Introduction to Data unfolding

GRK2044 annual meeting

Stefan Schmitt

Outline

- Introduction
- Matrix inversion, bin-by-bin, Likelihood fit
- Regularised unfolding methods
- Prediction error and related quantities
- Choice of regularisation parameters
  - Eigenvalue analysis
  - L-curve scan
  - SURE minimisation

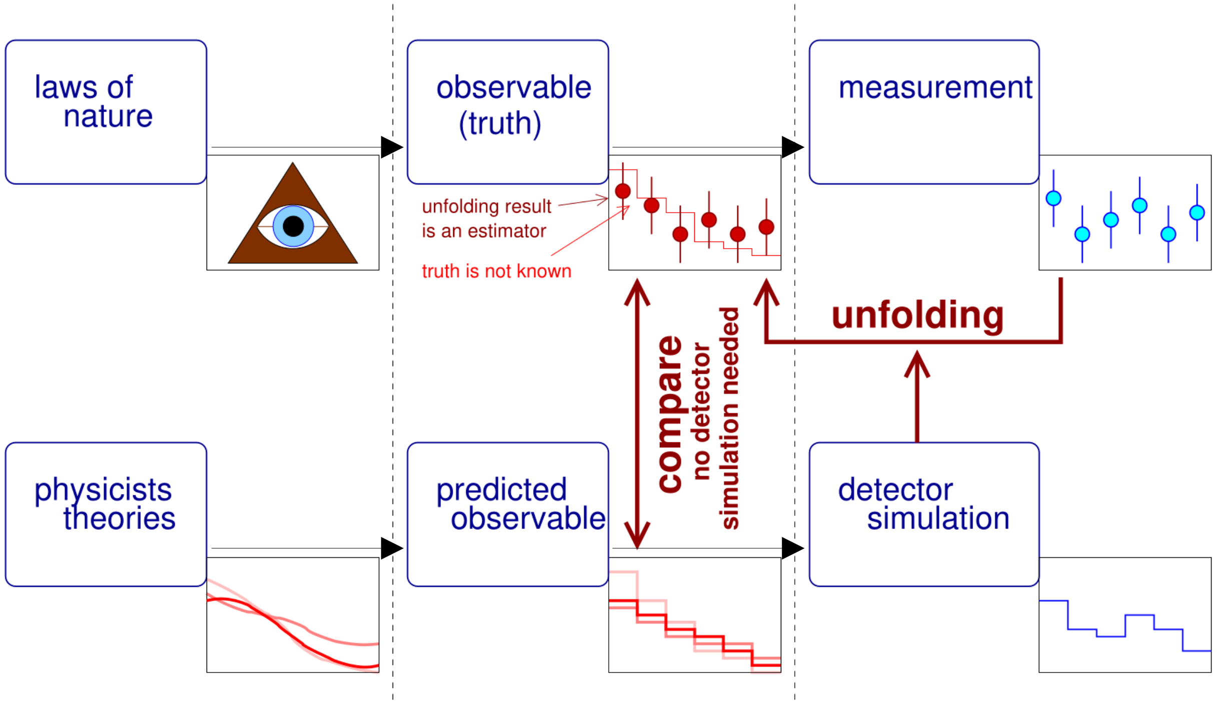
Exercises:

- lecture will be interrupted a few times for exercises
- Exercise results will be discussed in the lecture

Introduction

Unfolding: what this talk is about

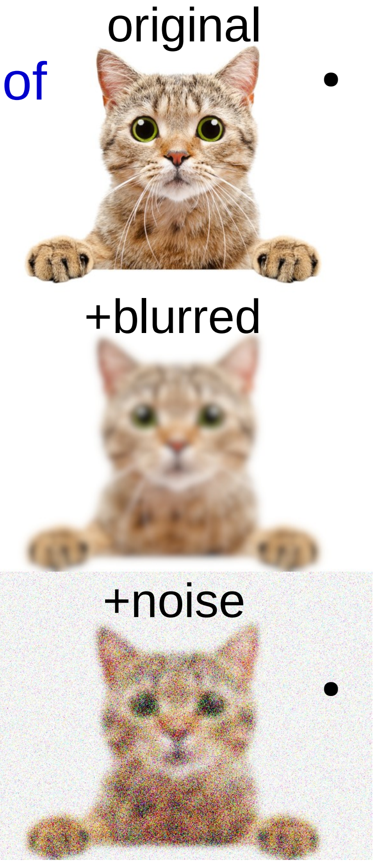
- Experimentalists record data which are “blurred” by detector effects
- Goal of data unfolding: present data independent of detector effects
- Decouples understanding of the detector from data interpretation



→ unfolded data are well suited for comparisons to (future) predictions

Folding and Unfolding examples

- Particle physics: measurement of a differential cross section as a function of a particle's energy
  - Particle count suffers from statistical fluctuations
  - Energy measurement is uncertain due to detector effects
- Unfold cross section: result independent of detector effects (still has "statistical uncertainties")

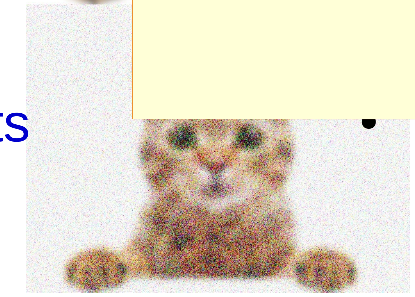


- Imaging (Astronomy, medical applications, etc)
  - Pixels are blurred (camera) or the detector only measures projections (tomography)
  - Light intensity in a detector element is uncertain (photon counting)
- Unfolding or "deconvolution": present the image without blurring

Folding and Unfolding examples

- Particle physics: measurement of a differential cross section as a function of a particle's energy
  - Particle count suffers from statistical fluctuations
  - Energy measurement is uncertain due to detector effects
- Unfold cross section: result independent of detector effects (still has "statistical uncertainties")

original

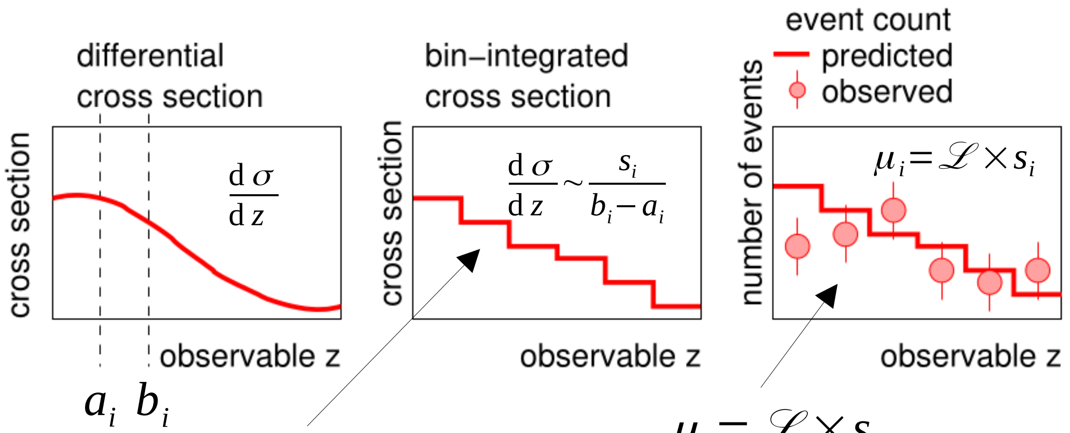


- Imaging (Astronomy, medical applications, etc)
  - Pixels are blurred (camera)
    y
    ons
    detector
    in (photon
- Unfolding or "deconvolution": present the image without blurring

This talk: examples will be from particles physics

Binning and Poisson statistics

- Particle physics: event counting
- Events are usually counted in bins i as a function of some observable
- Poisson probability distribution
- Expected event count is equal to cross section times integrated luminosity of the experiment
- Theorists can predict differential cross sections, so the outcome of the experiment can be used to test their models



$$s_i = \int_{a_i}^{b_i} [\frac{d\sigma}{dz}] dz$$

$$\mu_i = \mathcal{L} \times s_i$$

$\mathcal{L}$: integrated luminosity

μ_i : Poisson parameter

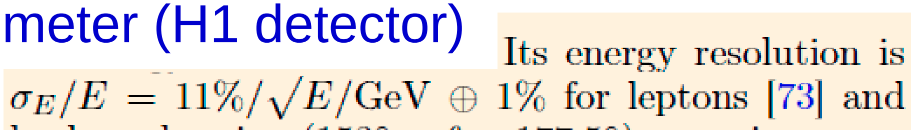
$$P(y_i; \mu_i) = \exp(-\mu_{ab}) \frac{\mu_i^{y_i}}{y_i!}$$

Expectation $E(y_i)=\mu_i$ is estimated from data as y_i

Variance $\sigma_i^2=\mu_i$ is also estimated from data as y_i

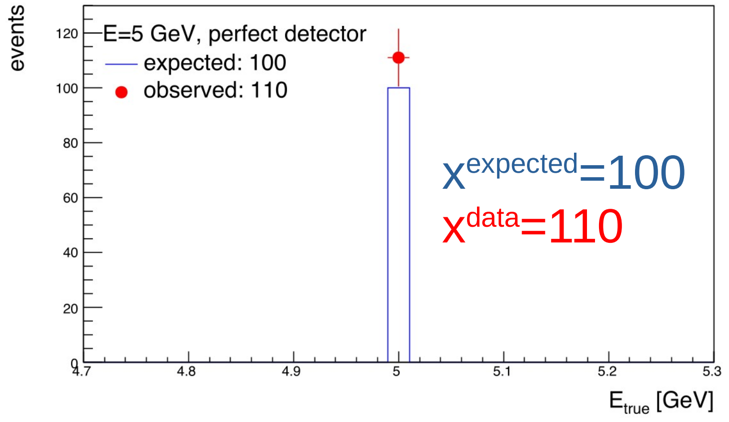
Detector effects (1)

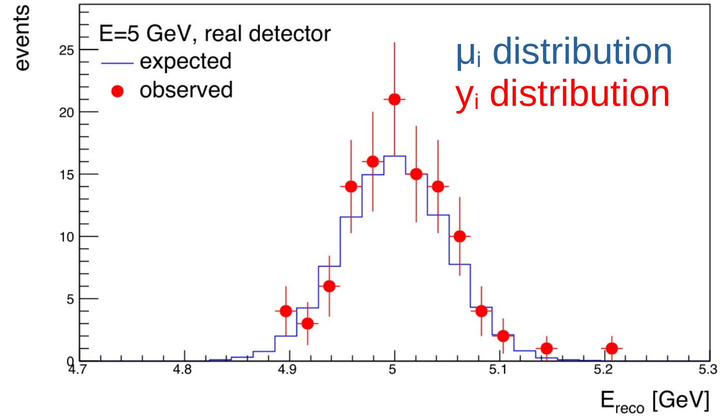
- Detectors have a finite resolution. Example: a calorimeter (H1 detector)



Its energy resolution is $\sigma_E/E = 11\%/\sqrt{E/\text{GeV}} \oplus 1\%$ for leptons [73] and

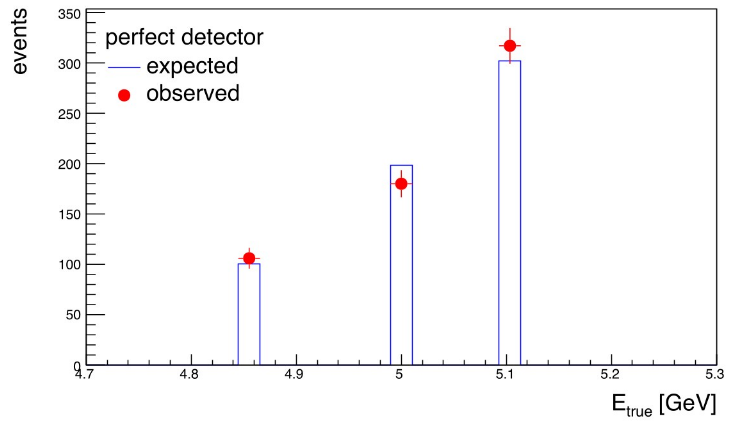
- Example: monoenergetic particle
  - Perfect detector: all events in a single bin, but number of events has statistical uncertainty
  - Real detector: events are spread over several bins. Event counts fluctuate around the expectation

