Unfolding methods in HEP

- Introduction on unfolding
- Example unfolding problem
- Unfolding methods
- Comparison

Extracted from slides:

T. Adye at PHYSTAT 2016 and K. Reygers lectures at Heilderbeg Univ.
S. Smitt and D. Britzger, DESY Stat School 2014
S. Schmitt , QCDHS conference in 2016

Unfolding

- Unfolding: estimate truth distribution from measurement, distorted by
  - detector effects
  - statistical fluctuations
- truth distribution: cross sections or similar quantities
- Unfolding is also referred to as “correction for detector effects”

- Integral equation of 1st kind

$$\int k(x,y) f(y)\, dy + \delta(x) = g(x)$$

given observations $g(x)$

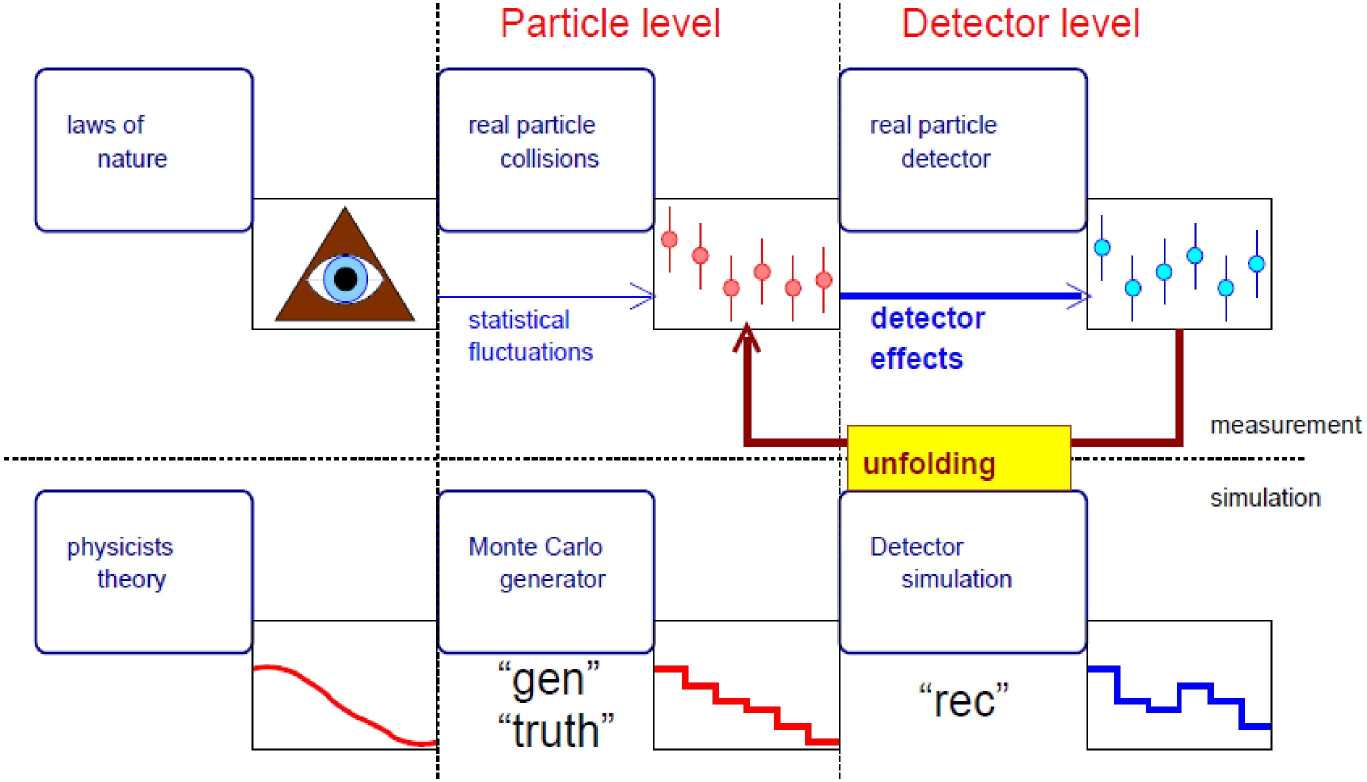
the kernel $k(x,y)$

and fluctuations $\delta(x)$

estimate the truth $f(y)$

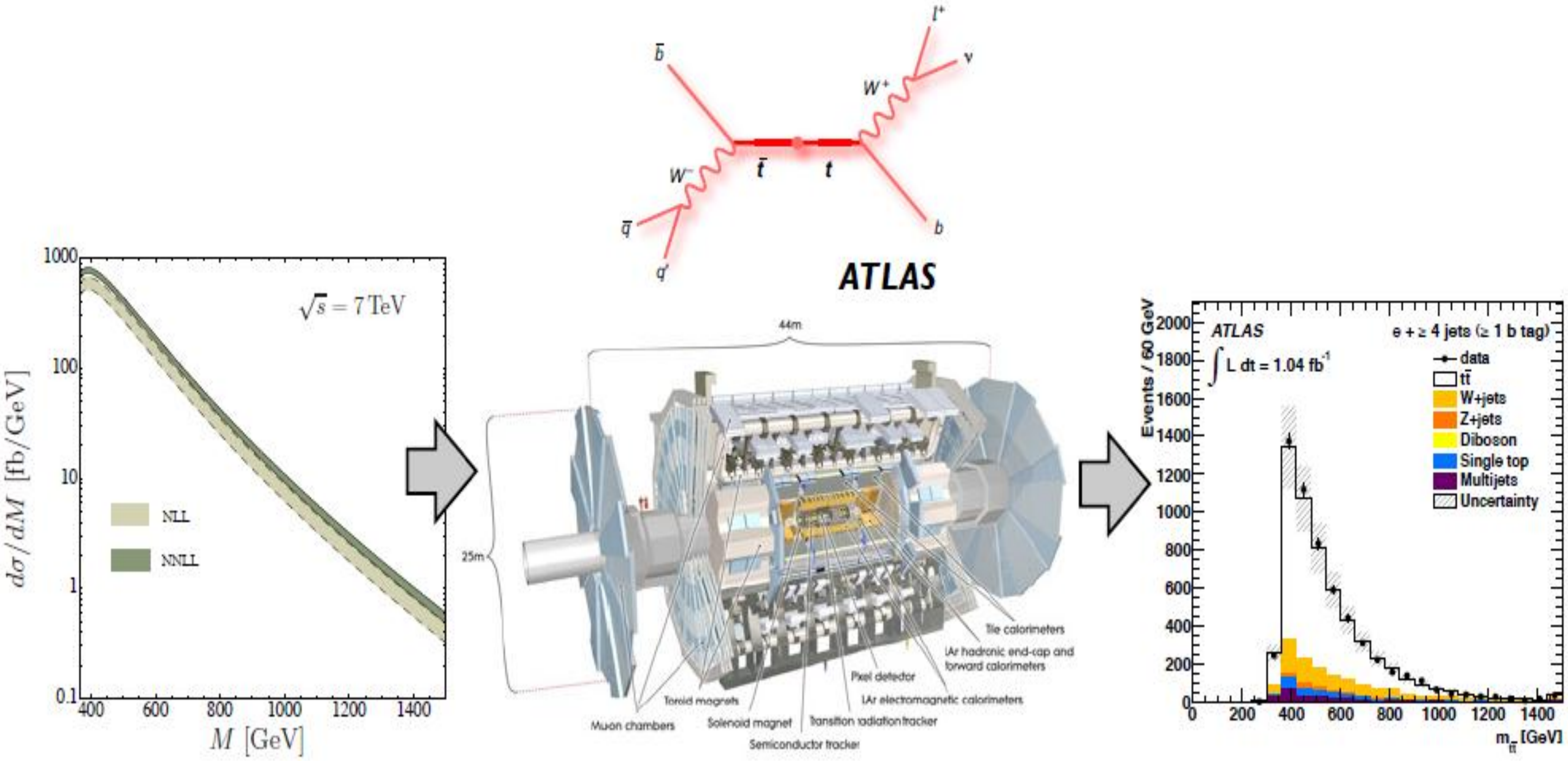
- k(x,y): detector effects, background, etc
- g(x) has uncertainties
- k(x,y) has syst. uncertainties → not covered in this talk

What is unfolding



- Obtain measurements independent of detector effects, using the simulation
- Propagate statistical uncertainties back to particle level
- Require results to be independent of theory assumptions

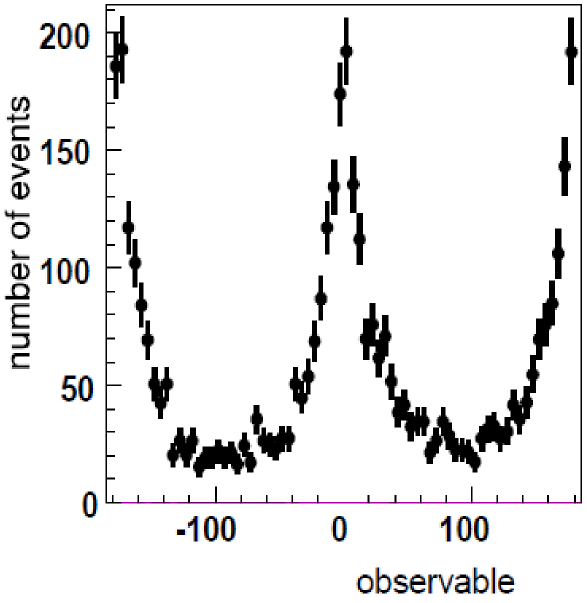
ATLAS analysis example

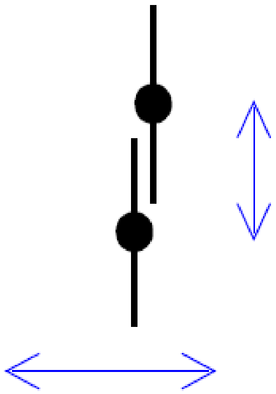


G. Aad *et al.* [ATLAS Collaboration], Eur. Phys. J. C **72** (2012) 2039, arXiv:1203.4211 [hep-ex].

Migrations and stat. fluctuations

Histogram of observed event counts is affected by statistical fluctuations (vertical axis) and detector effects (horizontal axis)





statistical fluctuations

migrations due to detector effects

Unfolding: correct for migration effects in the presence of statistical fluctuations

Result: estimator of the “truth” and its covariance matrix (statistical uncertainties)

Unfolding of binned measurement

- unfolding of binned (discrete) distributions, where bin-to-bin migrations are described by a matrix equation

$$\mu_i = \sum A_{ij} x_j + b_i$$

μ_i : expected measurement in bin i given the truth x

A_{ij} : probability of truth bin j to reconstruct in bin i

x_j : truth in bin j

b_i : background in bin i

$$A_{ij} = \frac{N_{ij}^{\text{MCreco,MCtruth}}}{N_j^{\text{MCtruth}}} \text{ is calculated from MC}$$

- Statistical fluctuations: the observations y_i are drawn from a Poisson distribution

$$P(y_i;\mu_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}$$

- Large sample limit: Gaussian distributions
- Correlated bins: multivariate Gaussians

Unfolding of binned measurement

- unfolding of binned (discrete) distributions, where bin-to-bin migrations are described by a matrix equation

- Statistical fluctuations: the observations y_i are drawn from a Poisson distribution

(truth+background) × detector × stat.fluctuations → measurement

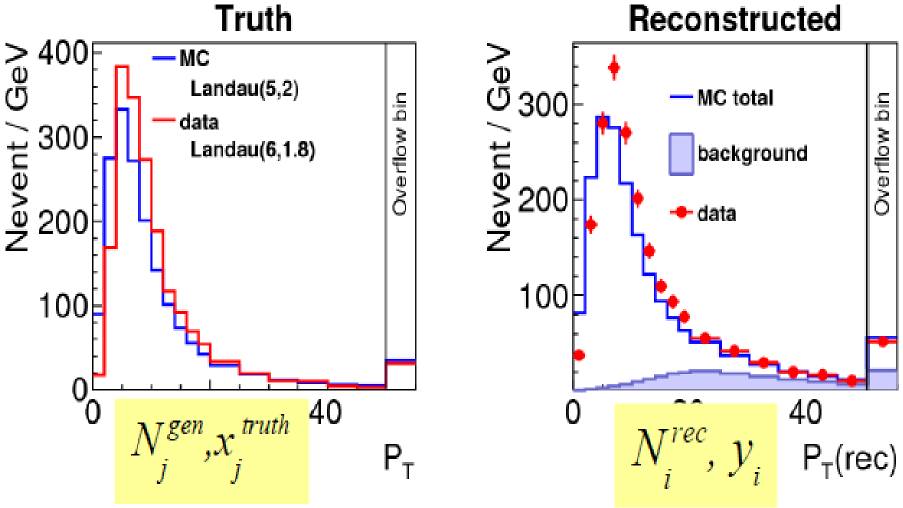
Result: estimator of truth ←unfolding algorithm ← measurement

Example of unfolding problem

arXiv:1611.01927

- Toy example to illustrate basic properties of unfolding algorithms
- Decay of a heavy particle into two light particles
- Light particles smeared by spatial and energy resolution
- Trigger threshold causes reconstruction inefficiency
- Background important at high P_T
- Variable bin size, overflow bin
- Goal: reconstruct P_T distribution

- Two samples of toy events
  - "data" P_T distribution following Landau(6,1.8)
  - "MC" P_T distribution following Landau(5,2)
- Background mainly at high P_T



N_j^{gen}, x_j^{truth}

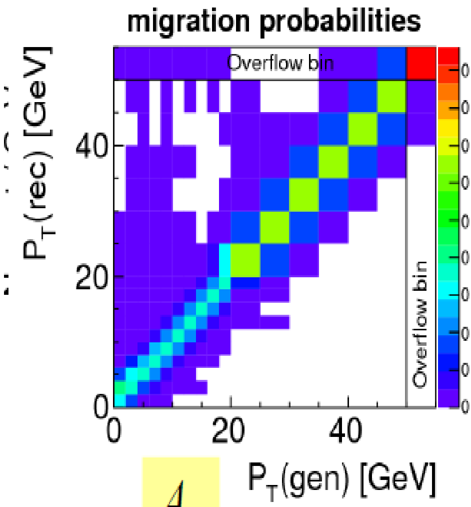
N_i^{rec}, y_i

Example of unfolding problem

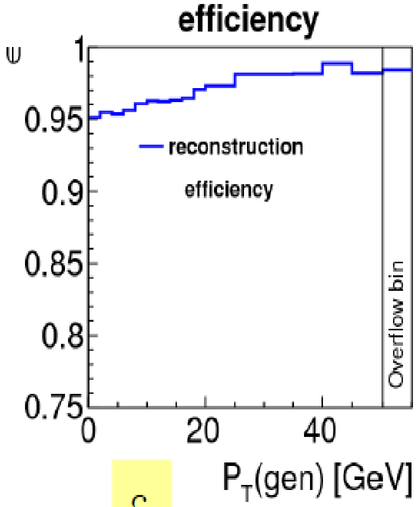
arXiv:1611.01927

- Toy example to illustrate basic properties of unfolding algorithms
- Decay of a heavy particle into two light particles
- Light particles smeared by spatial and energy resolution
- Trigger threshold causes reconstruction inefficiency
- Background important at high P_T
- Variable bin size, overflow bin
- Goal: reconstruct P_T distribution

- Significant migrations at low P_T
- Change of bin size leads to change in bin purity
- Efficiency >95%, not important for this study



A_{ij}



ε_j

How to test unfolding results?

