Science & Technology Facilities Council
Rutherford Appleton Laboratory

RooUnfold
unfolding framework and algorithms

Tim Adye
Rutherford Appleton Laboratory

ATLAS RAL Physics Meeting
20th May 2008

Outline

1. What is Unfolding?
   - and why might you want to do it?
2. Overview of a few techniques
   - Regularised unfolding
   - Iterative method
3. Some details
   - Filling the response matrix
   - Choice of regularisation parameter
4. RooUnfold package
   - Currently implements three algorithms with a common interface
5. Status and Plans
6. References

Unfolding

- In other fields known as “deconvolution” or “unsmearing”

- Given a “true” PDF in $\boldsymbol{\mu}$ that is corrupted by detector effects, described by a response function, $\mathbf{R}$, we measure a distribution in $\boldsymbol{\nu}$. For a binned distribution:

$$\nu_i = \sum_{j=1}^{M} R_{ij}\mu_j \qquad i = 1..N$$

- This may involve
  1. inefficiencies: lost events
  2. bias and smearing: events moving between bins (off-diagonal R_{ij})

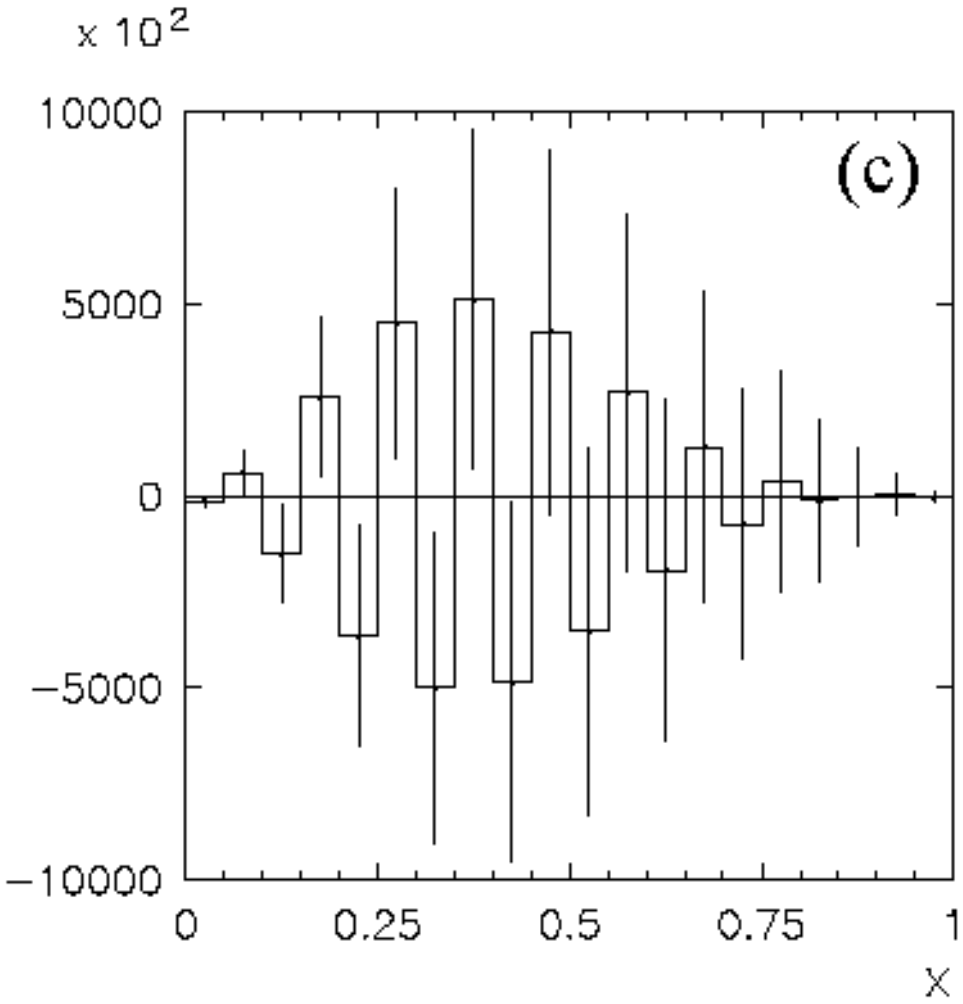
- With infinite statistics, it would be possible to recover the original PDF by inverting the response matrix