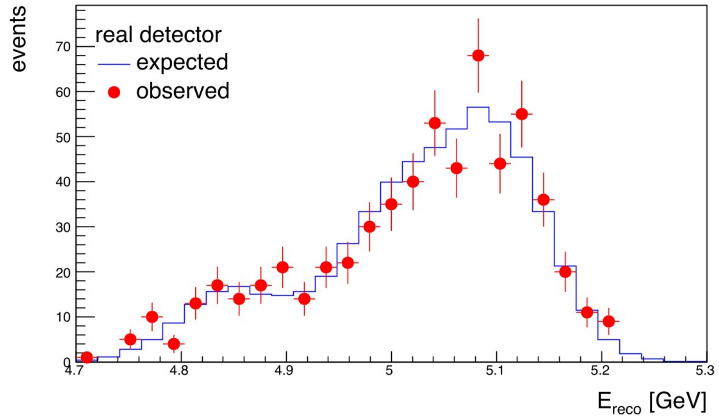




Detector effects (2)

- Again the calorimeter, but now there are three different energies, produced at different probabilities each
- The detector is mixing up all the bins
- In this example it could difficult to decide whether the “truth” energy spectrum really consisted of three lines or not





The folding equation

Folding equation

$$y_i \sim \mu_i = \sum_j A_{ij} x_j + b_i$$

y_i : observation

μ_i : expected measurement (unknown)

A_{ij} : matrix of probabilities

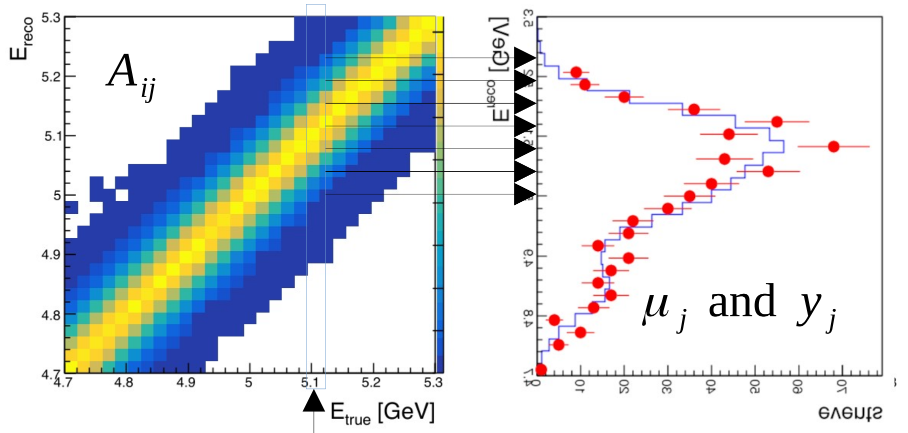
x_j : truth (unknown)

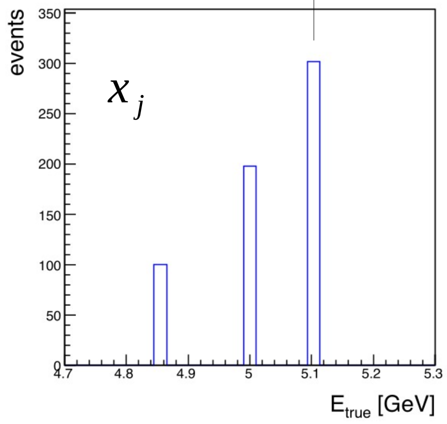
b_i : background

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

matrix notation (matrices, vectors in **bold**)

$$\boldsymbol{y} \sim \boldsymbol{\mu} = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{b}$$



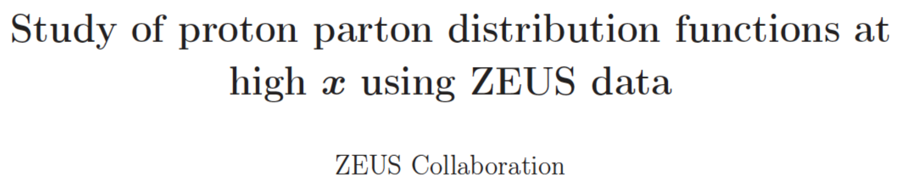


This example:
no background.

“Forward” folding

- Experimentalist can publish the measured **y-b**, the matrix **A** and the uncertainties of **y-b**
- Models predict **x**
- Test a theory: compare **y-b** to **Ax**$^{\text{model}}$
- Disadvantages:
  - theorists to deal with detectors
  - Can not compare a single bin x_i, only the full spectrum **x**
  - Anyway, this lecture is about ***un***folding, not *forward* folding

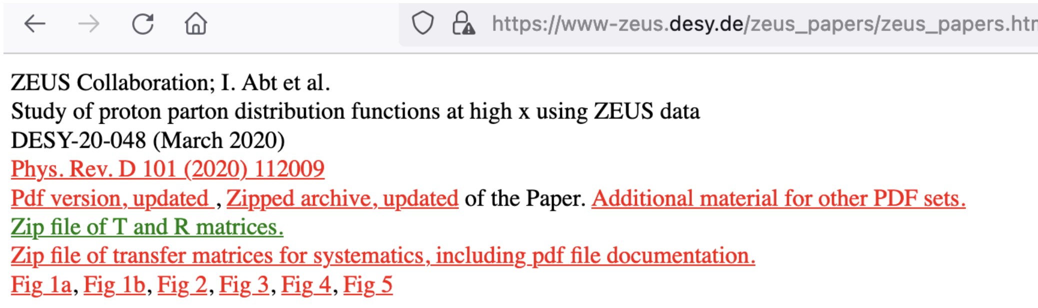
- Example paper: arXiv:2003.08742



Study of proton parton distribution functions at high x using ZEUS data

ZEUS Collaboration

- The relevant matrices are published, e.g. on their web-site:



https://www-zeus.desy.de/zeus_papers/zeus_papers.ht

ZEUS Collaboration; I. Abt et al.
Study of proton parton distribution functions at high x using ZEUS data
DESY-20-048 (March 2020)
Phys. Rev. D 101 (2020) 112009
Pdf version, updated , Zipped archive, updated of the Paper. Additional material for other PDF sets.
Zip file of T and R matrices.
Zip file of transfer matrices for systematics, including pdf file documentation.
Fig 1a, Fig 1b, Fig 2, Fig 3, Fig 4, Fig 5

Recall some basics about statistics

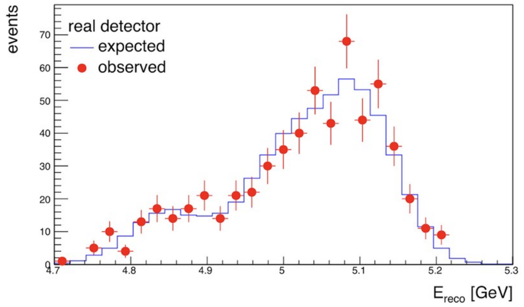
Probability density, random variables

- Probability distribution f(**y**)

  → quantifies probability to observe the data **y** (**y**: vector, dimension N)

  → f also depends on unknown parameters **x** (**x**: vector of dim M)
- Random variable **z**: a function depending on possible observations **y**
- Expectation value: $E(\boldsymbol{z}(\boldsymbol{y}))=\int \boldsymbol{z}(\boldsymbol{y}) f(\boldsymbol{y})\,d\,\boldsymbol{y}$
- Variance: $\mathrm{Var}(z)=\mathrm{E}((z-\mathrm{E}(z))^2)$
- “uncertainty”: $\sigma_z=\sqrt{\mathrm{Var}(z)}$
- Covariance: $\mathrm{Cov}(a,b)=\mathrm{E}((a-\mathrm{E}(a))(b-\mathrm{E}(b)))$
- Correlation coefficient: $\rho_{ij}=\frac{\mathrm{Cov}(a,b)}{\sigma_a\sigma_b}$

- Example: count events in N (mutually exclusive) bins



Poisson distribution:

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

Expectation value:

$$E(y_i)=\mu_i$$

Variance:

$$\mathrm{Var}(y_i)=\mu_i$$

independent bins:

$$\mathrm{Cov}(y_i,y_j)=\delta_{ij}\mu_i$$

- Event count → **y**
- Number of bins → N
- Parameters **μ**
- Large sample limit: Gaussian

  Gaussian mean (unknown): μ_i

  Gaussian variance (from data): $\sigma_i^2\sim y_i$

Likelihood function, estimators

- Likelihood function: the probability density, evaluated for a fixed observation $\mathbf{y}^{\text{data}}$ (it still depends on the unknown parameters $\mathbf{x}$)

$$L(\boldsymbol{x}) = f(\boldsymbol{y}^{\text{data}}; \boldsymbol{x})$$

- Parameter estimation: define an algorithm (a function) to estimate $\mathbf{x}$ from the observation $\mathbf{y}$ $\hat{\boldsymbol{x}}(\boldsymbol{y})$
- Example: maximum-likelihood fit

$$\left.\frac{\partial L}{\partial \boldsymbol{x}}\right|_{\hat{x}} = 0$$

- Bias of an estimator: $\beta(\hat{\mathbf{x}}) = E(\hat{\mathbf{x}} - \mathbf{x}^{\text{true}})$
- Unbiased estimator: $\beta(\hat{\mathbf{x}}) = 0$
- Well-known example: the maximum-likelihood estimator is unbiased
- Difficulty for practical applications: the parameters $\mathbf{x}^{\text{true}}$ are not known!

  → the “true” bias can not be calculated

Bootstrap and Toy experiments

- Toy and bootstrap are techniques used to estimate expectation values (and variances, covariances, etc)

Toy experiment:

- Have a model, with known parameters $\mathbf{x}^{\text{model}}$ and $\mathbf{b}$
- Expectation for $\mathbf{y}$: $\boldsymbol{\mu}=\boldsymbol{A}\boldsymbol{x}^{\text{model}}+\boldsymbol{b}$

Data bootstrap experiment

- Have the observation $\mathbf{y}^{\text{data}}$ (unknown truth $\mathbf{x}^{\text{true}}$)
- Estimate expectation $\boldsymbol{\mu}$: $\boldsymbol{\mu}=\boldsymbol{y}^{\text{data}}$

- Given $\boldsymbol{\mu}$, use pseudo-random numbers, generate new toy data $\mathbf{y}^{\text{toy}}$
- Repeat this N_{toy} times, estimate expectation value of $\mathbf{z}(\mathbf{y})$ as: $E(\boldsymbol{z})\sim\sum_{\text{toy}}\boldsymbol{z}(\boldsymbol{y}^{\text{toy}})/N_{\text{toy}}$

Very powerful tools, but with limitations:
Toy results depend on the given model parameters
Bootstrap results depend on the original data statistical fluctuations

Bootstrap is often used to estimate the result's covariance matrix

Unfolding methods

Unfolding algorithms discussed in this talk

- No tunable parameter
  - Matrix inversion
  - Bin-by-bin “correction”
  - Least-square or Likelihood fit

Part 1 (+exercises)

- With tunable parameter
  - Tikhonov regularisation
  - Iterative methods in general
  - EM Iterative method (“D’Agostini”)

Part 2 (+exercises)

Part3: how to choose the regularisation parameter (+exercises)

Unfolding methods without tunable parameter

Matrix inversion

- Folding equation $\boldsymbol{y} \sim \boldsymbol{\mu} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$
- Why not simply solve the equation for **x**?

$$\hat{\boldsymbol{x}}^{\text{invert}} = \boldsymbol{A}^{-1}(\boldsymbol{y} - \boldsymbol{b})$$

- Also want to know the resulting covariance (“uncertainties”)

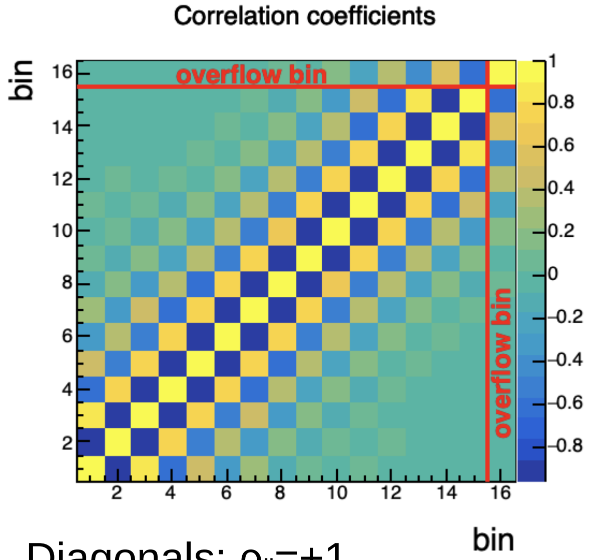
$$\text{Poisson: } \text{Cov}(y_k, y_l) = \text{diag}(\sqrt{y_k})$$