- Tests with real data
  - Look at (global) correlation coefficients
  - Trivial test: fold back unfolding result and compare to data

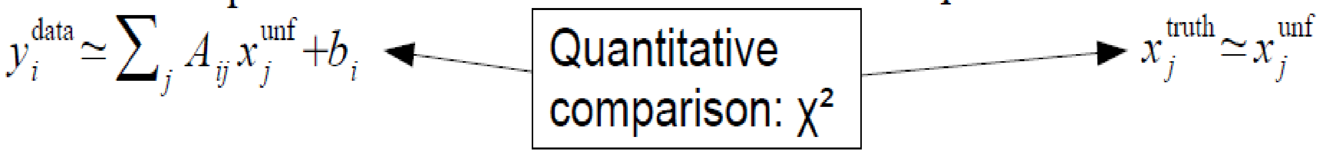
unfolding result: x_j^{unf}

fold back and compare to data:

- Test with Monte Carlo
  - Trivial test: response matrix and MC using the same truth
  - Non-trivial test: use different truth for response matrix and

unfold alternative MC (here: "data"): x_j^{unf}

compare to alternative MC truth:



$$y_i^{\text{data}} \simeq \sum_j A_{ij} x_j^{\text{unf}} + b_i$$

Quantitative comparison: χ²

$$x_j^{\text{truth}} \simeq x_j^{\text{unf}}$$

Look at average global correlation coefficients

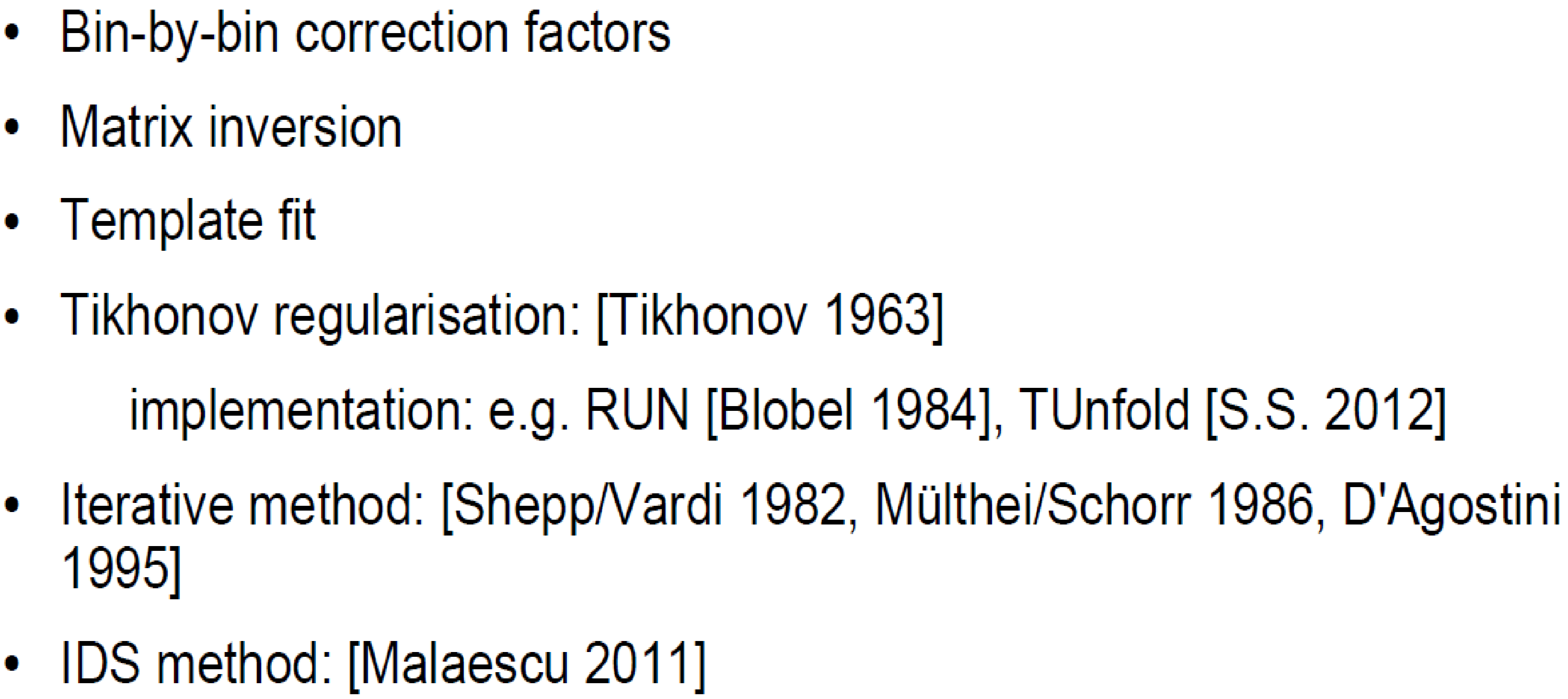
Compare folded result with data

Compare result to “data” truth

Extract “data” truth parameters using a fit

… plus many other things not discussed here, e.g. eigenvalue analysis

Unfolding methods



- Bin-by-bin correction factors
- Matrix inversion
- Template fit
- Tikhonov regularisation: [Tikhonov 1963]

  implementation: e.g. RUN [Blobel 1984], TUnfold [S.S. 2012]
- Iterative method: [Shepp/Vardi 1982, Mülthei/Schorr 1986, D'Agostini 1995]
- IDS method: [Malaescu 2011]

Bin-by-bin correction factors

- Very simple method:

$$x_i = (y_i - b_i)\frac{N_i^{\text{gen}}}{N_i^{\text{rec}}}$$

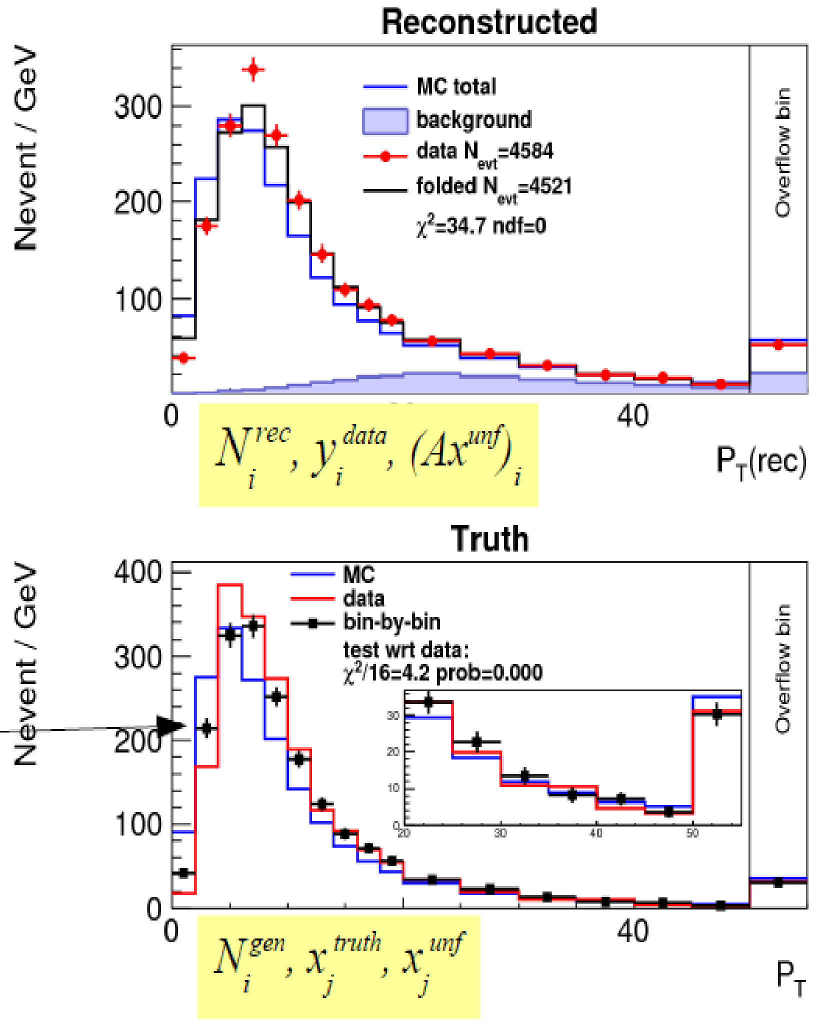
Correction factor

y_i : observed in bin i

b_i : expected backround in bin i

N_i^{gen} : MC truth in bin i

$N_i^{\text{rec}} = \sum_j A_{ij} N_i^{\text{gen}}$: MC reconstructed in bin i



Results “looks nice”

No statistical bin-to-bin correlations

but

Method is wrong, fails very basic tests

Simple „Bin-by-bin”: why is it wrong?

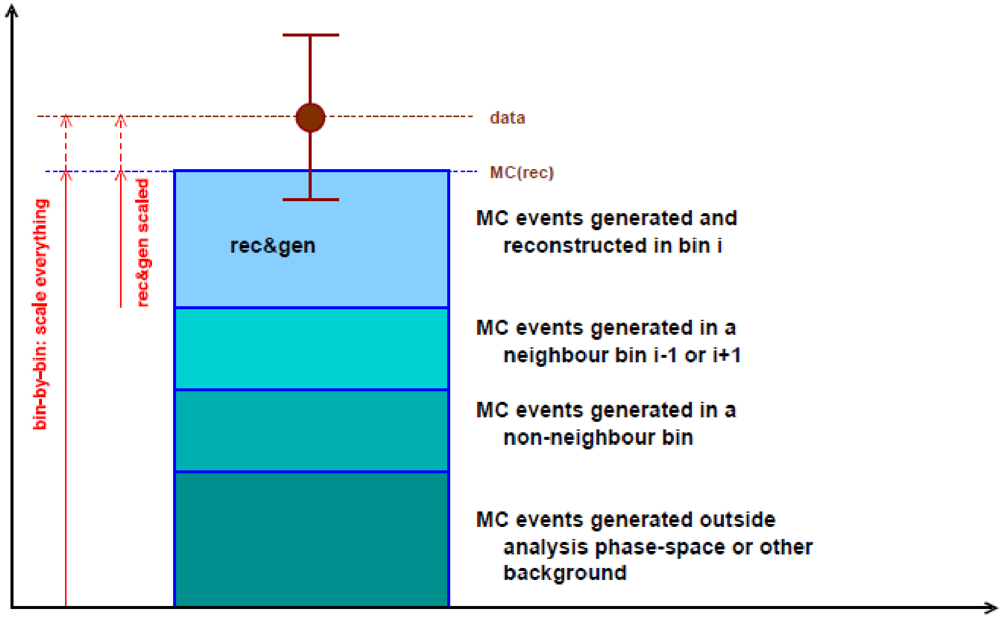
- Migrations are additive, while BBB correction is multiplicative → wrong type of correction

$$x_i^{\mathrm{BBB}} = x_i^{\mathrm{gen}} \frac{y_i^{\mathrm{data}}}{y_i^{\mathrm{rec}}}$$

- It should be:

$$x_i^{\mathrm{BBBSUB}} = x_i^{\mathrm{gen}} \frac{y_i^{\mathrm{data}} - \left(y_i^{\mathrm{rec}} - y_i^{\mathrm{rec\&gen}}\right)}{y_i^{\mathrm{rec\&gen}}}$$

$$= x_i^{\mathrm{gen}} \frac{y_i^{\mathrm{data}} - y_i^{\mathrm{rec}}\left(1 - P_i\right)}{y_i^{\mathrm{rec}} P_i}$$


data
MC(rec)
bin-by-bin: scale everything
rec&gen scaled
rec&gen
MC events generated and reconstructed in bin i
MC events generated in a neighbour bin i-1 or i+1
MC events generated in a non-neighbour bin
MC events generated outside analysis phase-space or other background

- Relevant quantity: purity

$$P_i = \frac{y_i^{\mathrm{rec\&gen}}}{y_i^{\mathrm{rec}}}$$

Matrix methods

- All matrix methods are based on the matrix of probabilities:

$$\text{Expected number of events in bin } i: \ \mu_i = \sum A_{ij} x_j^{\text{truth}}$$

- The A$_{ij}$ are calculated from Monte Carlo

$$A_{ij} = \frac{y_{ij}^{\text{rec,gen}}}{y_j^{\text{gen}}} \text{ and the reconstruction efficiencies are } \varepsilon_j = \sum_i A_{ij}$$

- A$_{ij}$ is normalized to the generated number of events in bin j, so it is (largely) model independent, only depends on the detector response.

Matrix inversion

- If the number of bins is equal on gen and rec level: A is a square matrix

  → invert it

folding equation: $y = Ax + b$

invert matrix: $x = A^{-1}(y - b)$

Covariance: $V_{xx} = A^{-1} V_{yy} (A^{-1})^T$

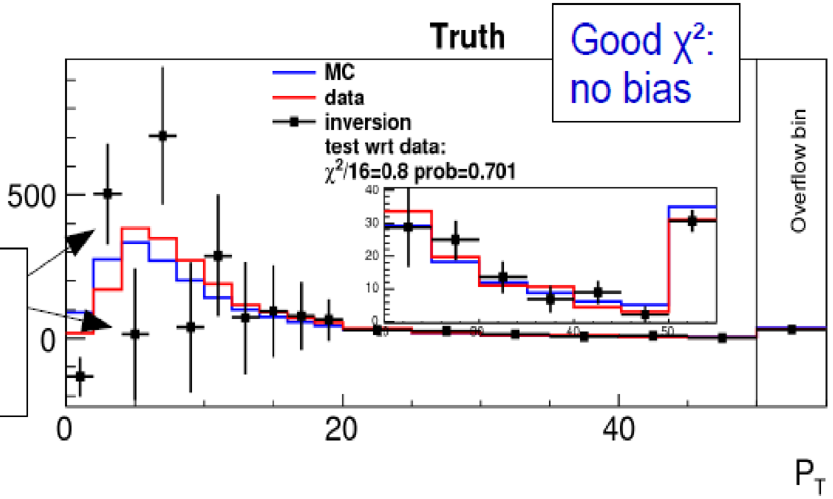
correlation coefficients: $\rho_{ij} = \frac{(V_{xx})_{ij}}{\sqrt{(V_{xx})_{ii}(V_{xx})_{jj}}}$

y : measurements

V_{yy} : covariance matrix of measurements

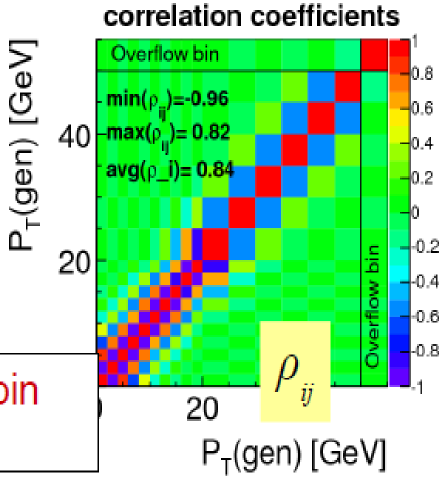
b : background

A : matrix of migrations

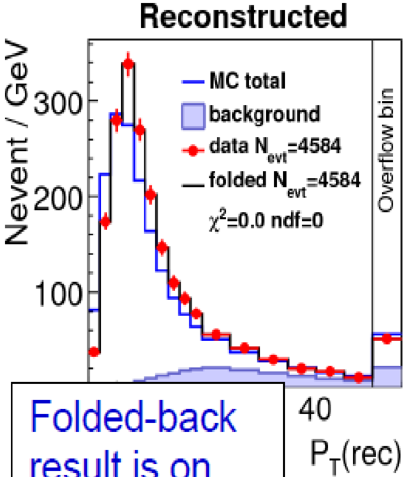


Good χ^2: no bias

Unfolded result exhibits bin-to-bin oscillations



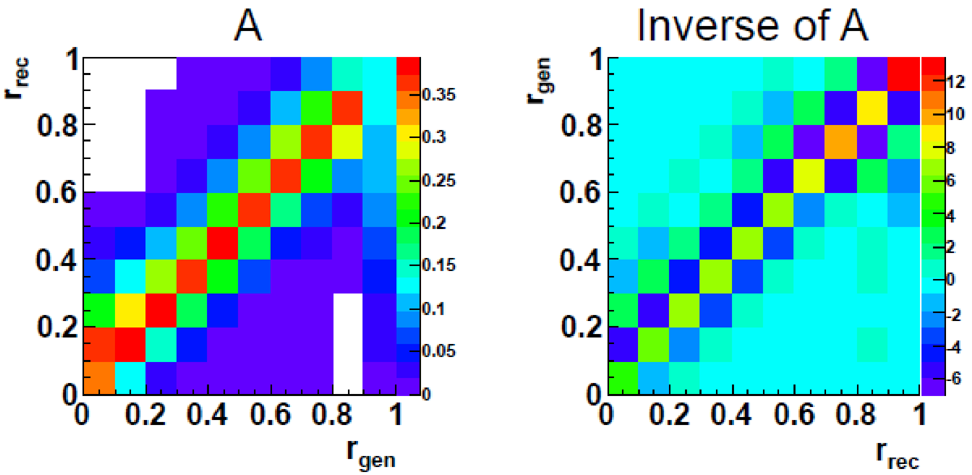
Large bin-to-bin correlations



Folded-back result is on the data

Cause of large fluctuations

- Matrix inversion: creates large negative off-diagonals
  → statistical fluctuations of the data are amplified
- Possible improvements
  - Avoid matrix inversion “Bayesian” or “Iterative”
  - Use more reconstructed bins → TFractionFitter, TUnfold
  - Regularisation:
    TSVDUnfold, TUnfold



Template fit

- Choose larger number of reconstructed bins than truth bins → least-square fit
- Idea: use more information → obtain better result?

