Unfolding algorithms and RooUnfold

- **Unfolding algorithms:**
  - **Extracted from slides by T. Adye at PHYSTAT 2016 and K. Reygers lectures at Heilderbeg Univ.**

Deconvolution

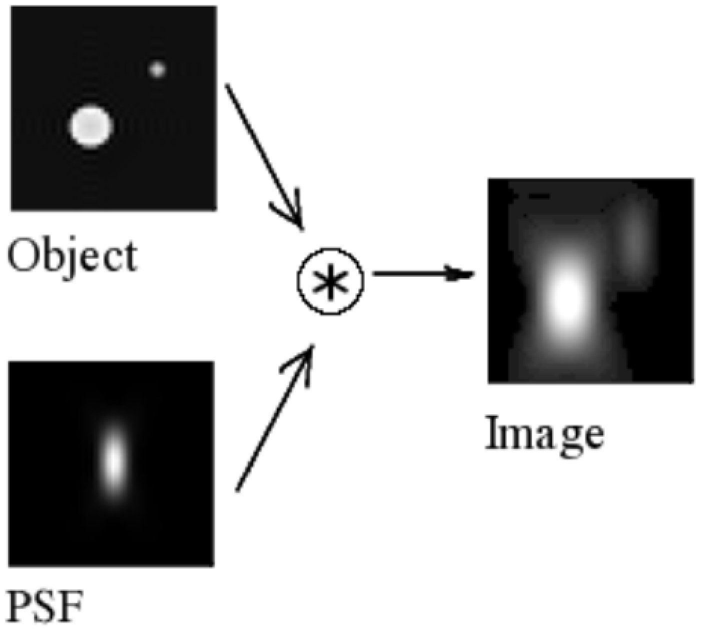
Finite resolution of the detector smears the quantities we're interested in.

Goal:
smeared information
→ original information

This is called *deconvolution* or *unfolding*

"Inverse problem"

Problem can be ill-posed in the sense that unfolded result can be very sensitive to small perturbations in the data



Example:
Smearing of a telescope image

https://en.wikipedia.org/wiki/Point_spread_function

To Unfold or not?

It's a lot of work, and often produces biased or otherwise unsatisfactory results. Moreover it's often unnecessary.

"Forward fitting" is much easier

- Take theory prediction
- Convolve it with the response of the detector
- Compare smeared theory directly with the data

When Unfolding Make Sense

1. Results from experiment A and B with different response function are to be compared

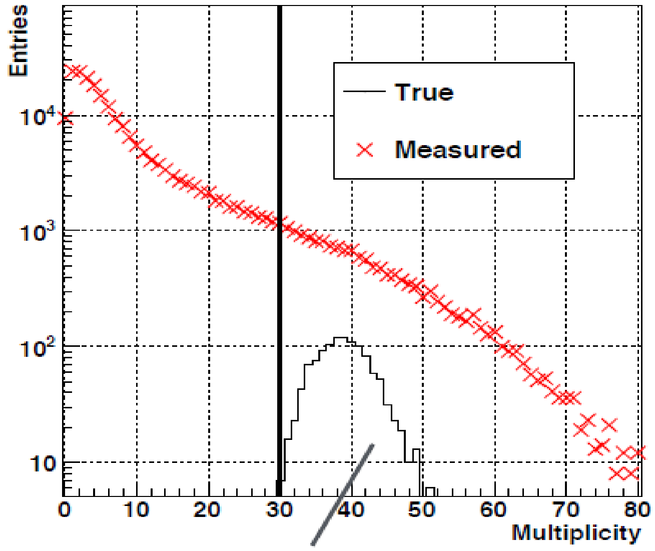
2. It is too complicated to publish the response function of the detector along with the data
   - Detector response might be very complex, e.g., time dependent
   - Sometimes computer code reflecting the response would have to be published
   - Danger that future users don't use the filter correctly

When Unfolding Make Sense

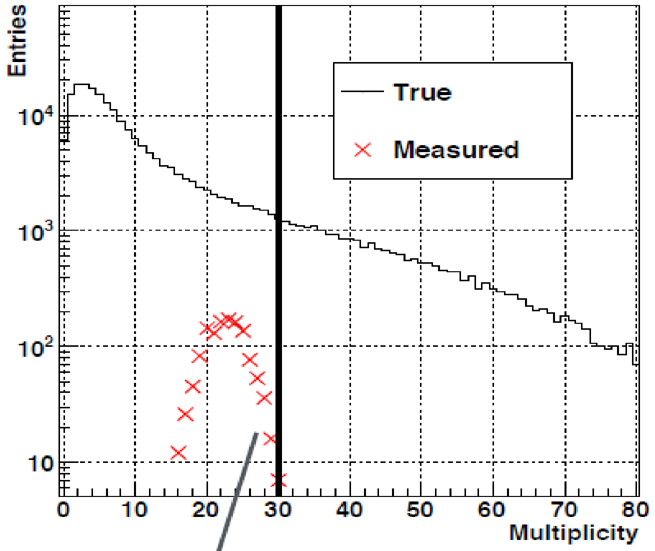
- Multiplicity distributions $P(N_{\mathrm{ch}})$
  - Measured multiplicity differs from true charged particle multiplicity due to detector effects (efficiency, fake hits, …)
- p_T spectra, e.g., π^0 spectrum measured with a calorimeter
  - finite energy resolution and shower overlaps in a calorimeter affect the p_T of the reconstructed shower

Example: multiplicity distributions in pp collisions

arXiv:0912.0023



true N's contributing to measured $N = 30$



measured N's for a true $N = 30$

Response Matrix (I)

Suppose that we deal with continues variables (e.g., transverse momentum)

$f_t(x_t)$	:	distribution of true values (normalized to unity)
$f_m(x_m)$	:	distribution of measured values (normalized to unity)
$f_b(x_m)$	:	distribution of background (normalized to unity)

Response function *R*:

$$R(x_m|x_t) = \underbrace{r(x_m|x_t)}_{\text{"smearing"}} \times \underbrace{\varepsilon(x_t)}_{\text{"efficiency"}}$$

probability (density) to observe x_m given x_t

By construction, one has

$$\int_{\Omega_m} r(x_m|x_t)\,\mathrm{d}x_m = 1$$

Response Matrix (II)

Further definitions:

m_{tot}	:	number of true events
n_{tot}	:	number of measured events
b_{tot}	:	number of background events

$$\mu_{\text{tot}} = E[m_{\text{tot}}], \quad \nu_{\text{tot}} = E[n_{\text{tot}}], \quad \beta_{\text{tot}} = E[b_{\text{tot}}]$$

It is practical to work with discrete bins. E.g., probability to find true in bin *j*:

$$p_j = \int_{\text{bin } j} \mathrm{d}x_t \, f_t(x_t), \quad \mu_j = \mu_{\text{tot}} \times p_j$$

Ignoring backgrounds, the measured number of entries in bin *i* is:

$$\begin{aligned} \nu_i = \mu_{\text{tot}} \int_{\Omega_t} \mathrm{d}x_t \, & \text{Prob}(x_m \text{ in } i | \text{true } x_t, \text{ detected}) \\ & \times \text{Prob}(\text{detect } x_t) \times \text{Prob}(\text{produce } x_t) \\ = \mu_{\text{tot}} \int_{\text{bin } i} \mathrm{d}x_m & \int_{\Omega_t} \mathrm{d}x_t \, r(x_m|x_t) \varepsilon(x_t) f_t(x_t) \end{aligned}$$

Response Matrix (III)

Further definitions:

$$\nu_i = \mu_{\text{tot}} \int_{\text{bin } i} \mathrm{d}x_m \sum_{j=1}^{M} \int_{\text{bin } j} \mathrm{d}x_t \, r(x_m|x_t)\varepsilon(x_t) f_t(x_t)$$

$$= \sum_{j=1}^{M} \int_{\text{bin } i} dx_m \int_{\text{bin } j} \mathrm{d}x_t \, \frac{r(x_m|x_t)\varepsilon(x_t) f_t(x_t)}{\mu_j/\mu_{\text{tot}}} \mu_j$$