Covariance of $\hat{\boldsymbol{x}}$:

$$\boldsymbol{V}_{\boldsymbol{x}}^{\text{invert}} = \boldsymbol{A}^{-1} \boldsymbol{V}_{\boldsymbol{y}} (\boldsymbol{A}^{-1})^T$$

- Data bins **y** are uncorrelated, but result bins $\mathbf{x}^{\text{invert}}$ are highly (anti-)correlated

Example correlation coefficients



Diagonals: $\rho_{ii}=+1$
For $i \neq j$:
$\rho_{ij}>0$ correlation
$\rho_{ij}<0$ anti-correlation
Values $|\rho_{ij}|>0.8$ are “large”

Least-square, likelihood fit

- Generalisation of matrix inversion: use more bins on vector **y** than **x**
- Idea: using more information (fine **y** bins) will improve the result on **x**
- Ansatz: determine maximum likelihood (minimum of neg. log L)

$$\left.\frac{\partial[-2\log L]}{\partial \boldsymbol{x}}\right|_{\hat{x}}=0$$

L : likelihood function, given the data $\boldsymbol{y}$

- Least-square fit (independent bins, large sample limit): also called χ² fit

$$\chi^2=-2\log L(\boldsymbol{x})=\sum_i\left(\frac{(\boldsymbol{A}\boldsymbol{x})_i+b_i-y_i}{\sigma_i}\right)^2+\text{const}$$

- Least-square solution:

$$\hat{\boldsymbol{x}}=(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A})^{-1}(\boldsymbol{A}^T\boldsymbol{W}(\boldsymbol{y}-\boldsymbol{b}))$$

$$\text{weight matrix } \boldsymbol{W}=\text{diag}\left(\frac{1}{\sigma_i^2}\right)$$

- Covariance of **x**

$$\boldsymbol{V}_{\boldsymbol{x}}=(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A})^{-1}$$

Bin-by-bin unfolding

- Idea: the observed data in bin i are distortions of the corresponding truth, and can be "corrected"

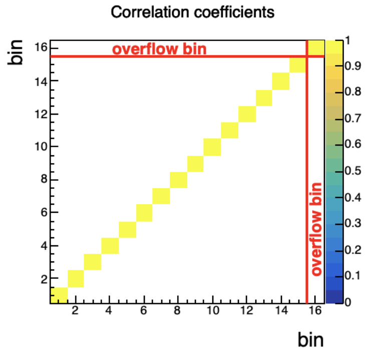
$$\hat{x}_i^{\mathrm{BBB}} = (y_i - b_i) \times f_i$$

correction factor f_i

$$f_i = \frac{x_i^{\mathrm{model}}}{y_i^{\mathrm{model}} - b_i} = \frac{x_i^{\mathrm{model}}}{(A\boldsymbol{x}^{\mathrm{model}})_i}$$

- The correction factors depend on a physics model

Each bin is treated separately, result has no statistical correlations



- How large is the model dependence introduced by this method?
- This is a general question for all unfolding methods
- One way to look at this: the prediction error

The per-bin observed prediction error

- In statistics, “error” is the deviation of an observation from a prediction

  physicists often use “error” in a different meaning: Gaussian width, sqrt(variance)
- Unfolding: estimate **x** from **y** $\hat{\boldsymbol{x}}(\boldsymbol{y})$
- For error, get a prediction of **y** from the folding equation

$$\hat{\boldsymbol{y}}(\boldsymbol{y})=\boldsymbol{A}\hat{\boldsymbol{x}}(\boldsymbol{y})+\boldsymbol{b}$$

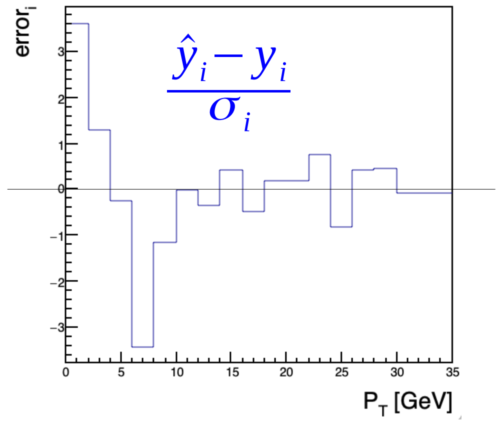
Fold back the unfolded data

- Per-bin prediction error, scaled to “uncertainty”:

$$\widehat{\text{error}}_i=\frac{\hat{y}_i-y_i}{\sigma_i}$$

- Compare the prediction to the data and relate it to the statistical uncertainty
- Deviations from zero of order unity or larger indicate a problem

Example



Notable deviation from zero in the first bins.
Difference much larger than statistical uncertainty → data tell us there is a problem with this unfolding method

The observed prediction error (squared)

- Observed prediction error (squared): condense difference between data and estimator in a single number

$$\widehat{\mathrm{err}} = -2\left[\ln L(\hat{\boldsymbol{y}}) - \ln L(\boldsymbol{y})\right] = \sum_i \left[\frac{y_i - \hat{y}_i}{\sigma_i}\right]^2$$

Sum of squares of the per-bin observed prediction error

- This is a model-independent estimator of the unfolding bias.
- Sometimes this is called “training error”: same “training” data are used to get both $\hat{\boldsymbol{y}}$ and $\widehat{\mathrm{err}}$
- Expectation: $\widehat{\mathrm{err}}$ should be small (close to zero)
- If **y** has more bins than **x**: expect $\widehat{\mathrm{err}}$ to be close to N_y-N_x

Exercises 1-6

Exercises: download and install

- The exercises are done using the ROOT6 framework
- If you plan to work on the exercises, make sure to have ROOT6 installed
- The exercises require some files to be downloaded:

- Download the zip file
- Create a new directory
- Unzip the files in the new directory

https://www.desy.de/~sschmitt/GRK2044/tutorialUnfolding_V2.zip

- Solutions: https://www.desy.de/~sschmitt/GRK2044/tutorialSolutions.zip

Exercises: list of files

- There is one root file with histograms
- There is a library with functions which do not have to be modified (but can be interesting to look at)
- There are macros to get the exercises started
- Each macro will produce some plots
- The macros have to be modified and expanded during the exercises

tutorialIntroduction.pdf : brief documentation

tutorial_inputHisto.root ; histograms
tutorialLibrary.h tutorialLibrary.C : library

tutorialPlotInput.C : show the input data

tutorialOwnUnfoldingExample.C : exercise 1-6

tutorialOwnIteration.C : exercise 7,8
tutorialTikhonovExample.C : exercise 9,10

tutorialScanLCurve.C : exercise 11
tutorialScanSURE.C : exercise 12-14
tutorialFit.C : exercise 15

Exercises: the tutorialLibrary

Definitions are in tutorialLIbrary.h

Classes:

TutorialInput : loads all required input histogram into memory for unfolding
TutorialResult: holds the result of an unfolding algorithm
TutorialUnfoldingAlgorithm : base class to run an unfolding algorithm
TutorialUnfoldTikhonov : Tikhonov unfolding
TutorialIterativeUnfolding: generic iterative unfolding (can select step function)

Auxillary classes:
TutorialUnfoldEMstep : step function for EM iterative unfolding

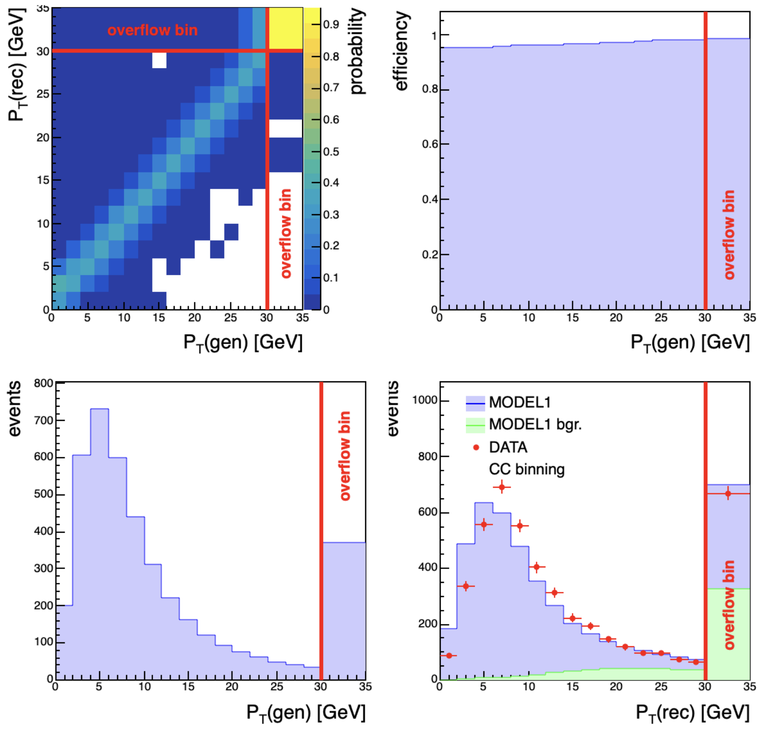
Namespaces:
TutorialPlotter : default plotting functions

Exercises: the input histograms

- The histograms include various types of distributions
- The class TutorialInput loads the required distributions such that they can be accessed by the unfolding algorithms and for plots
- TutorialPlotter::showInputPlots() can be used to visualize a set of input distributions

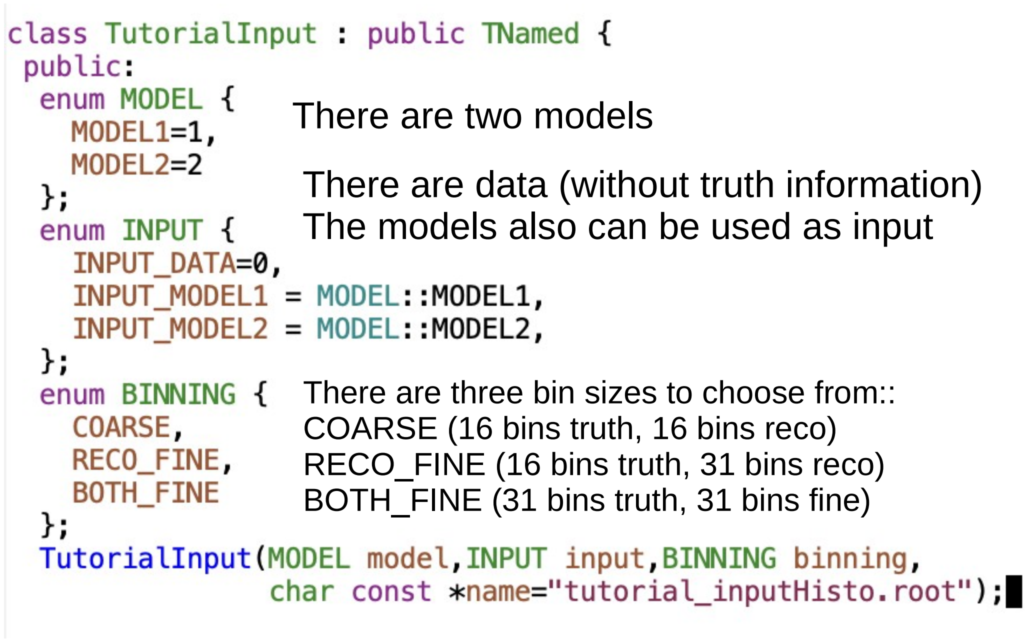
Example macro: in root type these commands:

```
.L tutorialLibrary.C+
.L tutorialPlotInput.C
tutorialPlotInput(1)
```



Exercises: choosing input variants

- When constructing TutorialInput, select the model, the input data for the unfolding, and the binning



```
class TutorialInput : public TNamed {
 public:
  enum MODEL {
    MODEL1=1,
    MODEL2=2
  };
  enum INPUT {
    INPUT_DATA=0,
    INPUT_MODEL1 = MODEL::MODEL1,
    INPUT_MODEL2 = MODEL::MODEL2,
  };
  enum BINNING {
    COARSE,
    RECO_FINE,
    BOTH_FINE
  };
  TutorialInput(MODEL model,INPUT input,BINNING binning,
                char const *name="tutorial_inputHisto.root");
```