$$\chi^2 = (y-b-Ax)^T V_{yy}^{-1} (y-b-Ax)$$

y : measurements

V_{yy} : covariance matrix of measurements

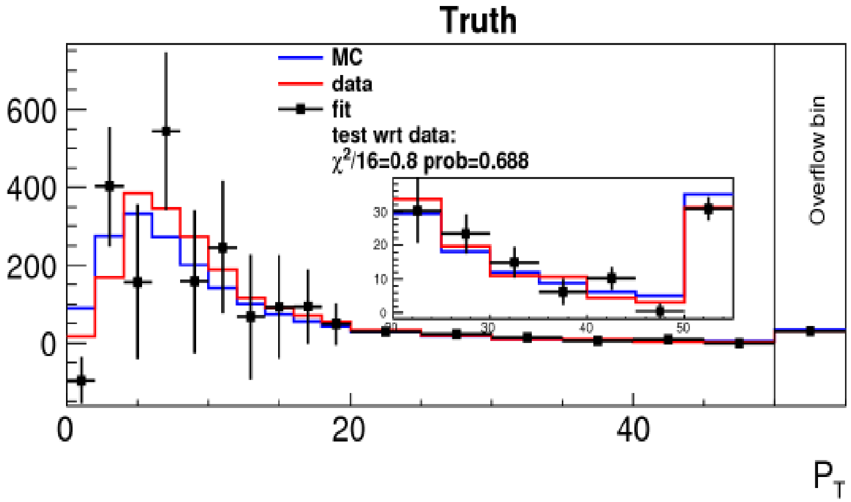
b : background

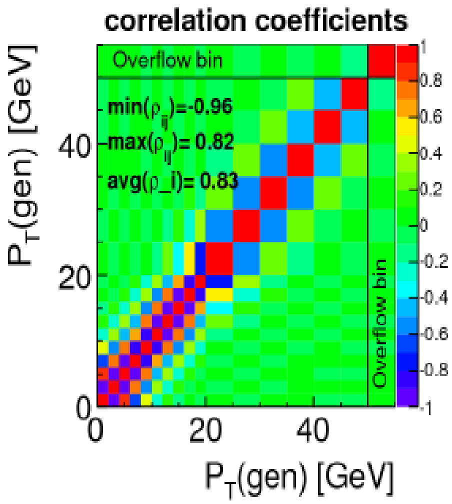
A : matrix of migrations

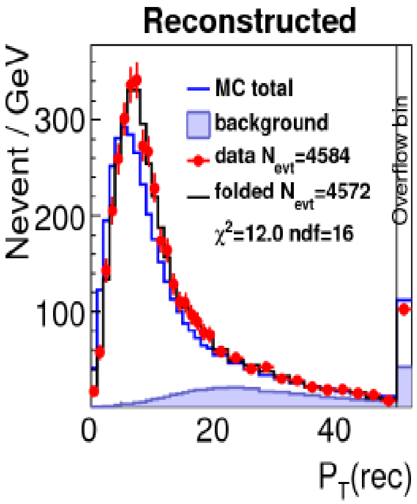
A_{ij} : MC template for truth bin j

$$x = (A^T V_{yy}^{-1} A)^{-1} A^T V_{yy}^{-1} (y-b)$$

covariance of x : $V_{xx} = (A^T V_{yy}^{-1} A)^{-1}$







Template fit

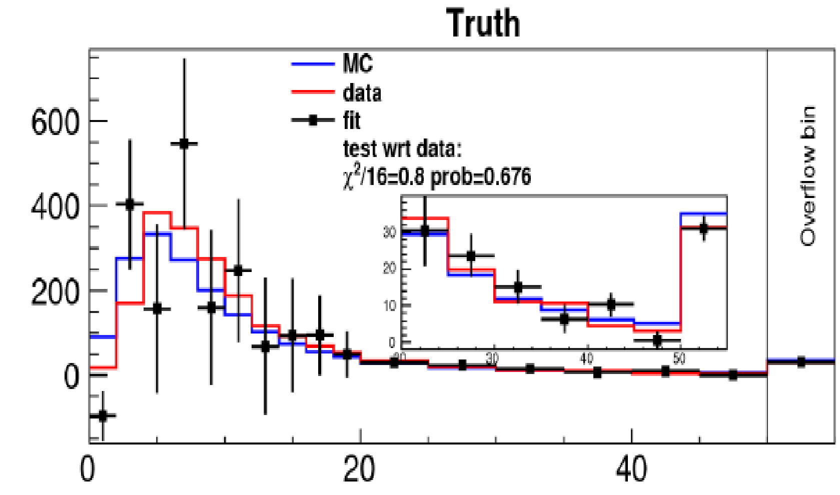
- Choose larger number of reconstructed bins than truth bins → least-square fit
- Idea: use more information → obtain better result

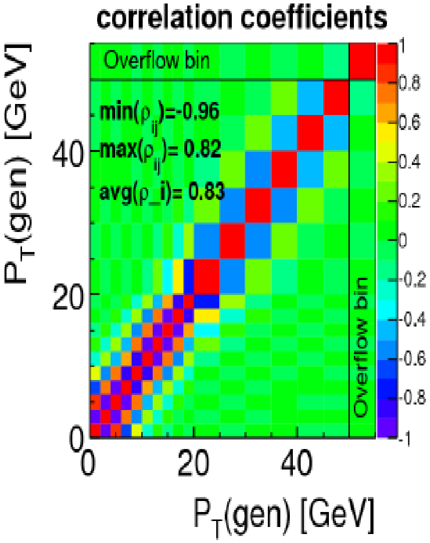
→ Result does not improve much over matrix inversion in this example

New problem: normalisation is not preserved [N_{data}=4584, N_{fold}=4572]

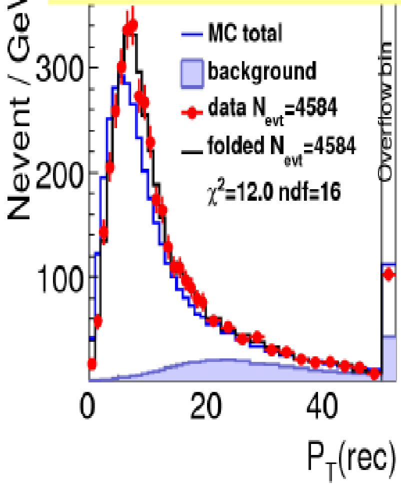
Well-known problem with least-square fits to Poisson-distributed data if sqrt(N) uncertainties are used

Can be improved by adding a constraint to the fit





2x more (finer) bins for reconstructed quantities



Template fit with area constraint

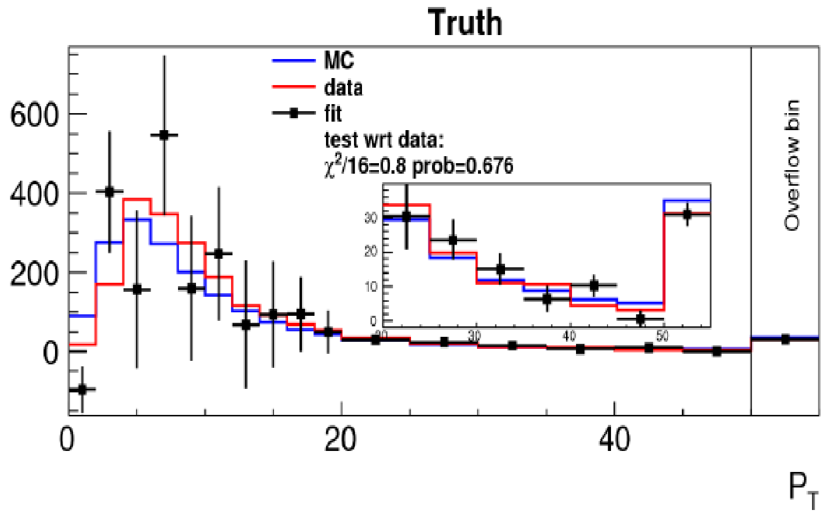
- Template with with constraint on the total number of events
- Basic idea: preserve normalisation for the folded-back result by adding the constraint

$$\sum (y_i - b_i) = \sum_{i,j} A_{ij} x_j$$

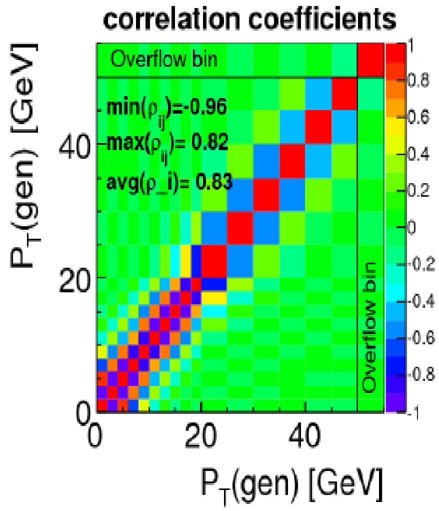
- Technical implementation: see TUnfold documentation

→ Result does not change much over unconstrained template fit, but normalisation is recovered

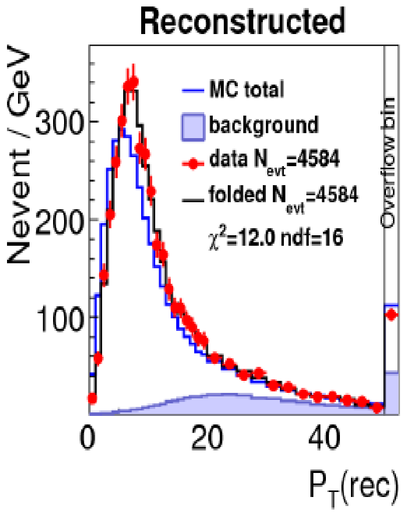
[N_{data} = N_{fold} =4584]



Truth



correlation coefficients



Reconstructed

Tikhonov regularisation

- Basic idea: add terms to the likelihood which damp oscillations in the result.

$$\chi^2 = (y - b - Ax)^T V_{yy}^{-1} (y - b - Ax) + \tau^2 (L(x - x_B))^T L(x - x_B)$$

y : measurements

V_{yy} : covariance matrix of measurements

b : background

A : matrix of migrations

x_B : regularisation bias

L : regularisation conditions

τ : regularisation strength

In addition, apply area constraint to preserve normalisation

- Regularisation bias x_B: set to zero or to MC truth
- Regularisation conditions L: set to unity matrix [or mimic second derivatives, “curvature”]
- Regularisation strength τ: “small” number

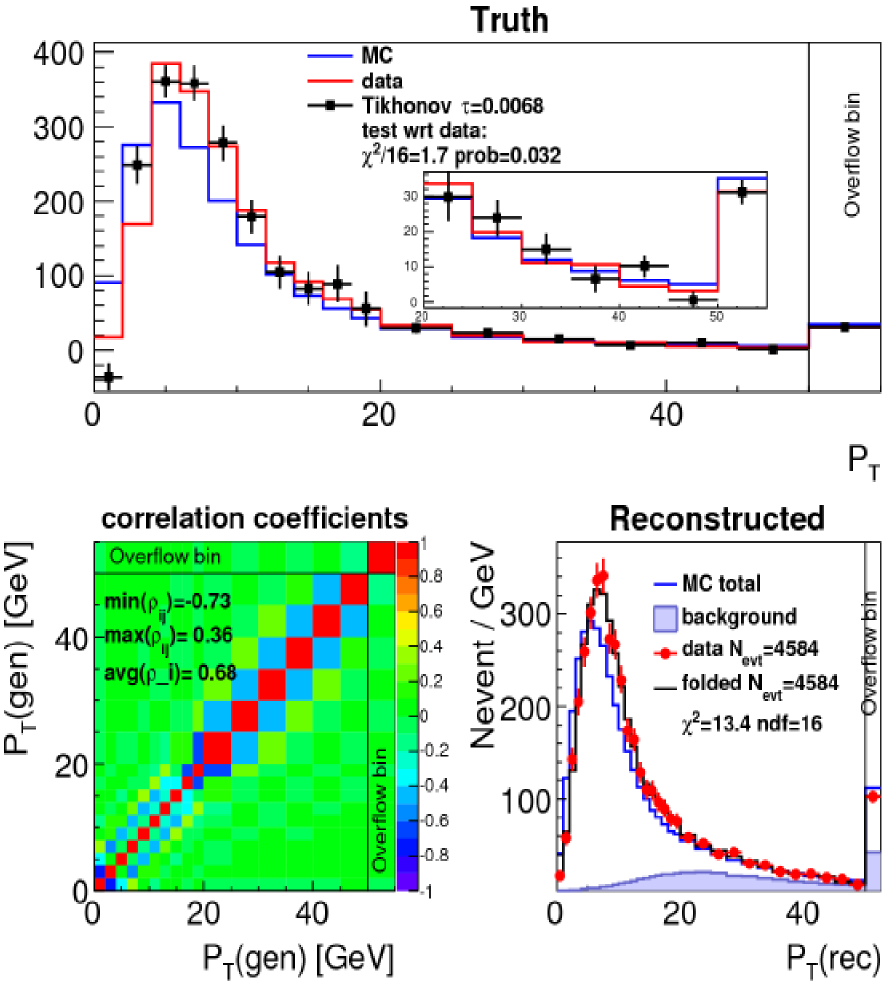
$$\tau \ll 1/\sigma$$

where σ~uncertainty after unfolding

Tikhonov regularisation (eg. TUnfold)

- Basic idea: add terms to the likelihood which damp oscillations in the result.
- This is working well: no oscillations, moderate correlations and uncertainties

  Basic tests look reasonable
- Question: objective to choose τ



Choice of the regularisation parameter

- Eigenvalue analysis (SVD)

  → not discussed

- Scan of parameter τ
  - L-curve scan
  - Scan of global correlation coefficients

- Other data driven methods (e.g. compare stat and syst errors, define convergence criteria) → not discussed

L-curve scan

- Algorithm is often used in medical image processing

for each τ repeat the unfolding:

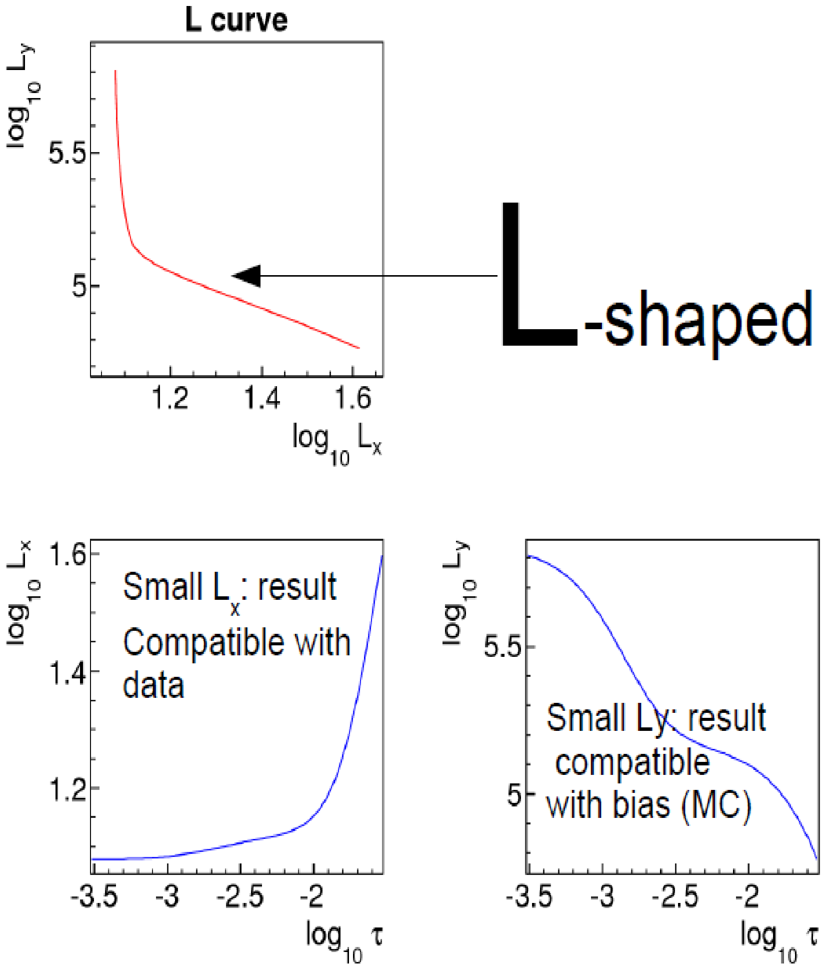
$$\chi^2 = (y-b-Ax)^T V_{yy}^{-1} (y-b-Ax)$$
$$+\tau^2 (L(x-x_B))^T L(x-x_B)$$
$$\equiv L_x + \tau^2 L_y$$

study parametric plot of: $\log L_x$ vs $\log L_y$

- Parametric plot is “L-shaped”

→ kink (largest curvature) defines τ

For a review, see: [P. C. Hansen 2000]



L-shaped

Scan of global correlation coeff.

- Global correlation coefficient (bin i)

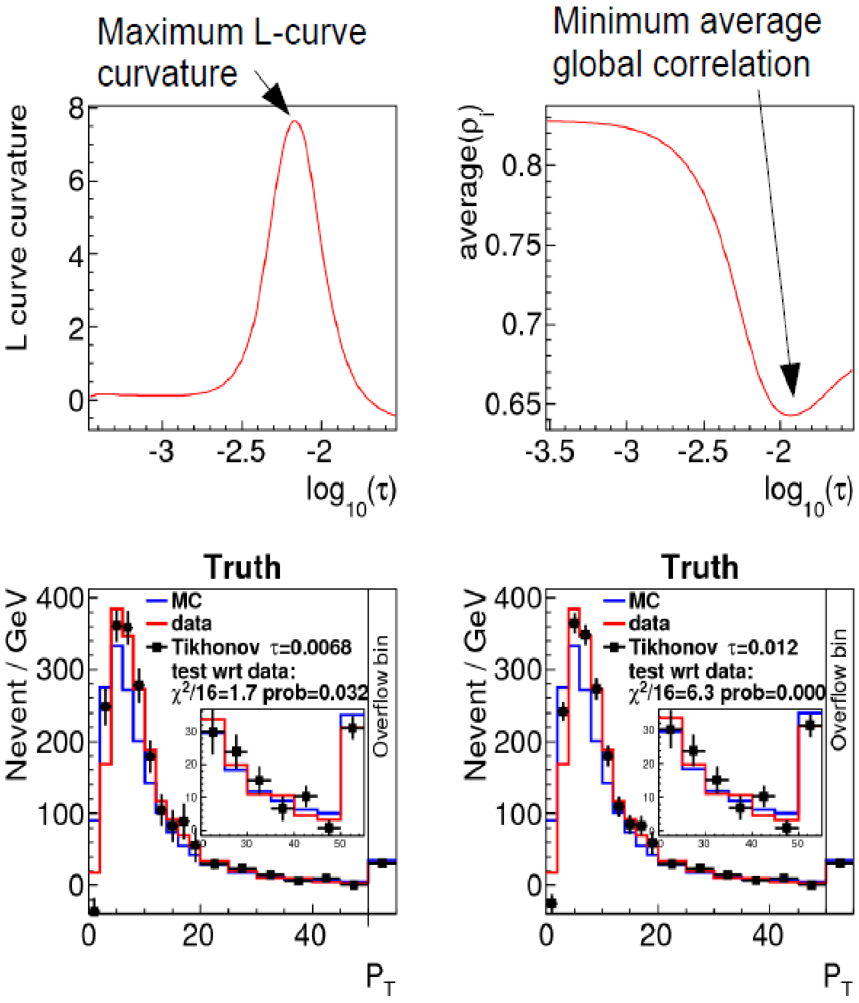
$$\rho_i = \sqrt{1 - \frac{1}{(V_{xx})_{ii}(V_{xx}^{-1})_{ii}}}$$

