Tools for event generator tuning and validation

Andy Buckley
Institute for Particle Physics Phenomenology,
Durham University, UK

Abstract
I describe the current status of MCnet tools for validating the performance of event generator simulations against data, and for tuning their phenomenological free parameters. For validation, the Rivet toolkit is now a mature and complete system, with a large library of prominent benchmark analyses. For tuning, the Professor system has recently completed its first tunes of Pythia 6, with substantial improvements on the existing default tune and potential to greatly aid the setup of new generators for LHC studies.

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 tweaked if the generator is to describe experimental data. Such parameters may be found in most aspects of generator codes, from choices of Λ_{QCD} and $p_\perp$ cutoff in the perturbative parton cascade, to the non-perturbative hadronisation process. These latter account for the majority of parameters, since the models are deeply phenomenological, typically invoking a slew of numbers to describe not only the kinematic distribution of $p_\perp$ in hadron fragmentation, but also 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 $\mathcal{O}(10)$ and $\mathcal{O}(30)$ may be of particular importance for physics studies.

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. Additionally, it is important that this tuning is performed against a wide range of experimental analyses, since otherwise parameters to which the selective analyses are insensitive will wander freely and may drive unconsidered observables to bad or even unphysical places. This requires a systematic and global approach to generator tuning: accordingly, I will summarise the current state of tools for systematically validating and tuning event generator parameters, and the first results of such systematic tunings.

2 Validation tools: Rivet

The Rivet library is a successor to the successful HERA-oriented generator analysis library, HZTool [5]. Like its predecessor, the one library contains both a library of experimental analyses and tools for calculating physical observables. It is written in object-oriented C++ and there is strong emphasis on the following features:

- strict generator-independence: analyses are strictly performed on HepMC [6] event record objects with no knowledge of or ability to influence the generator behaviour;

- experimental reference data files are included for each standard analysis, and are used to ensure that analysis data binnings match their experimental counterparts as well as for fit comparisons;
- computational results are automatically cached for use between different analyses, using an infrastructure mechanism based on *projection* classes;
- clean, transparent and flexible programming interface: while much of the complexity is hidden, analyses retain a clear algorithmic structure rather than attempting to hide everything behind "magic" configuration files.

The "projection" objects used to compute complex observables are now a fairly complete set:

- various ways to obtain final state particles: all, charged only, excluding certain particles, with $p_\perp$ and rapidity cuts, etc.;
- event shapes: sphericity, thrust, Parisi C & D parameters, jet hemispheres;
- jet algorithms: CDF and DØ legacy cones, Durham/JADE, and $k_\perp$, anti-$k_\perp$, SISCONE, CDF "JETCLU" etc. from FastJet [7];
- miscellaneous: jet shapes, isolation calculators, primary and secondary vertex finders, DIS kinematics transforms, hadron decay finder, etc.

The set of standard analyses has also grown with time and is now particularly well-populated with analyses from the LEP and Tevatron experiments:

- LEP: ALEPH and DELPHI event shape analyses; ALEPH, DELPHI and PDG hadron multiplicities, strange baryons; DELPHI and OPAL b-fragmentation analyses;
- Tevatron: CDF underlying event analyses (from 2001, 2004 & 2008); CDF and DØ EW boson $p_\perp$ analyses; CDF and DØ QCD colour coherence, jet decorrelation, jet shapes, Z+jets, inclusive jet cross-section;
- HERA: H1 energy flow and charged particle spectra; ZEUS dijet photoproduction.

In addition, users can write their own analyses using the Rivet projections without needing to modify the Rivet source, by using Rivet's plugin system. We encourage such privately-implemented analyses to be submitted for inclusion in the main Rivet distribution, and would particularly welcome QCD analyses from HERA, b-factory and RHIC p-p experiments.

