Systematic event generator tuning with Professor

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

Karlsruhe, December 4, 2009

Deutsche Forschungsgemeinschaft DFG

MCnet

Generator tuning through the ages

- By eye: lots of time and manpower, hard to repeat

Generator tuning through the ages

- By eye: lots of time and manpower, hard to repeat
- Brute-force grid-scans: tough in higher dimensions of parameter space

Generator tuning through the ages

- By eye: lots of time and manpower, hard to repeat
- Brute-force grid-scans: tough in higher dimensions of parameter space
- Genetic algorithm: burns too much CPU, no convergence defined (yet)

but:

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

Generator tuning through the ages

- By eye: lots of time and manpower, hard to repeat
- Brute-force grid-scans: tough in higher dimensions of parameter space
- Genetic algorithm: burns too much CPU, no convergence defined (yet)
- systematically:
  - Bin-wise interpolation of MC generator response and χ^2 minimization (DELPHI 1995, Hamacher et al.)

but:

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

Generator tuning through the ages

- By eye: lots of time and manpower, hard to repeat
- Brute-force grid-scans: tough in higher dimensions of parameter space
- Genetic algorithm: burns too much CPU, no convergence defined (yet)
- 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

A tool for systematic generator tuning

DELPHI 1995, Hamacher et al.: bin-wise interpolation of MC generator response and χ^2 minimization

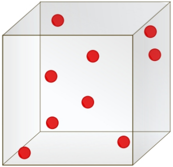
Professor (arXiv:0907.2973, arXiv:0906.0075, arXiv:0902.4403)

"PROcedure For EStimating Systematic errORrs"

- Pick up DELPHI idea
- Use flexible `python` interface
- Use quadratic or cubic interpolations
- Respond to new (LHC) data quickly
- Validation of results possible in many ways
- NEW: GUI tool, uncertainty estimates

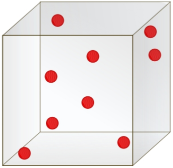
Tuning procedure in Professor (1D, 1Bin)

1. Random sampling: N parameter points in n-dimensional space



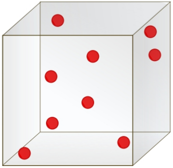
Tuning procedure in Professor (1D, 1Bin)

1. Random sampling: N parameter points in n-dimensional space



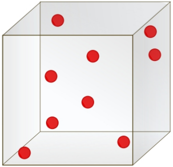
Tuning procedure in Professor (1D, 1Bin)

1. Random sampling: N parameter points in n-dimensional space



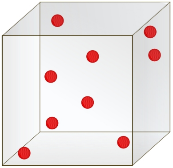
Tuning procedure in Professor (1D, 1Bin)

1. Random sampling: N parameter points in n-dimensional space



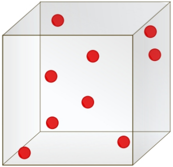
Tuning procedure in Professor (1D, 1Bin)

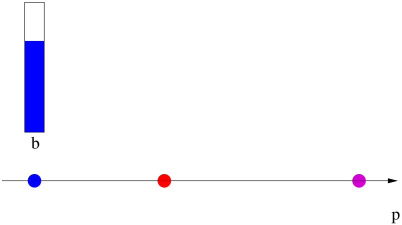
1. Random sampling: N parameter points in n-dimensional space



Tuning procedure in Professor (1D, 1Bin)

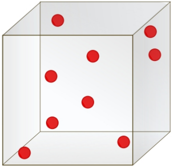
1. Random sampling: N parameter points in n-dimensional space
2. Run generator and fill histograms

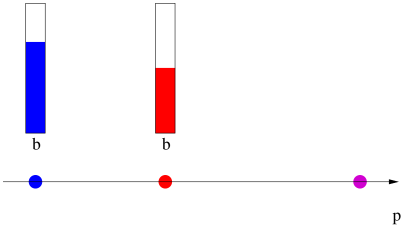




Tuning procedure in Professor (1D, 1Bin)

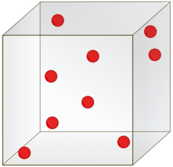
1. Random sampling: N parameter points in n-dimensional space
2. Run generator and fill histograms

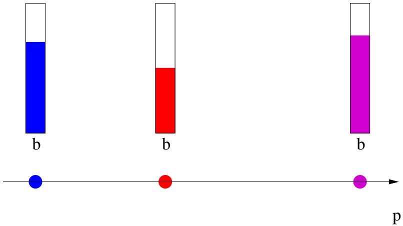




Tuning procedure in Professor (1D, 1Bin)

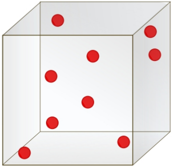
1. Random sampling: N parameter points in n-dimensional space
2. Run generator and fill histograms

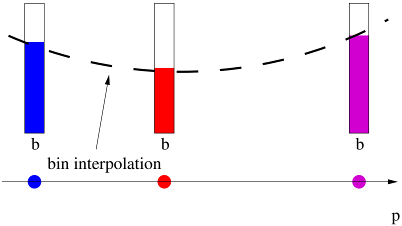




Tuning procedure in Professor (1D, 1Bin)

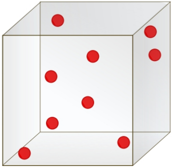
1. Random sampling: N parameter points in n-dimensional space
2. Run generator and fill histograms
3. For each bin: use N points to fit interpolation (2nd or 3rd order polynomial)

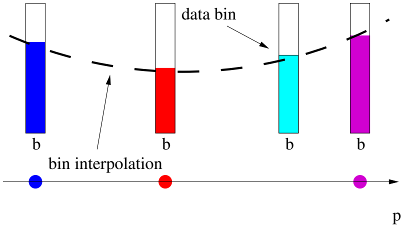




Tuning procedure in Professor (1D, 1Bin)

