MC tuning with Professor

Andy Buckley (Edinburgh), Hendrik Hoeth (Durham),
Heiko Lacker (HU Berlin) , Holger Schulz (HU Berlin),
Jan Eike von Seggern (DESY HH)

February 10, 2011
Workshop on Precision Measurements of α_s

Introduction

- Extraction of α_s from eventshape data requires reliable models to correct for hadronisation effects
- However, hadronisation based on phenomenological models
  - Lund string model (Pythia)
  - Cluster models (Sherpa, Herwig)
- Lots of a priori unknown parameters
- $\Rightarrow$ "tuning" of those parameters essential to get usable predictions

Tuning through the ages

- Manual tunes: lots of time and manpower or tuning experience of a life-time
- Grid-scans, genetic algorithm: tough in D>2, slow, not very flexible
- systematically:
  - Bin-wise interpolation of MC generator response and χ^2 minimization (DELPHI 1995, Hamacher et al.)
  - 2^{nd} order polynomials account for parameter correlations

but:

- Code (`fortran`) not sufficiently flexible
- Restricted to 2^{nd} order polynomial for bin-wise interpolation

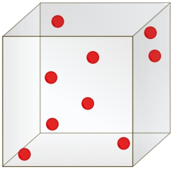
Better use Professor arXiv:0907.2973, arXiv:0906.0075, arXiv:0902.4403

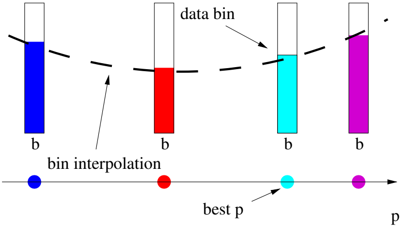
"PROcedure For EStimating Systematic errORs"

- Pick up DELPHI idea, add much more functionality
- Python - implementation (scripts and API), actively developed
- Allows for systematic checks

Tuning procedure in Professor (1D, 1Bin)

1. Random sampling: N parameter points in n-dimensional space
2. Run generator and fill histograms
3. For each bin: use N points to fit interpolation (2nd or 3rd order polynomial)
4. Construct overall (now trivial) $\chi^2 \approx \sum_{bins} \frac{(interpolation - data)^2}{error^2}$
5. and Numerically *minimize* `pyMinuit, SciPy`





Interactivity

Key feature of Professor:

1. we are parameterising a very expensive function
2. input to that parameterisation can be trivially parallelised

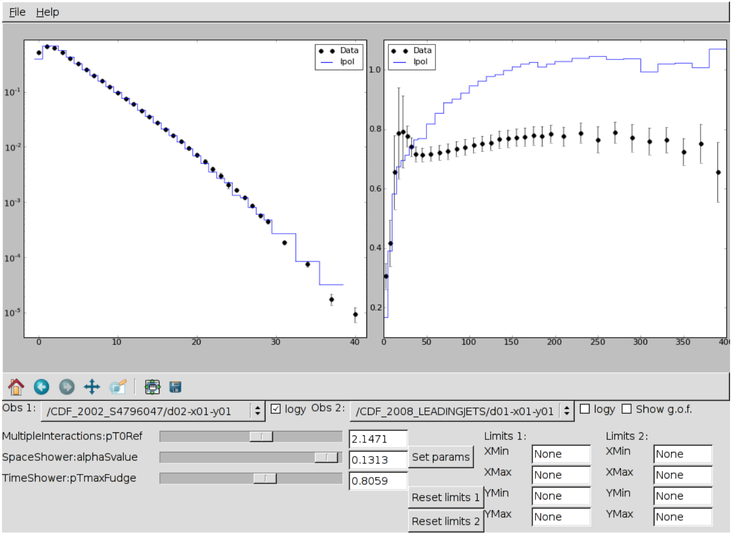
- Can parallelise parameterisation (for many run combinations)
- Optimisation, too

Parameterisation produces a fast, analytic "pseudo-generator"

- $\Rightarrow$ Can get a good approximation of what a generator will do when run for many hours/days with particular params, in < 1 second!