This may be written as

$$\nu_i = \sum_{j=1}^{M} R_{ij} \mu_j$$

with the components of the response matrix

$$R_{ij} = \frac{\int_{\text{bin } i} dx_m \int_{\text{bin } j} \mathrm{d}x_t \, r(x_m|x_t)\varepsilon(x_t) f_t(x_t)}{\int_{\text{bin } j} \mathrm{d}x_t f(x_t)}$$

Response Matrix (IV)

In other words:

$$R_{ij} = \mathrm{Prob}(\text{observed in bin } i | \text{true in bin } j)$$

Obviously, summing the response matrix over *i* gives the efficiency:

$$\sum_{i=1}^{N} R_{ij} = \varepsilon_j$$

In compact matrix form (including background):

$$\nu_i = \sum_{j=1}^{M} R_{ij}\mu_j + \beta_i \qquad \vec{\nu} = R\vec{\mu} + \vec{\beta}$$

Response matrix depends on $f_t(x_t)$ which we want to know. However, if we make the bins small enough $f_t(x_t) \approx$ const. within a bin and drops from the ratio:

$$R_{ij} = \frac{\int_{\text{bin } i} dx_m \int_{\text{bin } j} \mathrm{d}x_t \, r(x_m|x_t)\varepsilon(x_t) f_t(x_t)}{\int_{\text{bin } j} \mathrm{d}x_t f(x_t)} \approx \frac{1}{\Delta x_{t,j}} \int_{\text{bin } i} dx_m \int_{\text{bin } j} \mathrm{d}x_t \, r(x_m|x_t)\varepsilon(x_t)$$

Unfolding by Inverting Responce Matrix (I)

We have

$$\vec{\nu} = R\vec{\mu} + \vec{\beta}$$

Replace $\vec{\nu}$ by $\vec{n}$ to obtain and obvious estimator for the true distribution:

$$\hat{\vec{\mu}} = R^{-1}(\vec{n} - \vec{\beta})$$

This solution minimizes

$$\chi^2(\vec{\mu}) = (\vec{\nu}(\vec{\mu}) - \vec{n})^{\mathsf{T}} V^{-1}(\vec{\nu}(\vec{\mu}) - \vec{n}) \qquad \text{where} \qquad V_{i,j} = \mathrm{cov}[n_i, n_j]$$

It can be shown that the covariance matrix of the solution is given by

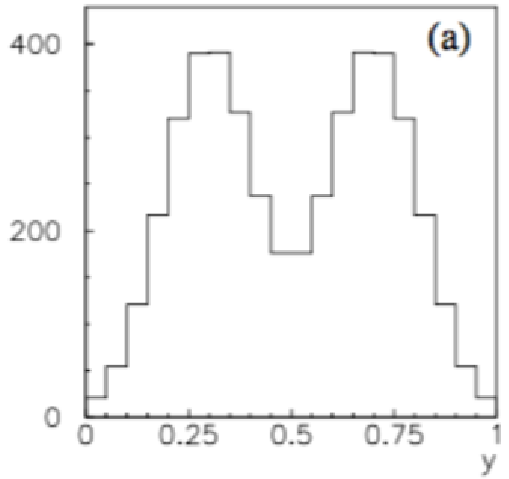
$$U = R^{-1}V(R^{-1})^{\mathsf{T}}$$

Unfolding by Inverting Responce Matrix (II)

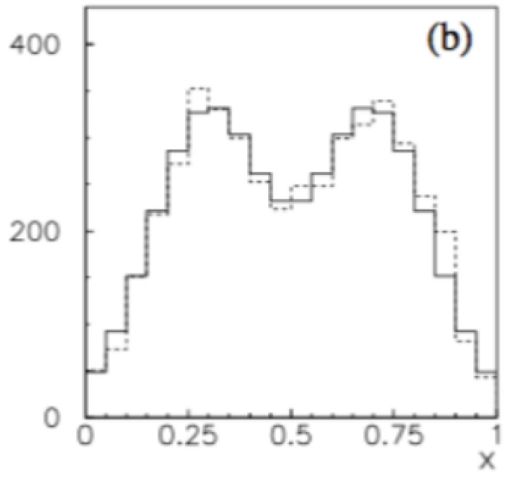
It can also be shown that matrix inversion is unbiased an has minimal variance.

This sounds good … let's try it.

Cowan, http://inspirehep.net/record/599644

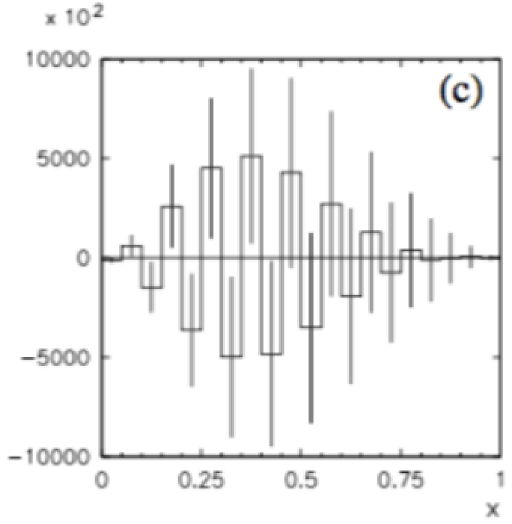


true distribution $\vec{\mu}$



solid line: $\vec{\nu} = R\vec{\mu}$

dashed line:
sampled data $\vec{n}$ from $\text{Poisson}(n_i, \nu_i)$



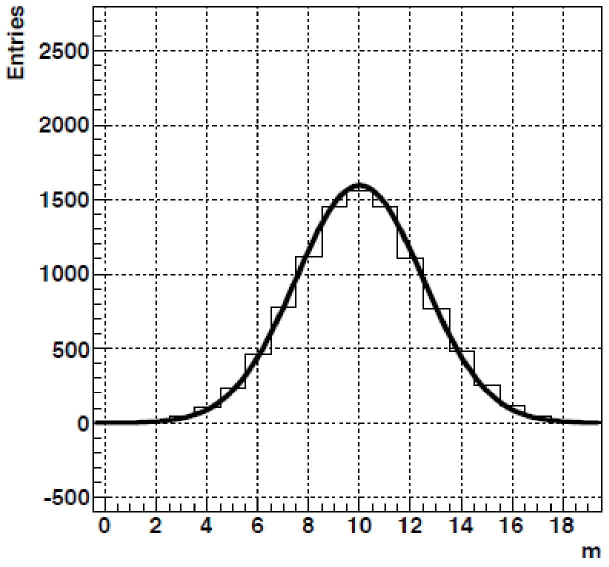
estimate $\hat{\vec{\mu}} = R^{-1}\vec{n}$

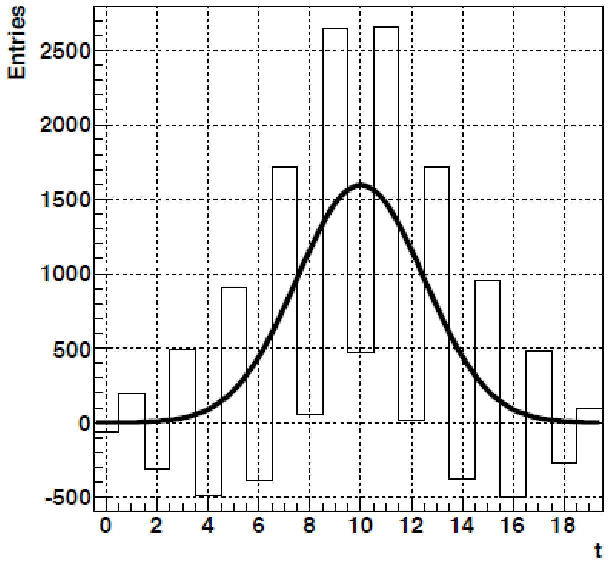
This looks like a disaster … unfolded distribution very different from the true one

Unfolding by Inverting Responce Matrix (III)

Another example:

$$R = \begin{pmatrix} 0.75 & 0.25 & 0 & & \cdots \\ 0.25 & 0.50 & 0.25 & 0 & \\ 0 & 0.25 & 0.50 & 0.25 & \\ & 0 & 0.25 & 0.50 & \\ \vdots & & & & \ddots \end{pmatrix} .$$





Same conclusion: we don't get the desired (smooth) answer

What's Wrong with the Matrix Inversion Method?