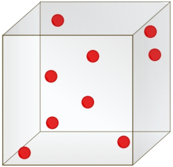
1. Random sampling: N parameter points in n-dimensional space
2. Run generator and fill histograms
3. For each bin: use N points to fit interpolation (2^{nd} or 3^{rd} order polynomial)
4. Construct overall (now trivial) $\chi^2 = \sum_{bins} \frac{(interpolation - data)^2}{error^2}$

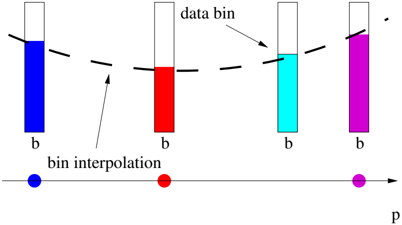




Tuning procedure in Professor (1D, 1Bin)

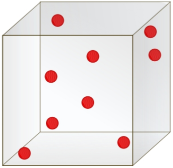
1. Random sampling: N parameter points in n-dimensional space
2. Run generator and fill histograms
3. For each bin: use N points to fit interpolation (2nd or 3rd order polynomial)
4. Construct overall (now trivial) $\chi^2 = \sum_{bins} \frac{(interpolation - data)^2}{error^2}$
5. Numerically minimize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

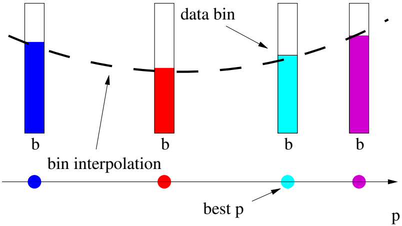




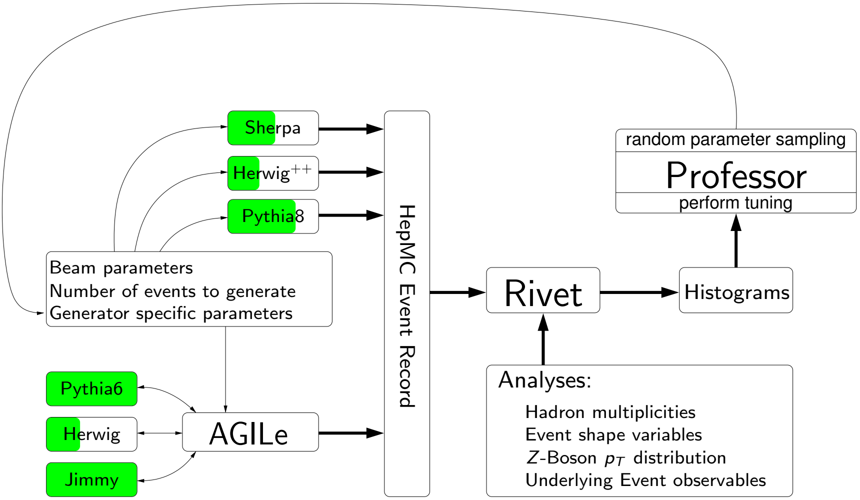
Tuning procedure in Professor (1D, 1Bin)

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 (2^{nd} or 3^{rd} order polynomial)
4. Construct overall (now trivial) $\chi^2 = \sum_{bins} \frac{(interpolation - data)^2}{error^2}$
5. Numerically minimize using `pyMinuit`, `SciPy`