$$\boldsymbol{\mu} = \mathbf{R}^{-1}\boldsymbol{\nu}$$

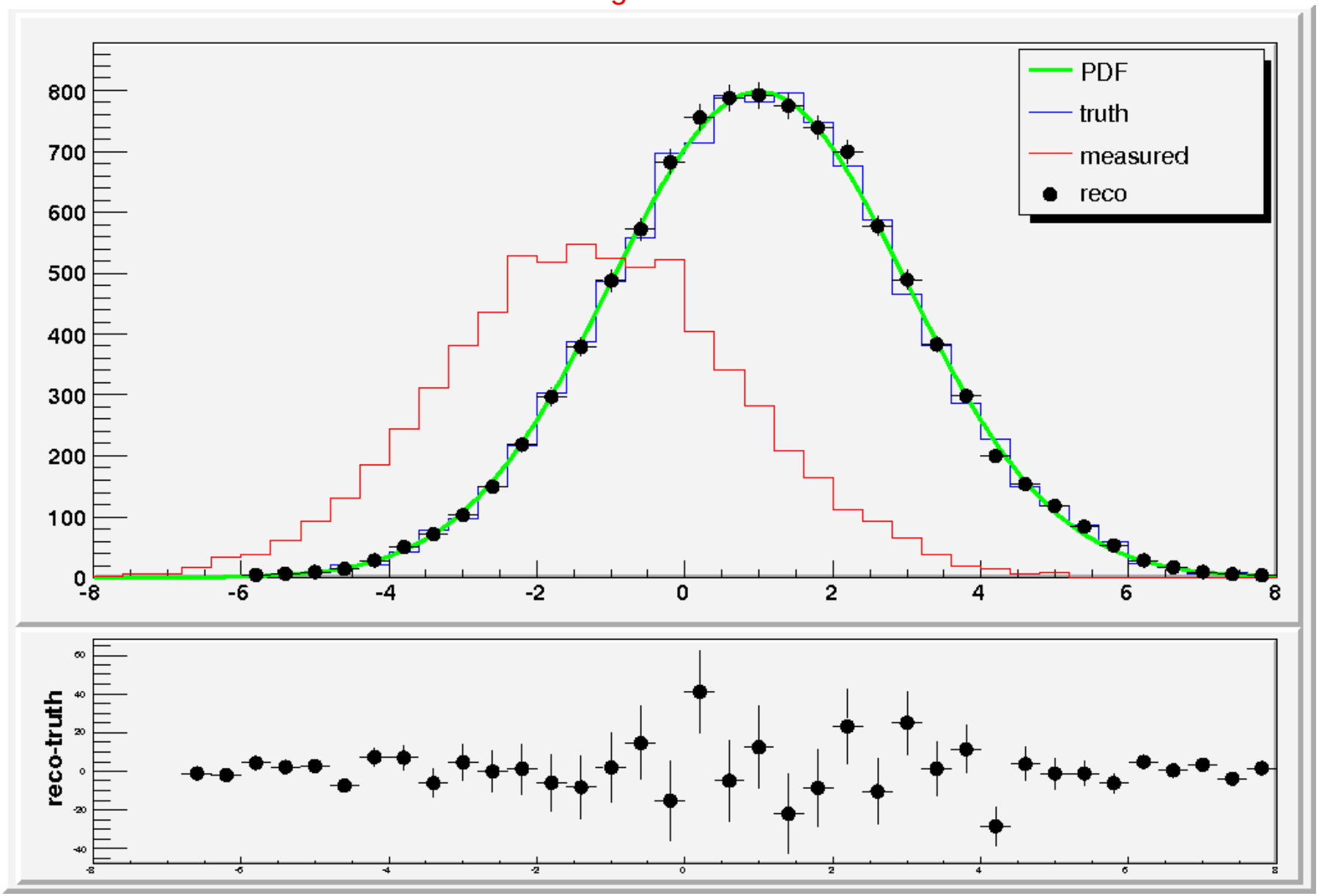
Not so simple…

- Unfortunately, if there are statistical fluctuations between bins this information is destroyed
  - Since $\mathbf{R}$ washes out statistical fluctuations, $\mathbf{R}^{-1}$ cannot distinguish between wildly fluctuating and smooth PDFs
    - Obtain large negative correlations between adjacent bins
    - Large fluctuations in reconstructed bin contents →
- Need some procedure to remove wildly fluctuating solutions
  1. Give added weight to "smoother" solutions
  2. Solve for $\boldsymbol{\mu}$ iteratively, starting with a reasonable guess and truncate iteration before it gets out of hand