Unbiased, minimum variance, actually also a ML estimator … all very nice!

The result is not wrong, it is just not desirable

- Does not really look like the original distribution
- Large correlation between bins

"Applying the response matrix R smears out fine structure
$\rightarrow$ applying R^{-1} creates (usually unwanted) structure"

More desirable solution by adding (smoothness) constraints.
However, this will produce a bias.

The art of unfolding is to find an acceptable balance between bias and smoothness.

Bin-by-Bin Method (I)

Used very often, but has issues …

Assume shape of true spectrum and determine correction factor for each bin (usually determined from Monte Carlo simulation):

$$\mu_i = C_i(n_i - \beta_i) \qquad\qquad C_i = \frac{\mu_i^{\mathrm{MC}}}{\nu_i^{\mathrm{MC}}}$$

Works if smearing (bin-to-bin sharing) is negligible, only loss due to finite efficiency:

$$R_{ij} \approx \delta_{ij}\varepsilon_j$$

Obviously works, too, if MC = nature.

Expectation value for corrected data:

$$E[\hat{\mu}_i] = C_i E[n_i - \beta_i] = C_i(\nu_i - \beta_i) \equiv C_i \nu_i^{\mathrm{sig}}$$

Bin-by-Bin Method (II)

Inserting the C_i's one can determine the bias:

$$E[\hat{\mu}_i] = \frac{\mu_i^{\mathrm{MC}}}{\nu_i^{\mathrm{MC}}} \nu_i^{\mathrm{sig}} = \underbrace{\left(\frac{\mu_i^{\mathrm{MC}}}{\nu_i^{\mathrm{MC}}} - \frac{\mu_i}{\nu_i^{\mathrm{sig}}} \right) \nu_i^{\mathrm{sig}}}_{\text{bias}} + \mu_i$$

no bias only if MC = nature

Covariance matrix of the corrected data (smearing fluctuations independent between bins)

$$U_{ij} = \mathrm{cov}[\hat{\mu}_i, \hat{\mu}_j] = C_i C_j \underbrace{\mathrm{cov}[n_i^{\mathrm{sig}}, n_j^{\mathrm{sig}}]}_{0 \text{ for } i \neq j} = C_i^2 \mathrm{Var}[n_i^{\mathrm{sig}}] \delta_{ij}$$

Iterative bin-by-by method

- Start with plausible guess of true spectrum
- Apply correction to measurement
- Generate new correction factors from corrected spectrum of previous iteration
- And so on … usually a few iterations sufficient

Regularized Unfolding

Matrix inversion is the maximum likelihood solution:

Independent Poisson fluctuations:

$$\ln L(\vec{\mu}) = \sum_{i=1}^{M} (n_i \ln \nu_i - \nu_i)$$

ML estimator:

$$\hat{\vec{\nu}} = \vec{n}$$

$$\rightarrow \hat{\vec{\mu}} = R^{-1}(\vec{n} - \vec{\beta})$$

Idea: accept solutions that are close to maximum likelihood estimate:

$$\ln L(\vec{\mu}) \geq \ln L(\vec{\mu}_{\text{max}}) - \Delta \ln L(\vec{\mu})$$

Define a smoothness function S that gets bigger when the unfolded solution becomes smoother.

The task then is to maximize

$$\Phi(\vec{\mu}) = \alpha \ln L(\vec{\mu}) + S(\mu)$$

α depends on $\Delta \ln \vec{\mu}$,
$\alpha \rightarrow \infty$ give ML solution

smoothness function

Tikhonov Regularization

Measure of smoothness = mean square of *k*-th derivative of deconvoluted function *f*:

$$S[f] = -\int \mathrm{d}x \left(\frac{\mathrm{d}^k f}{\mathrm{d}^k x} \right)^2 \qquad k = 1, 2, 3, \ldots$$

Minus sign makes *S* big when derivative is small

Tikhonov for *k* = 2 with log *L* = $-\chi^2/2$:

$$S(\vec{\mu}) = -\sum_{i=1}^{M-2} (-\mu_i + 2\mu_{i+1} - \mu_{i+2})^2$$

Implementation by A. Höcker, V. Kartvelishvili: *Singular Value Decomposition*
(NIM A372 (1996) 469, hep-ph/9509307, TSVDUnfold in ROOT)

Minimizes $$\chi^2(\vec{\mu}) + \tau \sum_i [(\mu_{i+1} - \mu_i) - (\mu_i - \mu_{i-1})]^2$$

Advice on how to choose τ in the paper

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

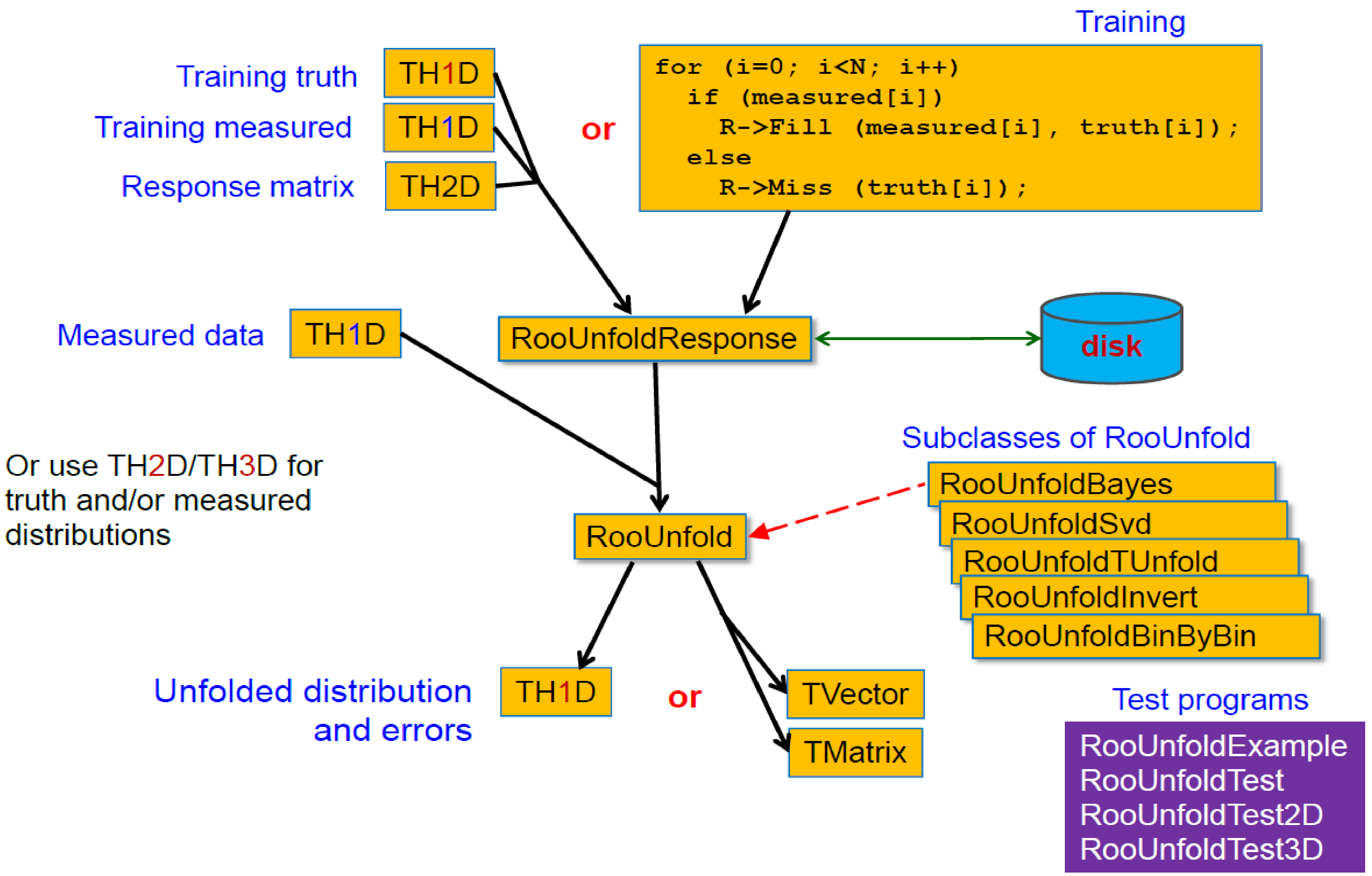
- Supports different binning scenarios
  - multi-dimensional distributions (1D, 2D, and 3D)
  - Different binning (or even dimensionality) for measured and truth
  - Option to include or exclude histogram under/overflow bins in the unfolding
- Supports different methods for error computation (simple switch). In order of increasing CPU time:
  - No error calculation (uses $\sqrt{N}$)
  - 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