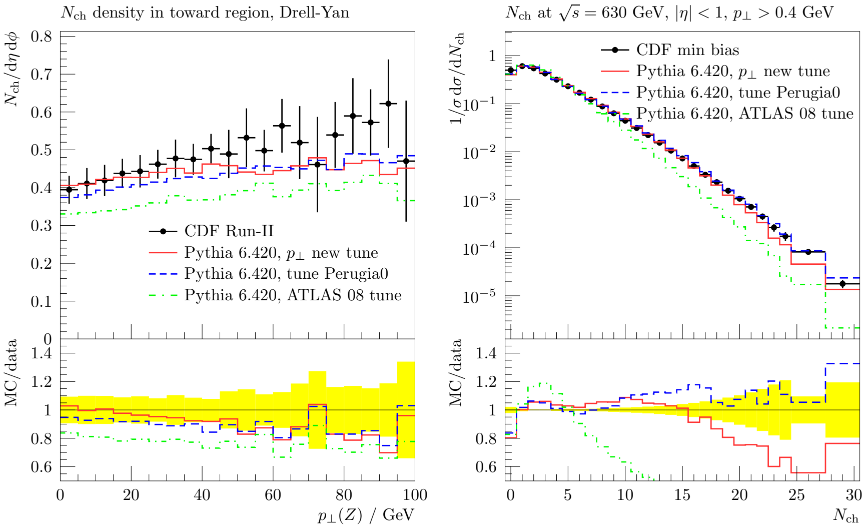
Work Flow and tuning status



Professor tunings

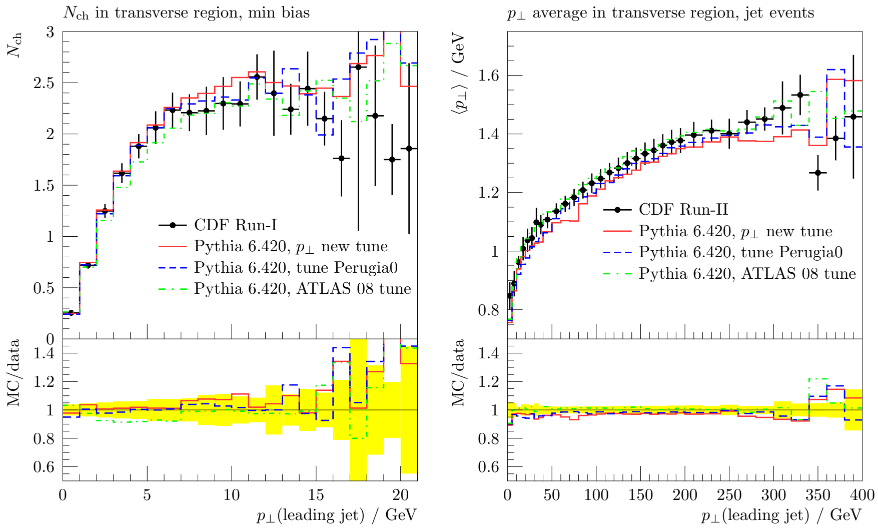
- Pythia6: simultaneous tuning of
  - 9 flavour parameters to LEP-data
  - 6 fragmentation parameters to LEP-data
  - 9 Underlying Event parameters to Tevatron-data
    - Repeated for several pdfs
- Pythia8
  - Repeated Pythia6-tune for flavour & fragmentation (new default)
  - Underlying Event curently broken/unusable in Pythia8
- Jimmy
  - Tuned Underlying Event parameters to Tevatron-data
- Sherpa
  - Currently, shower (AHADIC) is being tuned to LEP-data

Results



N_{ch} density in toward region, Drell-Yan
$N_{ch}/d\eta\,d\phi$
MC/data
$p_\perp(Z)$ / GeV
CDF Run-II
Pythia 6.420, $p_\perp$ new tune
Pythia 6.420, tune Perugia0
Pythia 6.420, ATLAS 08 tune
N_{ch} at $\sqrt{s} = 630$ GeV, $|\eta| < 1$, $p_\perp > 0.4$ GeV
$1/\sigma\, d\sigma/dN_{ch}$
CDF min bias
N_{ch}

Results



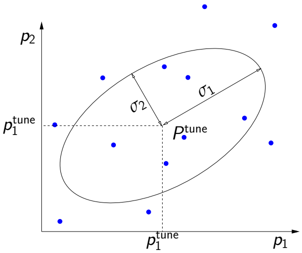
N_{ch} in transverse region, min bias
N_{ch}
MC/data
CDF Run-I
Pythia 6.420, $p_\perp$ new tune
Pythia 6.420, tune Perugia0
Pythia 6.420, ATLAS 08 tune
$p_\perp$(leading jet) / GeV
$p_\perp$ average in transverse region, jet events
$\langle p_\perp \rangle$ / GeV
CDF Run-II
Pythia 6.420, $p_\perp$ new tune
Pythia 6.420, tune Perugia0
Pythia 6.420, ATLAS 08 tune
$p_\perp$(leading jet) / GeV

Tuning uncertainties (work in progress)

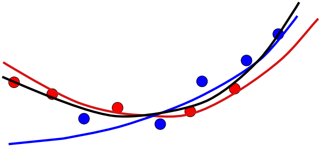
Goal: establish a robust estimate of tuning uncertainties (confidence-belt)

We currently study two different sources of tuning uncertainties:

- Statistical uncertainties → exploit covariance matrix returned by minimiser (inspired by NNPDF approach)

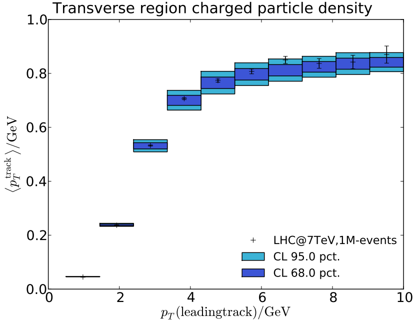


- Intrinsic systematics of the Professor method: freedom when parameterising generator response