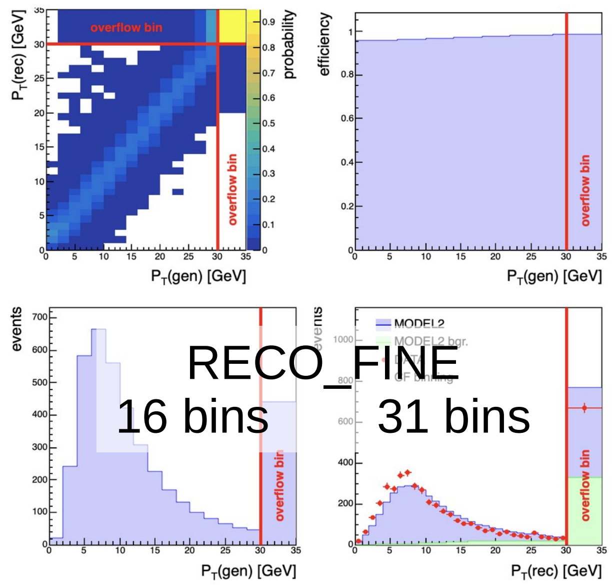
There are two models

There are data (without truth information)
The models also can be used as input

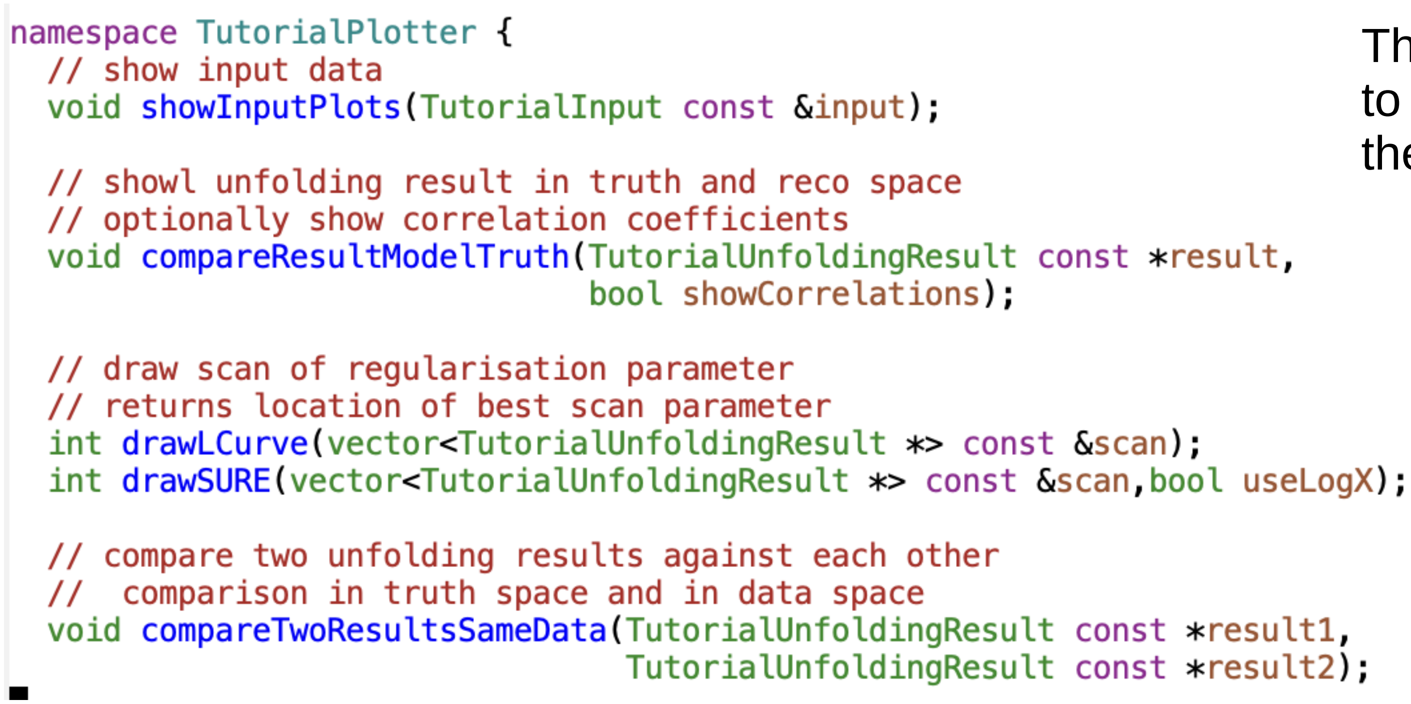
There are three bin sizes to choose from::
COARSE (16 bins truth, 16 bins reco)
RECO_FINE (16 bins truth, 31 bins reco)
BOTH_FINE (31 bins truth, 31 bins fine)

Example: tutorialPlotInput(2)

```
…
  case 2:
    input=new TutorialInput(TutorialInput::MODEL2,
                            TutorialInput::INPUT_DATA,
                            TutorialInput::RECO_FINE);
```



The tutorialPlotter utilities



```
namespace TutorialPlotter {
  // show input data
  void showInputPlots(TutorialInput const &input);

  // showl unfolding result in truth and reco space
  // optionally show correlation coefficients
  void compareResultModelTruth(TutorialUnfoldingResult const *result,
                               bool showCorrelations);

  // draw scan of regularisation parameter
  // returns location of best scan parameter
  int drawLCurve(vector<TutorialUnfoldingResult *> const &scan);
  int drawSURE(vector<TutorialUnfoldingResult *> const &scan,bool useLogX);

  // compare two unfolding results against each other
  //  comparison in truth space and in data space
  void compareTwoResultsSameData(TutorialUnfoldingResult const *result1,
                                 TutorialUnfoldingResult const *result2);
```

These utilities can be used to obtain sets of plots from the classes in the library.

Exercises 1-6

- Exercise 1: run tutorialOwnUnfoldingExample.C (matrix inversion), discuss the result
- Exercise 2: implement the bin-by-bin method, discuss the result
- Exercise 3: repeat (1-3) using MODEL2 for the unfolding, what changes?
- Exercise 4: plot the per-bin prediction error, compare Exercise 1, & 2.
  Hint: use TUnfoldingResult::getYhat(), define a new histogram or graph

$$\hat{x}_i^{\mathrm{BBB}} = (y_i - b_i) \times f_i$$

$$f_i = \frac{x_i^{\mathrm{model}}}{y_i^{\mathrm{model}} - b_i} = \frac{x_i^{\mathrm{model}}}{(A\,\boldsymbol{x}^{\mathrm{model}})_i}$$

$$\widehat{\mathrm{error}}_i = \frac{\hat{y}_i - y_i}{\sigma_i}$$

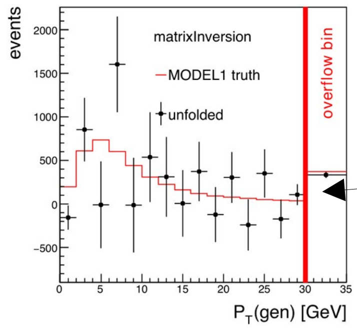
extra exercises, only if there is time left:

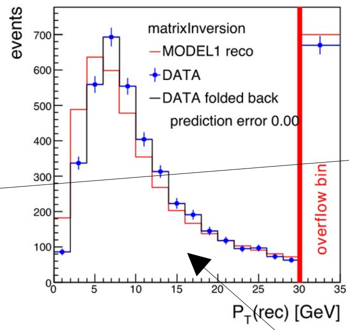
- Exercise 5*: implement the minimum χ².
  Use binning RECO_FINE and unfold the data. Compare to Exercise 1.
- Exercise 6*: repeat the likelihood fit, with MODEL2 and 1000 toy samples. Plot the observed prediction error, compare to χ² distribution (how many degrees of freedom?)

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A})^{-1} (\boldsymbol{A}^T \boldsymbol{W} (\boldsymbol{y} - \boldsymbol{b}))$$

weight matrix $\boldsymbol{W} = \mathrm{diag}\left(\frac{1}{\sigma_i^2}\right)$

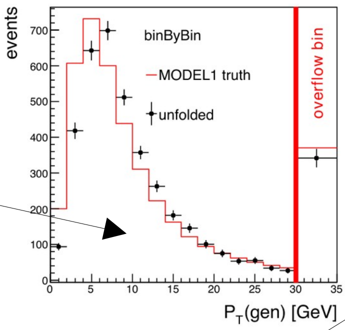
Discussion of exercises 1 and 2

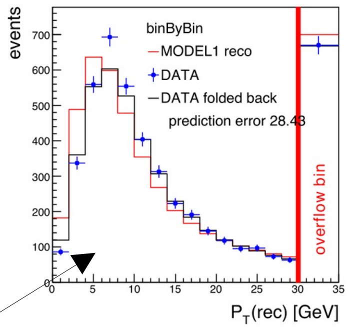


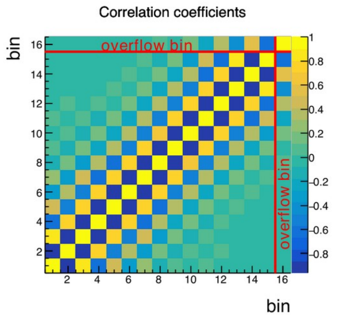


Unfolded data

- Matrix inversion: Wild oscillations
- Bin-by-bin: nice and smooth



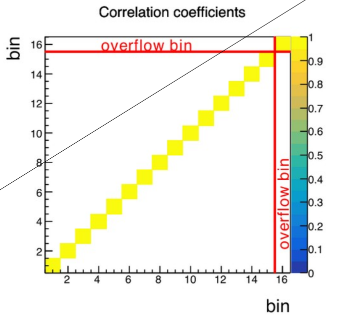




Exercise 1
Matrix inversion

Comparison at reco level

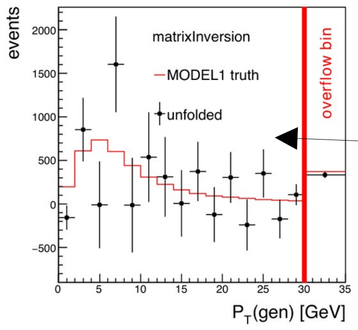
- Matrix inversion: on top of the data
- Bin-by-bin: significant differences to data

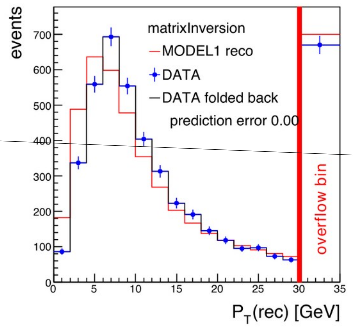


Exercise 2
bin-by-bin

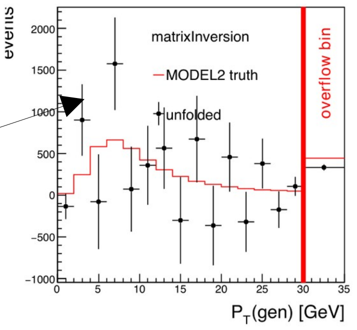
Is Bin-by-bin better (because it looks nicer)?

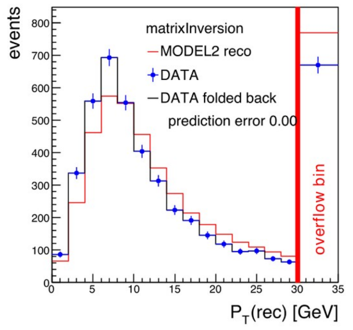
Discussion exercise 3 (matrix inversion)

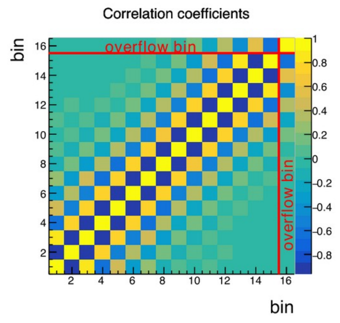




Unfolded data agree well. Only small differences, well hidden in the quoted statistical uncertainties.