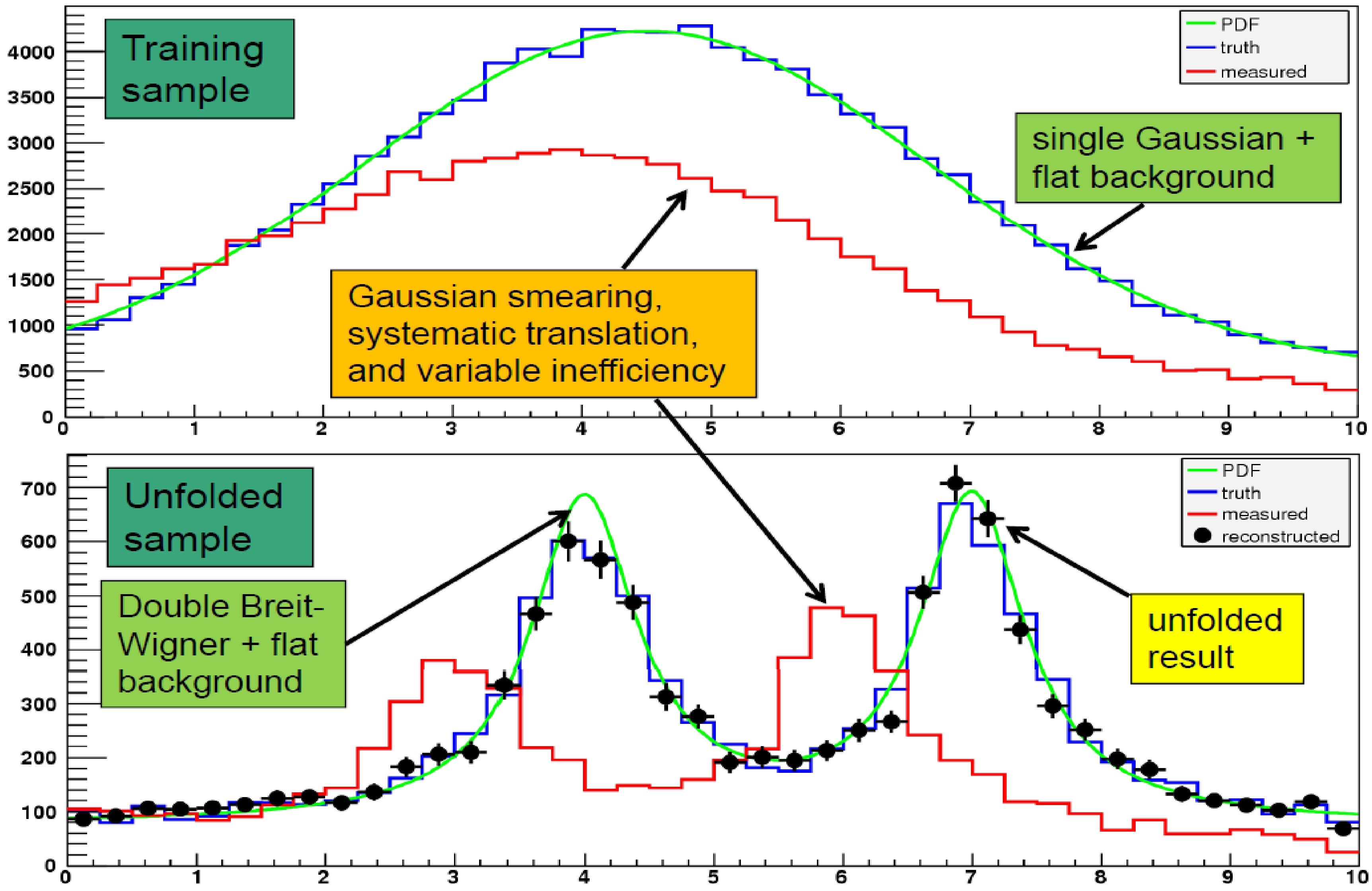
  and plotting results from a single command

RooUnfold classes

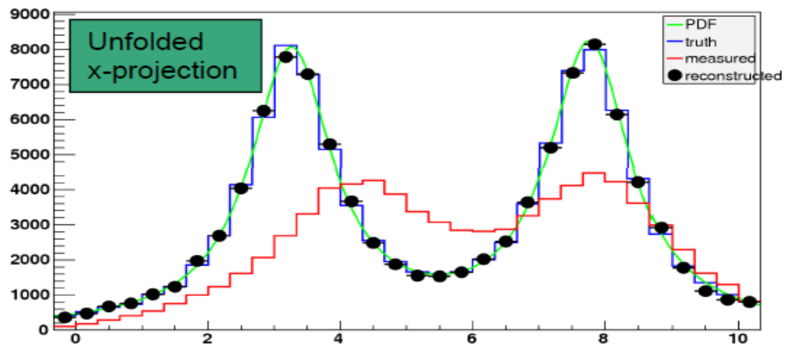


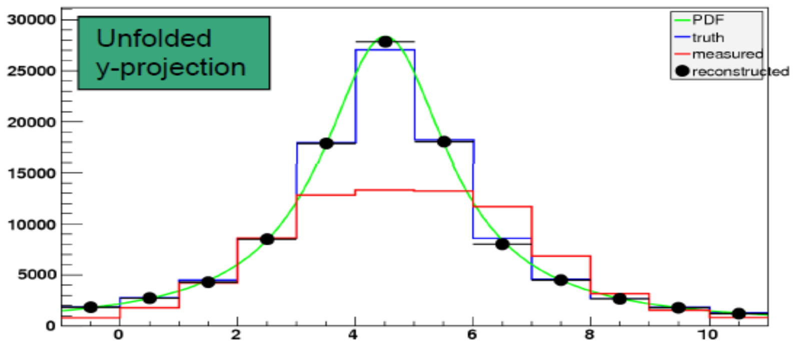
```
for (i=0; i<N; i++)
  if (measured[i])
    R->Fill (measured[i], truth[i]);
  else
    R->Miss (truth[i]);
```

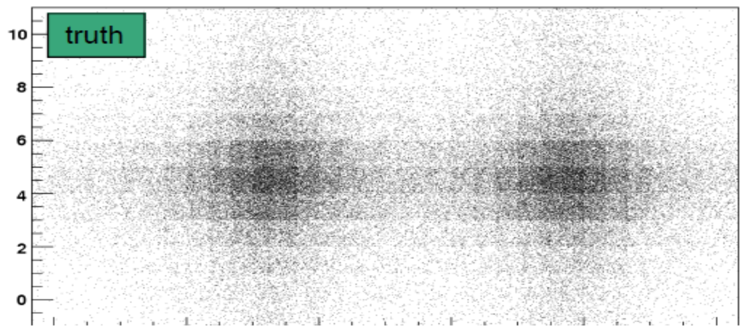
RooUnfold example (Bayes)