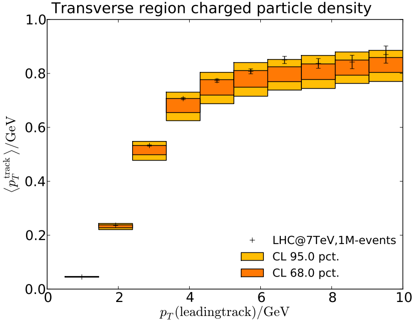


Confidence belt - without pseudodata

statistical uncertainties

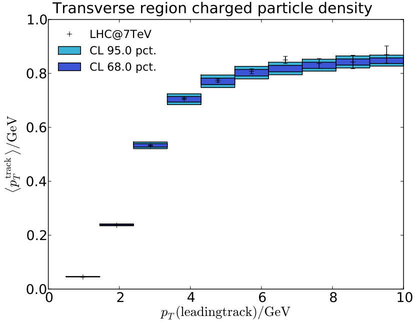


"systematic" uncertainties

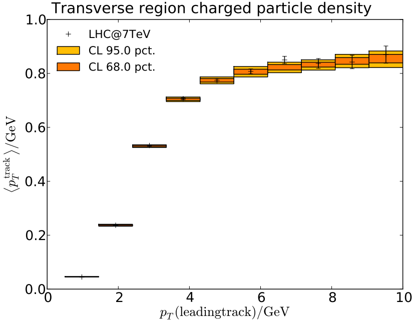


Confidence belt - adding pseudodata

statistical uncertainties



"systematic" uncertainties

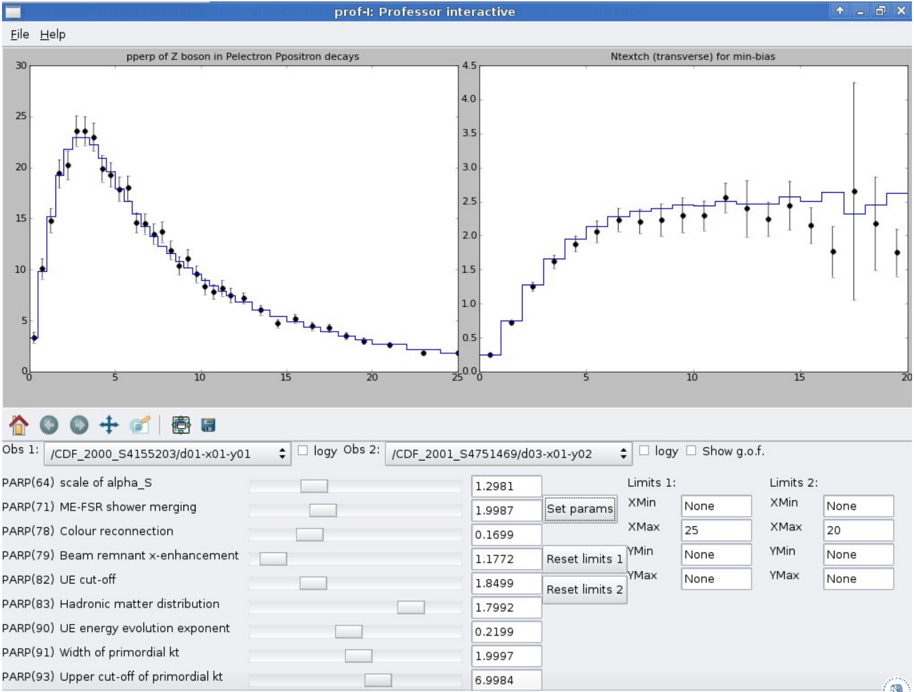


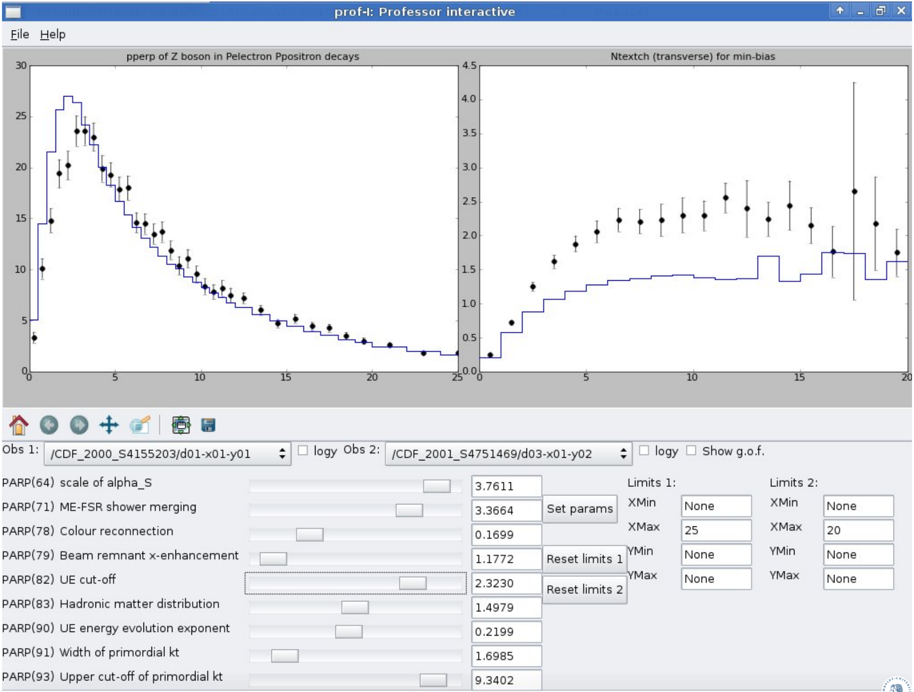
GUI-tool

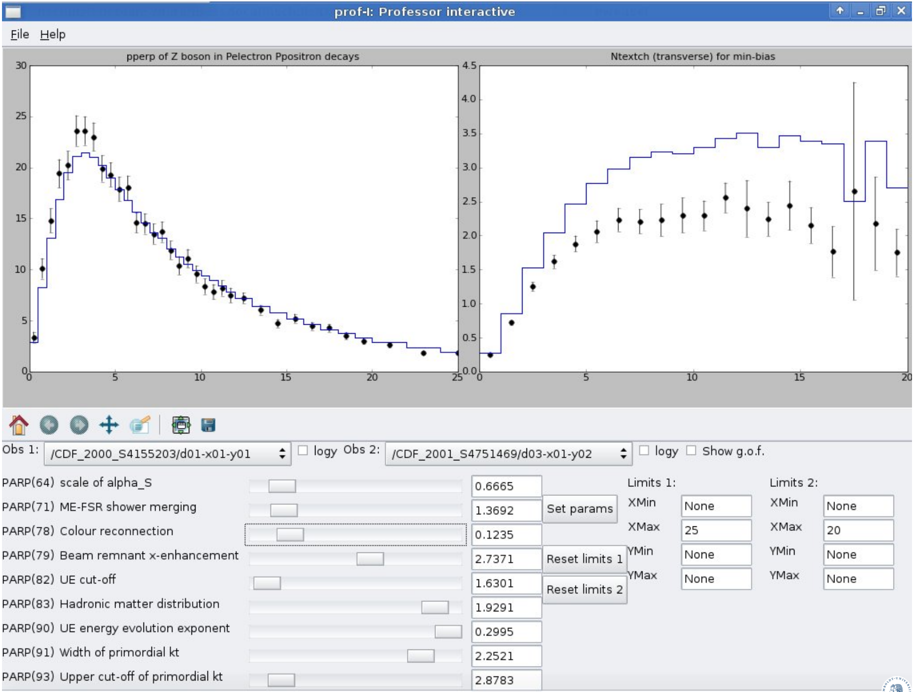
Once the generator is parameterised, we can easily produce (approximate) predictions of observables at any point in our parameter-space.

