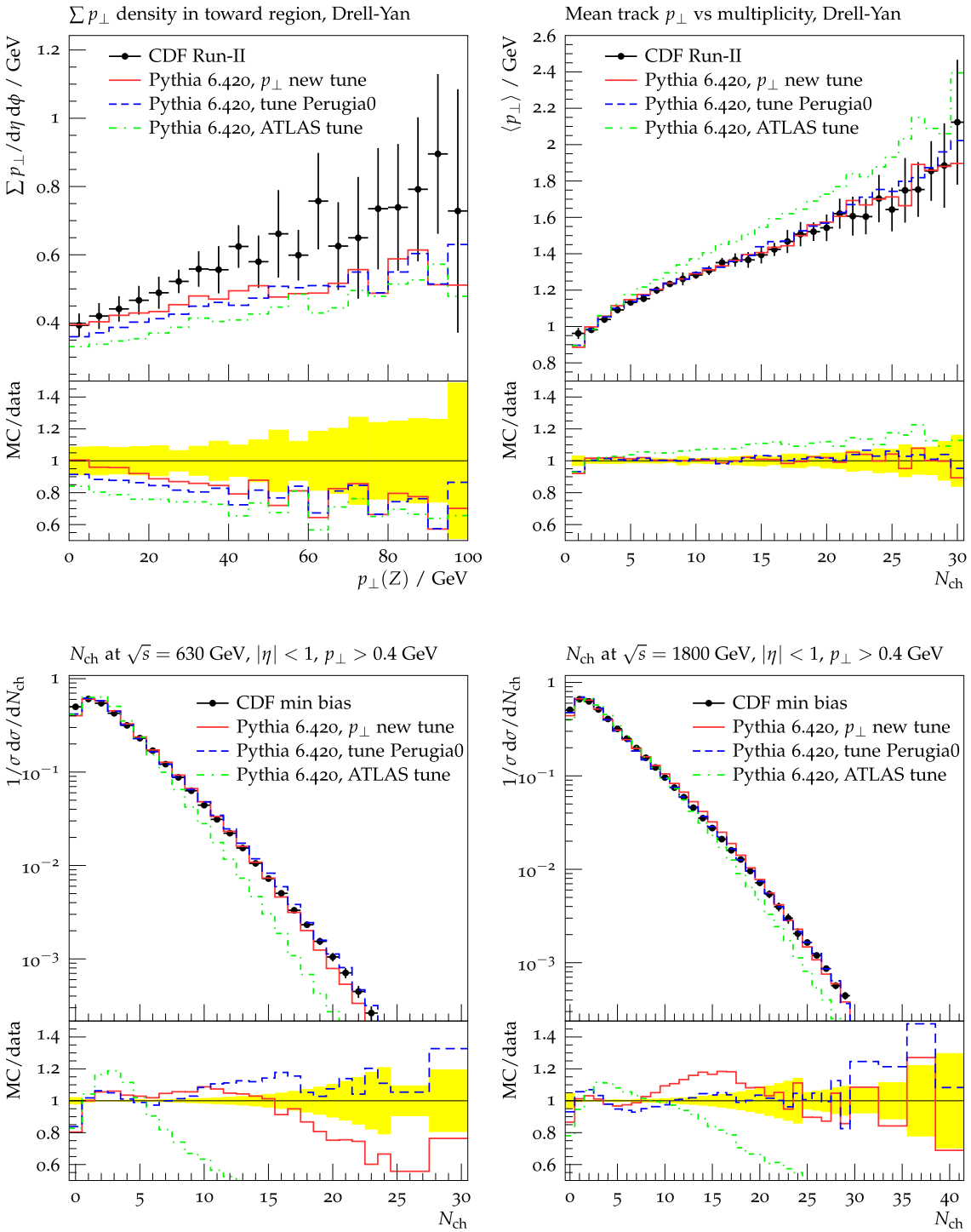


Fig. 11 Some more plots showing the behaviour of the interleaved MPI model and the $p_\perp$-ordered shower. The two *upper plots* focus on the underlying event in Drell-Yan [33]. On the *left* we see again that the new model underestimates the activity in Drell-Yan events (like in Fig. 10). Regardless of that, the *top right plot* shows that the average track $p_\perp$ as function of the charged multiplicity is described well, except by the ATLAS tune. The ATLAS tune also shows strong disagreement with the multiplicity distribution in minimum bias events, even at the reference energy of 1800 GeV, as shown in the lower two plots [31]

the model differences responsible for its presence/absence in the two Pythia models is not yet known in any detail.

It should be noted that the parameters PARP(71) and PARP(79) could not be constrained very well with the observables tuned to.