  3. Ignore bin-to-bin fluctuations altogether

What happens if you don't regularise



True Gaussian, with Gaussian smearing, systematic translation, and variable inefficiency – trained using a different Gaussian



So why don't we always do this?

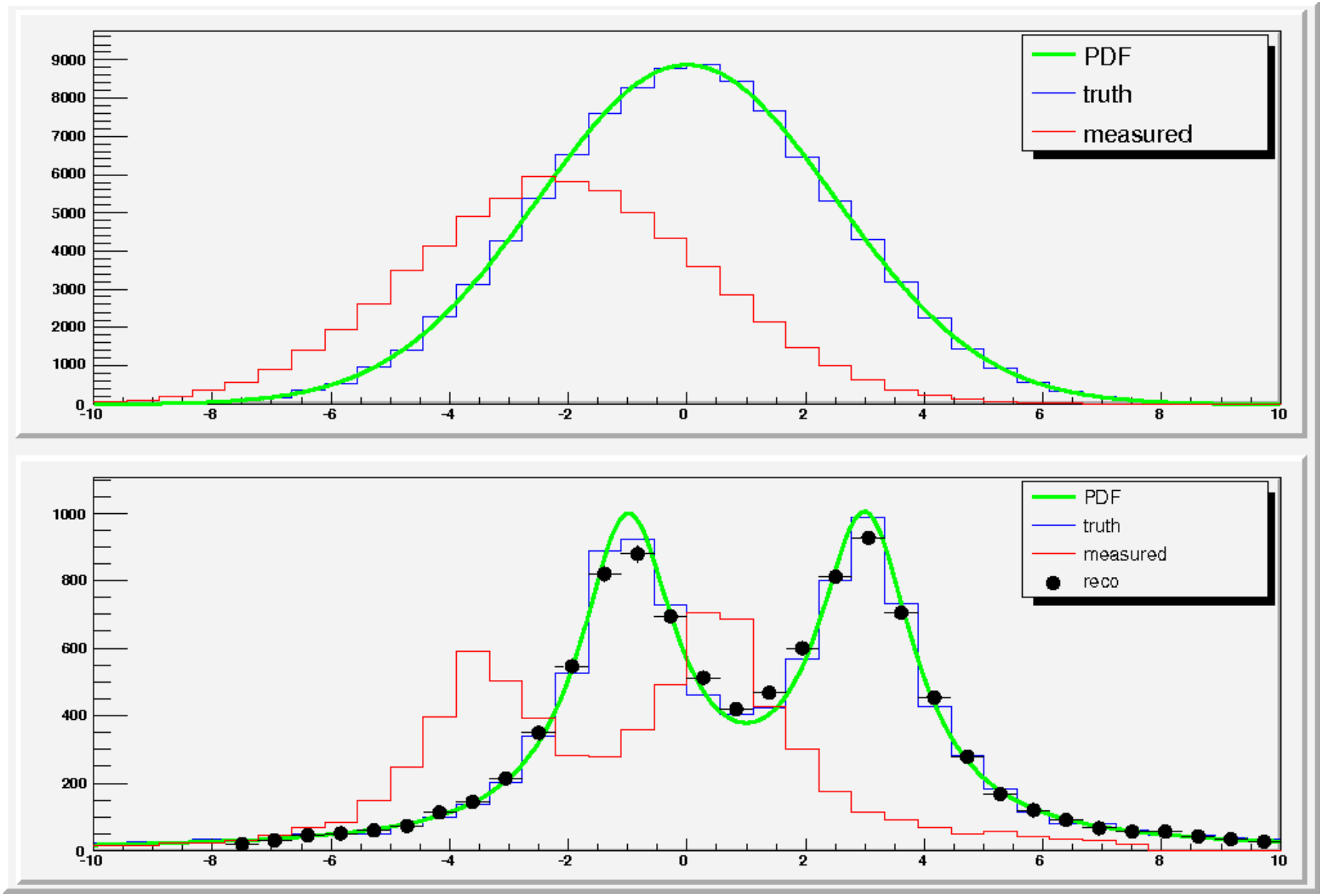
- If the true PDF and response function can be parameterised, then a Maximum Likelihood fit is usually more convenient
  - Directly returns parameters of interest
  - Does not require binning
- If the response matrix doesn't include smearing (ie. it's diagonal), then apply bin-by-bin efficiency correction directly
- If result is just needed for comparison (eg. with MC), could apply response function to MC
  - simpler than un-applying response to data