V_{xx} : result's covariance matrix

- Take average of all ρ_i and study dependence on τ → choose point with smallest avg(ρ_i)

(idea by V. Blobel/DESY)

- Comparison to L-curve scan: stronger regulatisation, more bias, smaller uncertainties & correlations



Iterative method

Ratio data to folded
→ iterate until ~1

$$x_j^{(N+1)} = x_j^{(N)} \sum_i \frac{A_{ij}}{\epsilon_j} \frac{y_i}{\sum_k A_{ik} x_k^{(N)}}$$

efficiency: $\epsilon_j = \sum_i A_{ij}$

start values: $x_j^{(-1)}$ [e.g. MC truth]

iterate until N is sufficiently large

- Original works by Shepp/Vardi 1982, Kondor 1983, Mülthei/Schorr 1987
- Re-invented by D'Agostini 1995 as "Iterative Bayesian unfolding"

Note: efficiency is absorbed in a redefinition of A, x in the original works: x'=εx and A'=A/ε

- Mathematical properties (Shepp/Vardi 1982 and Mülthei/Schorr 1987)
  - Ultimately converges to a maximum of the (Poisson) Likelihood

    → like matrix inversion but with all x≥0
  - Convergence is very slow
- Use in HEP:
  - Stop after N iterations → result will be "smooth" [regularized] but is biased to the start value

Regularisation strength:

Tikhonov: τ ↔ Iterative: N_{iter}

Iterative method with background

$$x_j^{(N+1)} = x_j^{(N)} \sum_i \frac{A_{ij}}{\epsilon_j} \frac{y_i - b_i}{\sum_k A_{ik} x_k^{(N)}}$$

efficiency: $\epsilon_j = \sum_i A_{ij}$

start values: $x_j^{(-1)}$ [e.g. MC truth]

OR

$$x_j^{(N+1)} = x_j^{(N)} \sum_i \frac{A_{ij}}{\epsilon_j} \frac{y_i}{\sum_k A_{ik} x_k^{(N)} + b_i}$$

efficiency: $\epsilon_j = \sum_i A_{ij}$

start values: $x_j^{(-1)}$ [e.g. MC truth]

- Background could be subtracted from the data
- Or: background could be added to the folded MC in the denominator. This guarantees the desired property x≥0
- D'Agostini suggests to include the background normalisation as extra bin x_{n+1}. This also guarantees x≥0 but results in an extra parameter → make sure to then include a background control bin in the set of measurement bins

Evaluation of the covariance matrix

- Matrix inversion methods (with or without Tikhonov regularisation): covariance matrix is calculated analytically

- Iterative methods: non-linear, covariance matrix calculation in general has to be done by other means

- Replica method
  - Apply statistical fluctuations on the data histogram

    → N replicas of the data
  - Repeat the unfolding for each replica
  - Covariance is estimated from RMS of the results

- Bootstrap method:

  similar idea, but based on events

  → test complete analysis chain

Iterative method: 0th iteration

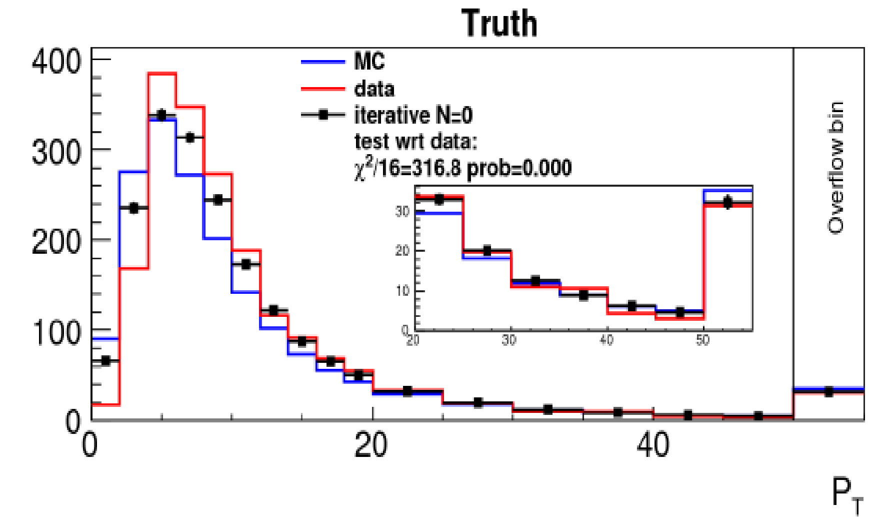
$$x_j^{(N+1)} = x_j^{(N)} \sum_i \frac{A_{ij}}{\epsilon_j} \frac{y_i}{\sum_k A_{ik} x_k^{(N)} + b_i}$$

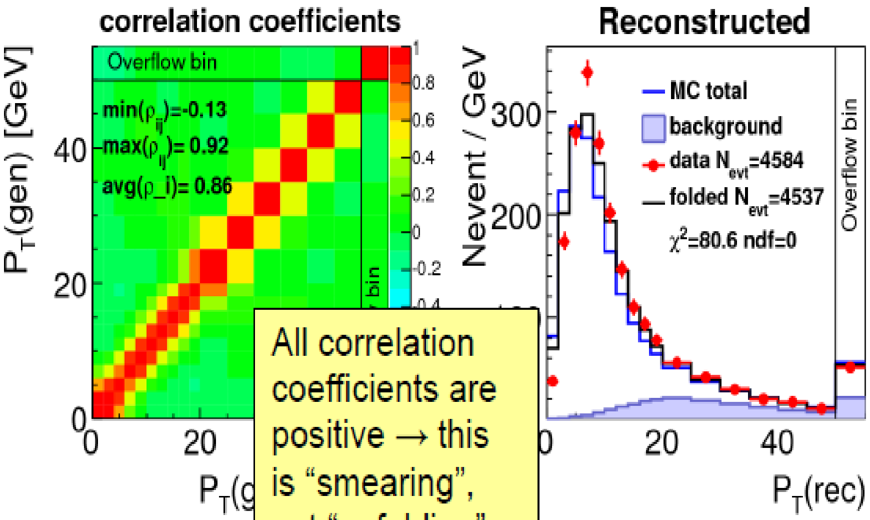
efficiency: $\epsilon_j = \sum_i A_{ij}$

start values $x_j^{(-1)}$ set to MC truth

- 0th iteration: “Bayesian unfolding” from 1995 D'Agostini paper
- Result “looks nice”, very small uncertianties, but fails all tests

→ the method has to be iterated





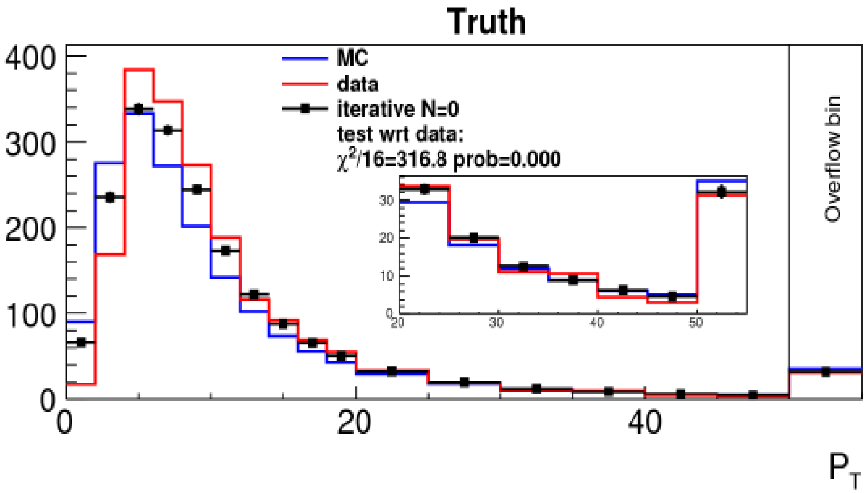
All correlation coefficients are positive → this is “smearing”, not “unfolding”

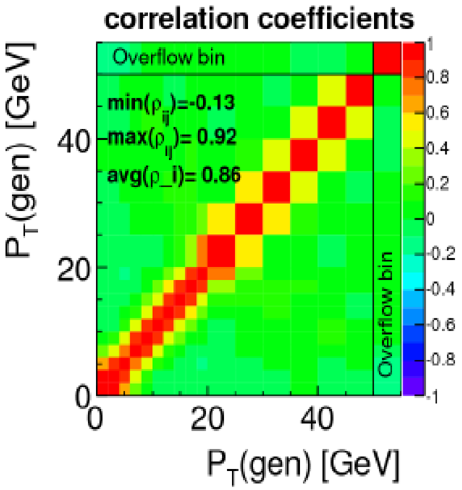
Iterative method: 1st iteration

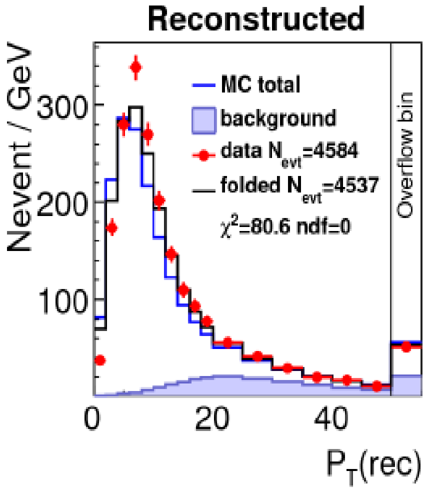
$$x_j^{(N+1)} = x_j^{(N)} \sum_i \frac{A_{ij}}{\epsilon_j} \frac{y_i}{\sum_k A_{ik} x_k^{(N)} + b_i}$$

- Convergence rate is expected to grow quadratically with the number of bins [Mülthei/Schorr 1987]
- Look at 1st iteration
  - Neighboring bins have positive correlation (expect: negative)
  - Shape not described
  - Folded-back different from data

    → have to iterate further



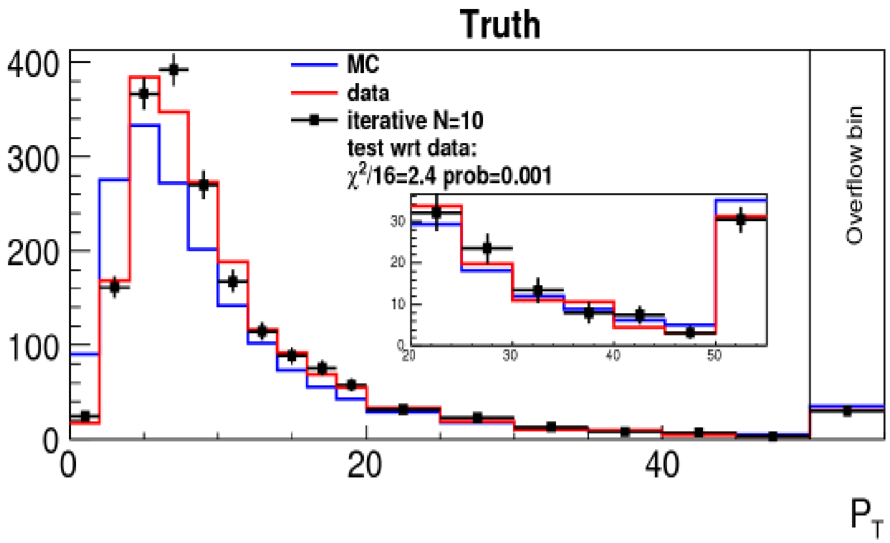


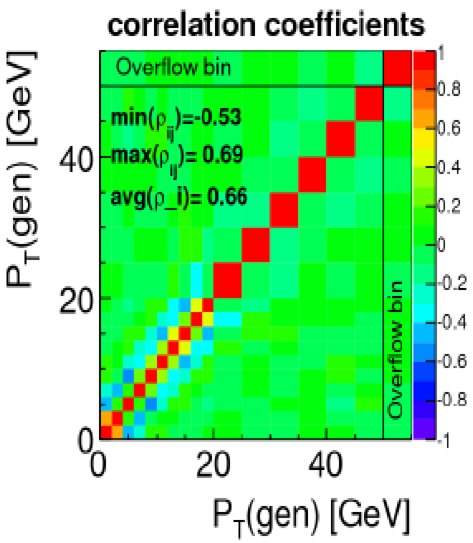


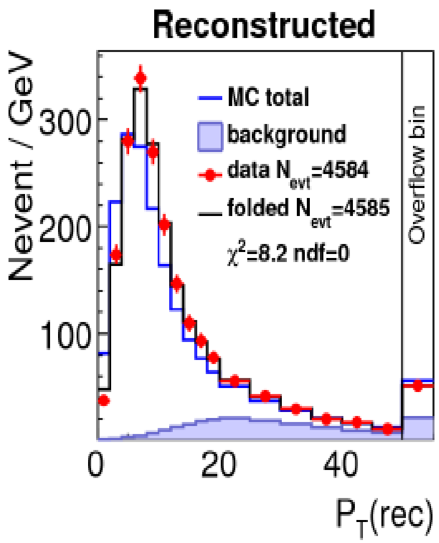
Iterative method: 10th iteration

$$x_j^{(N+1)} = x_j^{(N)} \sum_i \frac{A_{ij}}{\epsilon_j} \frac{y_i}{\sum_k A_{ik} x_k^{(N)} + b_i}$$

- Convergence rate is expected to grow quadratically with the number of bins [Mülthei/Schorr 1987]
- Look at 10th iteration
  - Similar to Tikhonov with strong regularisation



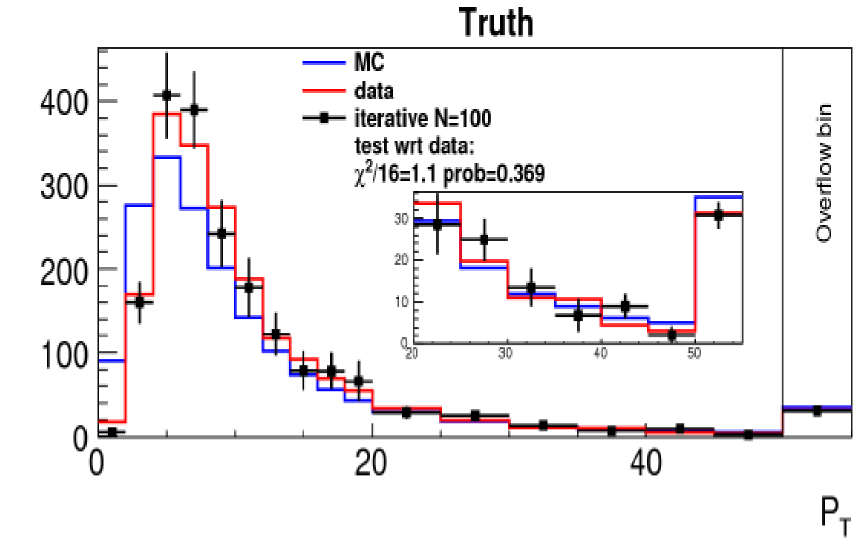


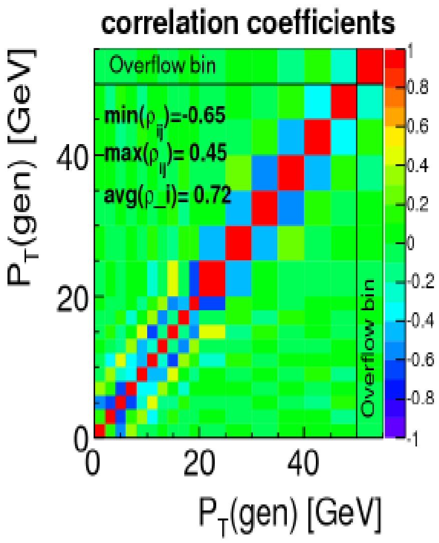


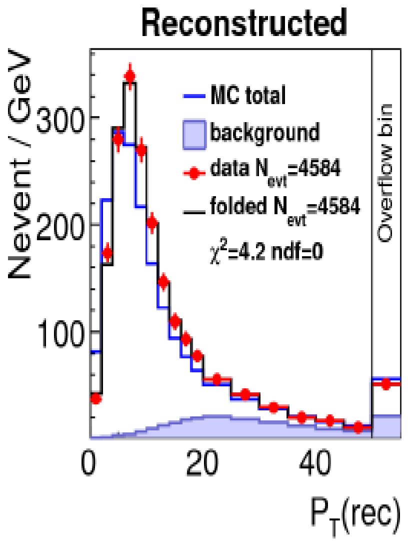
Iterative method: 100th iteration

$$x_j^{(N+1)} = x_j^{(N)} \sum_i \frac{A_{ij}}{\epsilon_j} \frac{y_i}{\sum_k A_{ik} x_k^{(N)} + b_i}$$

- Convergence rate is expected to grow quadratically with the number of bins [Mülthei/Schorr 1987]
- Look at 100th iteration
  - Similar to Tikhonov with weak regularisation