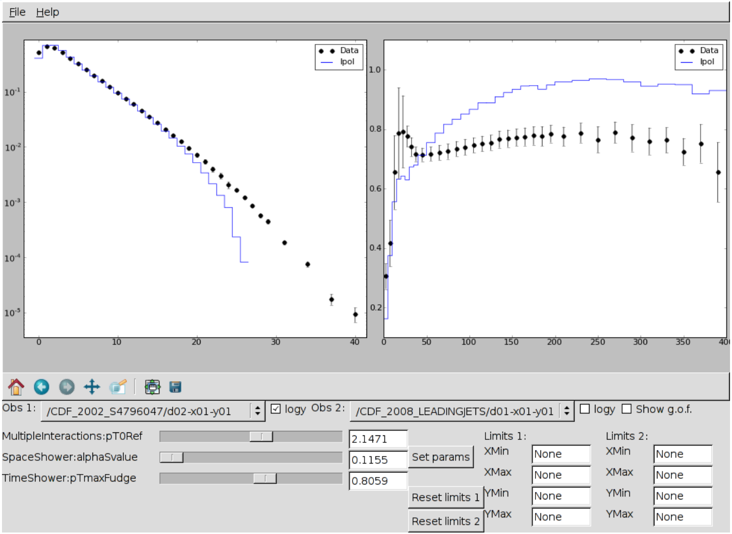
Why not make an interactive MC simulator?

PROF-I



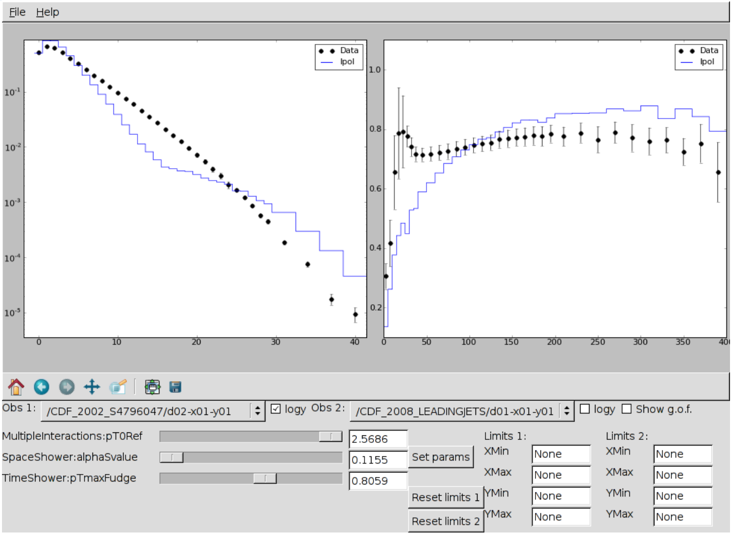
Screencast online: http://www.youtube.com/watch?v=qAJb418i_Qw

PROF-I



Screencast online: http://www.youtube.com/watch?v=qAJb418i_Qw

PROF-I



Screencast online: http://www.youtube.com/watch?v=qAJb418i_Qw

Professor news

- Version 1.2.0 just released
- Extensive documentation (scripts, API)
- Use cubic interpolations by default now
- Calculate “EigenTunes“
- Readily available on AFS:

`source/afs/cern.ch/sw/lcg/external/MCGenerators/professor/1.2.0/x86_64-slc5-gcc43-opt/setup.sh`

Observables and Weights

- This is what Professor minimises: $\chi^2(\vec{p}) = \sum_{\mathcal{O}} \sum_{b \in \mathcal{O}} w_b \frac{(f^{(b)}(\vec{p}) - \mathcal{R}_b)^2}{\Delta_b^2}$
- Slightly more art than science
- Garbage in, garbage out
- Use weights w_b to:
  - emphasize certain observables, e.g. $\langle N_{\mathrm{ch}} \rangle$
  - emphasize certain bins of an observable
  - exclude bins from the fit