With prof-I (based on wxPython) we can do this interactively:

- adjust sliders (1 per parameter)
- get observable predictions in real-time (running generator would take $\sim \mathcal{O}$(days) and compare with reference data
- investigate effect of parameter-shifts on two observables at a time


prof-I: Professor interactive
File
Help
pperp of Z boson in Pelectron Ppositron decays
Ntextch (transverse) for min-bias
Obs 1:
/CDF_2000_S4155203/d01-x01-y01
logy
Obs 2:
/CDF_2001_S4751469/d03-x01-y02
logy
Show g.o.f.
PARP(64) scale of alpha_S
1.2981
PARP(71) ME-FSR shower merging
1.9987
PARP(78) Colour reconnection
0.1699
PARP(79) Beam remnant x-enhancement
1.1772
PARP(82) UE cut-off
1.8499
PARP(83) Hadronic matter distribution
1.7992
PARP(90) UE energy evolution exponent
0.2199
PARP(91) Width of primordial kt
1.9997
PARP(93) Upper cut-off of primordial kt
6.9984
Set params
Reset limits 1
Reset limits 2
Limits 1:
XMin None
XMax 25
YMin None
YMax None
Limits 2:
XMin None
XMax 20
YMin None
YMax None


prof-I: Professor interactive
File
Help
pperp of Z boson in Pelectron Ppositron decays
Ntextch (transverse) for min-bias
Obs 1:
/CDF_2000_S4155203/d01-x01-y01
logy
Obs 2:
/CDF_2001_S4751469/d03-x01-y02
logy
Show g.o.f.
PARP(64) scale of alpha_S
3.7611
PARP(71) ME-FSR shower merging
3.3664
PARP(78) Colour reconnection
0.1699
PARP(79) Beam remnant x-enhancement
1.1772
PARP(82) UE cut-off
2.3230
PARP(83) Hadronic matter distribution
1.4979
PARP(90) UE energy evolution exponent
0.2199
PARP(91) Width of primordial kt
1.6985
PARP(93) Upper cut-off of primordial kt
9.3402
Set params
Reset limits 1
Reset limits 2
Limits 1:
XMin None
XMax 25
YMin None
YMax None
Limits 2:
XMin None
XMax 20
YMin None
YMax None


prof-I: Professor interactive
File Help
pperp of Z boson in Pelectron Ppositron decays
Ntextch (transverse) for min-bias
Obs 1: /CDF_2000_S4155203/d01-x01-y01
logy
Obs 2: /CDF_2001_S4751469/d03-x01-y02
logy
Show g.o.f.
PARP(64) scale of alpha_S 0.6665
PARP(71) ME-FSR shower merging 1.3692
PARP(78) Colour reconnection 0.1235
PARP(79) Beam remnant x-enhancement 2.7371
PARP(82) UE cut-off 1.6301
PARP(83) Hadronic matter distribution 1.9291
PARP(90) UE energy evolution exponent 0.2995
PARP(91) Width of primordial kt 2.2521
PARP(93) Upper cut-off of primordial kt 2.8783
Set params
Reset limits 1
Reset limits 2
Limits 1:
XMin None
XMax 25
YMin None
YMax None
Limits 2:
XMin None
XMax 20
YMin None
YMax None

Summary

- Professor is a mature tool for systematic generator tuning
- Have provided tunings for Pythia6/8, Sherpa, Jimmy
- Tunings systematically under control, repeatable
- Can study effects of parameter-shifts interactively
- Working on quantification of tuning uncertainties

Thank you!

A. Buckley et al., 2009

Systematic event generator tuning for the LHC

arXiv:0907.2973, accepted for publication in EPJC

- Website of the Professor project (plots, howtos, exercises, ...)

http://projects.hepforge.org/professor

If you are interested, contact us!

Backup

Professor tunes in Pythia 6:

```
6.4.20 : 20 February 2009
  - Comprehensive updates to PYTUNE, with the addition of the
    "Perugia" and "Pro" tunes, following the MPI workshop in Perugia
    in October 2008. The older tunes remain unaltered. The new
    available tunes in PYTUNE are:
    ===== Old UE, Q2-ordered showers ====================================
    --- Professor Tunes : 110+ (= 100+ with Professor's tune to LEP) ----
     110   A-Pro : Tune A, with LEP tune from Professor        (Oct 2008)
     111  AW-Pro : Tune AW, -"-                                (Oct 2008)
     112  BW-Pro : Tune BW, -"-                                (Oct 2008)
     113  DW-Pro : Tune DW, -"-                                (Oct 2008)
     114 DWT-Pro : Tune DWT, -"-                               (Oct 2008)
     115  QW-Pro : Tune QW, -"-                                (Oct 2008)
     116 ATLAS-DC2-Pro: ATLAS-DC2 / Rome, -"-                  (Oct 2008)
     117 ACR-Pro : Tune ACR, -"-                               (Oct 2008)
     118  D6-Pro : Tune D6, -"-                                (Oct 2008)
     119 D6T-Pro : Tune D6T, -"-                               (Oct 2008)
    --- Professor's Q2-ordered Perugia Tune : 129 -----------------------
     129 Pro-Q20 : Professor Q2-ordered tune                   (Feb 2009)
    ===== Intermediate and Hybrid Models ================================
     211 APT-Pro : Tune APT, with LEP tune from Professor      (Oct 2008)
    . . .
     ===== New UE, interleaved pT-ordered showers, annealing CR ==========
    --- Professor Tunes : 310+ (= 300+ with Professor's tune to LEP)
     310   S0-Pro : S0 with updated LEP pars from Professor    (Oct 2008)
     311   S1-Pro : S1 -"-                                     (Oct 2008)
     312   S2-Pro : S2 -"-                                     (Oct 2008)
     313  S0A-Pro : S0A -"-                                    (Oct 2008)
     314 NOCR-Pro : NOCR -"-                                   (Oct 2008)
     315  Old-Pro : Old -"-                                    (Oct 2008)
     . . .
    --- Professor's pT-ordered Perugia Tune : 329 -----------------------
     329 Pro-pT0   : Professor pT-ordered tune w. S0 CR model  (Feb 2009)
    =====================================================================
```

