Monte Carlo Tuning with Professor

Hendrik Hoeth
(Lunds Universitet)

Overview

Motivation	–	“why” and “what’s the problem”
Strategy	–	“how to tune”
Status	–	“where are we now”

Motivation

The phenomenological models in MC generators have free parameters which are a priori unknown.

The parameters are highly correlated $\Rightarrow$ can't be tuned one after the other.

Many parameters to be tuned $(\mathcal{O}(10))$.

Tuning all parameters at the same time puts us into a high dimensional parameter space.

Brute force approaches don't work: Running the MC generator takes too long for every point in the parameter space ($=$ setting of parameters).

A strategy

Predict the MC output for any parameter set,
then use the prediction for the fit.

1. Choose a tuning interval for the parameters, pick random points in parameter space and run the generator with these settings.

2. Interpolate between points $\Rightarrow$ prediction of the MC output at any specific parameter setting (for each bin of each observable).

3. Fit this prediction to data (minimal χ^2).

4. Repeat the fit for different combinations of observables.

5. Choose the nicest set of parameters.

Status

History

This has already been done by Delphi (*"Tuning and Test of Fragmentation Models Based on Identified Particles and Precision Event Shape Data"*, Z. Phys., C 73 (1996) 11–60).

The original program was called Professor (PROcedure For EStimating Systematic errORs).

The old code is not operational anymore – we reimplement the algorithm but keep the name.

Where we are

Actively working on the project: Andy Buckley (Durham), Eike von Seggern (Dresden), Heiko Lacker (Berlin), Hendrik Hoeth (Lund), Holger Schulz (Dresden).

We are using

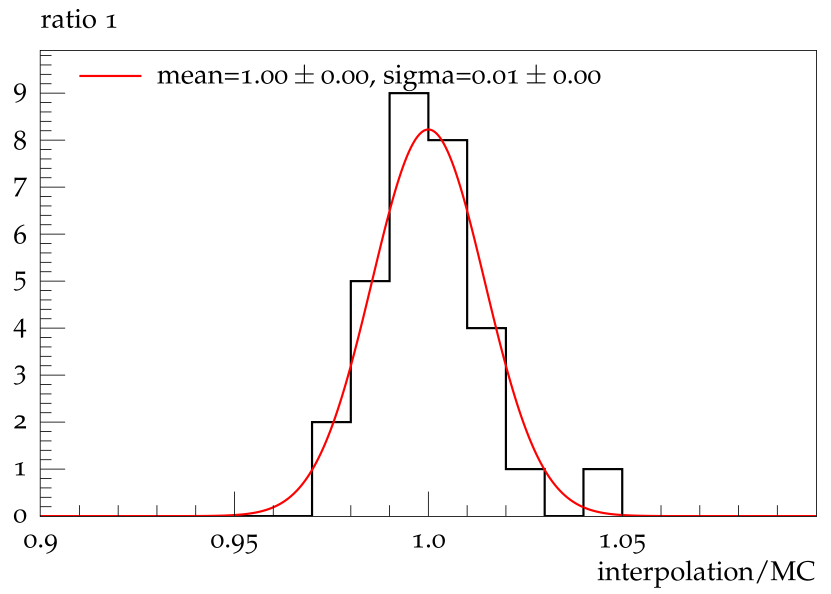
- python for our program
- Rivet for event generation
- libgsl and Minuit for interpolation and fitting

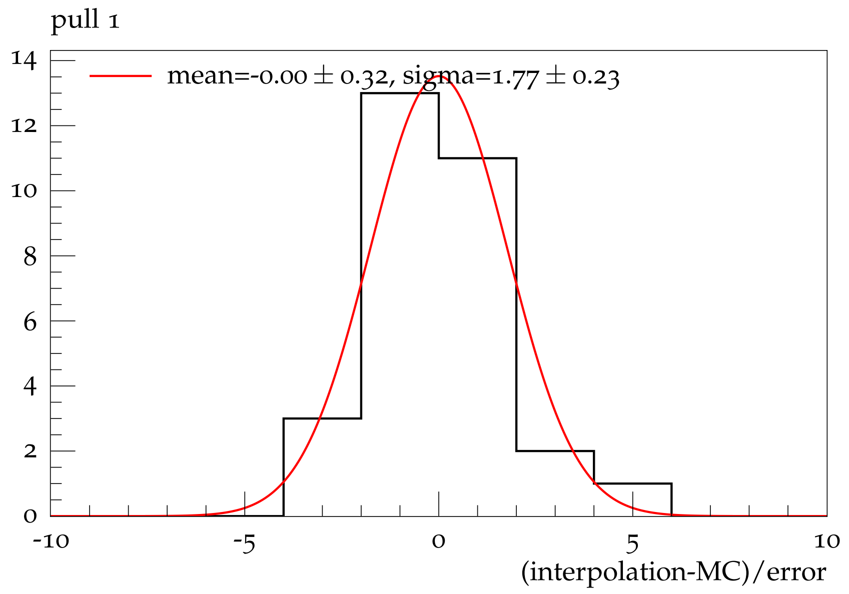
The reimplementation is (mostly) running and we are working on systematic studies.