Observable selection

Selecting what data to tune phenom. parameters to a priori difficult

Observable selection

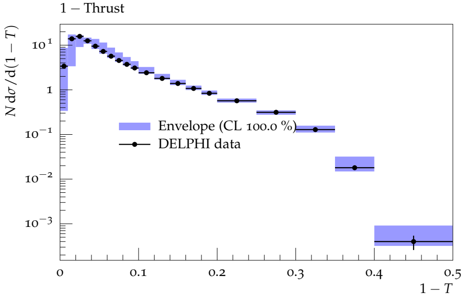
Selecting what data to tune phenom. parameters to a priori difficult

- Lots of thinking, reading and consultation of model authors

Observable selection

Selecting what data to tune phenom. parameters to a priori difficult

- Lots of thinking, reading and consultation of model authors
- Checking production (envelopes) $\rightarrow$ helps identify problematic regions



Observable selection

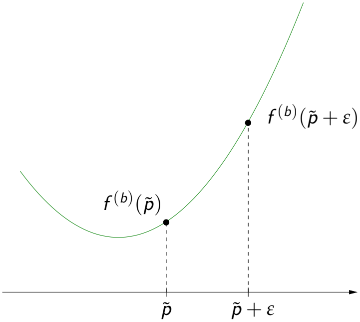
Selecting what data to tune phenom. parameters to a priori difficult

- Lots of thinking, reading and consultation of model authors
- Checking production (envelopes) $\rightarrow$ helps identify problematic regions
- Analysing sensitivity of observables to shifts in parameter space:
  "How much does the bin content change if I vary parameter i?"

Observable selection

Selecting what data to tune phenom. parameters to a priori difficult

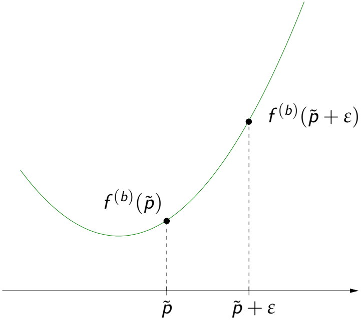
- Lots of thinking, reading and consultation of model authors
- Checking production (envelopes) $\rightarrow$ helps identify problematic regions
- Analysing sensitivity of observables to shifts in parameter space:
  "How much does the bin content change if I vary parameter i?"

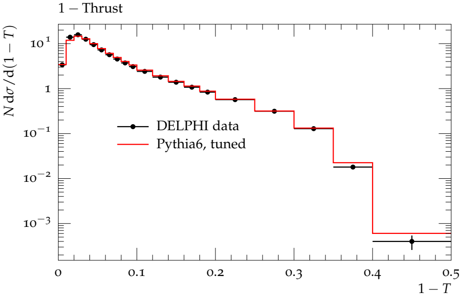


Observable selection

Selecting what data to tune phenom. parameters to a priori difficult

- Lots of thinking, reading and consultation of model authors
- Checking production (envelopes) $\rightarrow$ helps identify problematic regions
- Analysing sensitivity of observables to shifts in parameter space:
  "How much does the bin content change if I vary parameter i?"