Figures 7, 8, 9, 10, 11 show some comparisons between

our new tune and various other tunes. For the virtuality-ordered shower with the old MPI model we show Rick Field's tunes A [40] and DW [36] as references, since they are well-known and widely used. For the $p_\perp$-ordered shower and the new MPI framework we compare to Peter Skands' new "Perugia 0" tune [39]. We also include the 2008 ATLAS tune [28] in our comparison, since it is widely used at the LHC, but note that the energy scaling is strongly disfavoured by the existing data, as well as the issues discussed in Sect. 6.3.

6.5 Tune verification

Much effort has been put into the verification of the new tune. We have performed line-scan validations along the directions in parameter space that correspond to the largest and the smallest uncertainty based on the covariance matrix, $\mathcal{C}$, of the new tune's minimisation result. These directions should coincide with those where the GoF-function is very flat or very steep, respectively. The eigen-decomposition of $\mathcal{C}$ can be written as

$$\mathcal{C} = T^{\mathsf{T}} \Sigma T \tag{14}$$

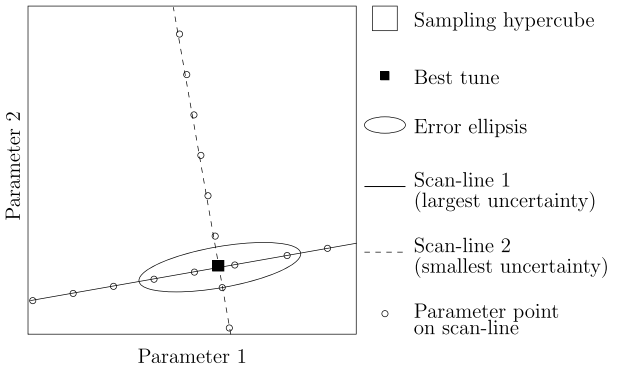
where T is a rotation matrix and Σ a diagonal matrix. The eigenvalues of Σ are related to the axes of the rotated hyper-ellipsoid of $\mathcal{C}$. The eigenvectors $\boldsymbol{\sigma}_{\text{max, min}}$ of the largest and the smallest (absolute) eigenvalues are rotated back into the original system and used to define the scan-lines:

$$\boldsymbol{d}_{\text{max, min}} = T \boldsymbol{\sigma}_{\text{max, min}}. \tag{15}$$

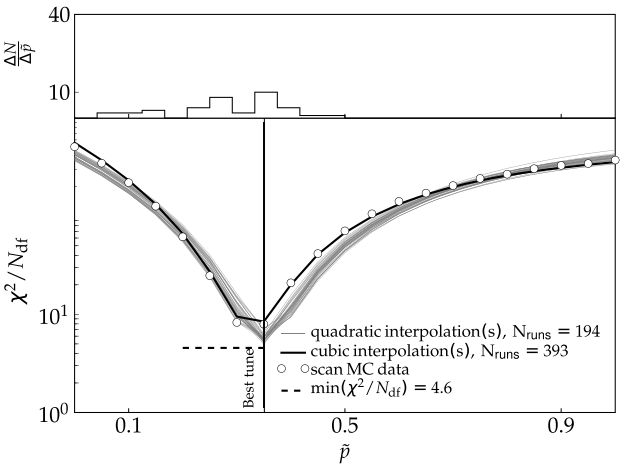
A schematic illustration of this procedure for a two dimensional case can be found in Fig. 12(a). It was checked that the parameter points sampled from these lines are kept in the region of interpolation. From the parameterisations available, only those where the obtained minima project onto the according scan line were chosen for the line-scan. The line-scans can be found in Figs. 12(b) and 12(c). It shows that the parameterisations and the actual generator response are in very good agreement, especially in the region of the minimum. However, in the case of Fig. 12(c) the quadratic interpolations seem to shift the line-scan a little to the left but the deviation is inside the parameter uncertainties indicated by the grey vertical band. In both cases we observe an even better description of the generator response by the cubic interpolation, especially in regions further away from the minimum.

6.6 Tuning stability

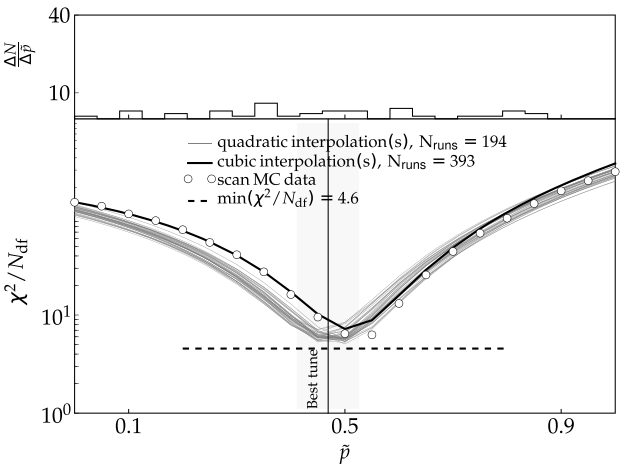
An example parameter-wise comparison of the goodness of fit of minimisation results obtained from 100 quadratic parameterisations that use 194 runs as well as from two parameterisations (one quadratic, one cubic) that use 393 runs