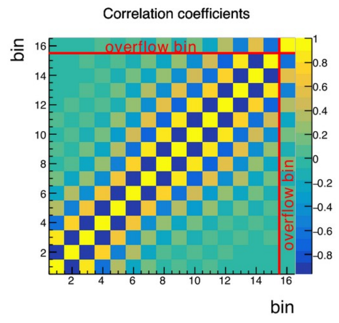




Exercise 1
Matrix inv.
MODEL1

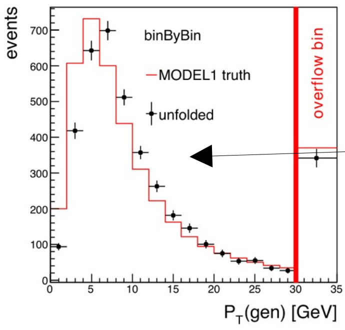
→ negligible model dependence

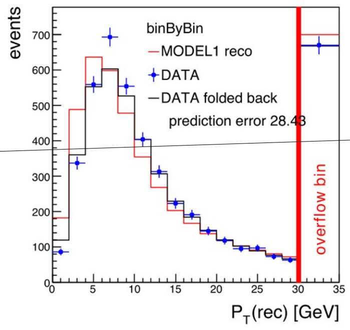
→ large statistical uncertainties, large (anti-)correlations



Exercise 4
Matrix inv.
MODEL2

Discussion exercise 3 (bin by bin)

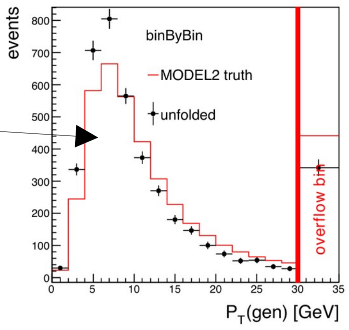


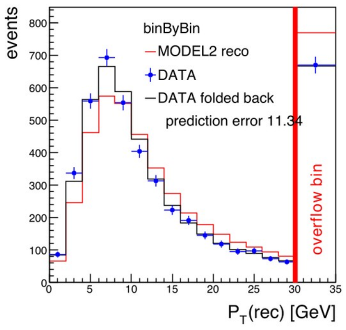


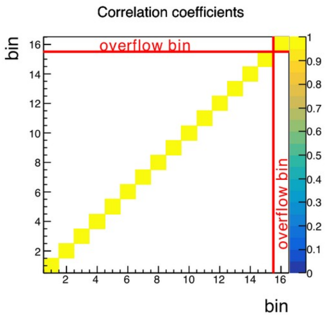
Unfolded data do not agree.

→ strong model dependence

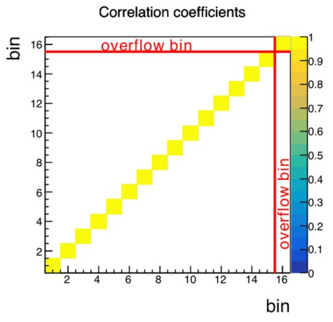
→ small stat. Uncertainties, zero correlations







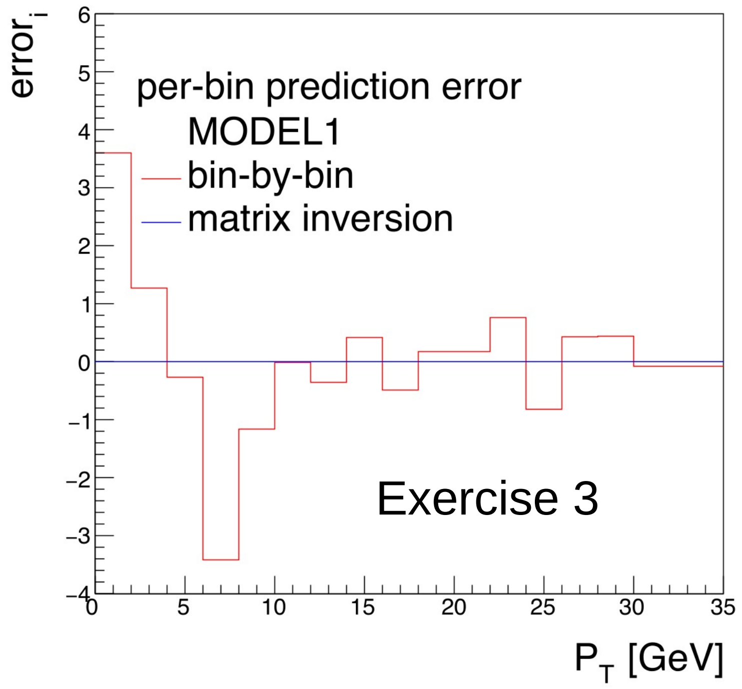
Exercise
Bin by bin
MODEL1



Exercise 4
Bin by bin
MODEL2

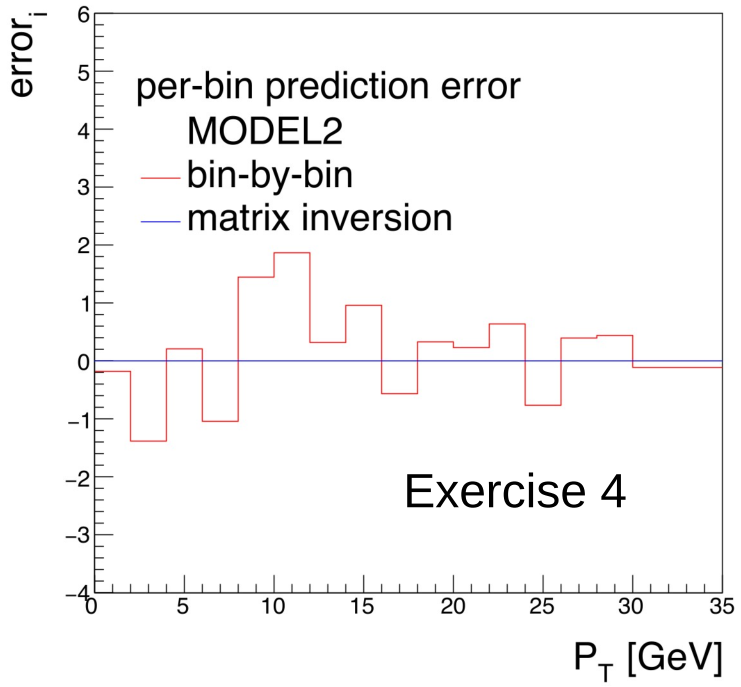
Conclusion: bin-by-bin gives “wrong” results.

Discussion of exercise 4

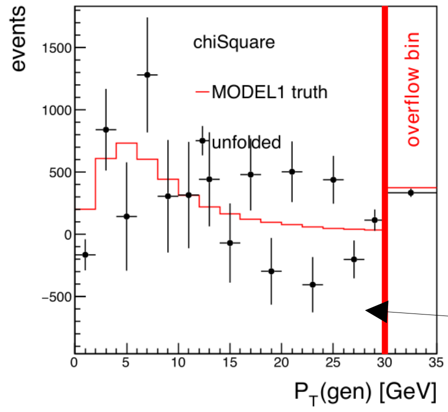


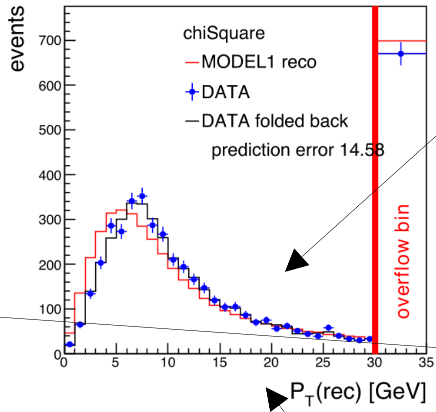
- Per-bin prediction error: very poor performance of the bin-by-bin algorithm

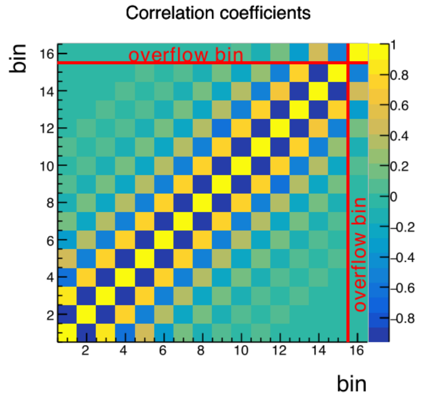
  Different results for different models
- Matrix inversion: zero per-bin prediction error, no model dependence



Discussion exercise 5





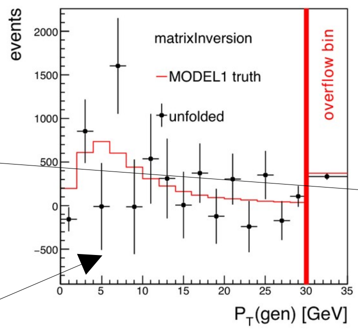


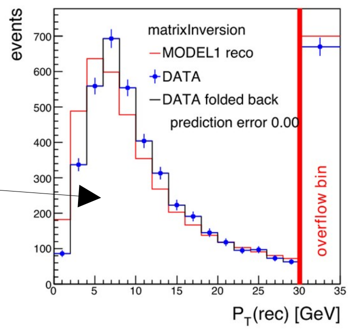
Use 31 reco bins as compared to 16 reco bins

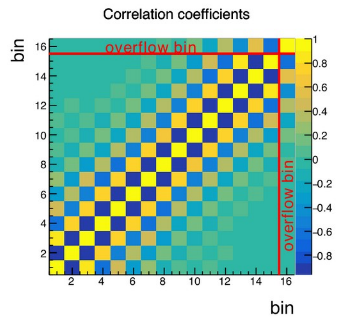
Results are very similar

Exercise 5
Maximum likelihood

Prediction has 16 degrees of freedom but data have 31 bins
→ data fluctuate around the folded-back result

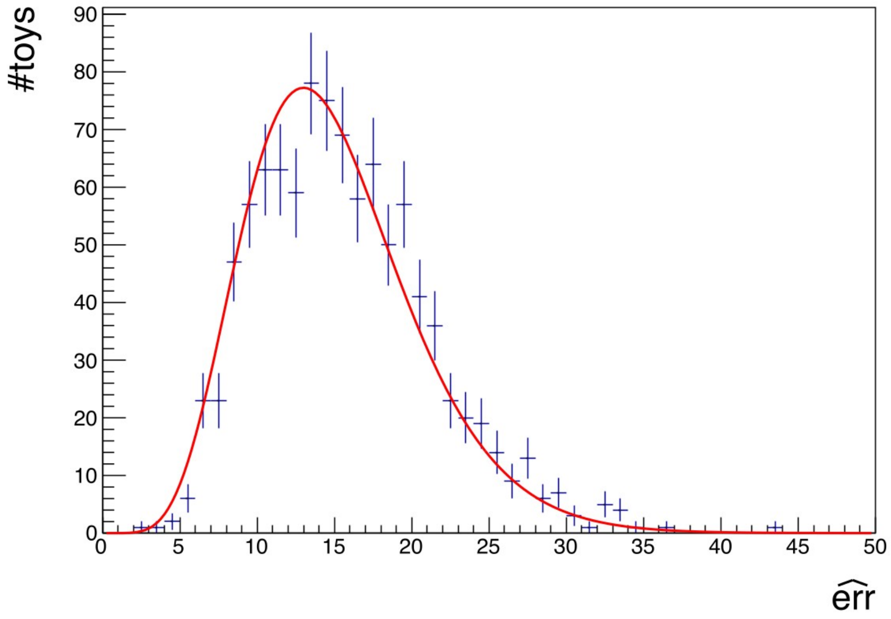






Exercise 1
Matrix inversion

Discussion exercise 6



- Plot of the prediction error follows χ^2 distribution with 15 degrees of freedom
- Expected from 31 data bins minus 16 truth bins (=15 fit parameters)
- Expected for maximum-likelihood fit in the large-sample limit
- unfolding algorithm: accept a somewhat increased prediction error in exchange for much reduced variances

Unfolding methods with tunable parameters

Iterative methods

- Idea: start with a prediction $\hat{\boldsymbol{x}}^{(-1)} = \hat{\boldsymbol{x}}^{\text{model}}$
- Iterative prescription "F" to improve this using the data

$$\hat{\boldsymbol{x}}^{(N+1)} = F\left(\hat{\boldsymbol{x}}^{(N)}, \boldsymbol{y}\right)$$

- Natural choice: an algorithm which converges for N→infinity to the maximum-likelihood solution
- Common choice: EM iterations

$$\hat{x}_j^{(N+1)} = \frac{\hat{x}_j^{(N)}}{\epsilon_j} \sum_i \frac{A_{ij} y_i}{\sum_k A_{ik} \hat{x}_k^{(N)} + b_i}$$

- The EM iterative method has proven properties: for Poisson-distributed data, it converges to the corresponding maximum-likelihood solution
- In Particle physics this is often called "D'Agostini" after his 1995 paper

For small "N" this method gives a solution biased to the prediction. For sufficiently large N, there is hope to have a small bias with still moderate statistical fluctuations.

Question: how many iterations?

Tikhonov regularisation

- Likelihood fit leads to large statistical fluctuations – can these be damped?
- Tikhonov regularisation: add a penalty term to suppress large deviations from a model

s=0: maximum-likelihood. Small, nonzero parameter s: solution without the large variances of the likelihood fit. The bias to the model grows with “s”.