- Use unfolding to recover theoretical distribution where
  - there is no a-priori parameterisation, and
  - it is needed for the result and not just comparison with MC, and
  - there is significant bin-to-bin migration of events

Response Matrix

- The response matrix may be known a-priori, but usually it is determined from Monte Carlo
  - this process is referred to “training”
  - to reduce systematic effects, use a training distribution close to the data
- For unfolding a 1D distribution, the response matrix can be represented as a 2D histogram
  - filled with MC values for ($x_{measured}$, x_{true})
  - each x_{true} column should be normalised to its reconstruction efficiency
    - an event is either measured with a value $x_{measured}$, or accounted for in the inefficiency

Double Breit-Wigner, with Gaussian smearing, systematic translation, and variable inefficiency – trained using a single Gaussian



Choice of Regularisation Parameter

- In both types of algorithm I will discuss, the regularisation parameter determines the relative weight placed on the data, compared to the training MC truth... or between statistical and systematic errors
  - One extreme favours the data, with the risk of statistical fluctuations being seen as true structure
    - has larger statistical errors – but these can be determined
    - in the limit, can be the same as matrix inversion, but numerical effects often appear first
  - The other extreme favours the training sample truth
    - if the MC truth is different from the data (as it surely will be, otherwise why do the experiment!), this will lead to larger systematic errors
- Of course, one chooses a value somewhere between these extremes
  - This can be optimised and tested with MC samples that are statistically and systematically independent of the training sample
    - Will depend on the number of events and binning
    - This step can usually be performed with toy MC samples

1. Regularised Unfolding

- Use Maximum Likelihood to fit smeared bin contents to measured data, but include regularisation function

$$\ln L'(\boldsymbol{\mu}) = \ln L(\boldsymbol{\mu}) + \alpha S(\boldsymbol{\mu})$$