While Rivet is primarily a library which can be used from within any analysis framework (for example, it is integrated into the Atlas experiment's framework), the primary usage method is via a small executable called rivetgun. This provides a frontend for reading in HepMC events from ASCII dump files and also for running generators "on the fly" via the AGILe interface library. This latter approach is particularly nice because there is no need to store large HepMC dump files and the corresponding lack of file I/O speeds up the analysis by a factor $\sim \mathcal{O}(10)$. In this mode, Rivet is ideal for parameter space scans, since generator parameters can be specified by name on the rivetgun command line and applied without recompilation. AGILe currently supports API-level interfaces to the Fortran HERWIG 6 [2] and Pythia 6 [1] generators (combined with the AlpGen [8] MLM multi-jet merging generator, the CHARYBDIS black hole generator [9], and the JIMMY hard underlying event generator [10] for HERWIG), plus the C++ generators Herwig++ [3], Sherpa [4] and Pythia 8 [11].

At the time of writing, the current version of Rivet is 1.1.0, with a 1.1.1 patch release pending. The main framework benefits of the 1.1.x series over 1.0.x are a safer and simpler mechanism for handling projection objects (massively simplifying many analyses), better compatibility of the AGILe loader with the standard LCG Genser packaging and a large number of new and improved analyses and projections. A "bootstrap" script is provided for easy setup. Anyone interested in using Rivet for generator validation should first visit the website http://projects.hepforge.org/rivet/.

Rivet is now a stable and powerful framework for generator analysis and we are looking forward to its increasing rôle in constraining generator tunings for background modelling in LHC high-$p_\perp$ physics. Future versions will see improvements aimed at high-statistics validation simulations, such as histogramming where statistical error merging is automatically correct, as well as the addition of more validation analyses.

3 Tuning tools: Professor

While Rivet provides a framework for comparing a given generator tuning to a wide range of experimental data, it provides no intrinsic mechanism for improving the quality of that tune. Historically, the uninspiring task of tuning generator parameters to data "by eye" has been the unhappy lot of experimental researchers, with the unsystematic nature of the study reflecting that significant improvements in quality of both life and tuning would have been possible. This call for an automated and systematic approach to tuning is taken up by a second new tool: Professor. This is written in Python code as a set of factorised scripts, using the SciPy numerical library [12] and an interface to rivetgun.

The rough formalism of systematic generator tuning is to define a goodness of fit function between the generated and reference data, and then to minimise that function. The intrinsic problem is that the true fit function is certainly not analytic and any iterative approach to minimisation will be doomed by the expense of evaluating the fit function at a new parameter-space point: this may well involve ten or more runs of the generator with 200k–2M events per run. Even assuming that such runs can be parallelised to the extent that only the longest determines the critical path, an intrinsically serial minimisation of $\mathcal{O}(1000)$ steps will still take many months. This is clearly not a realistic strategy!

The Professor approach, which is the latest in a lengthy but vague history of such efforts [13, 14], is to parameterise the fit function with a polynomial. In fact, since the fit function itself is expected to be complex and not readily parameterisable, there is a layer of indirection: the polynomial is actually fitted to the generator response of each observable bin, MC_b to the changes in the n-element parameter vector, $\vec{p}$. To account for lowest-order parameter correlations, a second-order polynomial is used,

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

where the shifted parameter vector $\vec{p}' \equiv \vec{p} - \vec{p}_0$, with $\vec{p}_0$ chosen as the centre of the parameter hypercube. A nice feature of using a polynomial fit function, other than its general-purpose robustness, is that the actual choice of the $\vec{p}_0$ is irrelevant: the result of a shift in central value is simply to redefine the coefficients, rather than change the functional form, but choosing a central value is numerically sensible.