In literature: sometimes s is named τ or τ^2
A more general formulation also includes a matrix L of regularisation conditions.

Minimize:

$$\underbrace{\sum_i \frac{\left(y_i-(\boldsymbol{Ax})_i-b_i\right)^2}{\sigma_i^2}}_{\chi^2 \text{ function}} + \underbrace{s\sum_j \left(x_j - x_j^{\text{model}}\right)^2}_{\text{penalty term}}$$

general regularisation condition:

$$\ldots + s\sum_j \left(\boldsymbol{L}\left(\boldsymbol{x}-\boldsymbol{x}_{\text{model}}\right)\right)_j^2$$

typical choice: curvature matrix

$$\boldsymbol{L}=\begin{pmatrix} 1 & -2 & 1 & 0 & \ldots \\ 0 & 1 & -2 & 1 & \\ \vdots & & & & \ddots \end{pmatrix}$$

Penalty grows with distance from model
Penalty is suppressed by s

Tikhonov eigenvalue analysis (1)

- Write the function F which is to be minimized in matrix form

$$F(\boldsymbol{x}) = (\boldsymbol{y} - \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})^T W (\boldsymbol{y} - \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) + s|\boldsymbol{x} - \boldsymbol{x}^{\text{model}}|^2$$

$\boldsymbol{W}$: (symmetric) weight matrix

- In our case, W is diagonal

$$\boldsymbol{W} = \text{diag}(1/\sigma_i^2)$$

- In general, W may include off-diagonal elements. However, it has to be symmetric and positive (all Eigenvalues >0).

Tikhonov eigenvalue analysis (2)

$$F(\boldsymbol{x}) = (\boldsymbol{y}-\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x})^T W (\boldsymbol{y}-\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}) + s|\boldsymbol{x}-\boldsymbol{x}^{\text{model}}|^2$$

- Diagonalize W

$$\boldsymbol{W} = \boldsymbol{O}_W \boldsymbol{D}_W \boldsymbol{O}_W^T$$

$\boldsymbol{O}_W$ is orthogonal, $\boldsymbol{D}_W$ is diagonal with $D_{W,ii} > 0$

$$F(\boldsymbol{x}) = |\sqrt{\boldsymbol{D}_W}\boldsymbol{O}_W^T(\boldsymbol{y}-\boldsymbol{b}) - \sqrt{\boldsymbol{D}_W}\boldsymbol{O}_W^T\boldsymbol{A}\,\boldsymbol{x}|^2 + s|\boldsymbol{x}-\boldsymbol{x}^{\text{model}}|^2$$

- Singular Value Decomposition of $\sqrt{\boldsymbol{D}_W}\boldsymbol{O}_W^T\boldsymbol{A}$

$$\sqrt{\boldsymbol{D}_W}\boldsymbol{O}_W^T\boldsymbol{A} = \boldsymbol{O}_1\boldsymbol{D}\boldsymbol{O}_2^T$$

$\boldsymbol{O}_1$ and $\boldsymbol{O}_2$ are ortogonal

$\boldsymbol{D}$ is rectangular and diagonal, $D_{ii} > 0$

$$F(\boldsymbol{x}) = |\boldsymbol{O}_1^T\sqrt{\boldsymbol{D}_W}\boldsymbol{O}_W^T(\boldsymbol{y}-\boldsymbol{b}) - \boldsymbol{D}\boldsymbol{O}_2^T\boldsymbol{x}|^2 + s|\boldsymbol{x}-\boldsymbol{x}^{\text{model}}|^2$$

- Substitute variables

$$\boldsymbol{z} = \boldsymbol{O}_1^T\sqrt{\boldsymbol{D}_W}\boldsymbol{O}_W^T(\boldsymbol{y}-\boldsymbol{b})$$

$$\boldsymbol{t} = \boldsymbol{O}_2^T\boldsymbol{x} \text{ and } \boldsymbol{t}^{\text{model}} = \boldsymbol{O}_2^T\boldsymbol{x}^{\text{model}}$$

- New Function to minimize:

$$F(\boldsymbol{t}) = |\boldsymbol{z}-\boldsymbol{D}\boldsymbol{t}|^2 + |\boldsymbol{t}-\boldsymbol{t}^{\text{model}}|^2$$

- Matrix **D** is diagonal

$$\boxed{\hat{t}_j = \frac{D_j z_j + s\ t_j^{\text{model}}}{D_j^2 + s}}$$

Tikhonov eigenvalue analysis summary

- The variables are transformed

$$\boldsymbol{z} = \boldsymbol{O}_1^T \sqrt{\boldsymbol{D}_W} \boldsymbol{O}_W^T (\boldsymbol{y} - \boldsymbol{b})$$

$$\boldsymbol{t} = \boldsymbol{O}_2^T \boldsymbol{x} \text{ and } \boldsymbol{t}^{\text{model}} = \boldsymbol{O}_2^T \boldsymbol{x}^{\text{model}}$$

- The transformed data **z** are often called “modes”. Resulting **t**:

$$\hat{t}_j = \frac{D_j z_j + s \ t_j^{\text{model}}}{D_j^2 + s}$$

for $s \rightarrow 0$

$$\hat{t}_j\Big|_{s=0} = \frac{z_j}{D_j}$$

for $s \rightarrow \infty$:

$$\hat{t}_j\Big|_{s=\infty} = t_j^{\text{model}}$$

- The transfomed data z_i have variance=1 ($\sigma_{z,i}$=1)
- Maximum-likelihood (s=0): the unfolded t_j are equal to z_j amplified by $1/D_j$
- Regularisation s>0 suppresses modes with $D_j^2 < s$
- Eigenvalue analysis: look at z_i and D_j, find a good compromize for s

The “SVD unfolding” uses a similar eigenvalue analysis, yet with non-diagonal regularisation pattern L

Exercises 7-10

Exercises 7-10

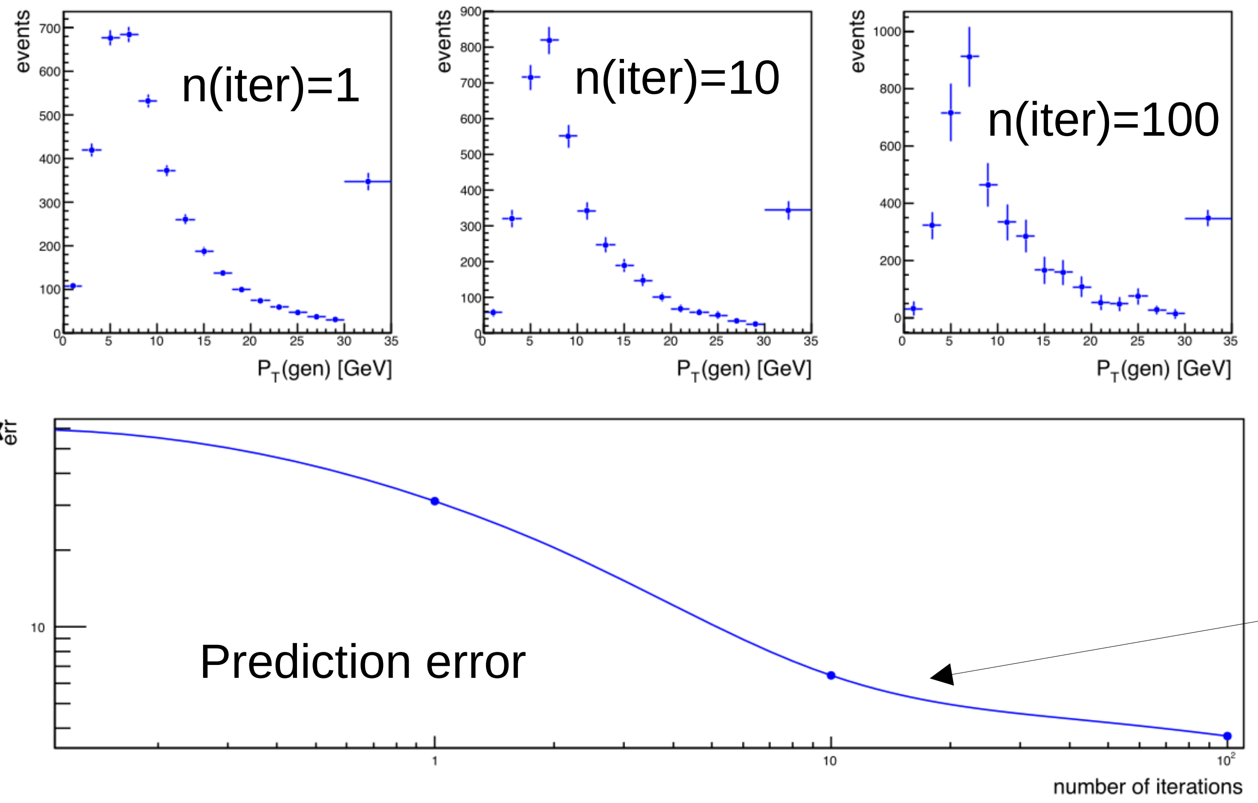
- Exercise 7: try tutorialOwnIteration.C Look at the unfolding result for various choices of the number of iterations (0..100). Plot observed prediction error as a function of n(iter)

  hint: TutorialUnfoldingResult::getPredictionError()

- Exercise 8: modify tutorialOwnIteration.C use the bin-by-bin method as step function. Repeat exercise 7 (with max. iterations=20)
- Exercise 9: try tutorialTikhonovExample.C Look at the unfolding result for various choices of s. Plot the observed prediction error as a function of s
- Exercise 10: modify tutorialTikhonovExample.C to plot the vector z and the eigenvalues D. Why are the dimension of z and D different? What could be a good choice of s, such that insignificant modes z_i are suppressed?

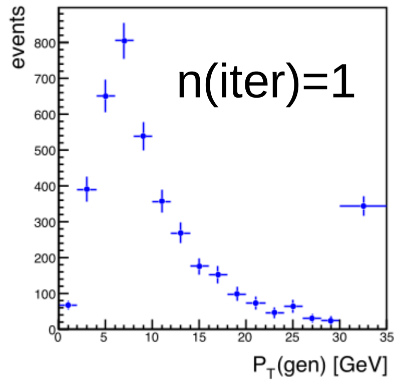
  Hint: use the methods TutorialUnfoldTikhonov::getEigenValues() and TutorialUnfoldTikhonov::getZ()

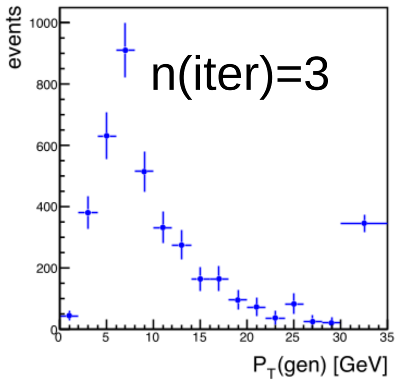
Exercise 7 discussion

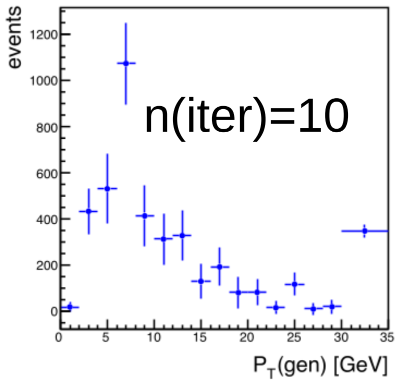


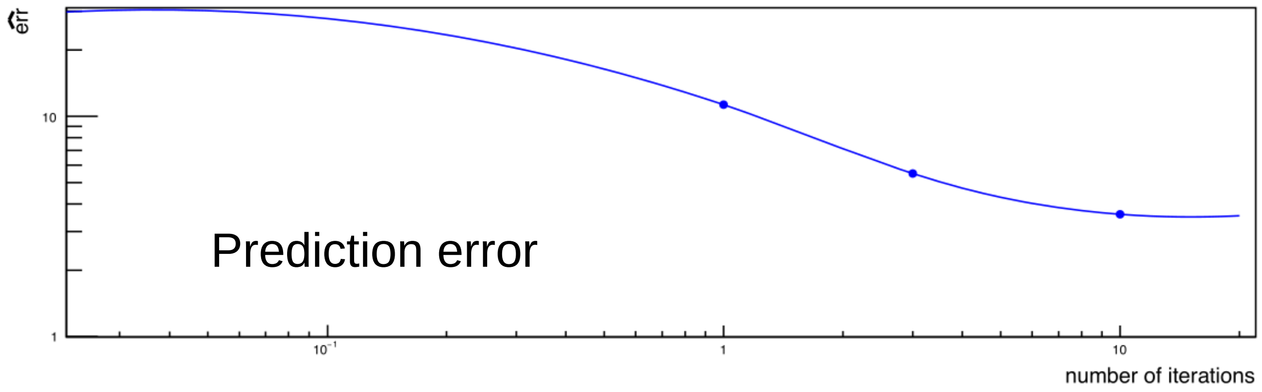
- Classical EM interations ("D'Agostini")
- Prediction error starts off high (large bias to model)
- Perhaps the point where it flattens out is a good choice?