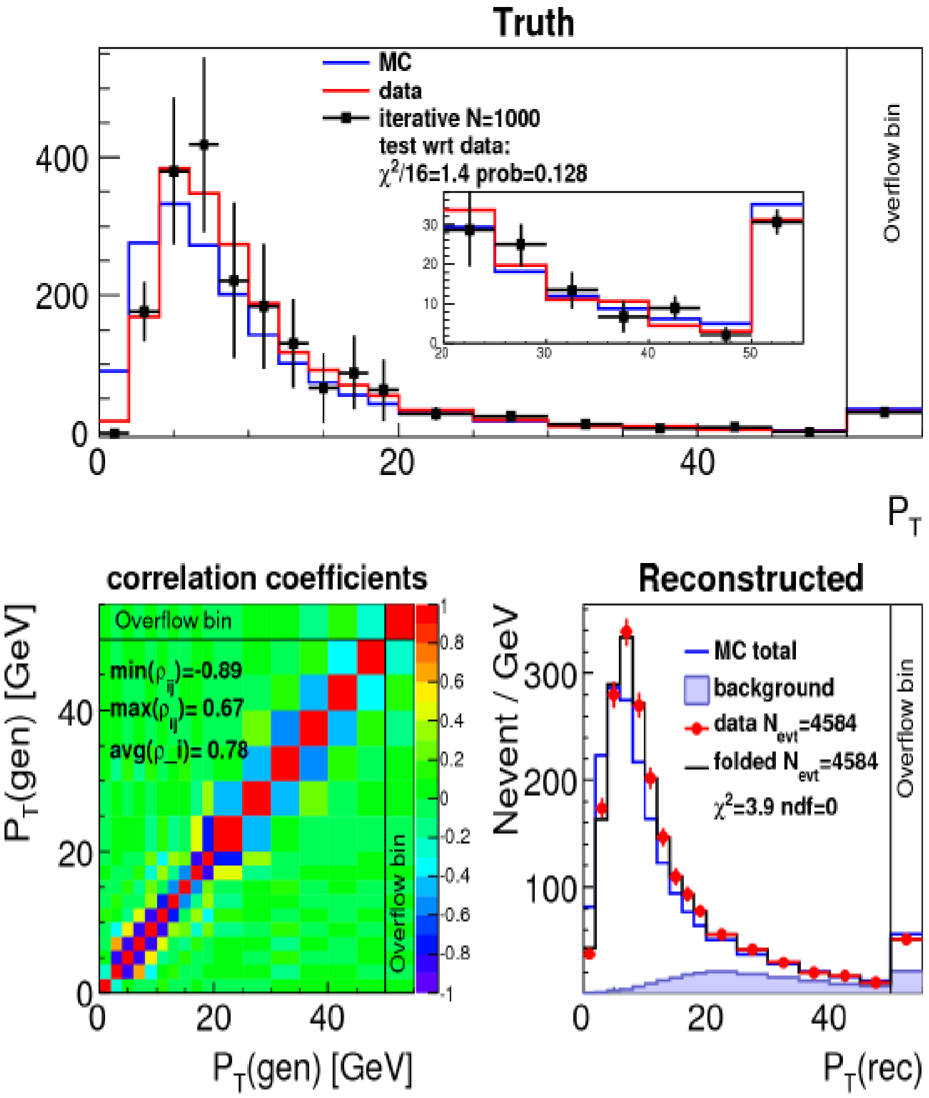


Iterative method: 1000th iteration

$$x_j^{(N+1)} = x_j^{(N)} \sum_i \frac{A_{ij}}{\epsilon_j} \frac{y_i}{\sum_k A_{ik} x_k^{(N)} + b_i}$$

- Convergence rate is expected to grow quadratically with the number of bins [Mülthei/Schorr 1987]

- Look at 1000th iteration
  - Similar to matrix inversion, but all guaranteed to be x≥0
  - Objective to choose number of iterations? Scan of correlation?

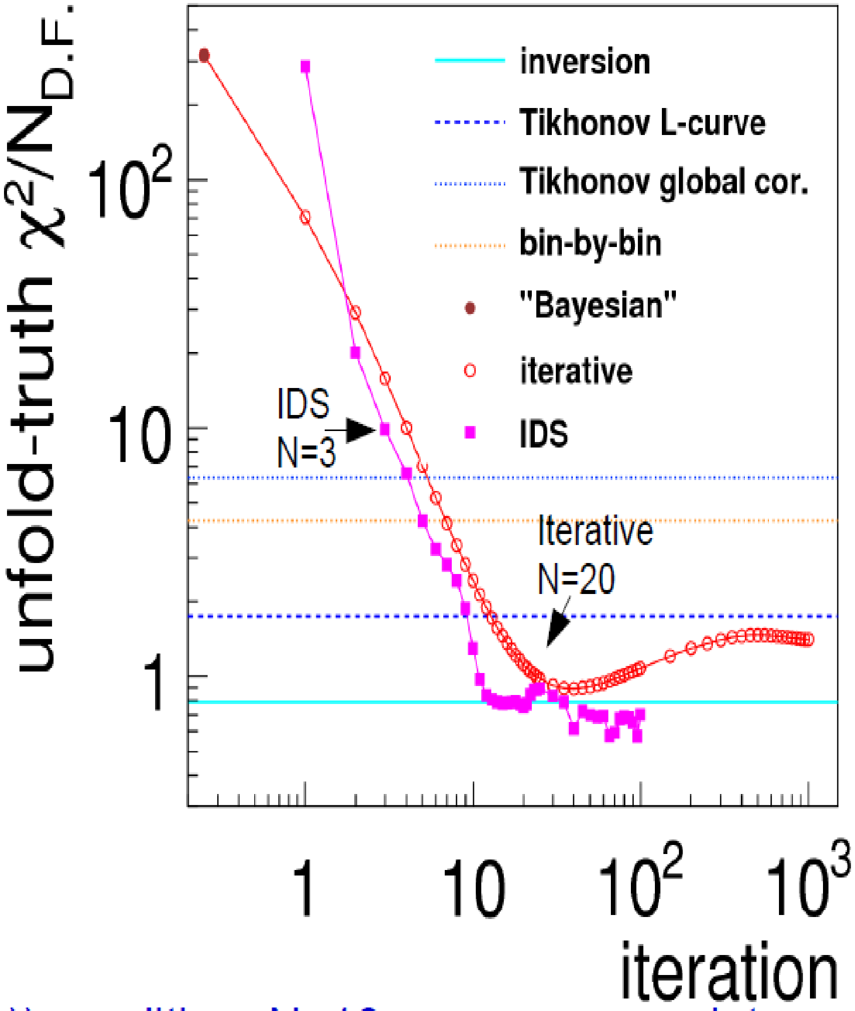


Comparison χ^2 vs. Data truth

- Test χ^2 of unfolded results against "data" truth
- For real analyses, such tests can be done by unfolding alternative truth models

Method	$X^2 / N_{D.F.}$
Tikhonov L-curve	1.75
Tikhonov min(avg(ρ_i))	6.30
bin-by-bin	4.24
iterative, N=20 min(avg(ρ_i))	1.12
IDS, N=3 min(avg(ρ_i))	9.88
IDS, N=11	0.97

- For the example studied, iterative+min(avg(ρ_i)) performs best



- IDS does not work with the min(avg(ρ_i)) condition, N>10 seems appropriate

Comparison vs. Data truth parameter

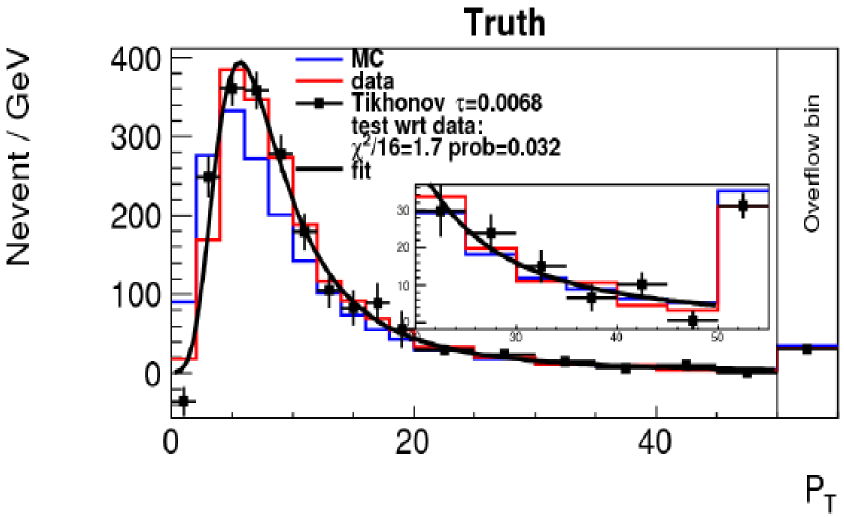
- Fit results by the analytic function use to generate the truth:

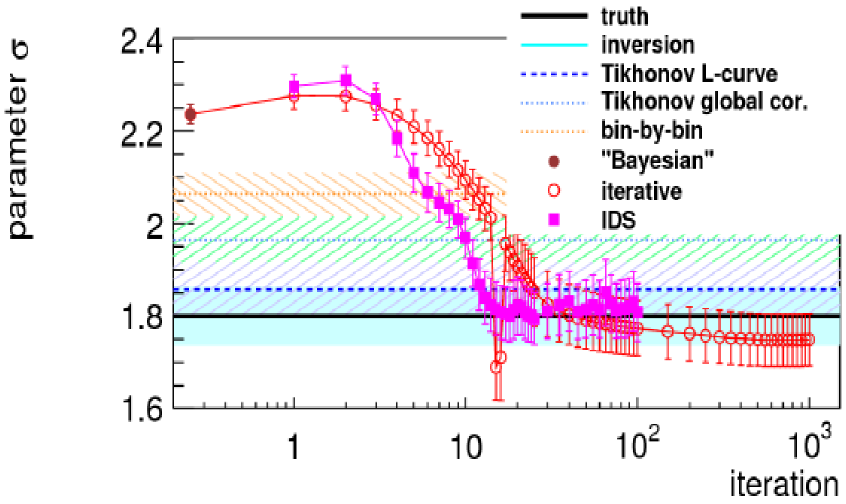
  Landau(μ,σ)

- Only the width σ is shown here (more difficult to fit)

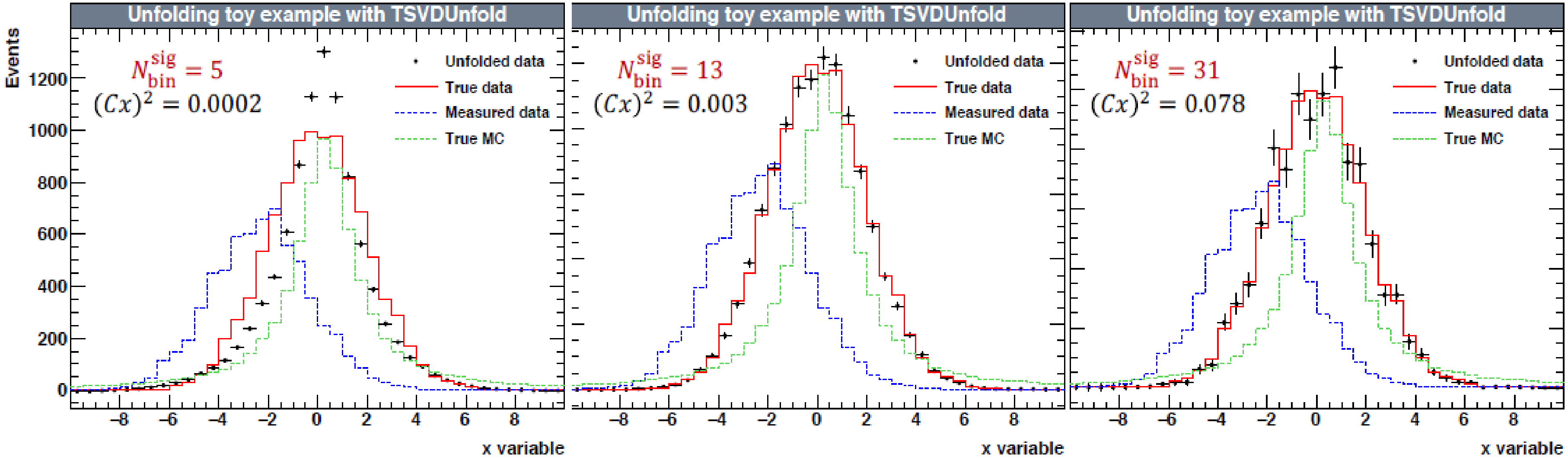
Method	fit of width σ
Tikhonov L-curve	1.858 ± 0.057
Tikhonov min(avg(ρ_i))	1.965 ± 0.049
bin-by-bin	2.064 ± 0.046
iterative, N=20 min(avg(ρ_i))	1.906 ± 0.071
IDS, N=3 min(avg(ρ_i))	2.268 ± 0.034
IDS, N=11	1.915 ± 0.050
truth	1.800

- For this test Tikhonov with L-curve is doing better than the iterative method





Over- and Under-Regularised Unfolding



Over-regularised

Best regularisation choice

Under-regularised

The parameter determines the strength of the regularization

- τ too small → oscillations
- τ too large → unfolded spectrum biased towards MC

Summary

- Unfolding: get measurements independent of the detector response
- Alternative: publish folding matrix with the result
- Many methods exist, only a few have been compared in this talk
- Big unfolding families investigated in this talk:
  - Matrix inversion +Tikhonov regularisation (parameter τ)
  - Iterative methods + truncation after N_{iter} steps
- Main question: how to choose the regularisation strength. Objectives studied in this talk: L-curve and scan of global correlation coefficients
- Tikhonov: L-curve scan is favored. Iterative: correlation scan seems to work
- Danger to obtain biased results if regularisation is too strong

RooUnfold package

- Provide a framework for different algorithms
  - Can compare performance directly, with common user code
    - RooUnfold takes care of different binning, normalisation, efficiency conventions
  - Can use common RooUnfold utilities
    - Write once, use for all algorithms
  - Currently implement or interface to iterative Bayes, SVD, TUnfold, unregularised matrix inversion, and bin-by-bin correction factors algorithms
- Simple OO design
  - "response matrix" object can be filled separately from training sample
    - in a different routine, or a different program (ROOT I/O support)
- Simple interface for the user
  - From program, ROOT/CINT script, or interactive ROOT prompt
  - Fill with histograms, vectors/matrices,… or direct methods:
    - `response->Fill(`x_{measured}, x_{true}`)` and `Miss(`x_{true}`)` methods takes care of normalisation
  - Results as a histogram with errors, or vector and covariance matrix

RooUnfold features

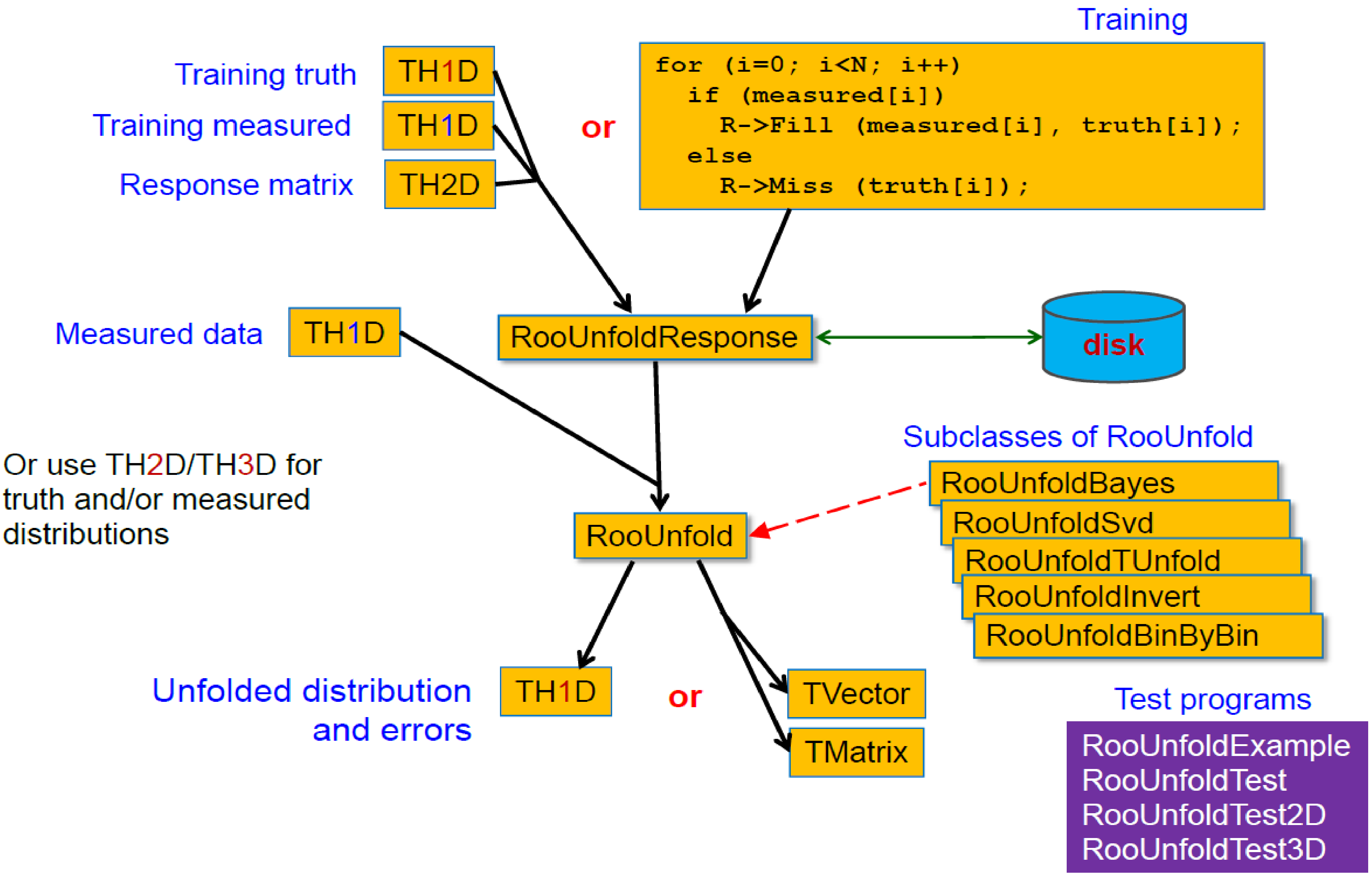
- Supports different binning scenarios
  - multi-dimensional distributions (1D, 2D, and 3D)
  - Different binning (or even dimensionality) for measured and truth
  - Option to include or exclude histogram under/overflow bins in the unfolding
- Supports different methods for error computation (simple switch). In order of increasing CPU time:
  - No error calculation (uses √N)
  - bin-by-bin errors (no correlations)
  - full covariance matrix from the propagation of measurement errors in the unfolding, or
  - covariance matrix from MC toys
    - useful to test error propagation and when it is inaccurate
- These details are handled by the framework, so don’t need to be implemented for each algorithm