(a) Determining directions of largest/smallest uncertainty



(b) Line-scan along direction of smallest uncertainty



(c) Line-scan along direction of largest uncertainty

Fig. 12 Line-scan validation of the best tuning estimate obtained with Professor for the underlying event ($p_\perp$-ordered shower). The sketch in (**a**) illustrates how the directions of largest and smallest uncertainty are found based on a best tune's covariance matrix for a toy two dimensional case—the real UE tune has 10 parameters. The line-scans are done along the direction of the smallest uncertainty in (**b**) and the largest uncertainty in (**c**), parameterised by $\tilde{p} \in [0, 1]$. The *histograms* on *top* of the line-scans show the distribution of minimisation results obtained with the corresponding parameterisations. The *gray band* in (**c**) indicates the parameter uncertainties (estimated by the minimiser) of the best tune estimate as quoted by Minuit. The corresponding band in (**b**) is a *thin line*, since the best tune value is very well-defined

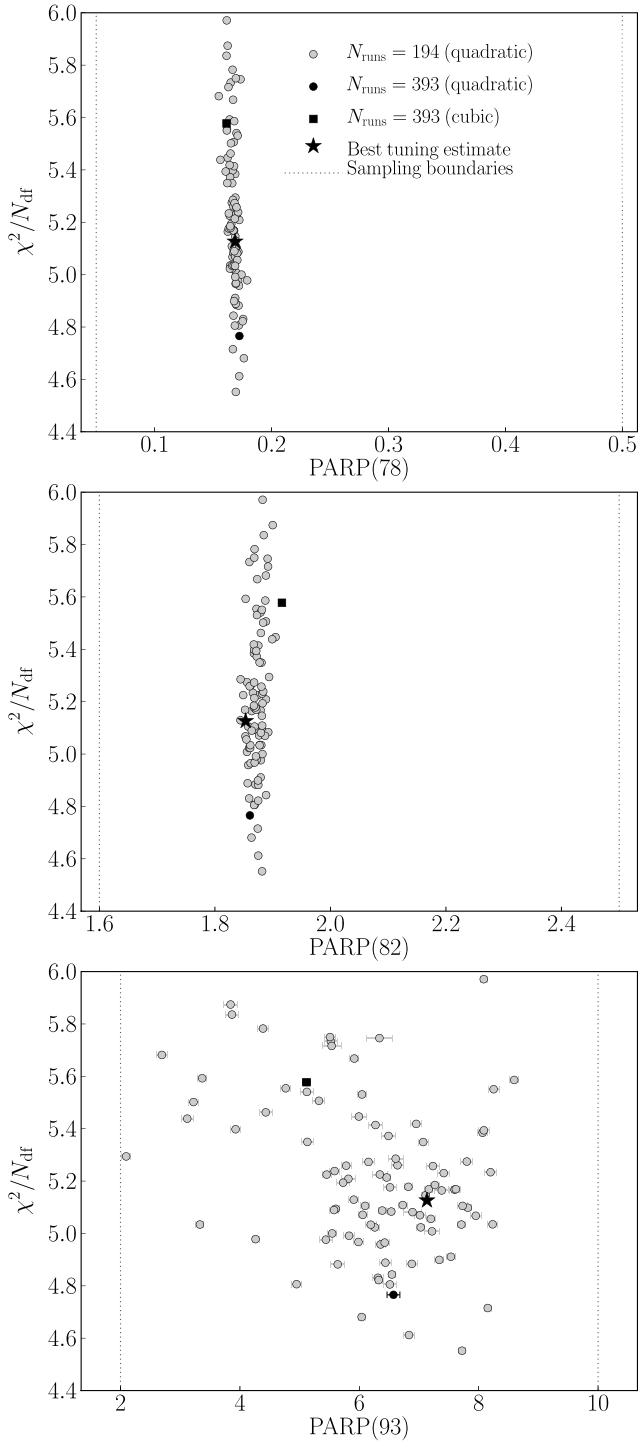


Fig. 13 Examples of parameter-wise distributions of minimisation results obtained for the underlying event ($p_\perp$-ordered shower) with the weights found in Table 8 and various N_{tune} and polynomial orders. For each parameter the goodness of fit of each minimisation result is projected on to the corresponding parameter axis. The minimum is clearly well-defined for some parameters like PARP(78) or PARP82; others are scattered over the whole parameter range, e.g. PARP(93), indicating that we are not sensitive to these parameters, i.e. they are not very important for a good description of the data we tuned to. The best tune estimate, indicated by a star, was derived from a quadratic parameterisation that uses 194 runs

can be found for the three parameters PARP(78), PARP(82) and PARP(93) in Fig. 13. In the case of PARP(78) we find a very well defined minimum, indicating that the used observables are sensitive to this parameter. A similar picture with a somewhat larger spread is found for PARP(82). Here, the cubic parameterisation predicts a slightly larger value than the quadratic ones do. PARP(93) does not show a well defined minimum, indicating that the observables we tuned to are not very sensitive to this parameter, i.e., that PARP(93) is not very important for a good description of the data. However the goodness of fit values suggest that a large value (>5 GeV) is preferred.