The coefficients are determined by randomly sampling the generator from N parameter space points in an n-dimensional parameter hypercube defined by the user. Each sampled point may actually consist of many generator runs, which are then merged into a single collection of simulation histograms. A simultaneous equations solution is possible if the number of runs is the same as the number of coefficients between the n parameters, i.e. $N = N_{\text{min}}^{(n)} = (2 + 3n + n^2)/2$. However, using this minimum number of runs introduces a systematic uncertainty, as we certainly do not expect the bin MC response to be a perfect polynomial. Here we are helped by the existence of the Moore–Penrose pseudoinverse: a generalisation of the normal matrix inverse to non-square matrices with the desirable feature that an over-constrained matrix will be inverted in a way which gives a least-squares best fit to the target vector. Even more helpful is that a standard singular value decomposition (SVD) procedure can be used to deterministically implement the pseudoinverse computation. Hence, we phrase the mapping on a bin-by-bin basis from coefficients C to generator values V as $PC = V$, where P is the parameter matrix to be pseudo-inverted. For a two parameter case, parameters $\in \{x, y\}$, the above may be explicitly written as

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

where the numerical subscripts indicate the N generator runs. Note that the columns of P include all $N_{\text{min}}^{(2)} = 6$ combinations of parameters in the polynomial, and that P is square (i.e. minimally pseudo-invertible) when $N = N_{\text{min}}^{(n)}$. Then $C = \tilde{\mathcal{I}}[P]\, V$, where $\tilde{\mathcal{I}}$ is the pseudoinverse operator.

Now that we have, in principle, a good parameterisation of the generator response to the parameters, $\vec{p}$, for each observable bin, b, it remains to construct a goodness of fit (GoF) function and minimise it. We choose the χ^2 function, but other GoF measures can certainly be used. Since the relative importance of various distributions in the observable set is a subjective thing — given 20 event shape distributions and one charged multiplicity, it is certainly 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 weights, $w_{\mathcal{O}}$, for each observable, $\mathcal{O}$, in our χ^2 definition:

$$\chi^2(\vec{p}) = \sum_{\mathcal{O}} w_{\mathcal{O}} \sum_{b \in \mathcal{O}} \frac{(f_b(\vec{p}) - \mathcal{R}_b)^2}{\Delta_b^2}, \tag{3}$$

where $\mathcal{R}_b$ is the reference value for bin b and the total error Δ_b is the sum in quadrature of the reference error and the statistical generator errors for bin b — in practise we attempt to generate enough data that the MC error is much smaller than the reference error for all bins.

The final stage of our procedure is to minimise this 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

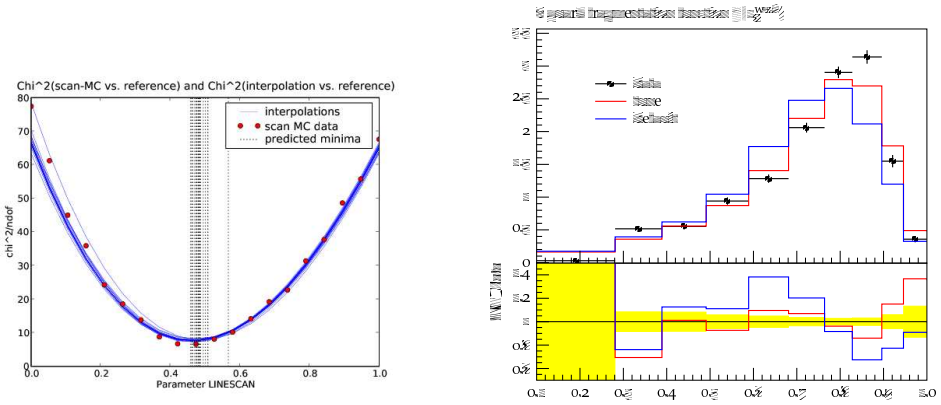

Fig. 1: (a) Parameter space line scan in χ^2, showing the agreement between Professor's predicted values (blue lines) and the true values (red dots). (b) Pythia 6 b-fragmentation functions, showing the improvements obtained using Professor (red) to tune the Bowler parameterisation against the default (blue).

hence we must finally resort to a numerical minimisation. We have implemented this in terms of minimisers from SciPy and also PyMinuit [15], with the latter's initial parameter space grid scan making it our preferred choice.