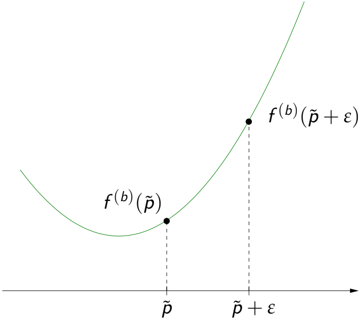


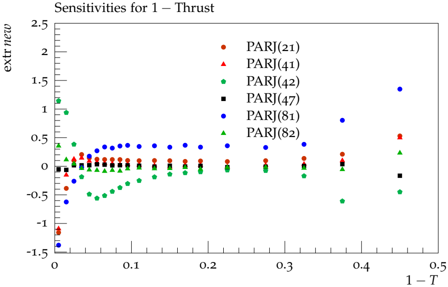


Observable selection

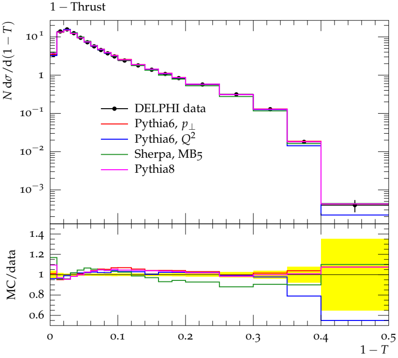
Selecting what data to tune phenom. parameters to a priori difficult

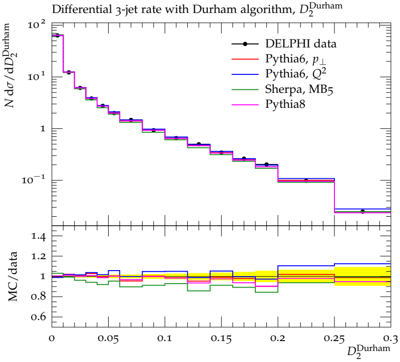
- Lots of thinking, reading and consultation of model authors
- Checking production (envelopes) $\rightarrow$ helps identify problematic regions
- Analysing sensitivity of observables to shifts in parameter space: "How much does the bin content change if I vary parameter i?"





Comparison of Professor tunes

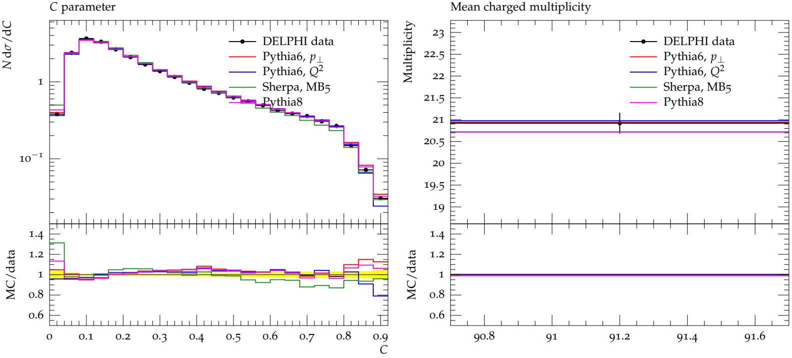




More plots online: http://users.hepforge.org/~holsch/AlphaSWorkshop/cmp2/plots.html

DELPHI data: Z.Phys.C73:11-60,1996

Comparison of Professor tunes

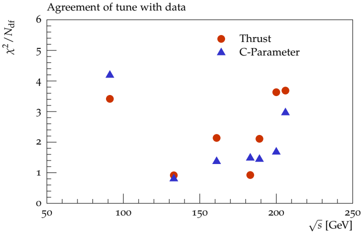


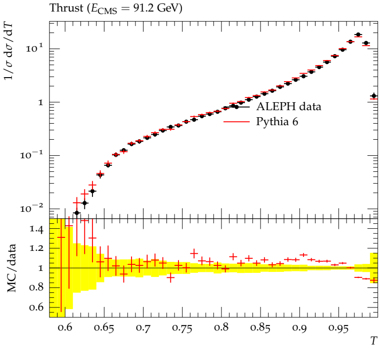
More plots online: http://users.hepforge.org/~holsch/AlphaSWorkshop/cmp2/plots.html