2nd order polynomial includes lowest-order correlations between parameters

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

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 $\vec{c}_b$: $N_{\text{min}}^{(n)} = 1 + n + n(n+1)/2$

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

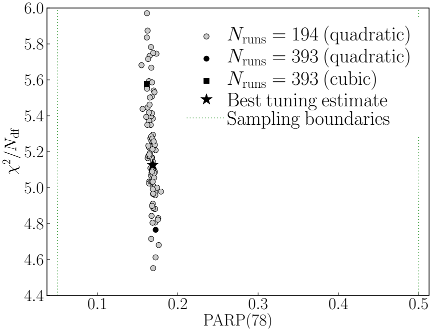
- Use Singular Value Decomposition (SVD), a general diagonalisation for all normal matrices M:$M = U\Sigma V^*$
- Method available in SciPy.linalg
- Minimal number of runs = number of coefficients in $\vec{c}_b$:
  $N_{\text{min}}^{(n)} = 1 + n + n(n+1)/2 + \underbrace{(n+1)(n+2)/6}_{\text{cubic only}}$

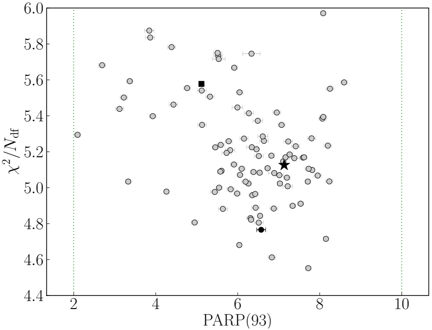
$$\boxed{\vec{c}_b = \tilde{\mathcal{I}}[\tilde{\mathbf{P}}]\vec{v}}$$

- 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

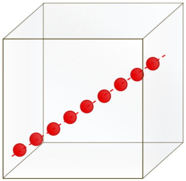
Num params, P	$N_2^{(P)}$ (2nd order)	$N_3^{(P)}$ (3rd order)
1	3	4
2	6	10
4	15	35
6	28	84
8	45	165
9	55	220

- If we have N generator runs: choose combinations of k ($k < N$) runs
- So far: $k \approx 3 \cdot N_{\text{min}}$ and $k/N \approx 0.66$
- Each combination → different parameterisation → (slightly) different minimisation result
- Investigate spread parameter-wise

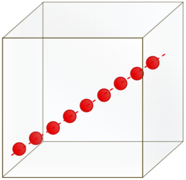


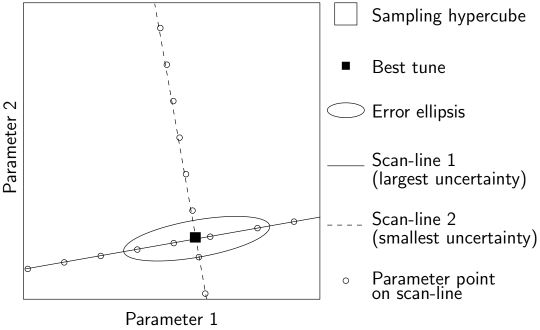


- Sample points from a straight line in parameter space
- Run generator, compare goodness of fit to that of parameterisation

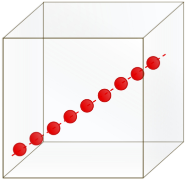


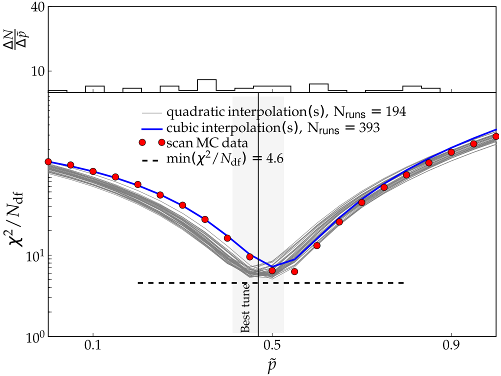
- Sample points from a straight line in parameter space
- Run generator, compare goodness of fit to that of parameterisation



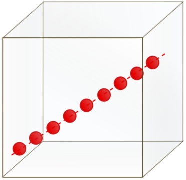


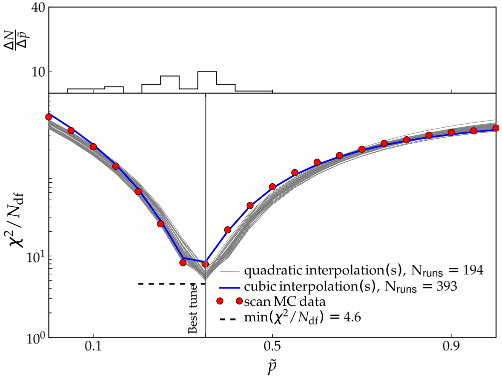
- Sample points from a straight line in parameter space
- Run generator, compare goodness of fit to that of parameterisation