It has been observed that the scattering of the minimisation results among each other is *larger* compared to the errors calculated by Minuit, indicating that for the tuning of the underlying event the choice of run-combinations is accompanied by a systematic error. This stands in opposition to the tunings of flavour- and fragmentation parameters.

7 Conclusions and outlook

MC generator tuning is currently in something of a boom era: there has been much activity in systematising tuning and validation in the past years, driven by the growing realisation that MPI effects will be a highly significant effect at the LHC and that existing data places only relatively weak constraints on their scale. In the last year, interest has been gradually converging on the Rivet and Professor tools; as demonstrated in this paper, these are now in a state where they can be used to achieve real physics goals and the Pythia 6 tunes described in Sect. 6 have been a significant success, bringing new accuracy, speed and systematic control to this previously vague topic.

Our development plans in the near future are primarily aimed at tuning of the newer C++ generator codes (Pythia 8, Sherpa & Herwig++) to e^+e^- and hadron collider data as we have done here for Pythia 6. As well as the hadron collider data shown here, Rivet's coverage of analyses is being extended to include low energy data from UA1, UA5, RHIC and other experiments. B-factory data, if it can be obtained, and HERA data via HZTool will help to improve further the MPI, parton shower, and hadronisation tunes and to challenge existing models. We have also been able to use Professor in the setup described here for similar tunings of Pythia 6 to alternative PDFs, to be described in a separate note.

Professor is under consideration within the LHC collaborations as part of the machinery to perform re-tunes of MC generators to early QCD-dominated data: this is a crucial step for understanding the underlying event at LHC energies, since low energy data provides little constraint on the evolution of the total pp cross-section. We are collaborating with both ATLAS and CMS to ensure that pre-publication

data can be best used to rapidly improve the simulation of backgrounds to new physics searches.

There is clearly more potential for exploration and innovation in connection with the Professor method and tools. One idea currently being investigated is the provision of representative "error tunes" (cf. "error PDFs") and uncertainty bands to give a quantitative estimate on how much models and tunes may be expected to deviate from data; this is particularly relevant for extrapolations such as the UE energy evolution. Other ideas include the use of Professor to make fast predictions of generator distributions for e.g. SUSY parameter space scans, optimisation of parameterised observables such as jet measures against user-defined goodness of fit functions, or even multidimensional fits in experimental analyses, e.g. a simultaneous fit of top mass and jet energy scale using MC templates. We look forward to challenging current MC models with the combination of LHC data and the new trend for statistically robust parameter exploration.

Acknowledgements We would like to thank Frank Krauss for convening the Professor collaboration and for myriad useful discussions. This work was supported in part by the MCnet European Union Marie Curie Research Training Network (contract MRTN-CT-2006-035606), which provided funding for collaboration meetings and attendance at research workshops such as ACAT08 and MPI@LHC.

Andy Buckley has been principally supported by a Special Project Grant from the UK Science & Technology Funding Council. Hendrik Hoeth acknowledges a MCnet postdoctoral fellowship.

Appendix A: Tables

Table 2 Tuned flavour parameters and their defaults

Parameter	Pythia 6.418 default	Final tune	
PARJ(1)	0.1	0.073	Diquark suppression
PARJ(2)	0.3	0.2	Strange suppression
PARJ(3)	0.4	0.94	Strange diquark suppression
PARJ(4)	0.05	0.032	Spin-1 diquark 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	0.63	η suppression
PARJ(26)	0.4	0.12	η' suppression

Table 3 Tuned fragmentation parameters and their defaults for the virtuality and $p_\perp$-ordered showers

Parameter	Pythia 6.418 default	Final tune (Q^2)	Final tune ($p_\perp$)	
MSTJ(11)	4	5	5	Frag. function
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	0.67	1.0	r_b
PARJ(81)	0.29	0.29	0.257	Λ_{QCD}
PARJ(82)	1	1.65	0.8	Shower cut-off

Table 4 Tuned parameters for the underlying event using the virtuality-ordered shower

Parameter	Pythia 6.418 default	Final tune	
PARP(62)	1.0	2.9	ISR cut-off
PARP(64)	1.0	0.14	ISR scale factor for α_S
PARP(67)	4.0	2.65	Max. virtuality
PARP(82)	2.0	1.9	$p_\perp^0$ at reference E_{cm}
PARP(83)	0.5	0.83	Matter distribution
PARP(84)	0.4	0.6	Matter distribution
PARP(85)	0.9	0.86	Colour connection
PARP(86)	1.0	0.93	Colour connection
PARP(90)	0.2	0.22	$p_\perp^0$ energy evolution
PARP(91)	2.0	2.1	Intrinsic $k_\perp$
PARP(93)	5.0	5.0	Intrinsic $k_\perp$ cut-off