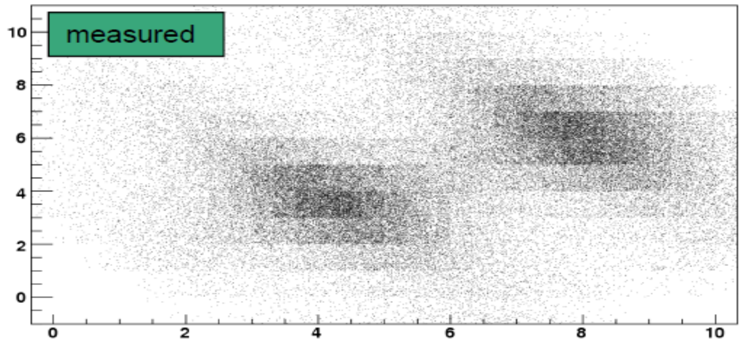


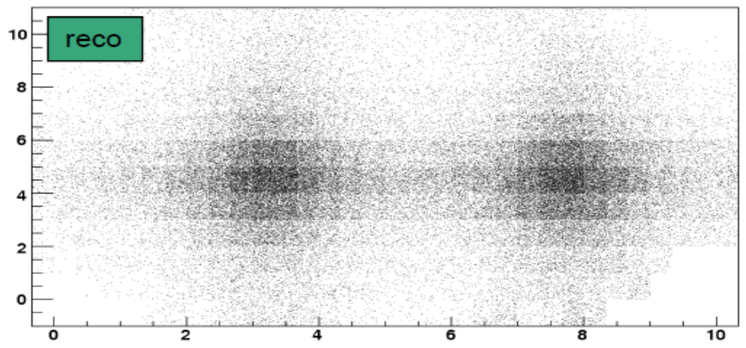
RooUnfold example (Bayes)











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 $\rightarrow$ result dominated by MC truth
    - k too large $\rightarrow$ 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

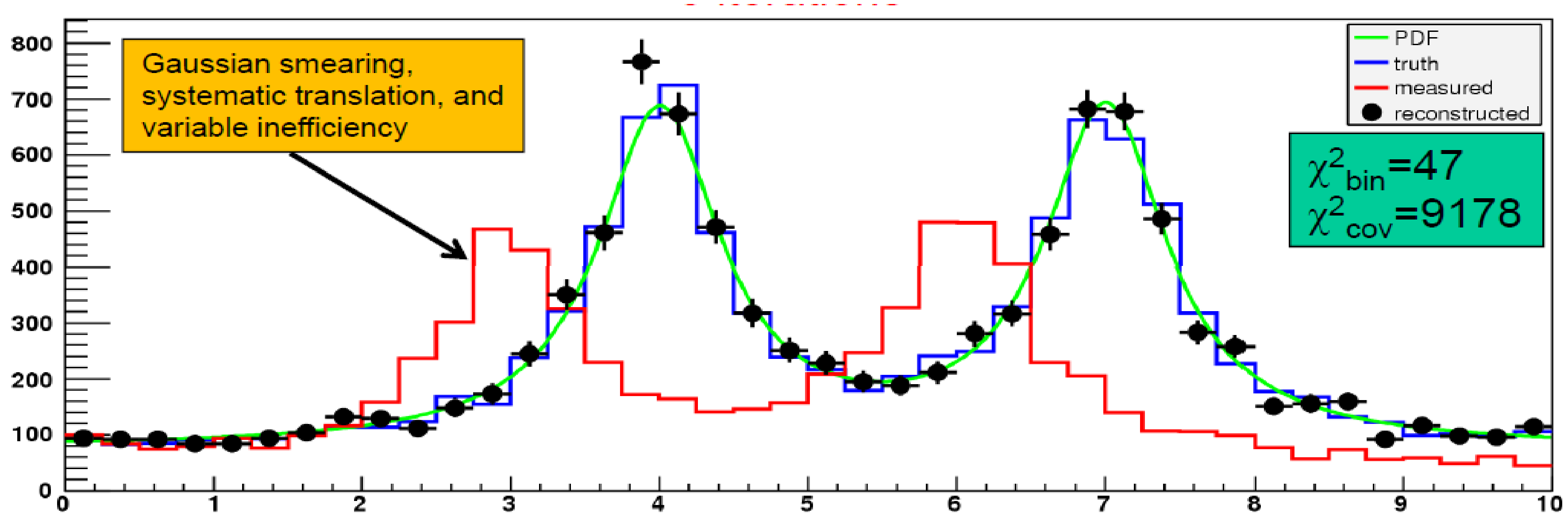
- Very simple algorithms
  - using bin-by-bin correction factors, with no inter-bin migration
  - using unregularised matrix inversion with singular value removal (TDecompSVD)

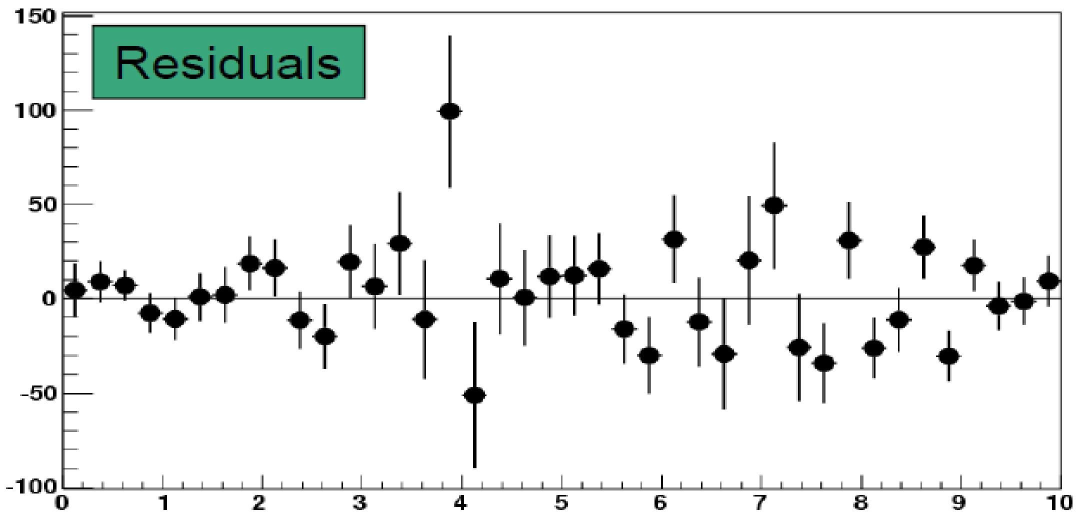
  are included for comparison – and to demonstrate why they should not be used in most cases!

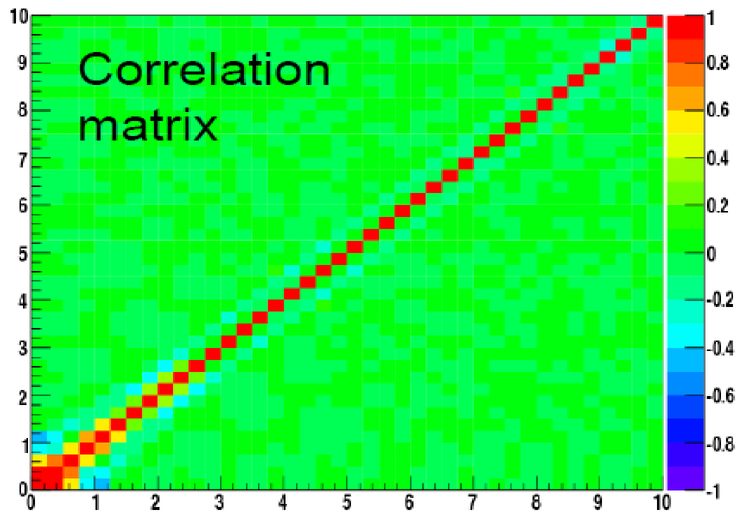
RooUnfold algorithms: comparison

- 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)