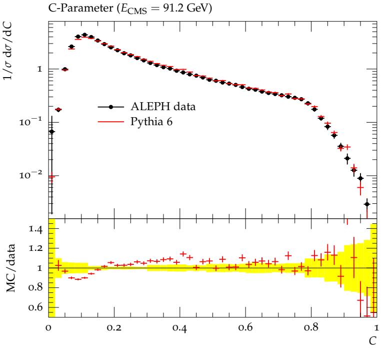
DELPHI data: Z.Phys.C73:11-60,1996

$\sqrt{s}$ DEPENDENCY

- Use Pythia 6 tuning obtained at $\sqrt{s} = 91.2\text{GeV}$
- Compare prediction at higher energies to ALEPH data
  `Eur.Phys.J.C35:457-486,2004`
- $\Rightarrow$ similar level of agreement

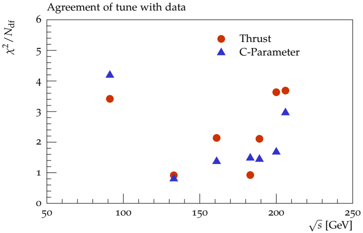


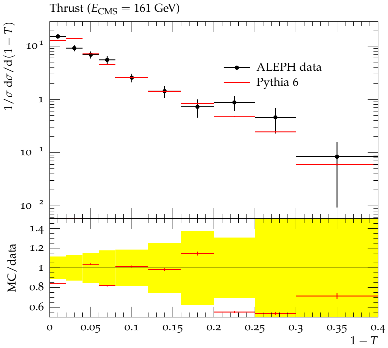


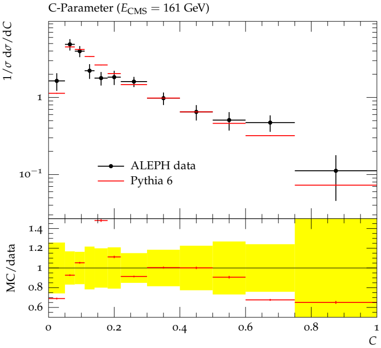


$\sqrt{s}$ DEPENDENCY

- Use Pythia 6 tuning obtained at $\sqrt{s} = 91.2\text{GeV}$
- Compare prediction at higher energies to ALEPH data
  Eur.Phys.J.C35:457-486,2004
- $\Rightarrow$ similar level of agreement

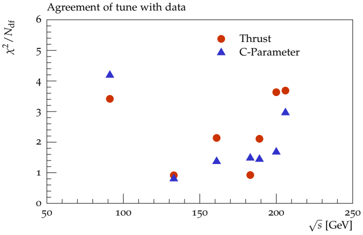


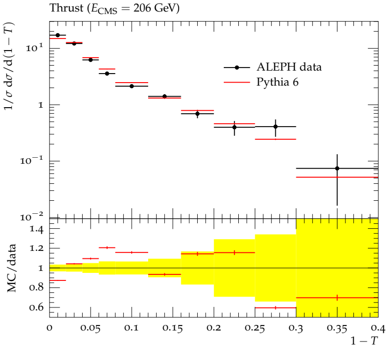


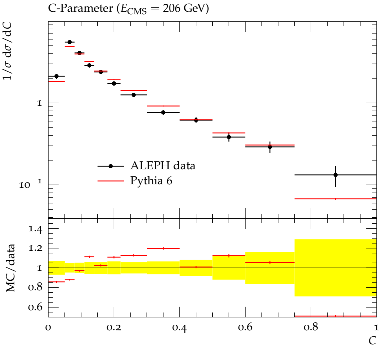


$\sqrt{s}$ DEPENDENCY

- Use Pythia 6 tuning obtained at $\sqrt{s} = 91.2\text{GeV}$
- Compare prediction at higher energies to ALEPH data
  Eur.Phys.J.C35:457-486,2004
- $\Rightarrow$ similar level of agreement