Table 5 Tuned parameters (upper table) and switches (lower table) for the underlying event using the $p_\perp$-ordered shower

Parameter	Pythia 6.418 default	Final tune	
PARP(64)	1.0	1.3	ISR scale factor for α_S
PARP(71)	4.0	2.0	Max. virtuality (non-s-channel)
PARP(78)	0.03	0.17	Colour reconnection in FSR
PARP(79)	2.0	1.18	Beam remnant x enhancement
PARP(80)	0.1	0.01	Beam remnant breakup suppression
PARP(82)	2.0	1.85	$p_\perp^0$ at reference E_{cm}
PARP(83)	1.8	1.8	Matter distribution
PARP(90)	0.16	0.22	$p_\perp^0$ energy evolution
PARP(91)	2.0	2.0	Intrinsic $k_\perp$
PARP(93)	5.0	7.0	Intrinsic $k_\perp$ cut-off

Switch	Value	Effect
MSTJ(41)	12	Switch on $p_\perp$-ordered shower
MSTP(51)	7	Use CTEQ5L
MSTP(52)	1	Use internal PDF set
MSTP(70)	2	Model for smooth $p_\perp^0$
MSTP(72)	0	FSR model
MSTP(81)	21	Turn on multiple interactions (new model)
MSTP(82)	5	Model of hadronic matter overlap
MSTP(88)	0	Quark junctions → diquark/Baryon model
MSTP(95)	6	Colour reconnection

Table 6 Observables and weights included in the flavour tune

Observable	Weight
b quark frag. function $f(x_B^{\mathrm{weak}})$	1
Mean of b quark frag. function $f(x_B^{\mathrm{weak}})$	1
uds events mean charged multiplicity	1
c events mean charged multiplicity	1
b events mean charged multiplicity	1
All events mean charged multiplicity	1
$\pi^\pm$ multiplicity	1
π^0 multiplicity	1
$\pi^0/\pi^\pm$ multiplicity ratio	6
$K^+/\pi^\pm$ multiplicity ratio	6
$K^0/\pi^\pm$ multiplicity ratio	6
$\eta/\pi^\pm$ multiplicity ratio	2
$\eta'(958)/\pi^\pm$ multiplicity ratio	1
$D^+/\pi^\pm$ multiplicity ratio	1
$D^0/\pi^\pm$ multiplicity ratio	1
$D_s^+/\pi^\pm$ multiplicity ratio	2
$(B^+, B_d^0)/\pi^\pm$ multiplicity ratio	1
$B^+/\pi^\pm$ multiplicity ratio	1
$B_s^0/\pi^\pm$ multiplicity ratio	2

Table 6 (*continued*)

Observable	Weight
$\rho^0(770)/\pi^\pm$ multiplicity ratio	9
$\rho^+(770)/\pi^\pm$ multiplicity ratio	9
$\omega(782)/\pi^\pm$ multiplicity ratio	9
$K^{*+}(892)/\pi^\pm$ multiplicity ratio	2
$K^{*0}(892)/\pi^\pm$ multiplicity ratio	2
$\phi(1020)/\pi^\pm$ multiplicity ratio	1
$D^{*+}(2010)/\pi^\pm$ multiplicity ratio	1
$D_s^{*+}(2112)/\pi^\pm$ multiplicity ratio	1
$B^*/\pi^\pm$ multiplicity ratio	1
$p/\pi^\pm$ multiplicity ratio	3
$\Lambda/\pi^\pm$ multiplicity ratio	4
$\Sigma^0/\pi^\pm$ multiplicity ratio	2
$\Sigma^{pm}/\pi^\pm$ multiplicity ratio	2
$\Xi^-/\pi^\pm$ multiplicity ratio	1
$\Delta^{++}(1232)/\pi^\pm$ multiplicity ratio	1
$\Sigma^{pm}(1385)/\pi^\pm$ multiplicity ratio	1