  where the regularisation parameter, $\boldsymbol{\alpha}$, controls the degree of smoothness (select $\boldsymbol{\alpha}$ to, eg., minimise mean squared error)
- Various choices of regularisation function, $\boldsymbol{S}$, are used
  - Tikhonov regularisation: minimise curvature
    - for some definition of curvature, eg. $S(\boldsymbol{\mu}) = -\sum_{i=2}^{M-1}[(\mu_{i+1} - \mu_i) - (\mu_i - \mu_{i-1})]^2$
    - Implemented as part of **`RUN`** by Volker Blobel
  - Maximum entropy: $S(\boldsymbol{\mu}) = -\sum_{i}^{M}(\mu_i / \mu_{\text{tot}})\ln(\mu_i / \mu_{\text{tot}})$
  - **`RooUnfHistoSvd`** by Kerstin Tackmann and Heiko Lacker (BaBar)
    - based on **`GURU`** by Andreas Höcker and Vakhtang Kartvelishvili
    - uses Singular Value Decomposition of the response matrix to simplify the regularisation process

2. Iterative method

- Uses Bayes' theorem to invert

$$R_{ij} = P(\text{observed in bin } i \mid \text{true value in bin } j)$$

and using an initial set of probabilities, $\boldsymbol{p_i}$ (eg. MC truth) obtain an improved estimate

$$\hat{\mu}_i = \frac{1}{\varepsilon_i} \sum_{j=1}^{N} \frac{R_{ij} p_i}{\sum_k R_{jk} p_k} n_j$$

- Repeating with new $\boldsymbol{p_i}$ from these new bin contents converges quite rapidly
  - Truncating the iteration prevents us seeing the bad effects of statistical fluctuations

- Fergus Wilson and I have implemented this method in ROOT/C++
  - Supports 1D, 2D, and 3D cases