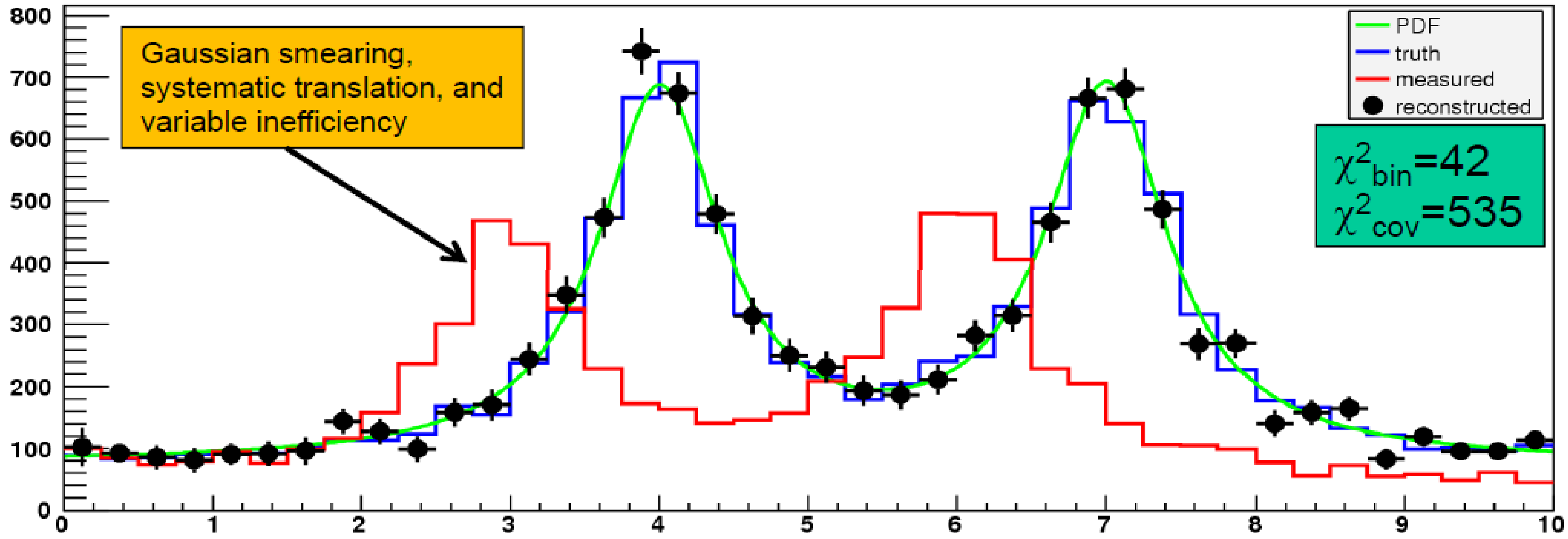
RooUnfold with Bayes algorithm (3 iterations)

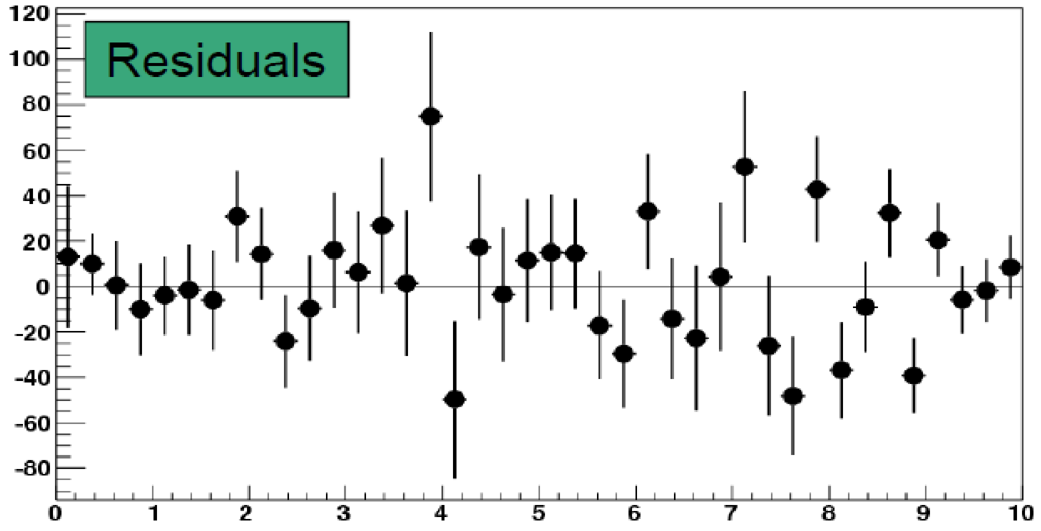


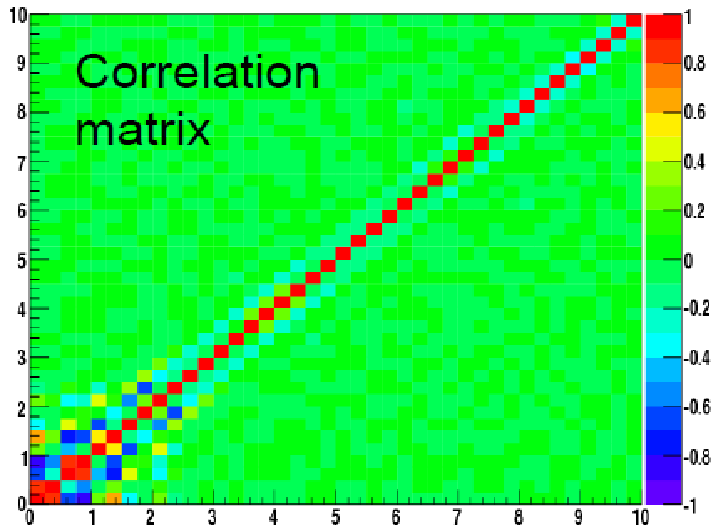




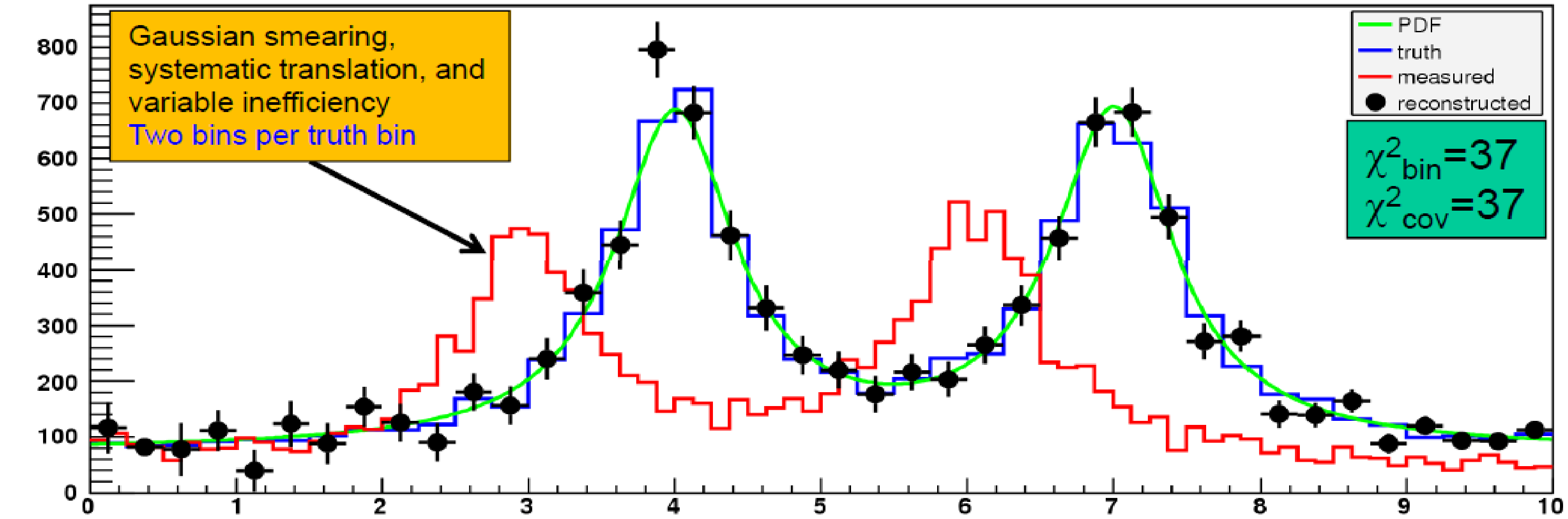
RooUnfold with SVD algorithm (k=30)