Table 7 Observables and weights included in the fragmentation tune

Observable	Weight (Q^2)	Weight ($p_\perp$)
$p_\perp^{\text{in}}$ w.r.t. thrust axes	1	2
$p_\perp^{\text{out}}$ w.r.t. thrust axes	1	1
$p_\perp^{\text{in}}$ w.r.t. sphericity axes	1	2
$p_\perp^{\text{out}}$ w.r.t. sphericity axes	1	1
Scaled momentum, $x_p = \|p\|/\|p_{\text{beam}}\|$	1	3
Log of scaled momentum, $\log 1/x_p$	1	3
Mean $p_\perp^{\text{out}}$ vs x_p		1
Mean $p_\perp$ vs x_p		1
$1 - \text{thrust}$, $1 - T$	1	6
Thrust major, M	1	4
Thrust minor, m	1	4
Oblateness $= M - m$	1	1
Sphericity, S	1	1
Aplanarity, A	1	1
Planarity, P	1	1
C parameter	1	1
D parameter	1	4
Energy-energy correlation, EEC		1
Mean charged multiplicity	160	181
b quark frag. function $f(x_B^{\text{weak}})$	1	2
Mean of b quark frag. function $f(x_B^{\text{weak}})$	1	4
uds events mean charged multiplicity	20	10
c events mean charged multiplicity	20	10
b events mean charged multiplicity	20	10
uds events scaled momentum, $x_p = \|p\|/\|p_{\text{beam}}\|$		1
c events scaled momentum, $x_p = \|p\|/\|p_{\text{beam}}\|$		1
b events scaled momentum, $x_p = \|p\|/\|p_{\text{beam}}\|$		1
uds events log of scaled momentum, $x_p = \|p\|/\|p_{beam}\|$		1
c events log of scaled momentum, $x_p = \|p\|/\|p_{\text{beam}}\|$		1
b events log of scaled momentum, $x_p = \|p\|/\|p_{\text{beam}}\|$		1

Table 8 Observables and weights used for the tuning of the underlying event

Observable	Weight
CDF underlying event in min-bias events:	
$p_\perp(Z)$	40 (Q^2) / 10 ($p_\perp$)
N_{ch} density vs leading jet $p_\perp$ (toward), min-bias	1
N_{ch} density vs leading jet $p_\perp$ (transverse), min-bias	1
N_{ch} density vs leading jet $p_\perp$ (away), min-bias	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (toward), min-bias	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (transverse), min-bias	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (away), min-bias	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (toward), JET20	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (transverse), JET20	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (away), JET20	1
$p_\perp$ distribution (transverse), leading $p_\perp > 30$ GeV	1
CDF multiplicity measurement:	
N_{ch} distribution at 630 GeV	2
N_{ch} distribution at 1800 GeV	2
CDF underlying event in leading jet events:	
N_{ch} density vs leading jet $p_\perp$ (transverse)	1
N_{ch} density vs leading jet $p_\perp$ (transMAX)	1
N_{ch} density vs leading jet $p_\perp$ (transMIN)	1
N_{ch} density vs leading jet $p_\perp$ (transDIF)	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (transverse)	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (transMAX)	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (transMIN)	1
$\sum p_\perp$ density vs leading jet $p_\perp$ (transDIF)	1
$\langle p_\perp \rangle$ (transverse)	1
CDF min-bias:	
$\langle p_\perp \rangle$ vs N_{ch}	2
CDF underlying event in Drell-Yan events analysis:	
N_{ch} density vs lepton pair $p_\perp$ (toward)	1
N_{ch} density vs lepton pair $p_\perp$ (transverse)	1
N_{ch} density vs lepton pair $p_\perp$ (transMAX)	1
N_{ch} density vs lepton pair $p_\perp$ (transMIN)	1
N_{ch} density vs lepton pair $p_\perp$ (transDIF)	1
N_{ch} density vs lepton pair $p_\perp$ (away)	1
$\sum p_\perp$ density vs lepton pair $p_\perp$ (toward)	1
$\sum p_\perp$ density vs lepton pair $p_\perp$ (transverse)	1
$\sum p_\perp$ density vs lepton pair $p_\perp$ (transMAX)	1
$\sum p_\perp$ density vs lepton pair $p_\perp$ (transMIN)	1
$\sum p_\perp$ density vs lepton pair $p_\perp$ (transDIF)	1
$\sum p_\perp$ density vs lepton pair $p_\perp$ (away)	1
$\langle p_\perp \rangle$ (toward)	1
$\langle p_\perp \rangle$ (transverse)	1
$\langle p_\perp \rangle$ (away)	1
$p_\perp^{\mathrm{max}}$ (toward)	1
$p_\perp^{\mathrm{max}}$ (transverse)	1

Table 8 (*continued*)

Observable	Weight
$p_\perp^{\max}$ (away)	1
$\langle p_\perp(\text{lepton pair})\rangle$ vs N_{ch}	1
$\langle p_\perp\rangle$ vs N_{ch}	1
$\langle p_\perp\rangle$ vs N_{ch}, $p_\perp(Z) < 10$ GeV	1
DØ dijet angular correlations:	
dijet azimuthal angle, $p_\perp^{\max} \in [75, 100]$ GeV	2
dijet azimuthal angle, $p_\perp^{\max} \in [100, 130]$ GeV	2
dijet azimuthal angle, $p_\perp^{\max} \in [130, 180]$ GeV	2
dijet azimuthal angle, $p_\perp^{\max} > 180$ GeV	2

Table 9 Correlation coefficients for flavour parameters as calculated by Minuit

	PARJ(1)	PARJ(2)	PARJ(3)	PARJ(4)	PARJ(11)
PARJ(1)	1	0.32	−0.75	−0.34	0.41
PARJ(2)		1	−0.39	−0.26	0.71
PARJ(3)			1	0.63	−0.35
PARJ(4)				1	−0.33
PARJ(11)					1