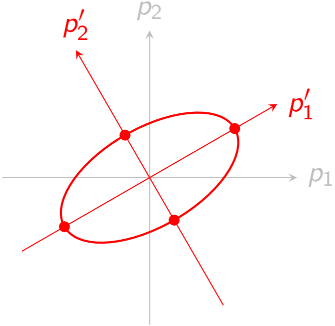


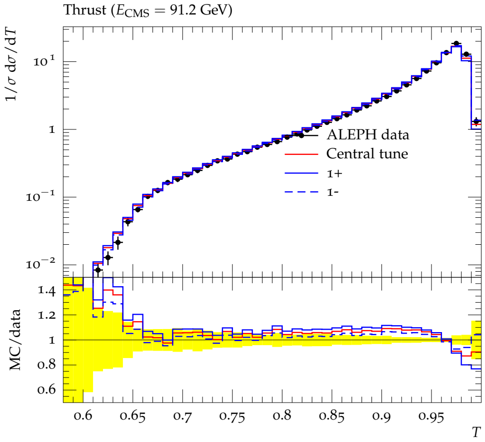
Eigentunes

Pick the extremal points of the χ^2 contour hyper-ellipsoid as representative tunes, cf. Hessian PDF errors.

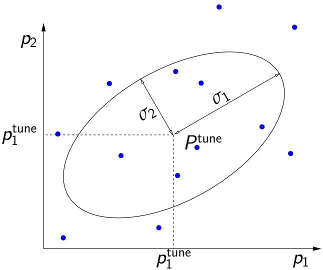
$\Rightarrow$ obtained Eigentunes stay consistent, respect correlations

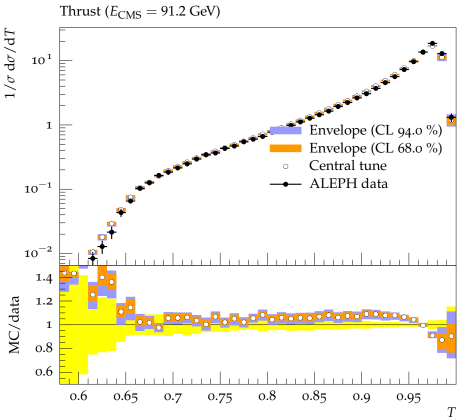
$\Rightarrow$ suitable for systematic variations





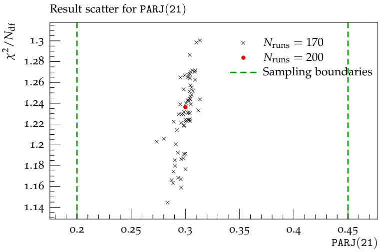
Tuning uncertainty I

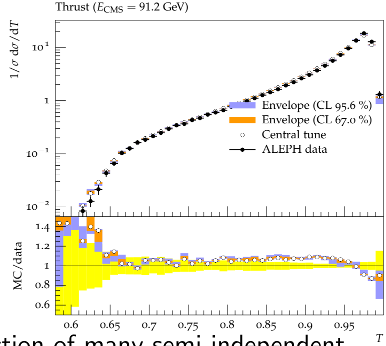




- Exploit minimiser covariance matrix, sample points from hyper-ellipsoid
- Inspired by NNPDF approach, fast parameterisation allows thousands of pseudo-MCs to be calculated at sampled parameter points
- Translate into *statistical* tuning uncertainty

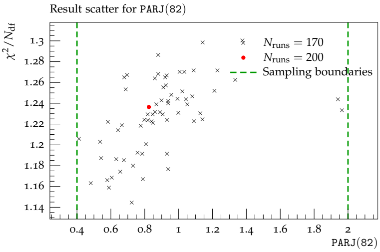
Tuning uncertainty II

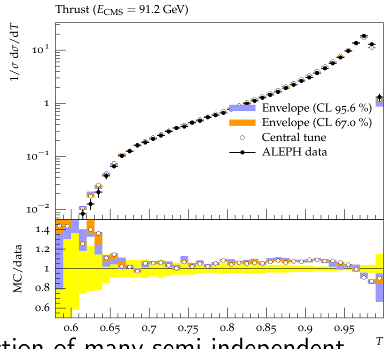




- Oversampling allows for construction of many semi-independent parameterisations
- Tunings (with same weights) yield sligthly different results $\Rightarrow$ tune-spread
- Can be investigated parameterwise (projection) - always used as sanity check
- But can also translate this into a “systematic” uncertainty of the method