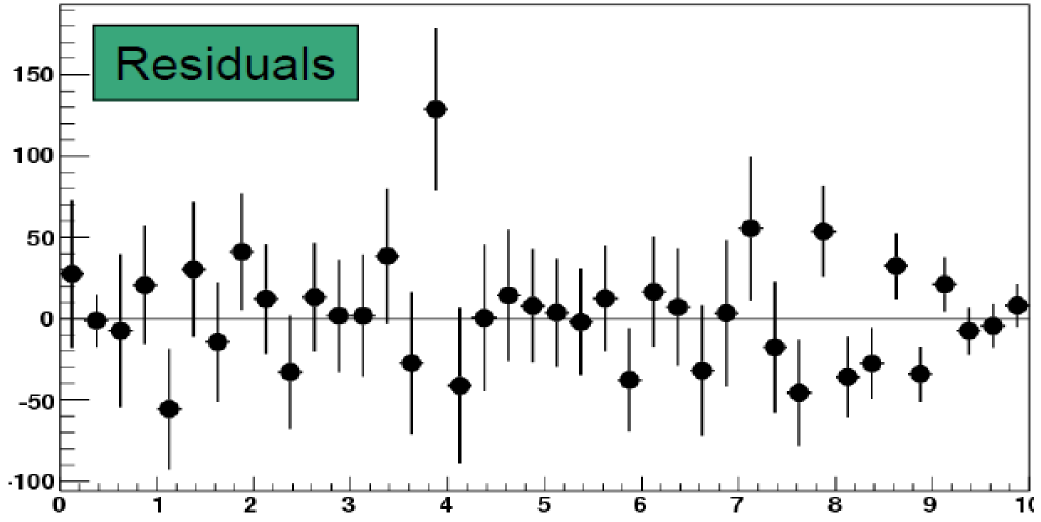


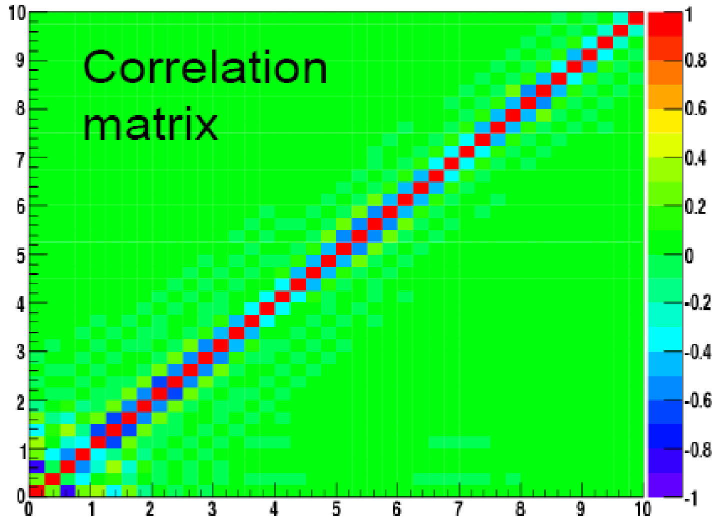




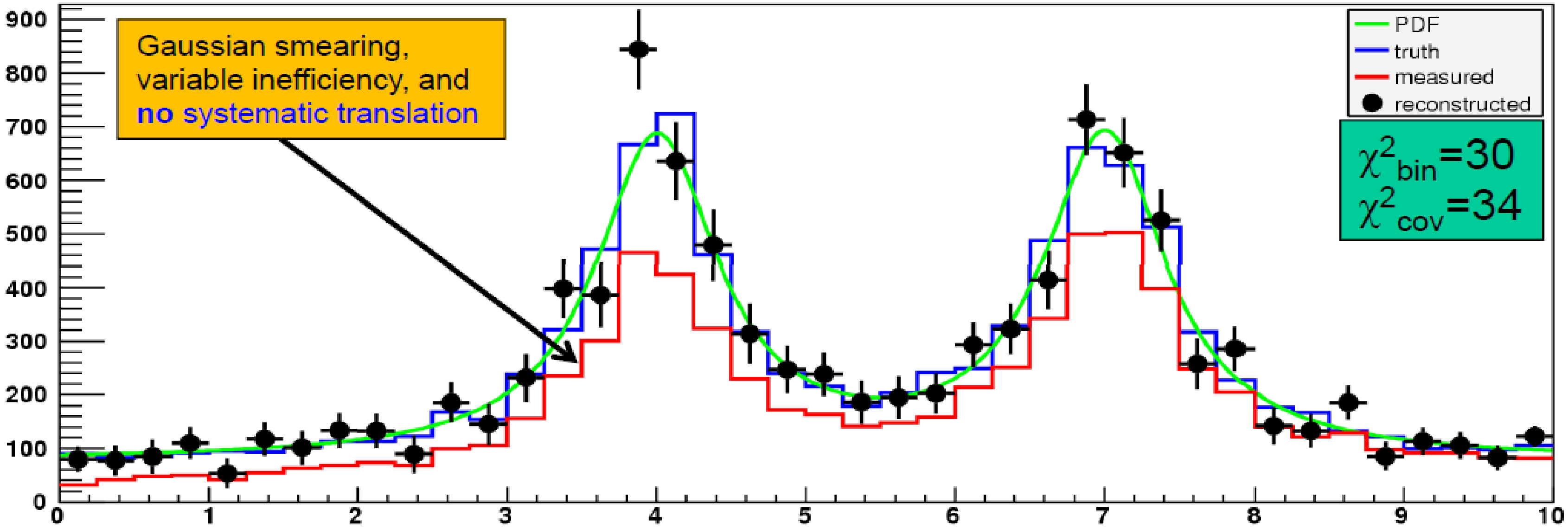
RooUnfold with TUnfold algorithm (τ=0.004)