	PARJ(12)	PARJ(13)	PARJ(25)	PARJ(26)
PARJ(1)	0.05	0.05	0.20	0.31
PARJ(2)	−0.08	0.13	0.44	0.43
PARJ(3)	0.04	−0.05	−0.15	−0.33
PARJ(4)	−0.04	−0.08	−0.50	−0.27
PARJ(11)	0.01	0.07	0.41	0.28
PARJ(12)	1	0.02	0.09	−0.04
PARJ(13)		1	−0.01	0.08
PARJ(25)			1	0.04
PARJ(26)				1

Table 10 Correlation coefficients for fragmentation parameters as calculated by Minuit

	PARJ(21)	PARJ(41)	PARJ(42)	PARJ(47)	PARJ(81)	PARJ(82)
PARJ(21)	1	0.55	0.40	0.22	−0.33	0.50
PARJ(41)		1	0.95	0.40	0.15	0.74
PARJ(42)			1	0.47	0.31	0.52
PARJ(47)				1	0.07	0.18
PARJ(81)					1	0.04
PARJ(82)						1

Table 11 Correlation coefficients for underlying event parameters ($p_\perp$-ordered shower)

	PARP(64)	PARP(71)	PARP(78)	PARP(79)	PARP(82)
PARP(64)	1	0.26	−0.17	−0.15	−0.65
PARP(71)		1	0.39	0.04	−0.26
PARP(78)			1	−0.45	−0.17
PARP(79)				1	0.15
PARP(82)					1

Table 11 (*continued*)

	PARP(83)	PARP(90)	PARP(91)	PARP(93)
PARP(64)	0.48	−0.18	−0.18	−0.19
PARP(71)	0.42	−0.18	−0.24	−0.38
PARP(78)	0.44	−0.12	−0.36	0.20
PARP(79)	−0.18	0.17	0.00	−0.26
PARP(82)	−0.58	0.65	0.15	0.10
PARP(83)	1	−0.21	−0.18	0.12
PARP(90)		1	−0.09	−0.08
PARP(91)			1	0.27
PARP(93)				1

Appendix B: Comparisons

Table 12 Mean hadron multiplicities in e^+e^- collisions at 91 GeV for data [25], Pythia 6.418 default and our tune using the virtuality-ordered shower. While there is a slight degradation in charm and bottom mesons, the strange sector is significantly improved (mesons and baryons), and also particles like ρ and ω clearly benefit from the tuning

Particle	Data	Pythia 6.418 default	Final tune
π^+	17.02 ± 0.19	17.10	17.23
π^0	9.42 ± 0.32	9.69	9.78
K^+	2.228 ± 0.059	2.311	2.126
K^0	2.049 ± 0.026	2.211	2.068
η	1.049 ± 0.08	1.012	1.014
$\eta'(958)$	0.152 ± 0.02	0.301	0.171
D^+	0.175 ± 0.016	0.166	0.219
D^0	0.454 ± 0.03	0.494	0.490
D_s^+	0.131 ± 0.021	0.128	0.101
B^+, B_d^0	0.33 ± 0.052	0.346	0.373
B_u^+	0.178 ± 0.006	0.173	0.186
B_s^0	0.057 ± 0.013	0.052	0.037
$\rho^0(770)$	1.231 ± 0.098	1.523	1.267
$\rho^+(770)$	2.4 ± 0.43	2.86	2.40
$\omega(782)$	1.016 ± 0.065	1.367	1.150
$K^{*+}(892)$	0.715 ± 0.059	1.111	0.740
$K^{*0}(892)$	0.738 ± 0.024	1.106	0.743
$\phi(1020)$	0.0963 ± 0.0032	0.1942	0.1006
$D^{*+}(2010)$	0.1937 ± 0.0057	0.2395	0.1974
$D_s^{*+}(2112)$	0.101 ± 0.048	0.090	0.058
B^*	0.288 ± 0.026	0.299	0.221
p	1.05 ± 0.032	1.221	1.117
Λ	0.3915 ± 0.0065	0.3922	0.3507
Σ^0	0.076 ± 0.011	0.075	0.096
Σ^-	0.081 ± 0.01	0.069	0.091
Σ^+	0.107 ± 0.011	0.074	0.095
$\Sigma^\pm$	0.174 ± 0.009	0.143	0.186
Ξ^-	0.0258 ± 0.001	0.0278	0.0282
$\Delta^{++}(1232)$	0.085 ± 0.014	0.193	0.147
$\Sigma^-(1385)$	0.024 ± 0.0017	0.037	0.025
$\Sigma^+(1385)$	0.0239 ± 0.0015	0.0389	0.0266
$\Sigma^\pm(1385)$	0.0462 ± 0.0028	0.0757	0.0516