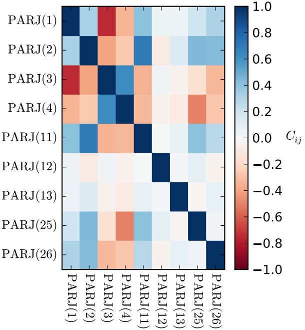


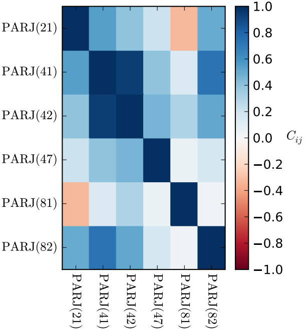
- Sample points from a straight line in parameter space
- Run generator, compare goodness of fit to that of parameterisation

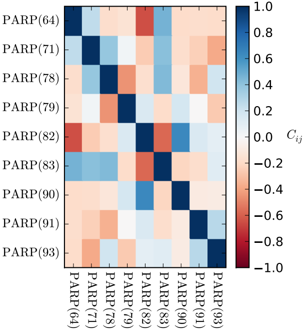




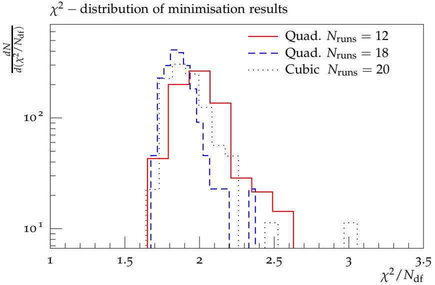
- Use covariance matrix determined by minimiser to calculate parameter-parameter correlations
- Display provided as colour map or table



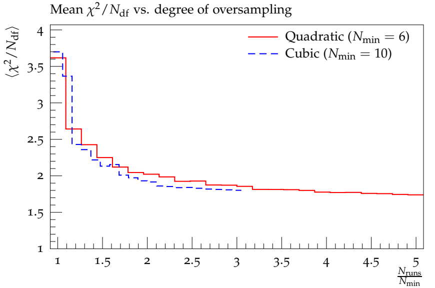




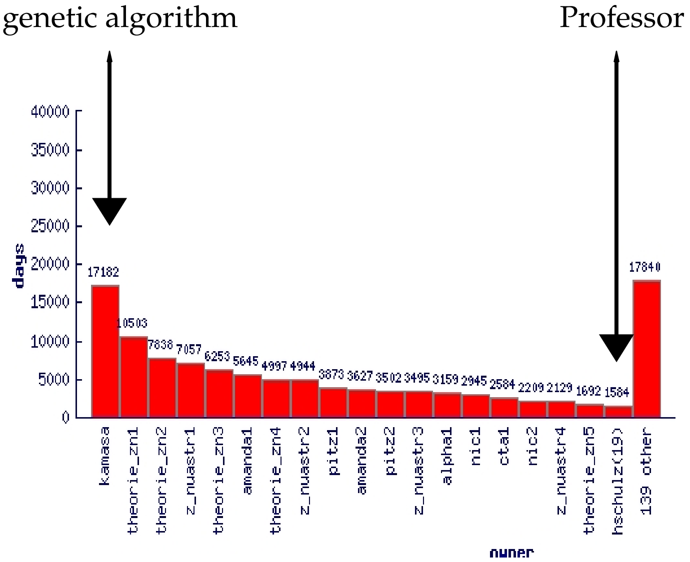
Observe lower χ^2/N_{df}-boundary:



Oversampling is neccesary (at least 2 to 3 times N_{min}):

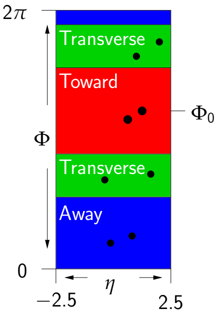


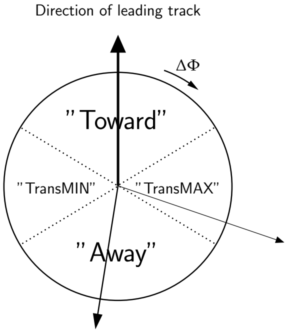
One year DESY (Zeuthen) batch farm usage, CPU-time per user (Top 20)



Leading track

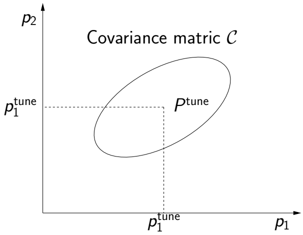
- Measure track-$p_\perp$ using only inner detector
- Identify leading track = largest $p_\perp$ in event $\rightarrow$ defines ϕ_0
- Define "transverse" region, measure N_{tracks}, scalar $p_\perp$-sum as function of $p_{\perp\text{, leading track}}$





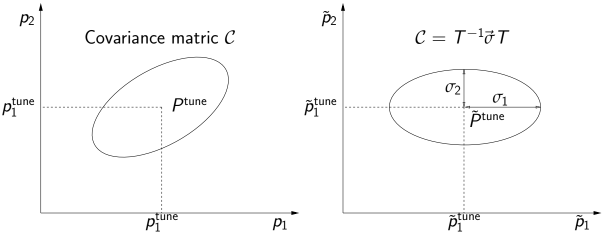
Tuning uncertainties

- get cov. matrix



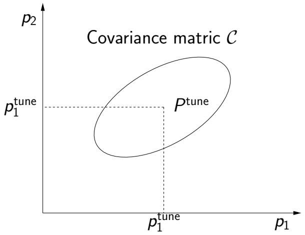
Tuning uncertainties

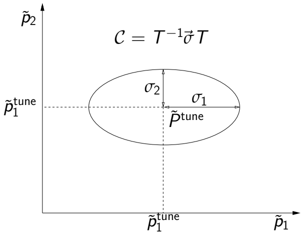
- get cov. matrix
- Eigendecomp.