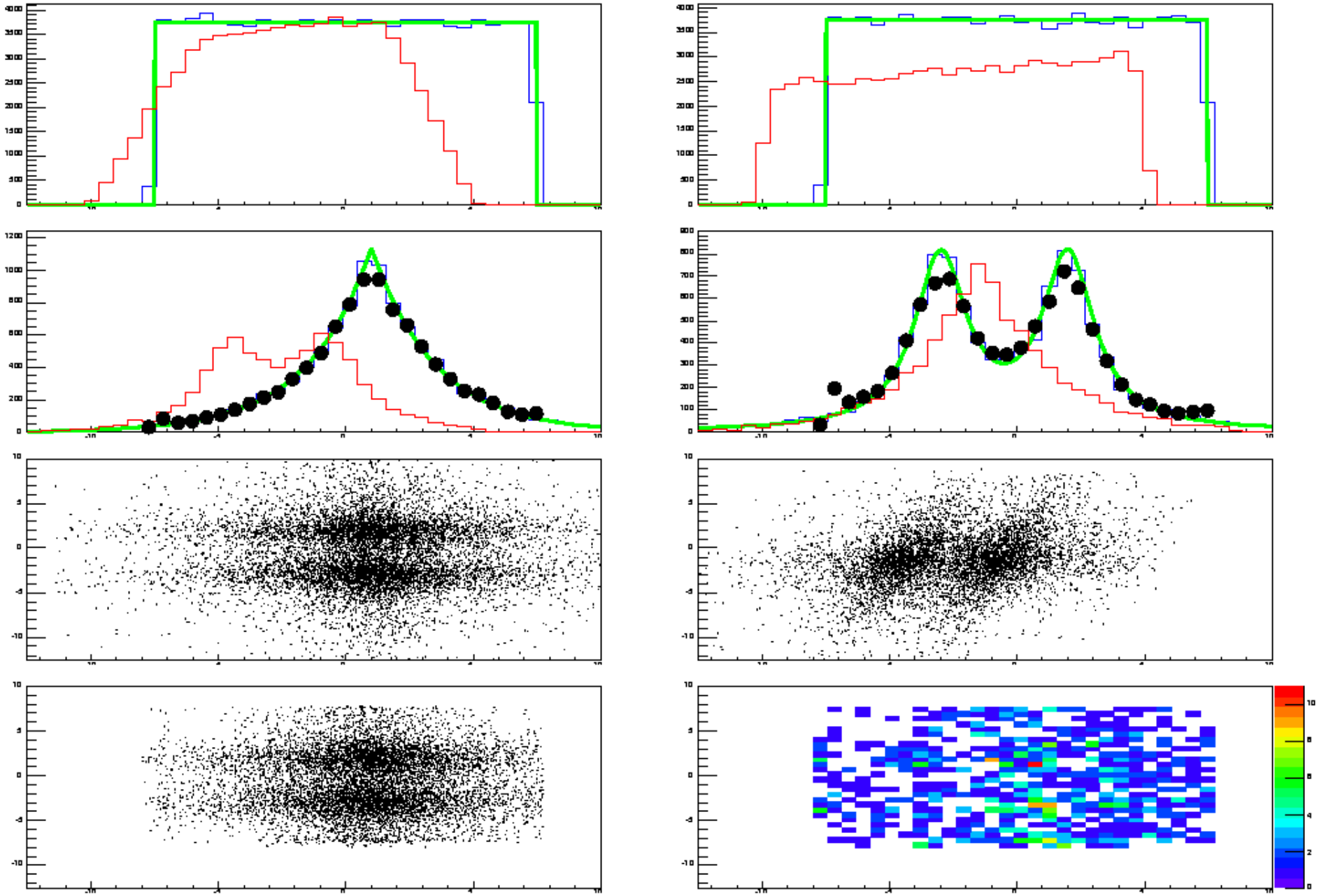
RooUnfold Package

- Make these different methods available as ROOT/C++ classes with a common interface to specify
  - unfolding method and parameters
  - response matrix
    - pass directly or fill from MC sample
    - RooUnfold takes care of normalisation
  - measured histogram
  - return reconstructed truth histogram and errors
    - full covariance matrix also available
  - Simplify handling of multiple dimensions
    - when supported by the underlying algorithm

- This should make it easy to try and compare different methods in your analysis

2D Unfolding Example

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



RooUnfold Classes

- RooUnfoldResponse
    - response matrix with various filling and access methods
    - create from MC, use on data (can be stored in a file)
- RooUnfold – unfolding algorithm base class
  - RooUnfoldBayes – Iterative method
  - RooUnfoldSvd – Inteface to RooUnfHistoSvd package
  - RooUnfoldBinByBin – Simple bin-by-bin method
    - Trivial implementation, but useful to compare with full unfolding

- RooUnfoldExample – Simple 1D example
- RooUnfoldTest and RooUnfoldTest2D
  - Test with different training and unfolding distributions

Plans and possible improvements

1. Simplify interface: new RooUnfoldDistribution class for more filling/output options
   - consistent handling of multi-dimensional unfolding, with any number of dimensions
   - allow access by histogram (THxD), vector (TVectorD), or matrix (TMatrixD)
     - Other data types, eg. float rather than double?
   - Should be mostly upwardly compatible so users don't have to change code
2. Add common tools, useful for all algorithms
   - Automatic calculation of figures of merit (eg. χ^2)
     - can also use standard ROOT functions on histograms
   - Simplify or automate selection of regularisation parameter
3. More algorithms?
   - Maximum entropy regularisation
   - Simple (if slow) matrix inversion without regularisation
     - perhaps useful with large statistics
   - Investigate techniques used in astrophysics, eg. CLEAN
4. Incorporate as an official ROOT package?

RooUnfold Status

- RooUnfold was originally developed in the BaBar framework.
- I have subsequently released a stand-alone version
  - This is what I will continue to develop, so it can be used everywhere
  - There seems to be some interest in the HEP community
    - ... at least judging by the number of questions from various experiments I have received
- Unfortunately, I have not had time for much development
  - So far, this has been a “spare time” activity for me
  - I am working with Fergus Wilson, who is interested in trying out some other algorithms

References

RooUnfold code, documentation, and references to unfolding reviews and techniques can be found on this web page

http://hepunx.rl.ac.uk/~adye/software/unfold/RooUnfold.html