Exercise 8 discussion



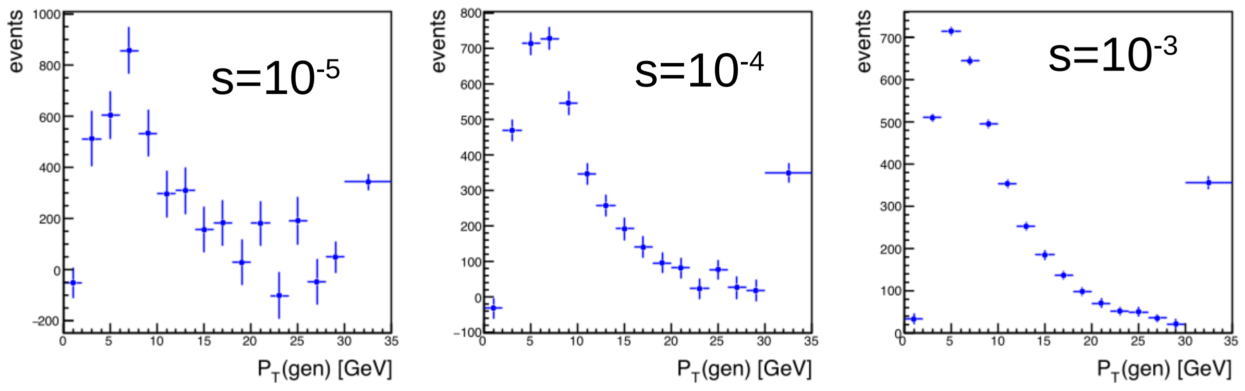


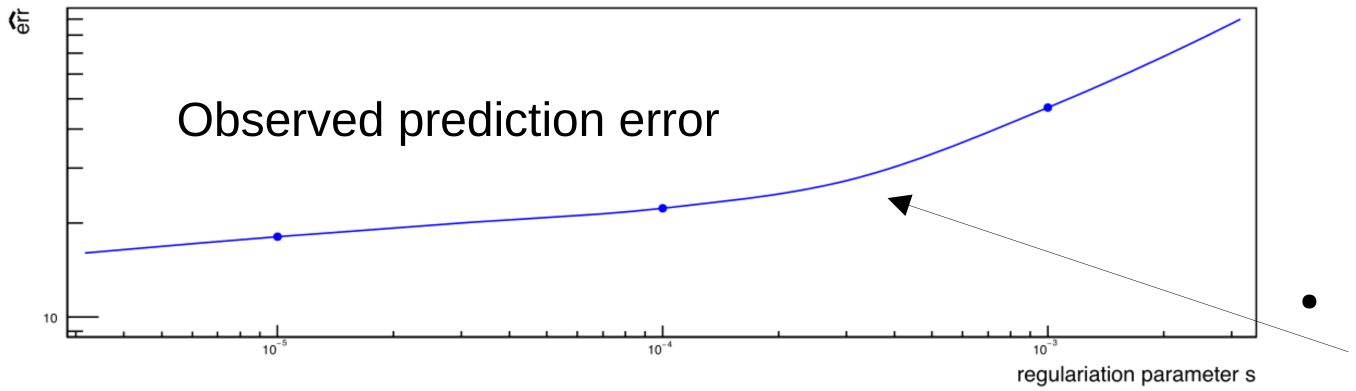




- Our “iterative bin-by-bin” behaves similar to the standard EM iterations: prediction error falls of with #iterations
- Seems to converge faster → less fine-tuning possible

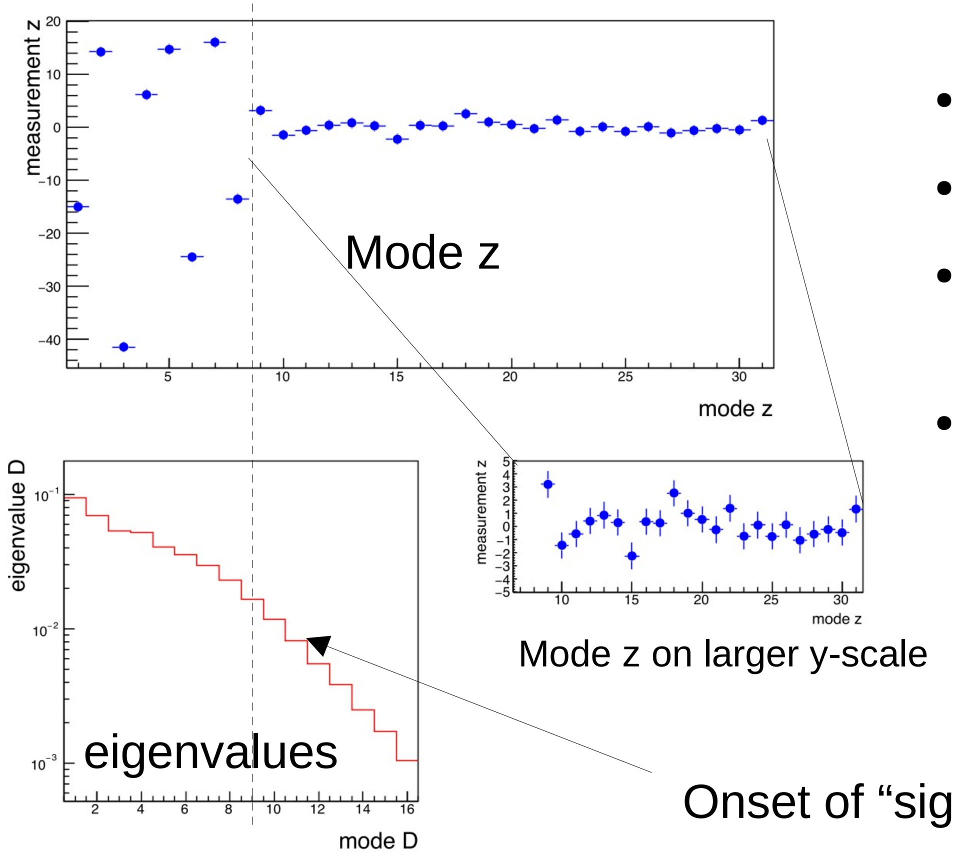
Exercise 9 discussion





- Tikhonov is “opposite” to “iterative”
  - Iterative starts with model, approaches non-regularized solution
  - Tikhonov starts with maximum-likelihood, approaches model with growing s
- Possible choice of s: kink of the curve?

Exercise 10 discussion



Mode z on larger y-scale

Onset of “significant” measurements z

- Dimension of z=31 (31 reco bins)
- 16 eigenvalues (16 truth bins)
- Modes z_j with j>16 do not contribute to the solution t
- First ~9 modes of z are significant, others fluctuate around zero → suppress these by choosing s large enough

→ Good choice of s is near the ninth Eigenvalue, $s \sim D_9^2 = 0.000138$

Choosing regularisation parameters

The L-curve method

- For Tikhonov method: can do eigenvalue analysis. Involves a choice on “which z is significant”
- Another method for Tikhonov regularisation: the L-curve method
- Tikhonov minimizes this function

$$\underbrace{\sum_i \frac{(y_i-(A\mathbf{x}+\mathbf{b})_i)^2}{\sigma_i^2}}_{L_x} + s \underbrace{\sum_j (x_j-(x_{\text{model}})_j)^2}_{L_y}$$

L_x: observed prediction error

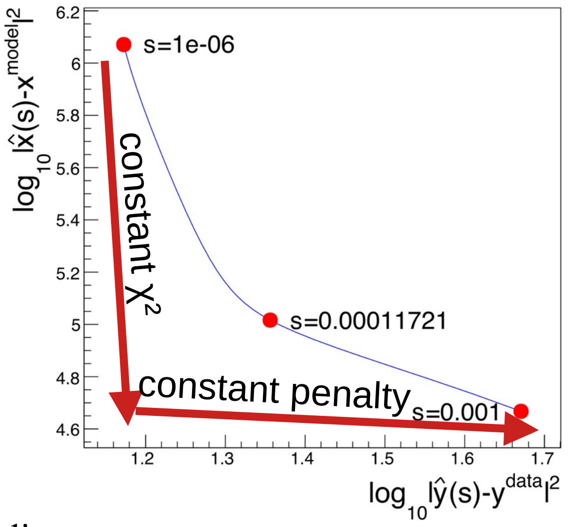
L_y: penalty term (excluding s)

- Parametric plot of

$$\log_{10} L_x(s) \text{ wrt } \log_{10} L_y(s)$$

is shaped like the letter L

- Select the geometrical “kink”



Geometrical curvature: inverse of radius

Note the different scales on x and y axis

The SURE method

- General data-driven method to select “best” parametric model: minimise SURE

  Stein’s Unbiased Risk Estimate

- Observed Prediction error

$$\widehat{\text{err}} = \sum_i \left[\frac{y_i - \hat{y}_i}{\sigma_i} \right]^2$$

- Effective number of degrees of freedom

$$\widehat{\text{DF}} = \sum_i \left[\frac{\partial \hat{y}_i}{\partial y_i} \right]$$

- SURE $\text{SURE} = \widehat{\text{err}} + 2\,\widehat{\text{DF}}$
- SURE probes the “true” prediction error

$$E(\text{SURE}) = E\left(\sum_i \left[\frac{\mu_i^{\text{true}} - \hat{y}_i}{\sigma_i} \right]^2 \right)$$

The expectation value of SURE is equal to the true squared error.

→ minimizing SURE is trying to minimize the “true” prediction error
The “true” prediction error is unknown. But SURE can estimate it from the data (model-independent)

Tikhonov number of degrees of freedom

- For Tikhonov, the variable DF can be expressed by the Eigenvalues

$$\widehat{\mathrm{DF}} = \sum_i \left[\frac{\partial \hat{z}_i}{\partial z_i}\right] = \sum_i \frac{D_i^2}{D_i^2 + s}$$

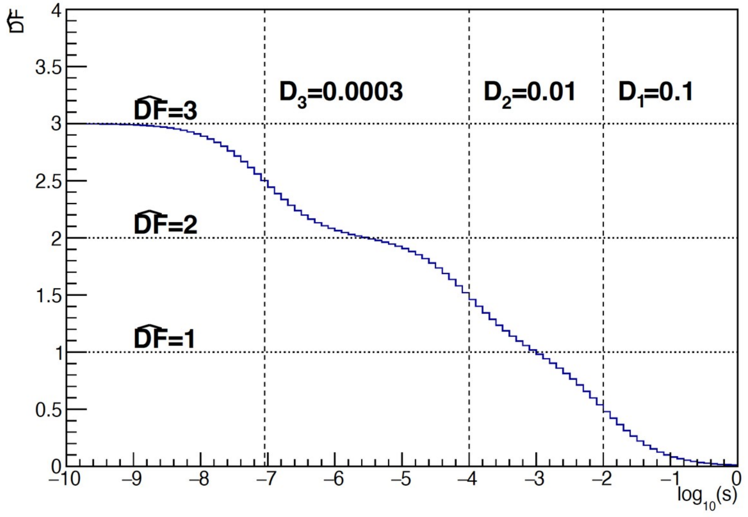
- It only depends on the Eigenvalues, not on the data z

s=0:

$$\widehat{\mathrm{DF}}\Big|_{s=0} = N_x$$

number of $\boldsymbol{x}$ bins

- For s>0: DF corresponds to the number of modes which contribute to the result (modes with $D_i^2 < s$ are suppressed)