Project web page: http://projects.hepforge.org/professor/

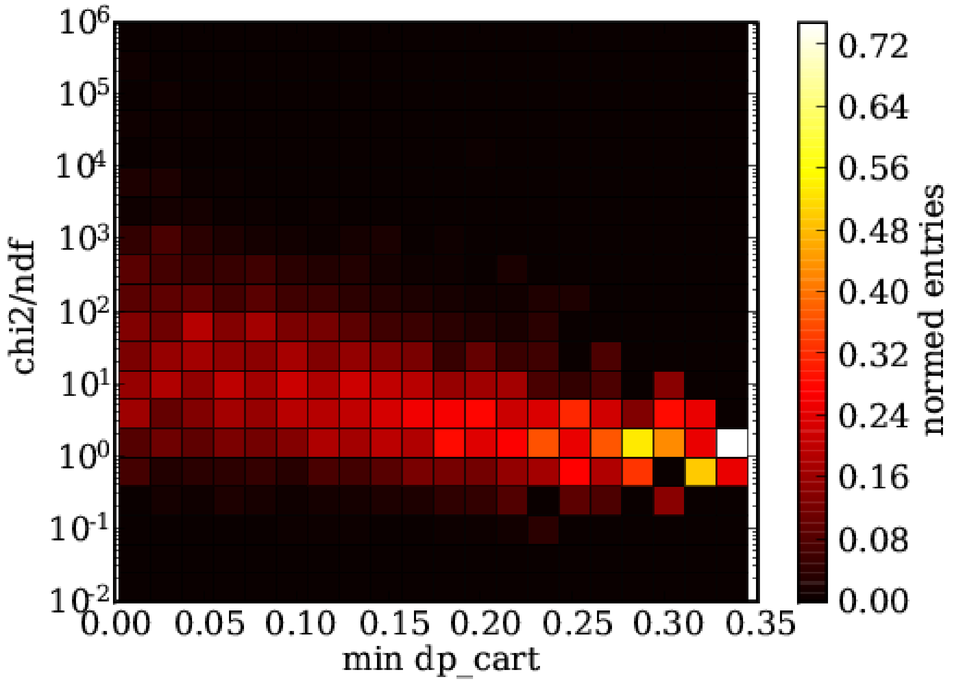
Examples of studies

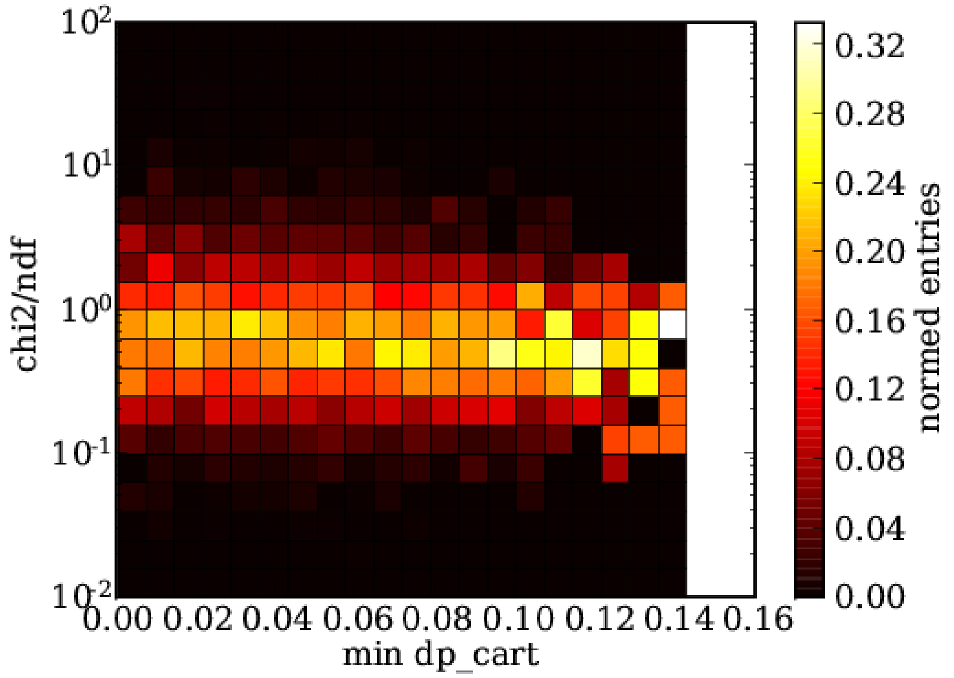
Compare the prediction with the generated MC output, for each bin of each observable:





Check interpolation quality (χ^2) dependence on distance between sampling points:





Try to tune Pythia with Pythia – generate “data” with some parameter settings and reproduce those settings with Professor.