Tuning uncertainty II





- Oversampling allows for construction of many semi-independent parameterisations
- Tunings (with same weights) yield sligthly different results ⇒ tune-spread
- Can be investigated parameterwise (projection) - always used as sanity check
- But can also translate this into a "systematic" uncertainty of the method

Summary

- Professor tunes using event-shapes available for Pythia 6, Pythia 8 and Sherpa
- In general very good agreement with e^+e^--data
- In case of Pythia 6 different tunes for different showers needed $(Q^2, p_\perp)$
- No strong evidence found for $\sqrt{s}$-dependence
- Eigentunes and retunes for different showers allow systematic checks
- Tuning uncertainty estimates available, so far no real use-case
- What do you need?

Thank you!

Backup

Tuning of event shapes - Pythia 6

- First: tune flavour-parameters to hadron-multiplities at LEP
- Second: tune string-fragmentation parameters to eventshapes for both $p_\perp$ and Q^2 ordered showers

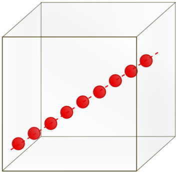
Parameter	Tune (Q^2)	Tune ($p_\perp$)	
PARJ(21)	0.325	0.313	String tension σ_q
PARJ(41)	0.5	0.49	Lund frag. param a
PARJ(42)	0.6	1.2	Lund frag. param b
PARJ(47)	0.67	1.0	Heavy quark frag.
PARJ(81)	0.29	0.257	Λ_{QCD}
PARJ(82)	1.65	0.8	Shower cut-off

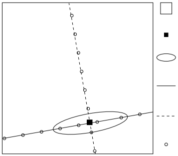
- Eventshapes tuned to: Thrust, Planarity, Sphericity, C-Parameter, ...
- The same strategy was applied when tuning Sherpa and Pythia 8
- Sherpa tune described here:

  http://projects.hepforge.org/professor/diplomathesis_jev_seggern.pdf

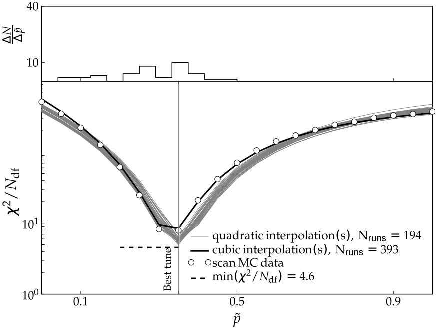
Systematic checks

- Sample params from straight hyperline through χ^2 valley



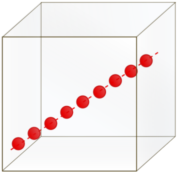


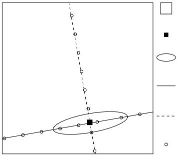
- Calculate and compare χ^2 of parameterisation with "true" MC response