RooUnfold testing

- Calculates resolutions, pulls, and χ^2
- Includes a toy MC test framework, allowing selection of different
  - PDFs and PDF parameters
  - binning
  - 1D, 2D, 3D tests
  - unfolding methods and parameters
  - Test procedures for the regularisation parameter and errors

  and plotting results from a single command

RooUnfold classes



Training

Training truth TH1D

Training measured TH1D

Response matrix TH2D

or

```
for (i=0; i<N; i++)
  if (measured[i])
    R->Fill (measured[i], truth[i]);
  else
    R->Miss (truth[i]);
```

Measured data TH1D

RooUnfoldResponse

disk

Or use TH2D/TH3D for truth and/or measured distributions

Subclasses of RooUnfold

RooUnfoldBayes

RooUnfoldSvd

RooUnfoldTUnfold

RooUnfoldInvert

RooUnfoldBinByBin

RooUnfold

Unfolded distribution and errors TH1D

or

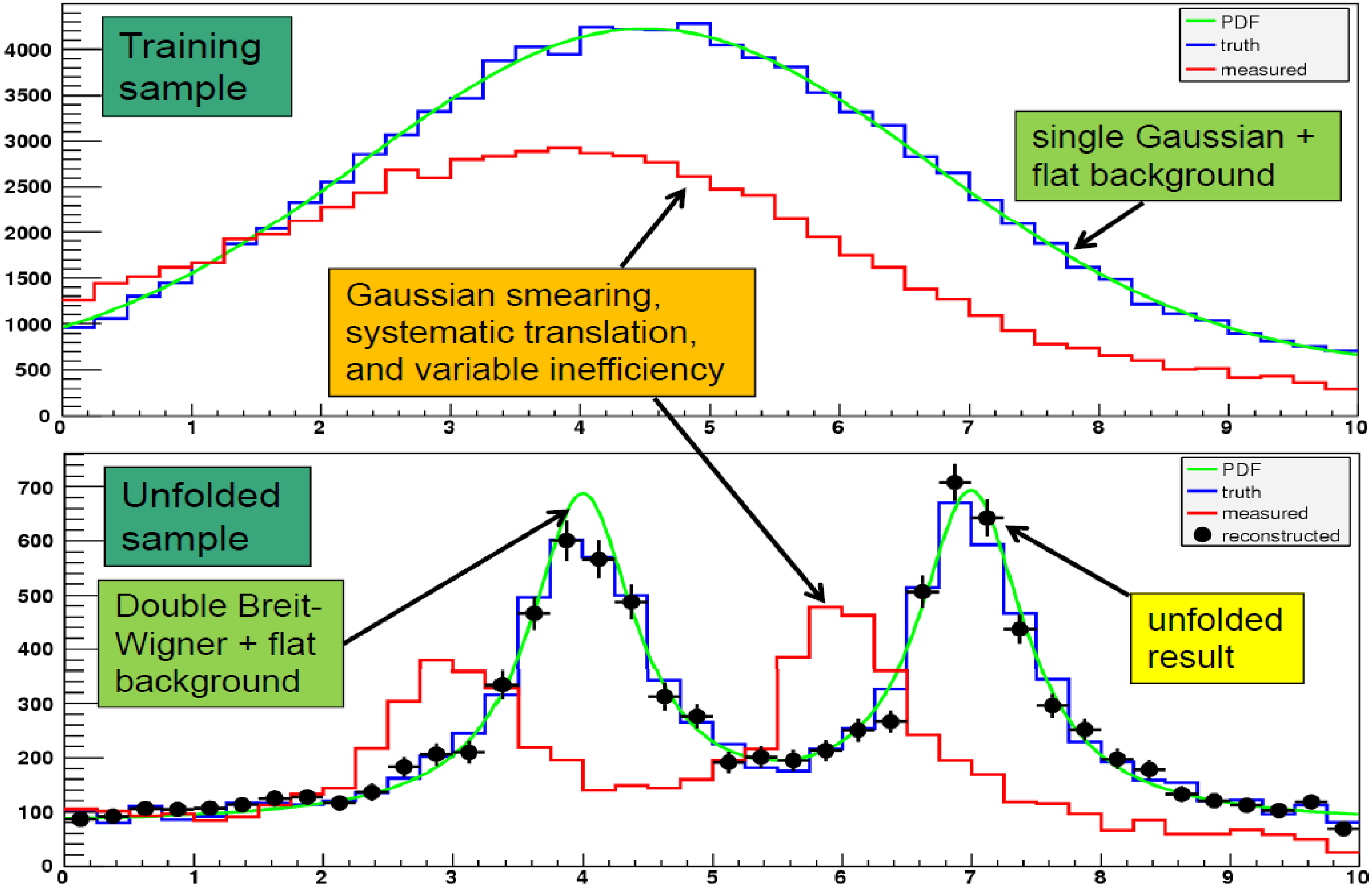
TVector

TMatrix

Test programs

RooUnfoldExample
RooUnfoldTest
RooUnfoldTest2D
RooUnfoldTest3D

RooUnfold example (Bayes)



Training sample

single Gaussian + flat background

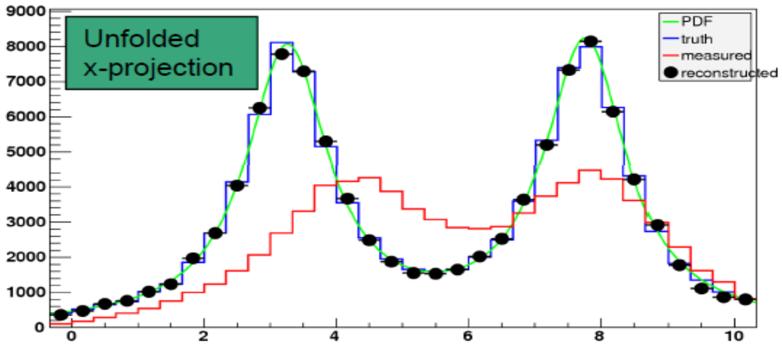
Gaussian smearing, systematic translation, and variable inefficiency

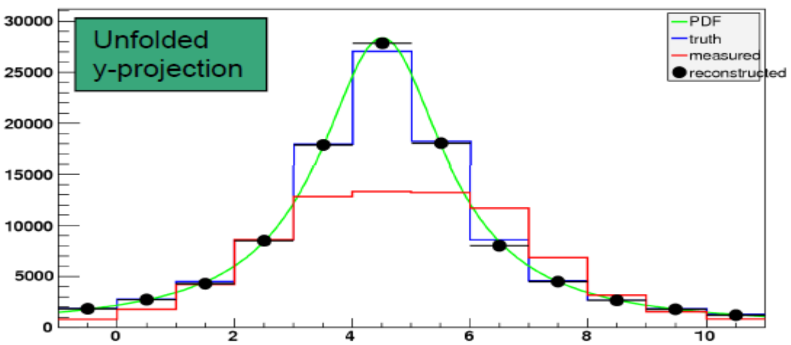
Unfolded sample

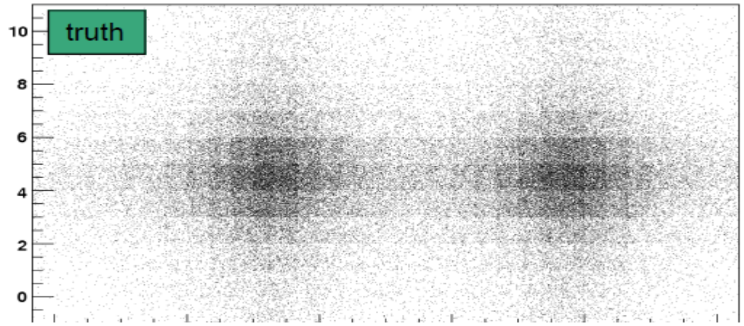
Double Breit-Wigner + flat background

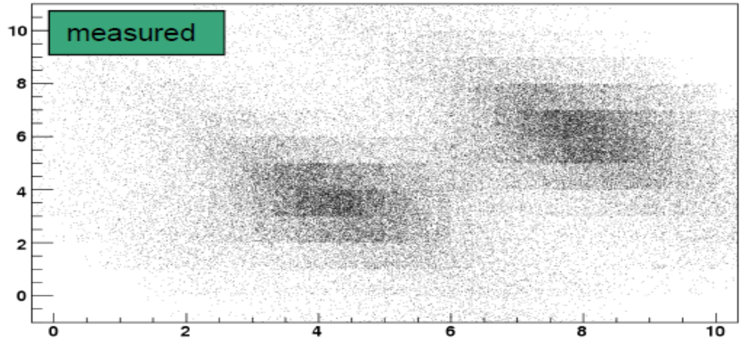
unfolded result

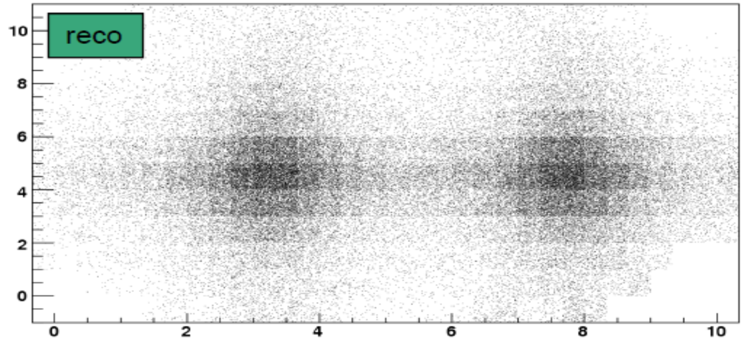
RooUnfold example (Bayes)







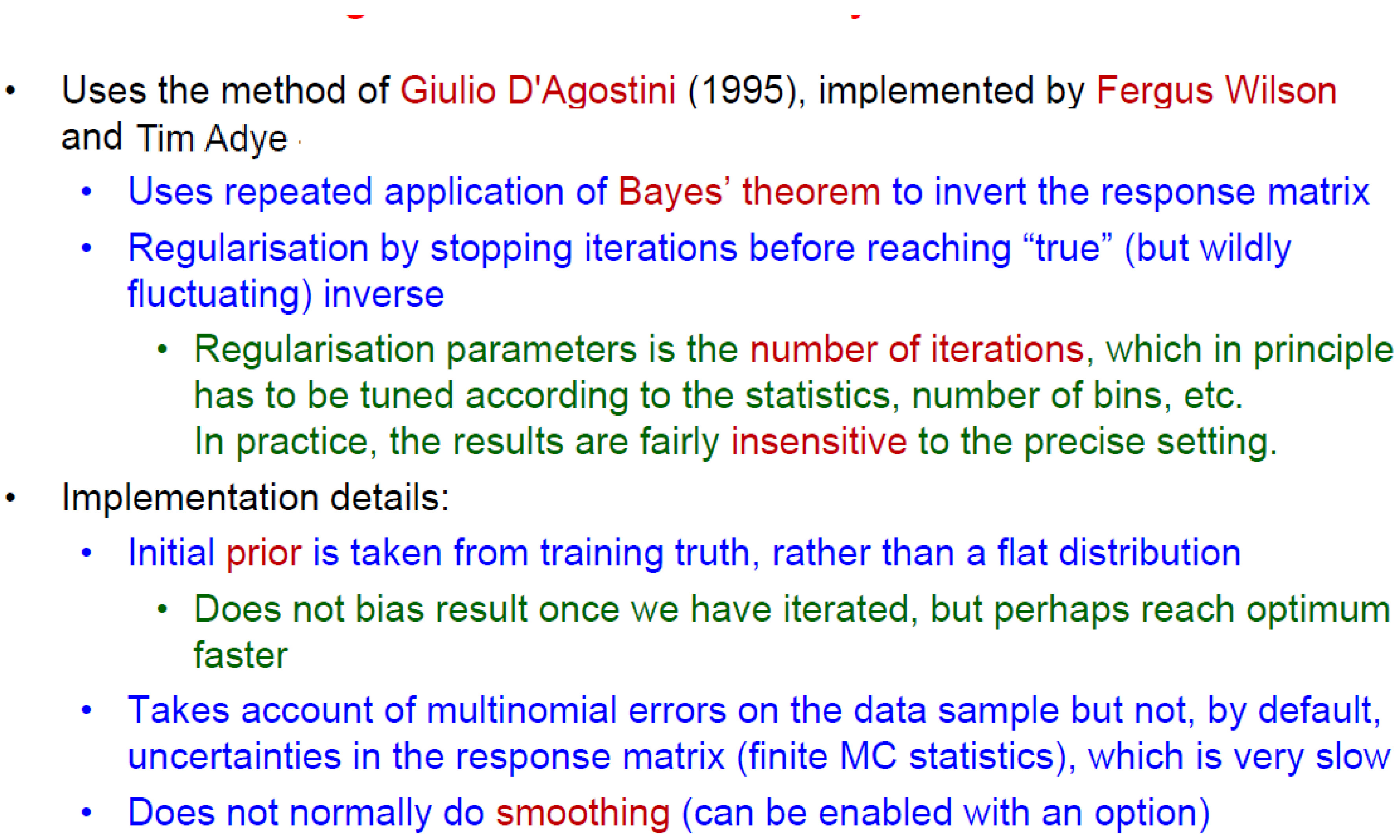




2D unfolding

2D Smearing, bias, variable efficiency, and variable rotation

RooUnfold algorithms: Iterative Bayes



- Uses the method of Giulio D'Agostini (1995), implemented by Fergus Wilson and Tim Adye
  - Uses repeated application of Bayes' theorem to invert the response matrix
  - Regularisation by stopping iterations before reaching "true" (but wildly fluctuating) inverse
    - Regularisation parameters is the number of iterations, which in principle has to be tuned according to the statistics, number of bins, etc.
      In practice, the results are fairly insensitive to the precise setting.
- Implementation details:
  - Initial prior is taken from training truth, rather than a flat distribution
    - Does not bias result once we have iterated, but perhaps reach optimum faster
  - Takes account of multinomial errors on the data sample but not, by default, uncertainties in the response matrix (finite MC statistics), which is very slow
  - Does not normally do smoothing (can be enabled with an option)

RooUnfold algorithms: SVD

- Uses the method of Andreas Höcker and Vato Kartvelishvili
- Obtains inverse of response matrix using singular value decomposition
  - Use number-of-events matrix to keep track of MC uncertainties
- Regularisation with a smooth cut-off on small singular value contributions (these correspond to high-frequency fluctuations)
  - Replace $s_i^2 \rightarrow s_i^2 / (s_i^2 + s_k^2)$
  - k determines the relative contributions of MC truth and data
    - k too small → result dominated by MC truth
    - k too large → result dominated by statistical fluctuations
  - k needs to be tuned for the particular type of distribution, number of bins, and approximate sample size
- Unfolded error matrix includes effect of finite MC training statistics (usually small)

RooUnfold algorithms: TUnfold

- Uses the TUnfold method implemented by Stefan Schmitt and included in ROOT
  - RooUnfold includes an interface to this class

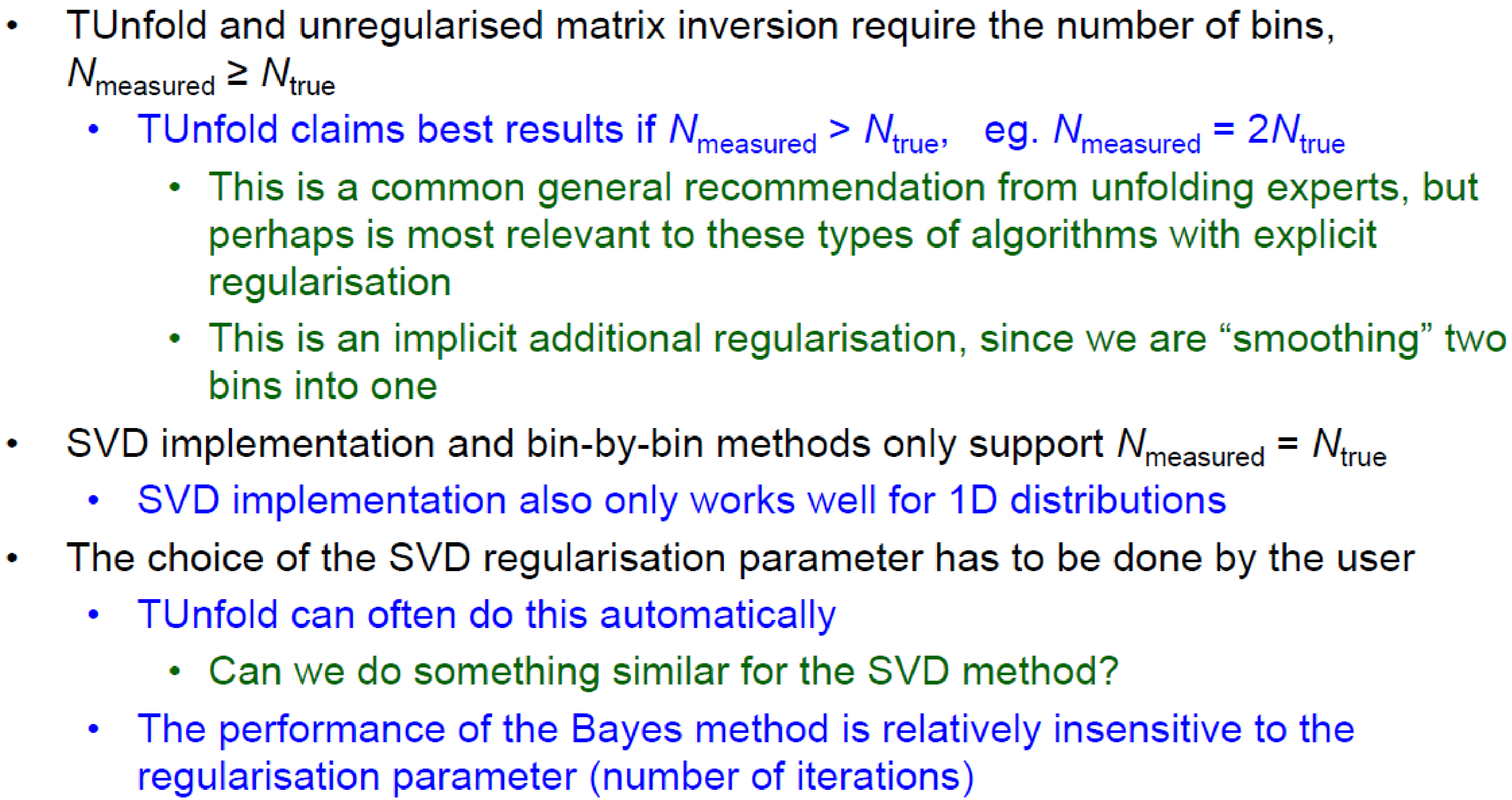
- Performs a matrix inversion with 0-, 1-, or 2-order polynomial regularisation of neighbouring bins
  - RooUnfold automatically takes care of packing 2D and 3D distributions and creating the appropriate regularisation matrix required by TUnfold
- TUnfold can determine an optimal regularisation parameter (τ) by scanning the "L-curve" of $\log_{10}(\chi^2)$ vs $\log_{10}(\tau)$.