We did this with up to four parameters:

parameter	input	fit result
PARJ(21)	0.42890	0.44
PARJ(41)	0.25458	0.24
PARJ(81)	0.39228	0.42
PARJ(82)	6.72443	6.40

What’s missing, next steps

- Some observables needed for a real tuning are still missing in Rivet (e. g. jet rates).
- Implementation of the observables in Rivet should be checked before the tuning.
- Code cleanup (not a show stopper, though).
- We plan to tune an old Pythia version (6.2?) to compare it to the Delphi tuning.
- Real tuning (Pythia 8?) hopefully by the end of February.

Backup slides

1. Choosing parameters

Pick the parameters you want to tune:

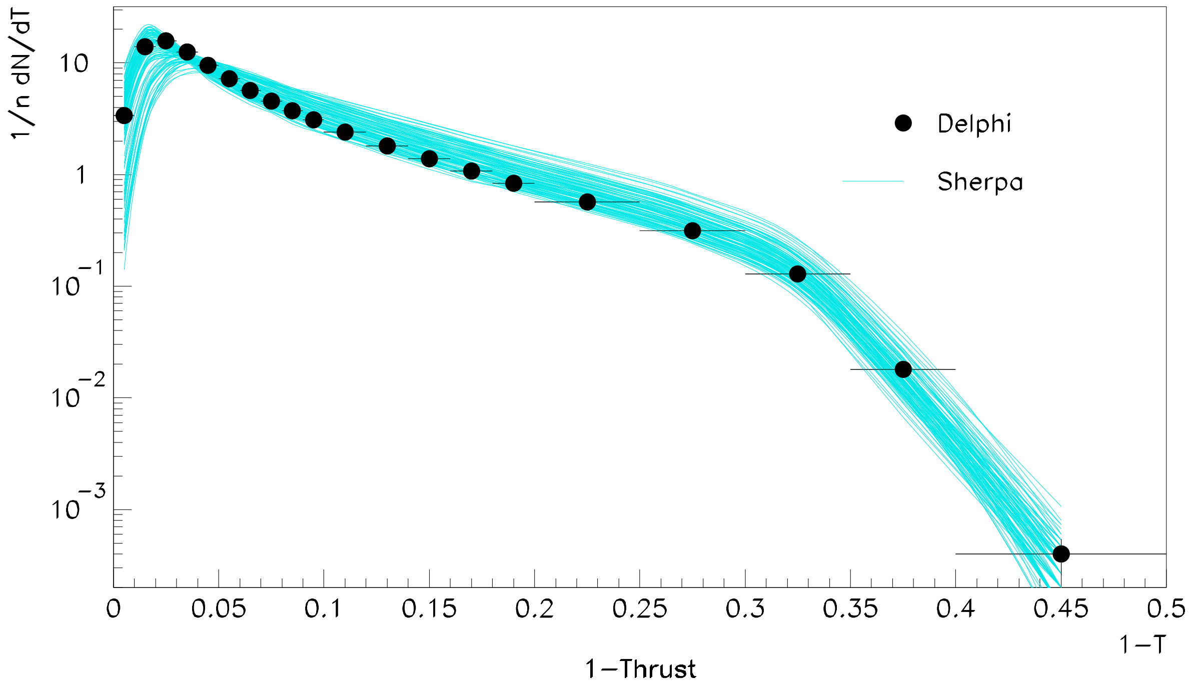
- Tune everything that is important.
- But remember: Each additional parameter adds one dimension to the parameter space.

Define parameter intervals:

- Make the interval large enough so that the result will not be outside.
- But remember: Cutting down 10 intervals by 10 % shrinks the volume of the parameter space by 2/3.

Now pick random points in parameter space and run the generator for each setting.

Calculating observables yields plots like this:



Every line corresponds to a certain setting.

2. Predict the Monte Carlo

Get a bin by bin prediction for the MC response as function of the parameter set $\vec{p} = (p_1, p_2, \ldots, p_n)$.