Finally, on obtaining a predicted best tune point from Professor, it is prudent to check the result. This can be done directly with rivetgun, and Professor also has a line scan feature which allows scans along arbitrary straight lines in parameter space, which is useful to verify that the χ^2 behaves as interpolated and to explicitly compare default tunes to predicted tunes. Such a line scan can be seen in Fig. 1(a). We have explicitly checked the robustness of the polynomial and the random distribution of sampling points against various skewed test distributions and the behaviour is robust. We have also found it to be useful to over-sample by a considerable fraction, and then to perform the χ^2 minimisation for a large number of distinct run-combinations, $N_{\text{min}}^{(n)} < N_{\text{tune}} \leq N$, which gives a systematic control on interpolation errors and usually a better performance than just using $N_{\text{tune}} = N$.[1]

The focus in testing and commissioning the Professor system has until recently been focused on Pythia 6 tunes against LEP data [16]. 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 happy with the minima. Eventually we decided to split the tuning into a two-stage procedure where flavour-sensitive fragmentation parameters were tuned first to provide a base on which to tune the semi-factorised kinematic parameters of the shower and hadronisation. The result has been a dramatic improvement of the Pythia 6 identified particle multiplicity spectra, without losing the event shape descriptions (originally tuned by DELPHI's version of the same procedure), and a major improvement of the b-fragmentation function as seen in Fig. 1(b).[2] This tune will be adopted as the default parameter set for the next release of

[1] Note that the tuning runs need a significant degree of variation, i.e. $N_{\text{tune}} \ll N$ for most of the tune run-combinations.

[2] Note that interpolation methods cannot deal with discrete settings such as the choice of functional form of b-fragmentation function. This required several parallel tunes with different values of the discrete parameter.

Pythia 6.

4 Conclusions

To conclude, the situation is looking positive for MC generator tuning at present: the Rivet and Professor tools are now in a state where they can be used to achieve real physics goals and the Pythia 6 tune described here (using both tools) has been a significant success. Development plans in the near future are very much aimed at getting the same tuning machinery to work for hadron collider studies, in particular initial state radiation (ISR) and underlying event (UE) physics. We aim to present tunes of C++ generators to LEP data shortly, along with first studies of interpolation-based tunes to CDF underlying event data. Finally, we are keen to constrain fragmentation and UE hadron physics for the LHC, using b-factory, RHIC and early LHC data.

References

[1] T. Sjostrand and S. Mrenna and P. Skands, JHEP **05**, 026 (2006), hep-ph/0603175.

[2] B. Webber and others, JHEP **01**, 010 (2001), hep-ph/0011363, HERWIG Collaboration.

[3] P. Richardson *et al.*, (2008), 0803.0883, Herwig++ Collaboration.

[4] F. Krauss and others, JHEP **02**, 056 (2004), hep-ph/0311263, Sherpa Collaboration.

[5] J. Bromley *et al.*, (1995), ZEUS and H1 Collaborations.

[6] M. Dobbs and J. B. Hansen, Comput. Phys. Commun. **134**, 41 (2001).

[7] M. Cacciari and G. Salam and G.Soyez, (2006), hep-ph/0607071.

[8] M. Mangano *et al.*, JHEP **07**, 001 (2003), hep-ph/0206293.

[9] C. Harris, P. Richardson, and B. Webber, JHEP **08**, 033 (2003), hep-ph/0307305.

[10] J. Butterworth, J. Forshaw, and M. Seymour, Z. Phys. **C72**, 637 (1996), hep-ph/9601371.

[11] T. Sjostrand, S. Mrenna, and P. Skands, Comput. Phys. Commun. **178**, 852 (2008), 0710.3820.

[12] SciPy website: http://www.scipy.org.

[13] DELPHI Collaboration, K. Hamacher *et al.*, Z. Phys. **C73**, 11 (1996).

[14] K. Hamacher and M. Weierstall, (1995), hep-ex/9511011.

[15] PyMinuit website: http://code.google.com/p/pyminuit/.

[16] A. Buckley *et al.*, Professor Collaboration, in preparation.