References

1. T. Sjostrand, S. Mrenna, P. Skands, J. High Energy Phys. **05**, 026 (2006). hep-ph/0603175
2. G. Corcella et al., hep-ph/0210213 (2002)
3. M. Bähr et al. (Herwig++ Collaboration), 0812.0529 (2008)
4. T. Gleisberg et al., 0811.4622 (2008)
5. M. Althoff et al. (TASSO Collaboration), Z. Phys. C **26**, 157 (1984)
6. W. Braunschweig et al. (TASSO Collaboration), Z. Phys. C **41**, 359 (1988)
7. D. Buskulic et al. (ALEPH Collaboration), Z. Phys. C **55**, 209 (1992)
8. R. Barate et al. (ALEPH Collaboration), Phys. Rep. **294**, 1 (1998)
9. K. Hamacher, M. Weierstall, hep-ex/9511011 (1995)
10. P. Abreu et al. (DELPHI Collaboration), Z. Phys. C **73**, 11 (1996)
11. B.M. Waugh et al., hep-ph/0605034 (2006)
12. J. Bromley et al. (ZEUS and H1 Collaborations), (1995)
13. M. Dobbs, J.B. Hansen, Comput. Phys. Commun. **134**, 41 (2001)
14. M. Cacciari, G. Salam, G. Soyez, hep-ph/0607071 (2006)
15. M.L. Mangano, M. Moretti, F. Piccinini, R. Pittau, A.D. Polosa, J. Hihg Energy Phys. **07**, 001 (2003). hep-ph/0206293
16. C.M. Harris, P. Richardson, B.R. Webber, J. High Energy Phys. **08**, 033 (2003). hep-ph/0307305
17. J.M. Butterworth, J.R. Forshaw, M.H. Seymour, Z. Phys. C **72**, 637 (1996). hep-ph/9601371
18. A.E. Albert, *Regression and the Moore-Penrose Pseudoinverse* (Academic Press, New York, 1972)
19. K.M. Brown, Computer oriented methods for fitting tabular data in the linear and nonlinear least squares sense, in *AFIPS '72 (Fall, part II): Proceedings of the December 5–7, 1972, Fall Joint Computer Conference, Part II*, New York, NY, USA (ACM, 1972), pp. 1309–1315
20. T.E. Oliphant, Comput. Sci. Eng. **9**, 10 (2007). NumPy website: http://numpy.scipy.org/
21. E. Jones et al., SciPy: Open source scientific tools for Python. SciPy website: http://www.scipy.org/
22. T. Rudd, M. Orr, C. Esterbrook, I. Bicking, Cheetah: Python template engine. Cheetah website: http://www.cheetahtemplate.org/
23. J. Cohen-Tanugi, PyMinuit2, http://code.google.com/p/pyminuit2/
24. F. James, M. Roos, Comput. Phys. Commun. **10**, 343 (1975)
25. C. Amsler et al. (Particle Data Group Collaboration), Phys. Lett. B **667**, 1 (2008)
26. G. Barker et al. (DELPHI Collaboration), DELPHI 2002-069 CONF 603 (2002)
27. K. Ackerstaff et al. (OPAL Collaboration), Eur. Phys. J. C **7**, 369 (1999). hep-ex/9807004
28. A. Moraes (ATLAS Collaboration), ATL-PHYS-PROC-2009-045 (2009)
29. T. Affolder et al. (CDF Collaboration), Phys. Rev. Lett. **84**, 845 (2000). hep-ex/0001021
30. T. Affolder et al. (CDF Collaboration), Phys. Rev. D **65**, 092002 (2002)
31. D. Acosta et al. (CDF Collaboration), Phys. Rev. D **65**, 072005 (2002)
32. T. Aaltonen et al. (CDF Collaboration), 0904.1098 (2009)
33. D. Kar, R. Field, (CDF Collaboration), CDF Note 9351 (2008)
34. R. Field (CDF Collaboration), The Underlying Event and Comparisons with MC, First International Workshop on Multiple Partonic Interactions at the LHC (2008)
35. V.M. Abazov et al. (D0 Collaboration), Phys. Rev. Lett. **94**, 221801 (2005). hep-ex/0409040
36. R. Field, "Minimum Bias" Collisions at CDF and CMS, talk presented at the CMS Meeting on Min-Bias Monte-Carlo Tunes, CERN, 28 June 2006
37. M. Sandhoff, P. Skands, hep-ph/0604120. Presented at Les Houches Workshop on Physics at TeV Colliders, Les Houches, France, 2–20 May 2005
38. P. Skands, D. Wicke, Eur. Phys. J. C **52**, 133 (2007). hep-ph/0703081
39. P.Z. Skands, 0905.3418 (2009)
40. R. Field, Min-Bias and the Underlying Event at the Tevatron and the LHC, talk presented at the Fermilab ME/MC Tuning Workshop, 4 October 2002