Using a second order polynomial takes the correlations between the parameters into account:

$$X_{\mathrm{MC}}(p_1, p_2, \ldots, p_n) = \\ A_0 + \sum_{i=1}^{n} B_i p_i + \sum_{i=1}^{n} C_i p_i^2 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} D_{ij} p_i p_j + \cdots$$

3. Fit the prediction to data

Having A_0, B_i, C_i and D_{ij} we can predict the MC response for any set of parameters very fast. This prediction can be fitted to data, minimising the χ^2:

$$\chi^2(\vec{p}) = \sum_{\text{observables}} \sum_{\text{bins}} \left(\frac{X_{\text{data}} - X_{\text{MC}}(\vec{p})}{\sigma_{\text{data}}} \right)^2$$

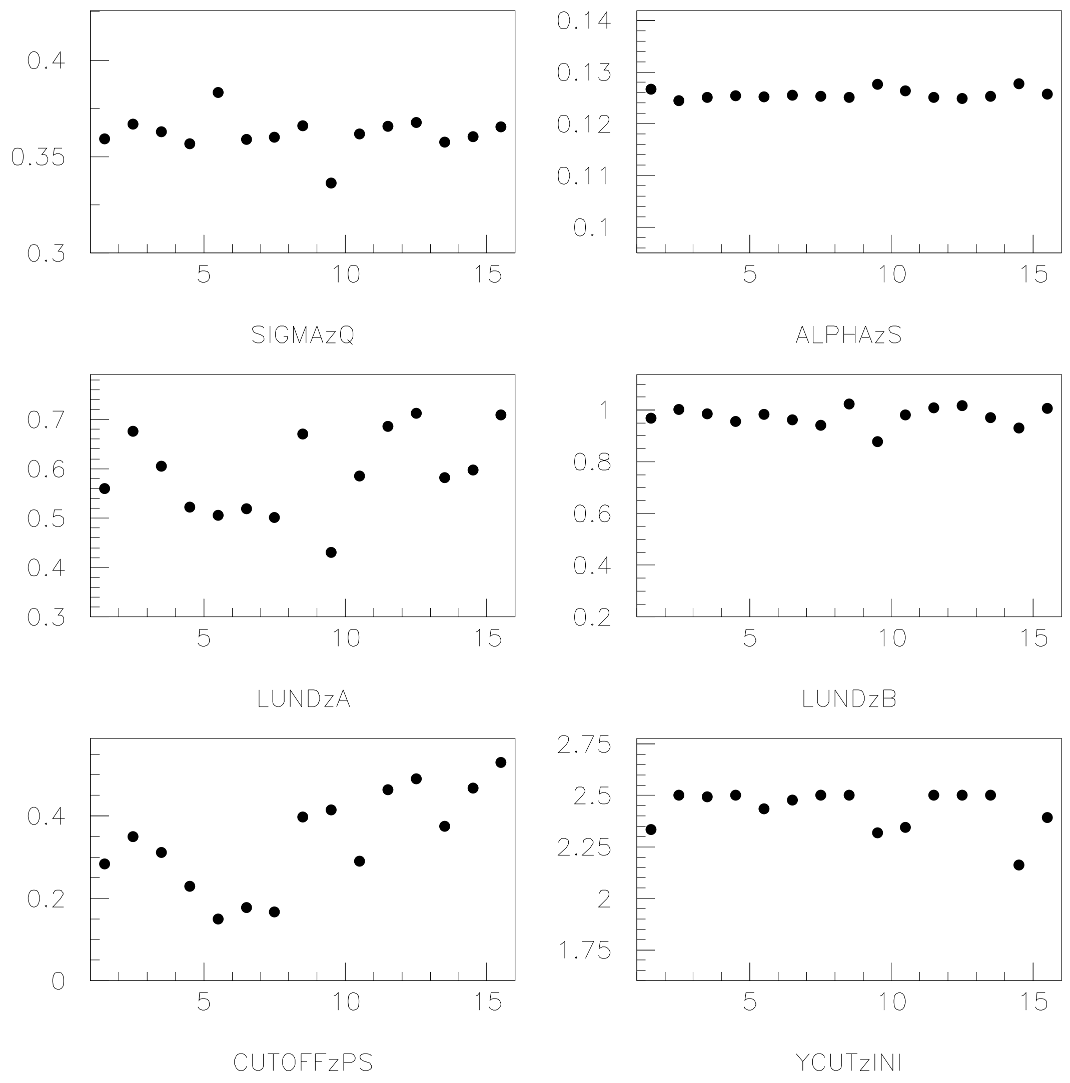
Include all the relevant data distributions in the fit!

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

4. Use different data sets

Using different combinations of observables yield different optimisation results.

Learn something about correlations and stability of the tuning.



What data should we use ...?

It depends . . .

In general: Use observables which are physically related to the parameters you want to tune!

Examples:

For the parton shower use event shapes like Thrust, Sphericity, Planarity, Major, Minor, (differential) jet rates, p_t spectra, N_{ch}, . . .

For hadronisation use identified particle spectra, multiplicities . . .

… and what data is available?

- LEP has published *lots* of high precision data.
- There is good data from SLD and the DESY experiments.
- There is very little useful data from the Tevatron!

To compare data to MC either the data needs to be acceptance corrected or the MC needs to be folded with the detector response. *Most of the published Tevatron data satisfies neither condition and can't be compared to anything!*