RooUnfold algorithms: Unregularised

- Very simple algorithms
  - using bin-by-bin correction factors, with no inter-bin migration
  - using unregularised matrix inversion with singular value removal (TDecompSVD)

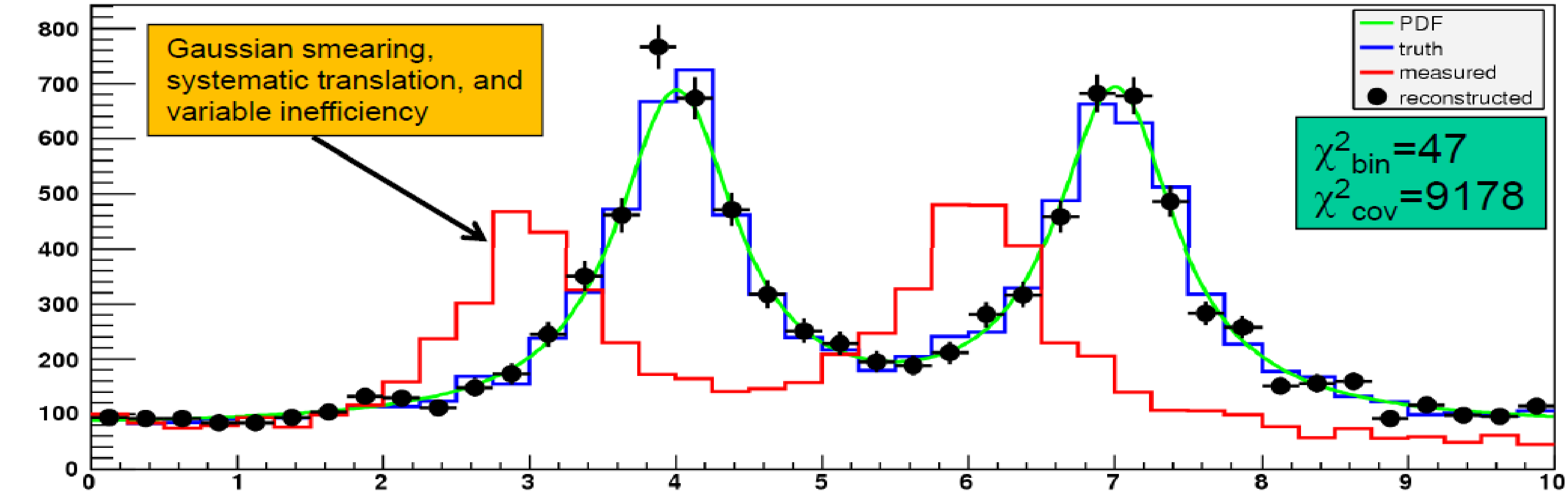
  are included for comparison – and to demonstrate why they should not be used in most cases!

RooUnfold algorithms: comparison



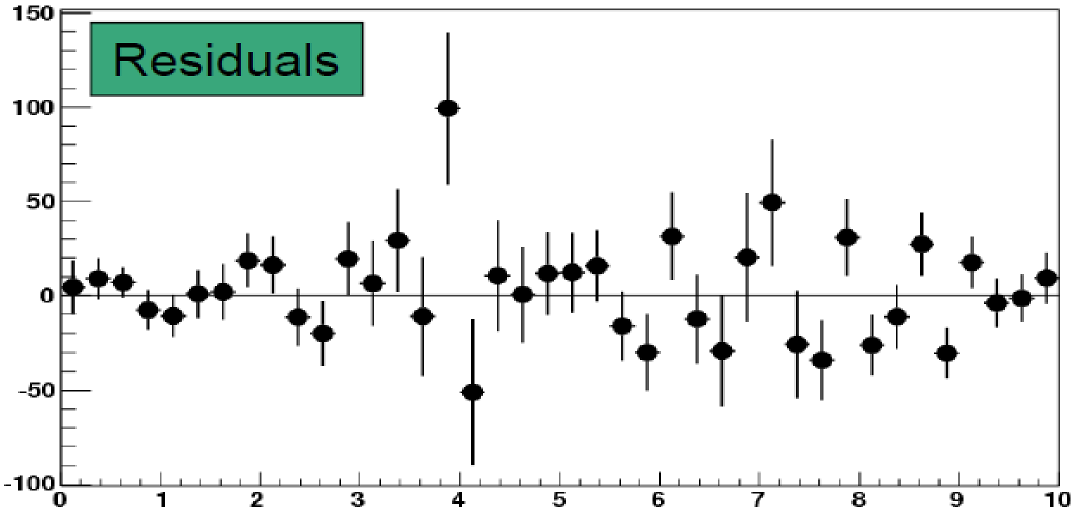
- TUnfold and unregularised matrix inversion require the number of bins, $N_{\text{measured}} \geq N_{\text{true}}$
  - TUnfold claims best results if $N_{\text{measured}} > N_{\text{true}}$, eg. $N_{\text{measured}} = 2N_{\text{true}}$
    - This is a common general recommendation from unfolding experts, but perhaps is most relevant to these types of algorithms with explicit regularisation
    - This is an implicit additional regularisation, since we are "smoothing" two bins into one
- SVD implementation and bin-by-bin methods only support $N_{\text{measured}} = N_{\text{true}}$
  - SVD implementation also only works well for 1D distributions
- The choice of the SVD regularisation parameter has to be done by the user
  - TUnfold can often do this automatically
    - Can we do something similar for the SVD method?
  - The performance of the Bayes method is relatively insensitive to the regularisation parameter (number of iterations)

RooUnfold with Bayes algorithm (3 iterations)

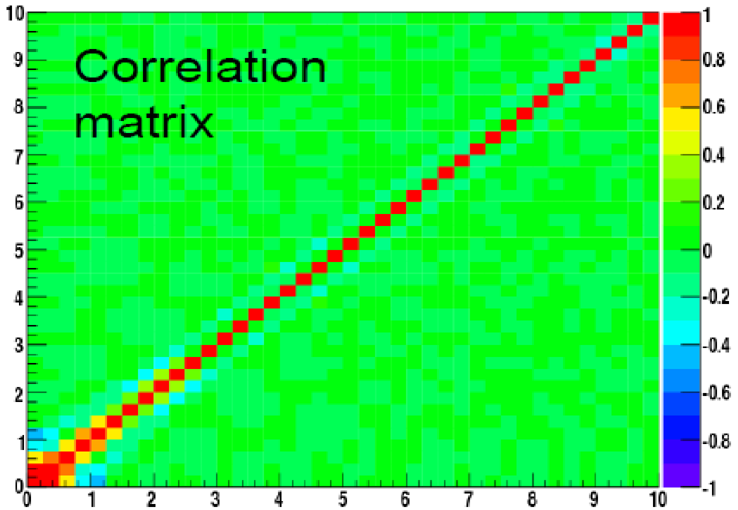


Gaussian smearing, systematic translation, and variable inefficiency

χ^2_{bin}=47
χ^2_{cov}=9178

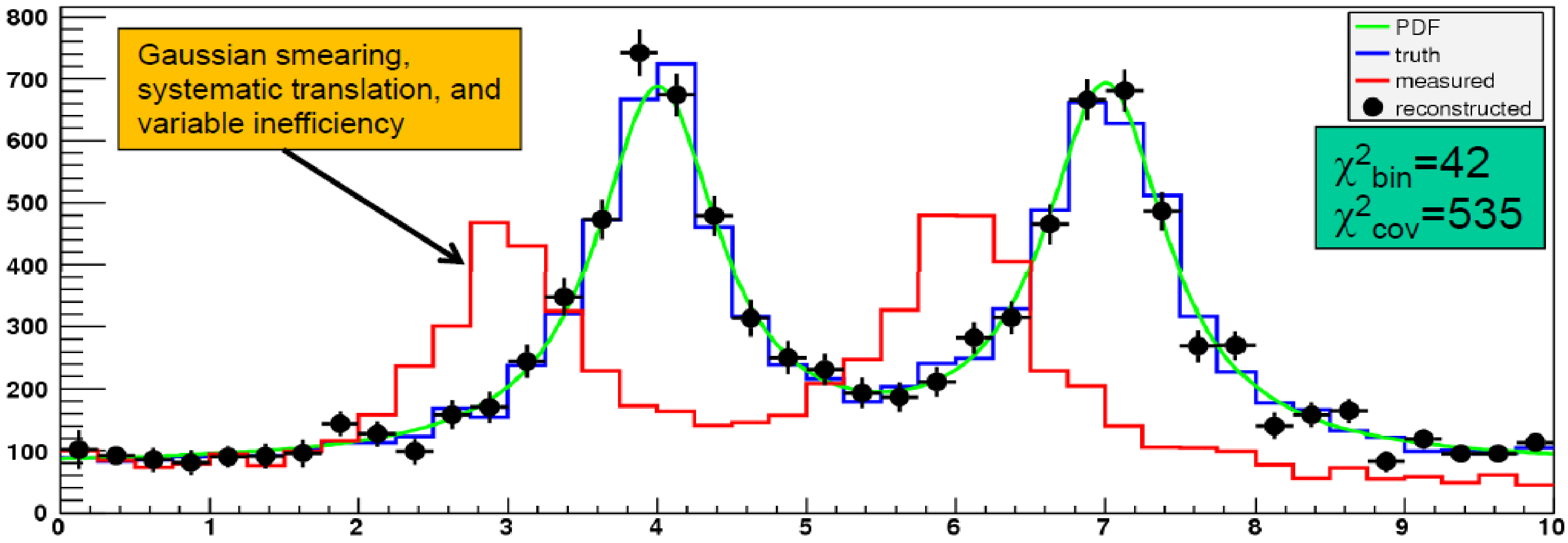


Residuals



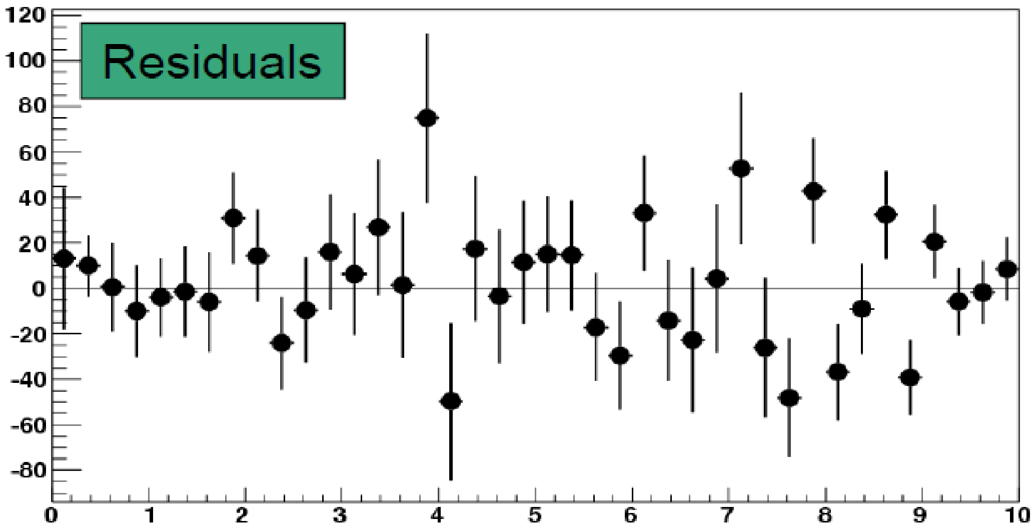
Correlation matrix

RooUnfold with SVD algorithm (k=30)

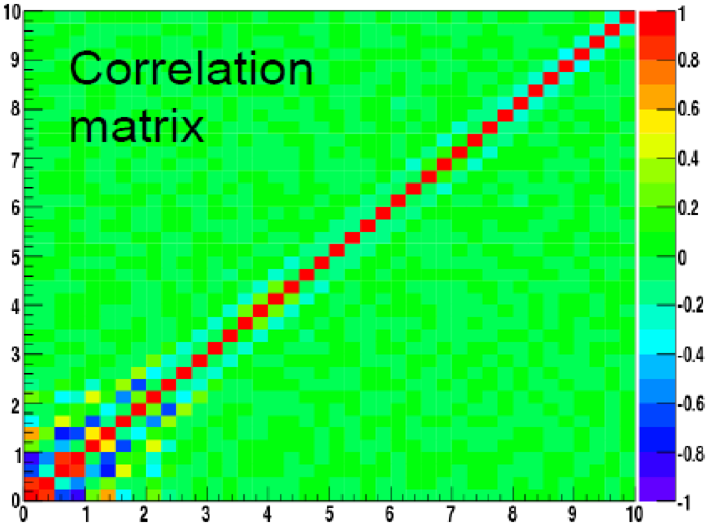


Gaussian smearing, systematic translation, and variable inefficiency

χ^2_{bin}=42
χ^2_{cov}=535

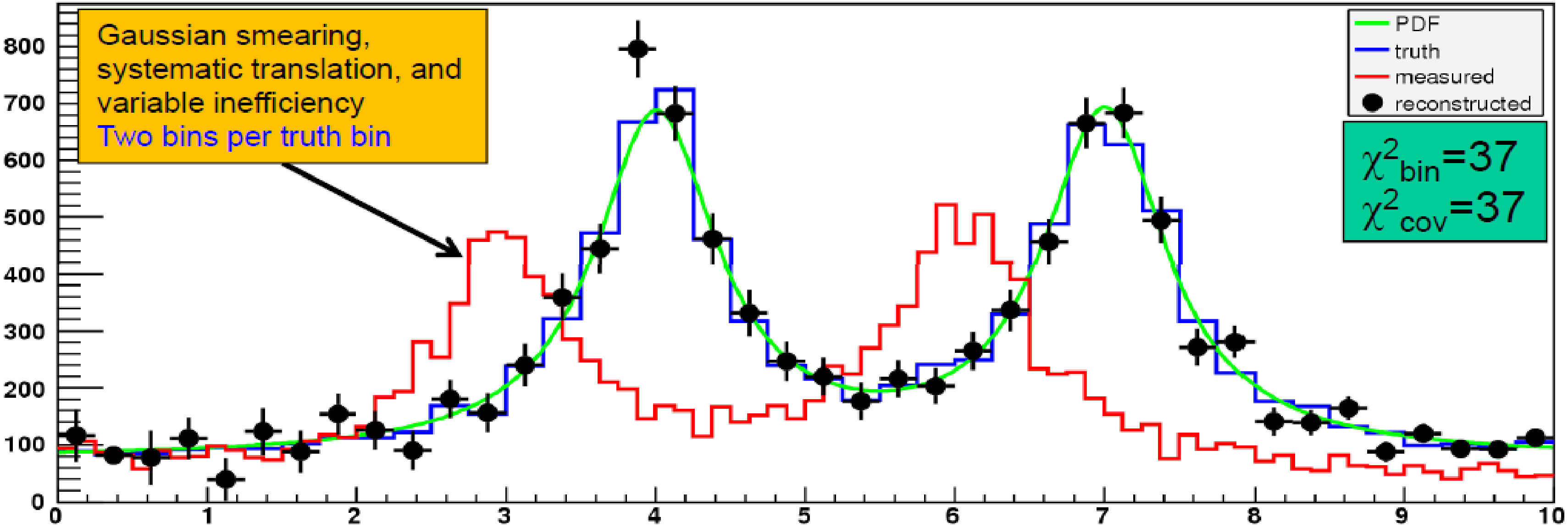


Residuals



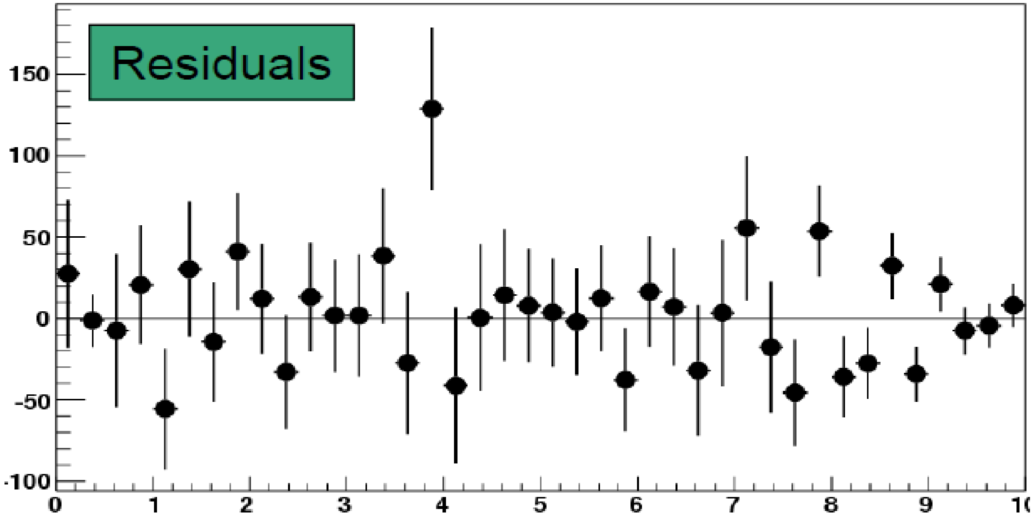
Correlation matrix

RooUnfold with TUnfold algorithm (τ=0.004)