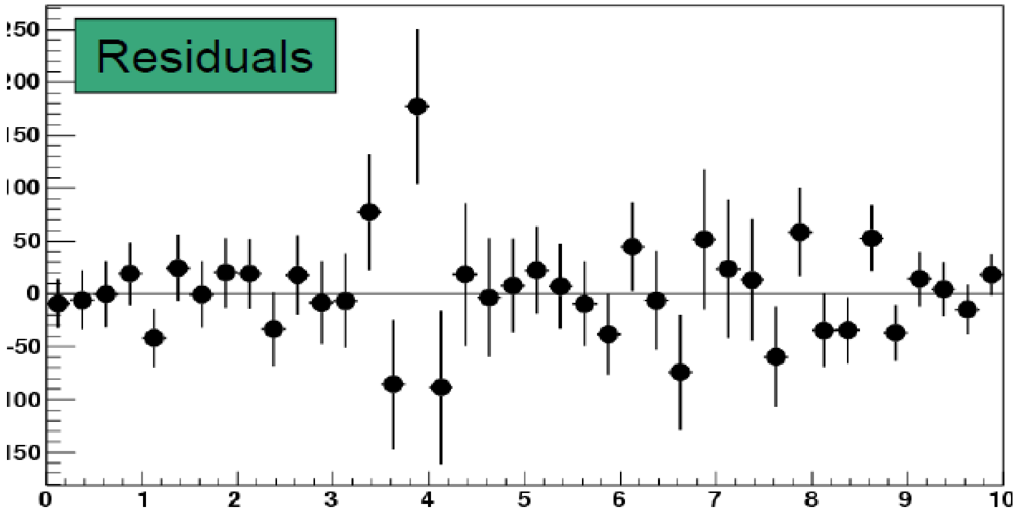


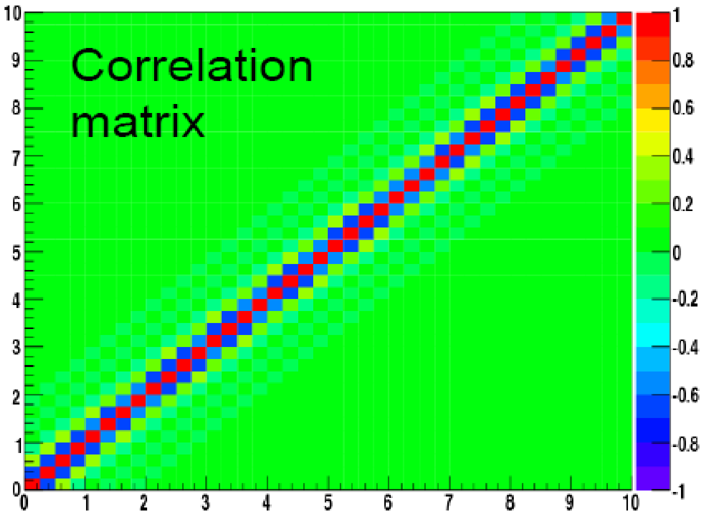




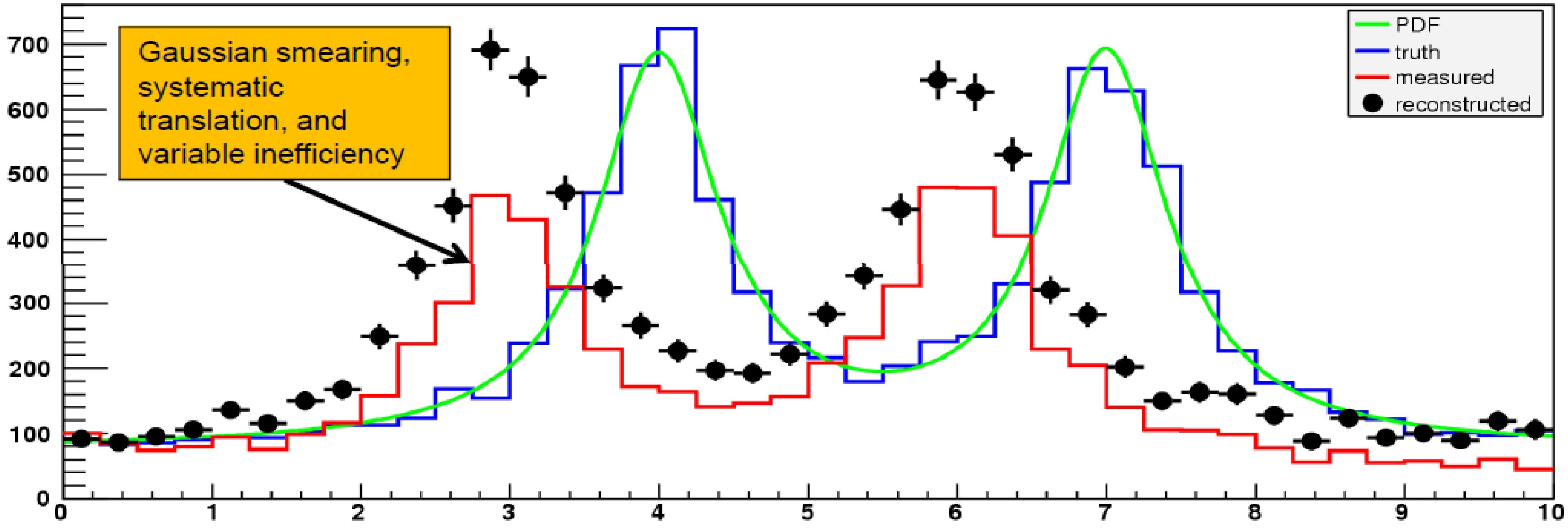
Unregularised matrix inversion

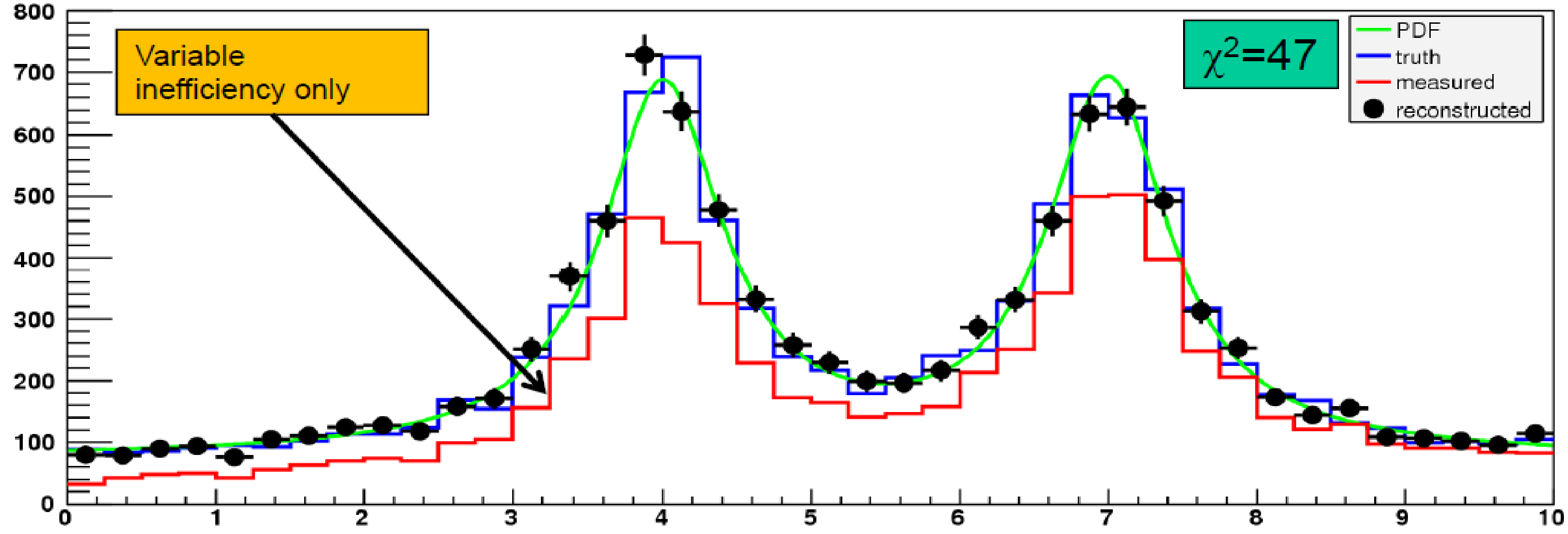






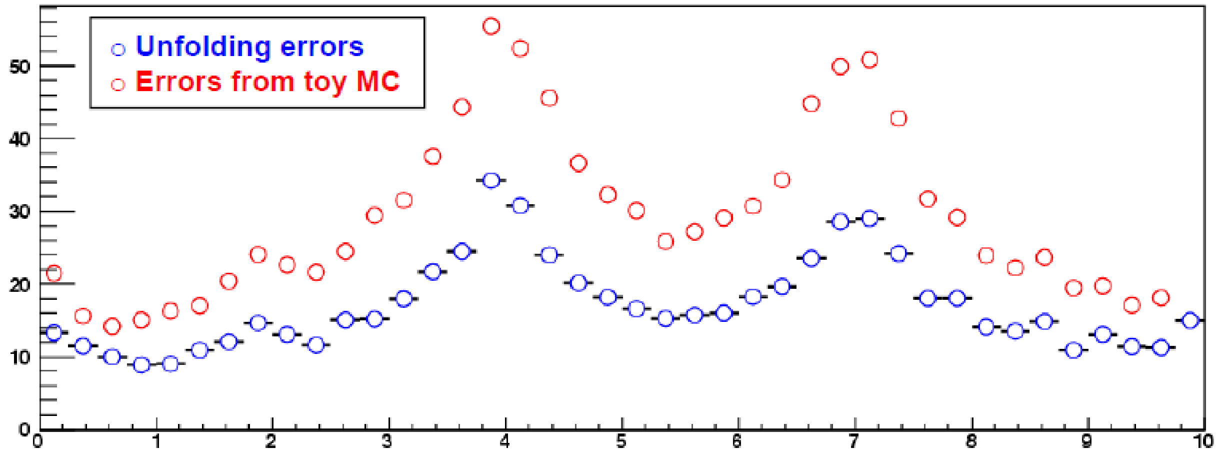
Simple correction factors





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 $\rightarrow$ 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