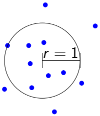


Tuning uncertainties

- get cov. matrix
- Eigendecomp.
- Sample from $\mathcal{G}^N$

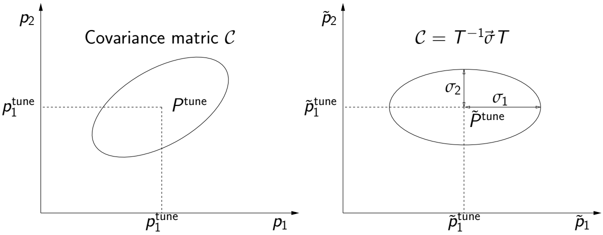


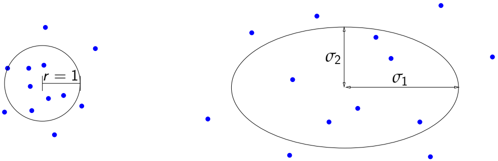




Tuning uncertainties

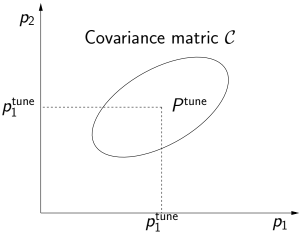
- get cov. matrix
- Eigendecomp.
- Sample from $\mathcal{G}^N$
- Stretch by σ_i

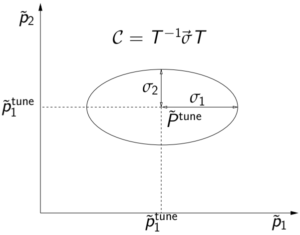


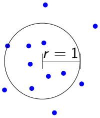


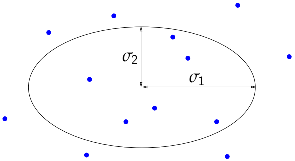
Tuning uncertainties

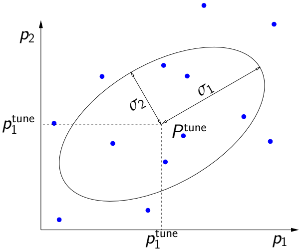
- get cov. matrix
- Eigendecomp.
- Sample from $\mathcal{G}^N$
- Stretch by σ_i
- Rotate back with T





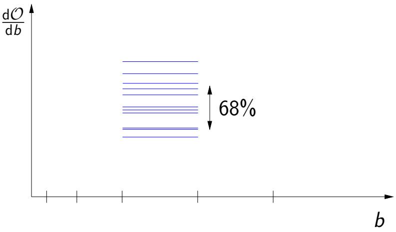






Confidence belt construction

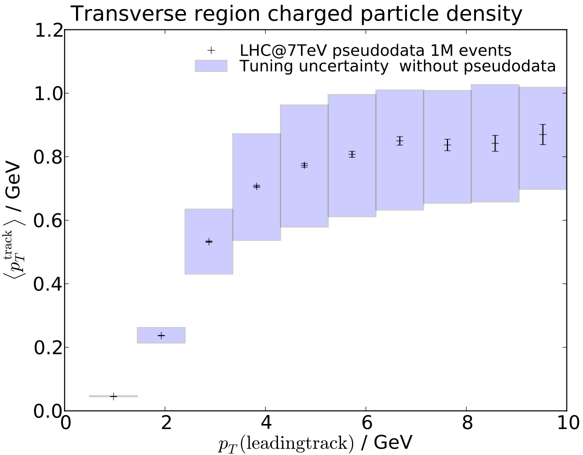
1. Estimate tuning uncertainties: use points sampled from ellipse
2. Run generator or use parameterisation to get bin-content prediction
3. For each bin b and each observable $\mathcal{O}$: determine central 68, 95 pct.



$\longrightarrow$ Use confidence belts for early data sensitivity studies, i.e., if we add LHC (pseudo-) data to the existing tune, does the confidence belt shrink? If so, consider corresponding measurement worthwile for early data.

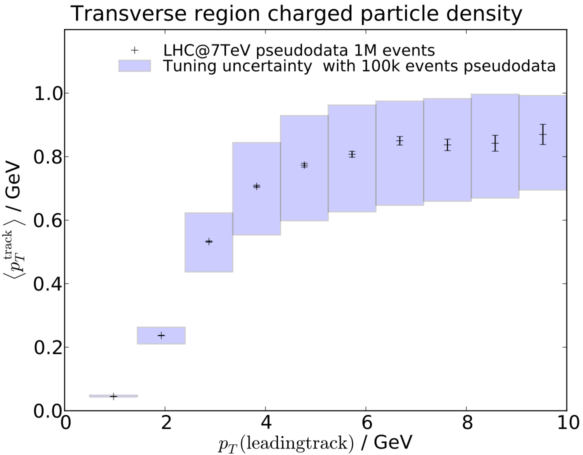
Leading track sensitivity study (generator level)

Pythia6 (tune 329) prediction for the LHC ($\sqrt{s}$ =7 TeV)



Leading track sensitivity study (generator level)

Pythia6 (tune 329) + 100k events of pseudo-data



Leading track sensitivity study (generator level)

Pythia6 (tune 329) + 1M events of pseudo-data