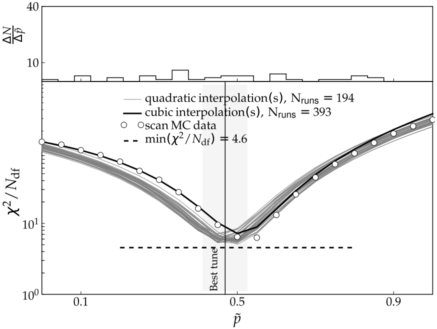
Systematic checks

- Sample params from straight hyperline through χ^2 valley

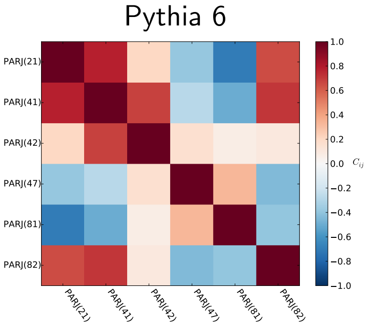


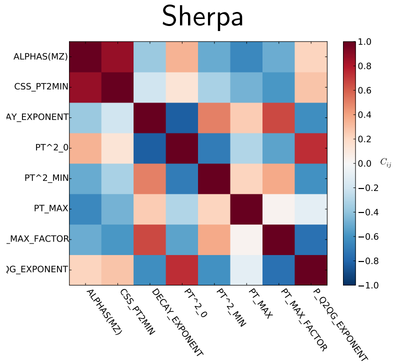


- Calculate and compare χ^2 of parameterisation with "true" MC response



Parameter correlations





2nd order polynomial includes lowest-order correlations between parameters

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

Now use N generator runs, i.e. N different parameter sets x,y:

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

Therefore: $\boxed{\vec{c}_b = \tilde{\mathcal{I}}[\tilde{\mathbf{P}}]\vec{v}}$ where $\tilde{\mathcal{I}}$ is the pseudoinverse operator.

$$\boxed{\vec{c}_b = \tilde{\mathcal{I}}[\tilde{\mathbf{P}}]\vec{v}}$$

- Use Singular Value Decomposition (SVD), a general diagonalisation for all normal matrices M: $M = U\Sigma V^*$
- Method available in SciPy.linalg
- Minimal number of runs = number of coefficients in $\vec{c}_b$:
$N_{\text{min}}^{(n)} = 1 + n + n(n+1)/2 + \underbrace{(n+1)(n+2)/6}_{\text{cubic only}}$

$\boxed{\vec{c}_b = \tilde{\mathcal{I}}[\tilde{\mathbf{P}}]\vec{v}}$

- Use Singular Value Decomposition (SVD), a general diagonalisation for all normal matrices M:$M = U\Sigma V^*$
- Method available in SciPy.linalg
- Minimal number of runs = number of coefficients in $\vec{c}_b$:
  $N_{\text{min}}^{(n)} = 1 + n + n(n+1)/2 + \underbrace{(n+1)(n+2)/6}_{\text{cubic only}}$
- Oversampling by a factor of three has proven to be much better

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
9	55	220
10	66	286

Weights used for Pythia 6 frag. tune

Observable	Weight (Q^2)	Weight ($p_\perp$)
$p_\perp^{\mathrm{in}}$ w.r.t. Thrust axes	1	2
$p_\perp^{\mathrm{out}}$ w.r.t. Thrust axes	1	1
$p_\perp^{\mathrm{in}}$ w.r.t. Sphericity axes	1	2
$p_\perp^{\mathrm{out}}$ w.r.t. Sphericity axes	1	1
Scaled momentum, $x_p = \lvert p\rvert / \lvert p_{\mathrm{beam}}\rvert$	1	3
Log of scaled momentum, $\log 1/x_p$	1	3
Mean $p_\perp^{\mathrm{out}}$ vs x_p		1
Mean $p_\perp$ vs x_p		1
$1 -$ 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 $f(x_{\mathrm{B}}^{\mathrm{weak}})$	1	2
Mean of b quark frag. function $f(x_{\mathrm{B}}^{\mathrm{weak}})$	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, $x_p = \lvert p\rvert / \lvert p_{\mathrm{beam}}\rvert$		1
c events scaled momentum, $x_p = \lvert p\rvert / \lvert p_{\mathrm{beam}}\rvert$		1
b events scaled momentum, $x_p = \lvert p\rvert / \lvert p_{\mathrm{beam}}\rvert$		1
uds events log of scaled momentum, $x_p = \lvert p\rvert / \lvert p_{beam}\rvert$		1
c events log of scaled momentum, $x_p = \lvert p\rvert / \lvert p_{\mathrm{beam}}\rvert$		1
b events log of scaled momentum, $x_p = \lvert p\rvert / \lvert p_{\mathrm{beam}}\rvert$		1