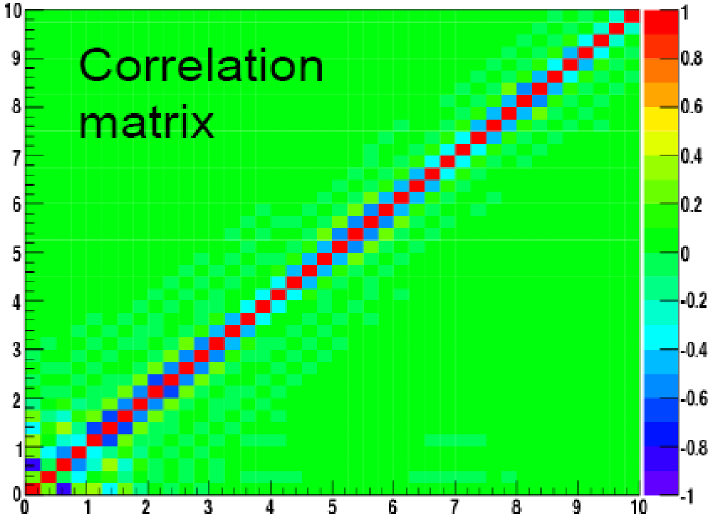


Gaussian smearing, systematic translation, and variable inefficiency

Two bins per truth bin

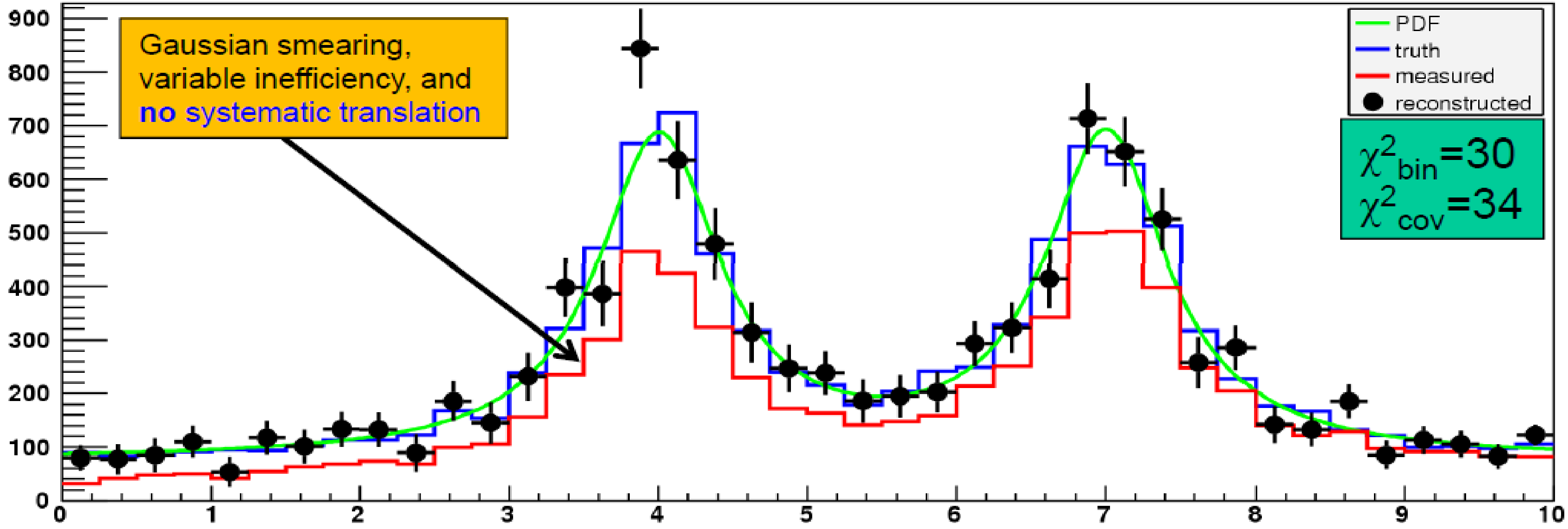


Residuals



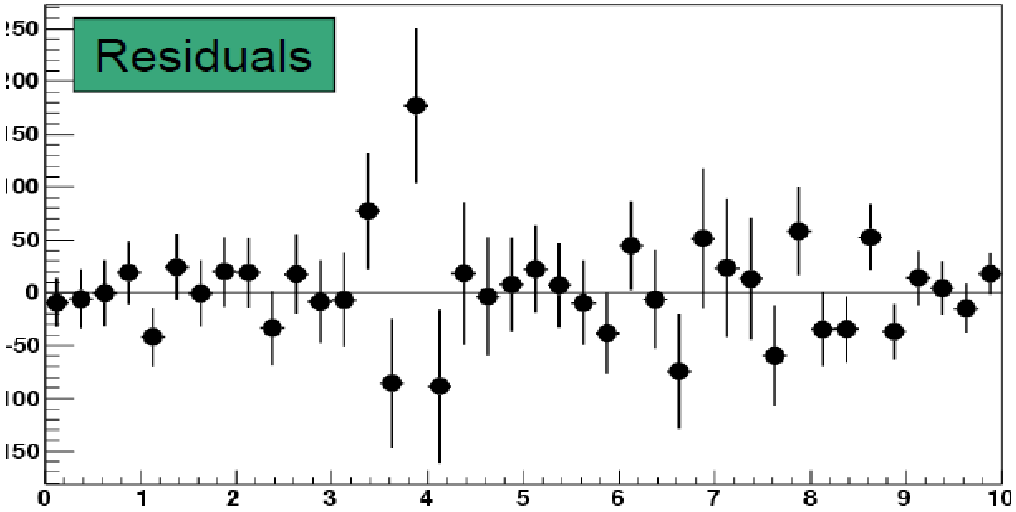
Correlation matrix

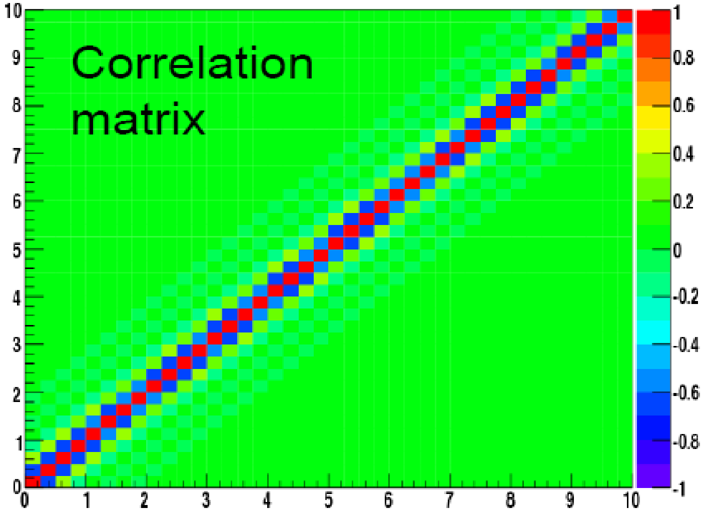
Unregularised matrix inversion



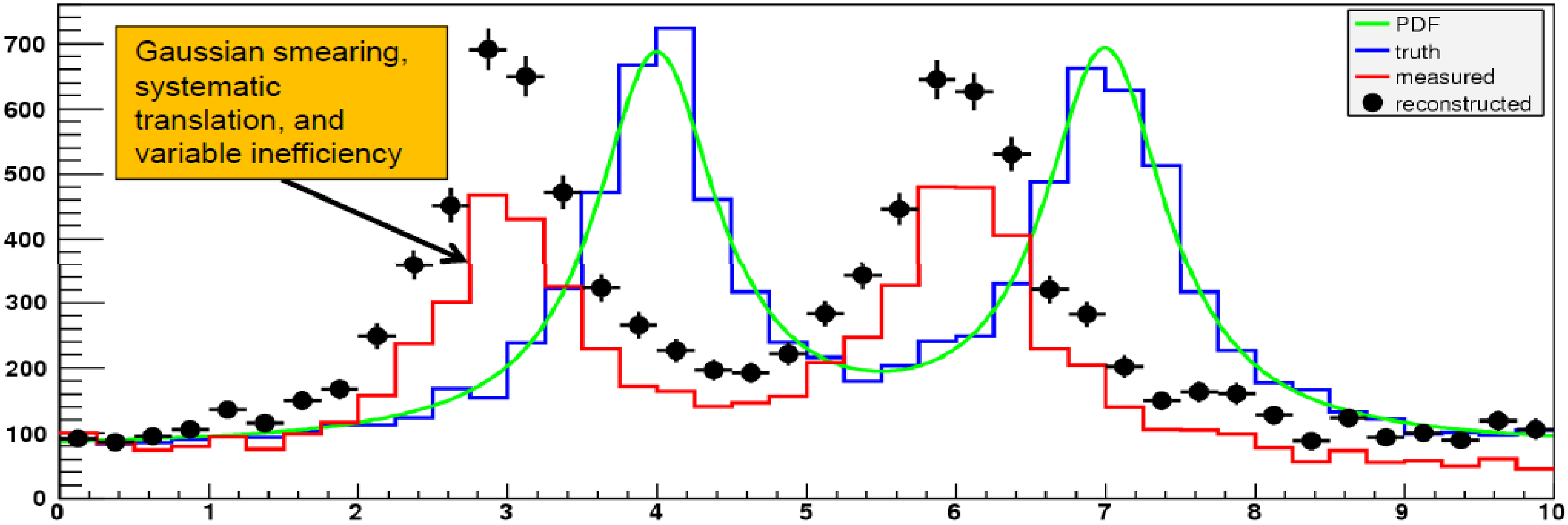
Gaussian smearing, variable inefficiency, and **no systematic translation**

χ^2_{bin}=30
χ^2_{cov}=34

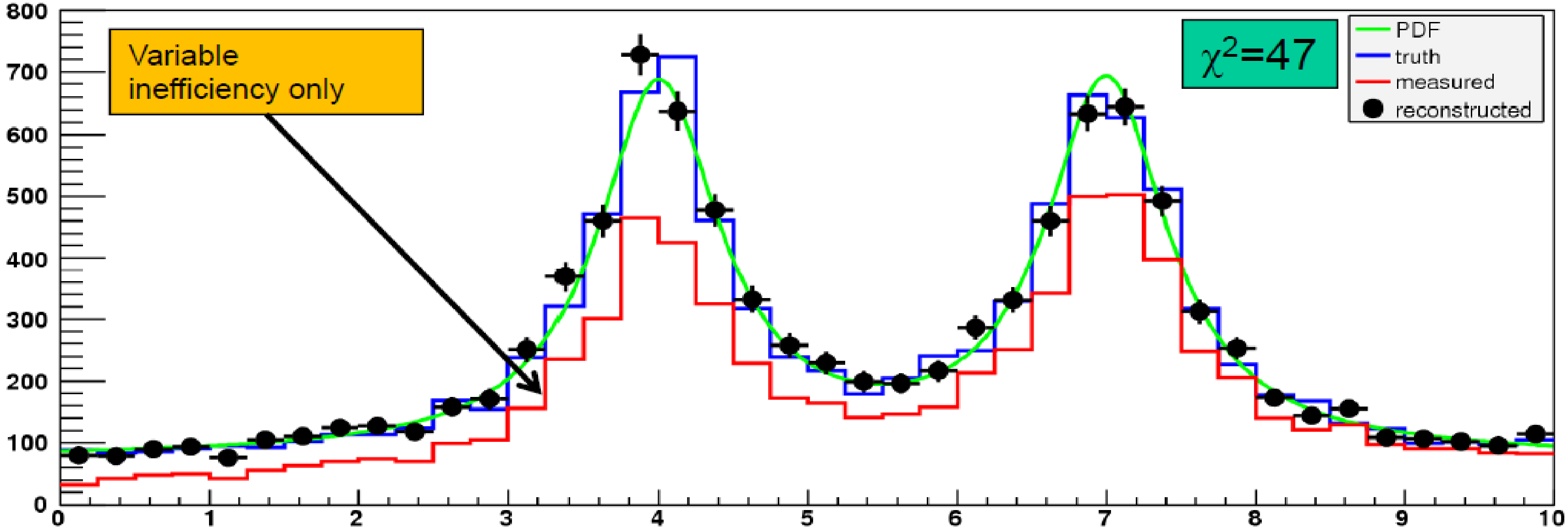




Simple correction factors



Gaussian smearing, systematic translation, and variable inefficiency

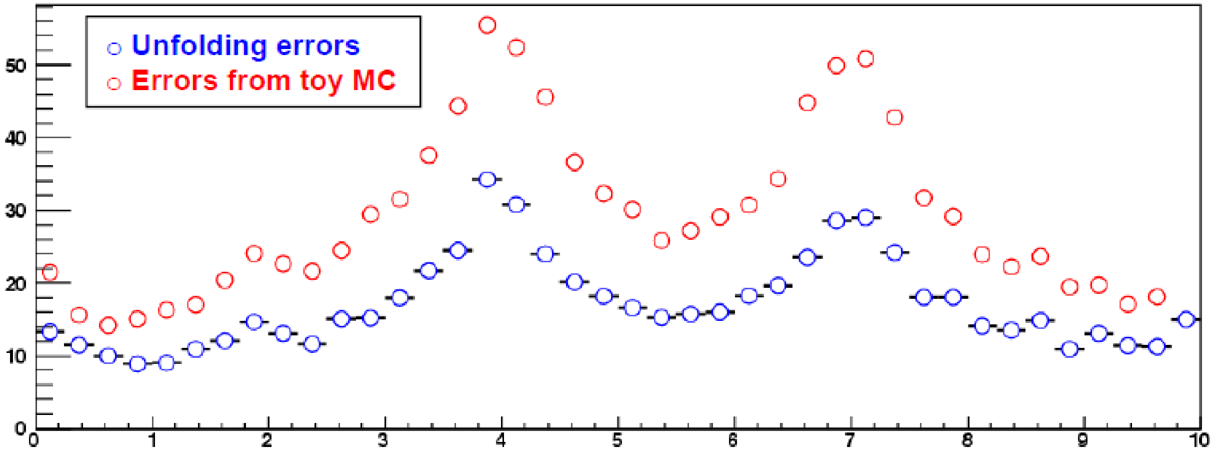


Variable inefficiency only

$\chi^2=47$

Unfolding errors

- All methods return a full covariance matrix of the errors on the unfolded histogram due to uncertainties on the measured distribution.
    - This is often calculated by propagation of errors
        - but not always possible if there are non-linearities or other problems, eg. the iterations in the Bayes method are not handled in D'Agostini's formalism:



- RooUnfold allows the covariance matrix to be calculated from toy MC instead
    - provides a cross-check of the error propagation or replace it if there are problems

Bin-to-bin correlations

- Regularisation introduces inevitable correlations between bins in the unfolded distribution
  - To calculate a correct χ^2, one has to invert the covariance matrix:
    $\chi^2 = (x_m - x_t)^T \mathbf{V}^{-1} (x_m - x_t)$
- However, in many cases, the covariance matrix is poorly conditioned, which makes calculating the inverse problematic
  - Inverting a poorly conditioned matrix involves subtracting large, but very similar numbers, leading to significant effects due to the machine precision

- In any case, χ^2 may not be the best figure of merit
  - could improve χ^2 by relaxing regularisation → larger errors, but also larger residuals
  - Is there a better figure of merit?

Which Method To Choose?

There is no "best" method. Depends on the analysis.

Main questions:

How to choose regularization parameters?

After how many iterations to stop in the iterative Bayesian unfolding?

Danger: Regularization and early stopping in iterative unfolding introduce a bias

Don't forget:
it some cases it is most useful to publish folding matrix with the result

References

- Bayesian:
  - Nucl.Instrum.Meth. A362 (1995) 487-498
  - arXiv:1010.0632
- TSVDUnfold
  - Nucl.Instrum.Meth.A372 (1996) 469-481
- TUnfold
  - JINST 7 (2012) T10003 [arXiv:1205.6201]