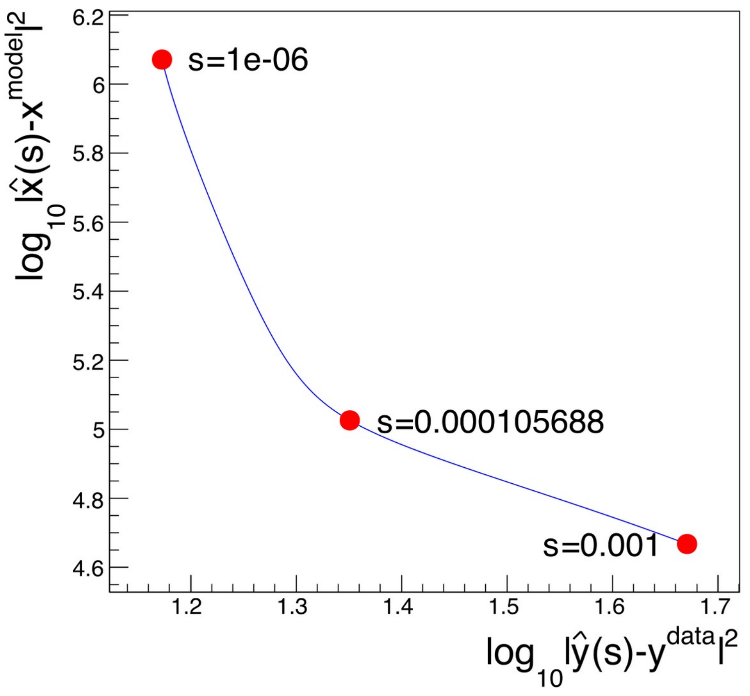
Example: 3 EV
If the EVs are well separated, there are steps in the DF function, by one unit at each EV

Exercise 11-15

Exercise 11-15

- Exercise 11: try out the L-curve scan tutorialScanLCurve.C. Best s?
- Exercise 12: try out the SURE minimisation tutorialScanSURE.C. Best s?
- Exercise 13a: modify tutorialScanSURE.C to apply the SURE scan to the EM iterative method. What is the best number of iterations?
- Exercise 13b repeat 13a with COARSE binning
- Exercise 14: modify tutorialScanSURE.C to apply the SURE scan to the bin-by-bin iterative method. What is the best number of iterations?
- Exercise 15: try/modify the macro tutorialFit.C to fit the respective optimized unfolding result with a function. Compare the results to the “data” truth: peak=6 GeV, width=1.8 GeV

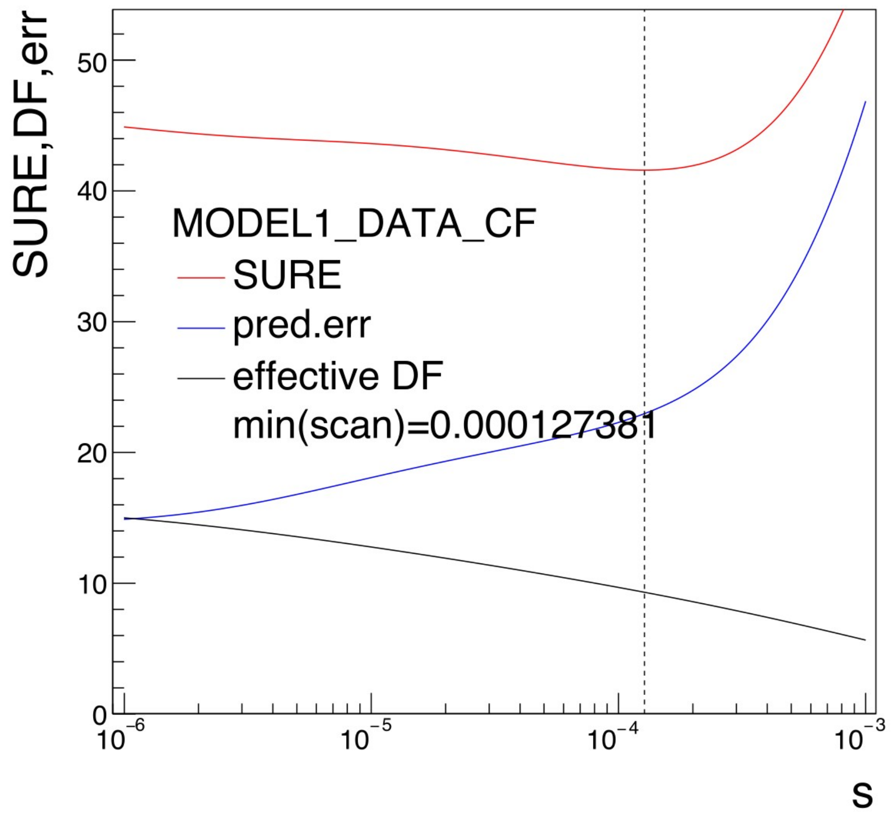
Exercise 11 discussion



- The algorithm puts a point at the “kink” position
- No interpolation is done.
- Best choice of s from l-curve

  s=0.000106

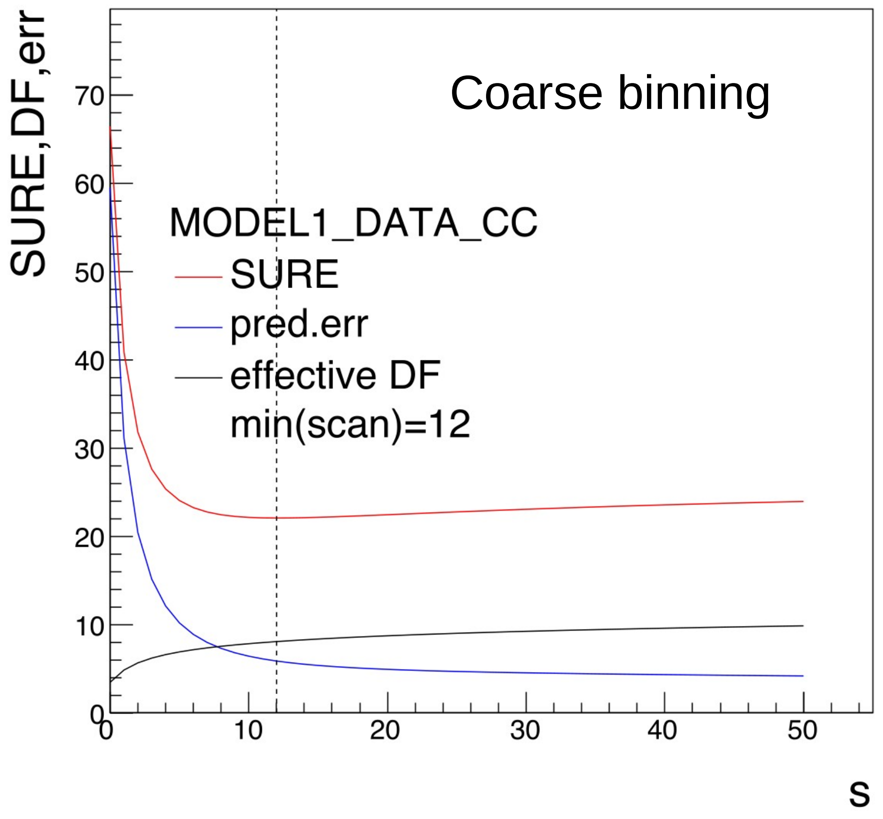
Exercise 12 discusison



- SURE at work:
  - Prediction error increases with s
  - Effective DF decreases with s
  - SURE has a minimum
- Again, no interpolation is done
- Best s from SURE:

  s=0.000127

Exercise 13 a+b discussion

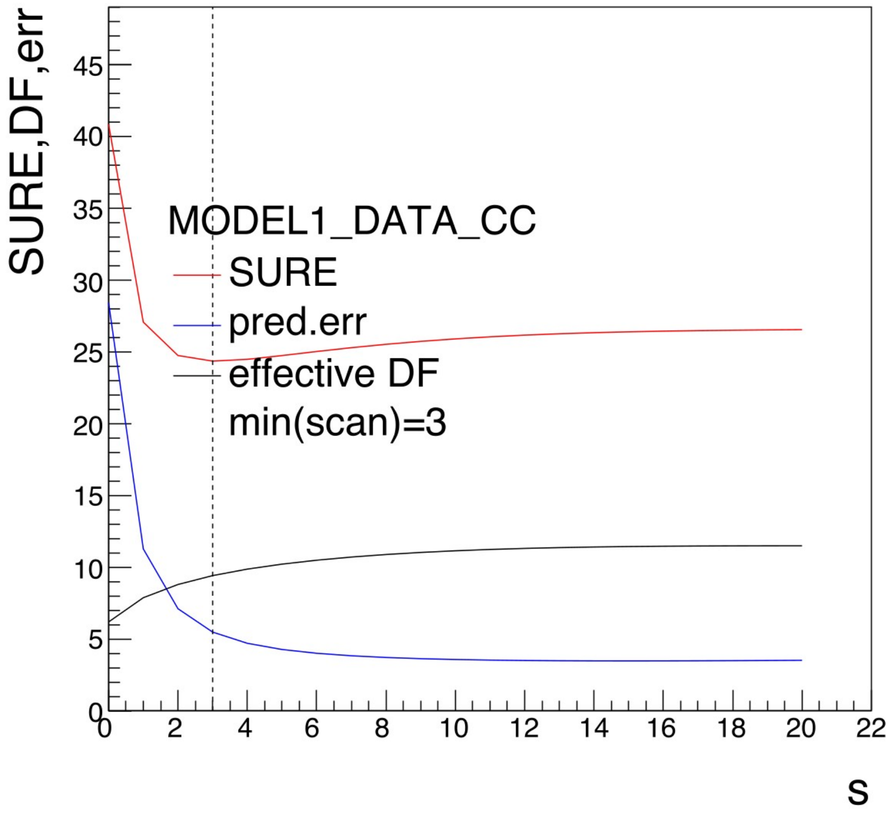


- Iterative method starts with large prediction error and small DF
- DF increases with the number of iterations, pred. error decreases
- Similar result for RECO_FINE binning
- Minimum SURE reached for

  nIter=12 (coarse binning)

  nIter=10 (fine binning)

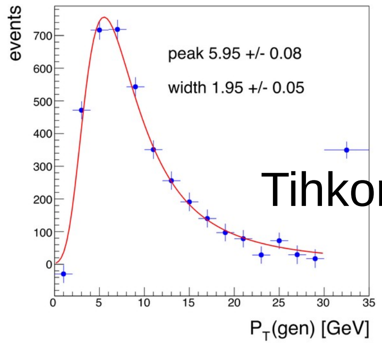
Exercise 14 discussion

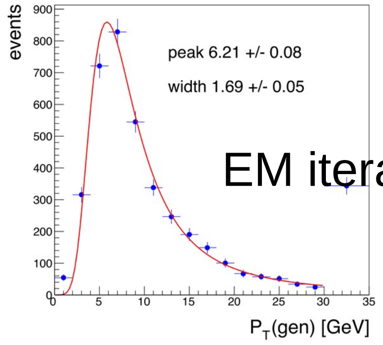


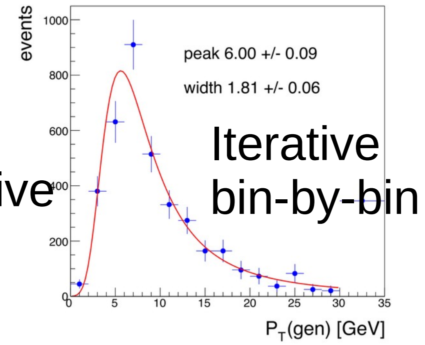
- Similar to EM iterations, but minimum is reached faster
- Minimum SURE reached at

  nIter=3
- This method would be difficult to fine-tune (large change from iteration to iteration)

Exercise 15 discussion







- SURE results are not too far apart from each other
- Fit parameters vary a bit
- Summary table shows that bin-by-bin has largest bias: width comes out is far to large

algorithm	peak	width
data truth	6	1.8
maximum L	6.06+/-0.09	1.75+/-0.07
Bin-by-bin	5.91+/-0.08	2.08+/-0.05
Tikhonov SURE	5.95+/-0.08	1.95+/-0.05
EM iterative SURE [C]	6.21+/-0.08	1.69+/-0.05
EM iterative SURE [F]	6.20+/-0.08	1.69+/-0.05
Iterative BBB SURE	6.00+/-0.09	1.81+/-0.06

Some practical hints for unfolding

Determining the matrix of probabilities

- In particle physics, the matrix A often is determined using Monte Carlo (MC) simulations
- Simulations use:
  - Models for the unknown process (to be measured)
  - Models for hadronisation and QCD
  - Models for the detector response
- Alternatives for less complex setups: measure response for known test data, use known response function, etc.

- MC method: draw events according to some high-dimensional probability distribution
- Simulates a large number of events
- For each simulated event, the truth bin j and the observed bin i are known

count events in $j : x_j^{\mathrm{MC}}$

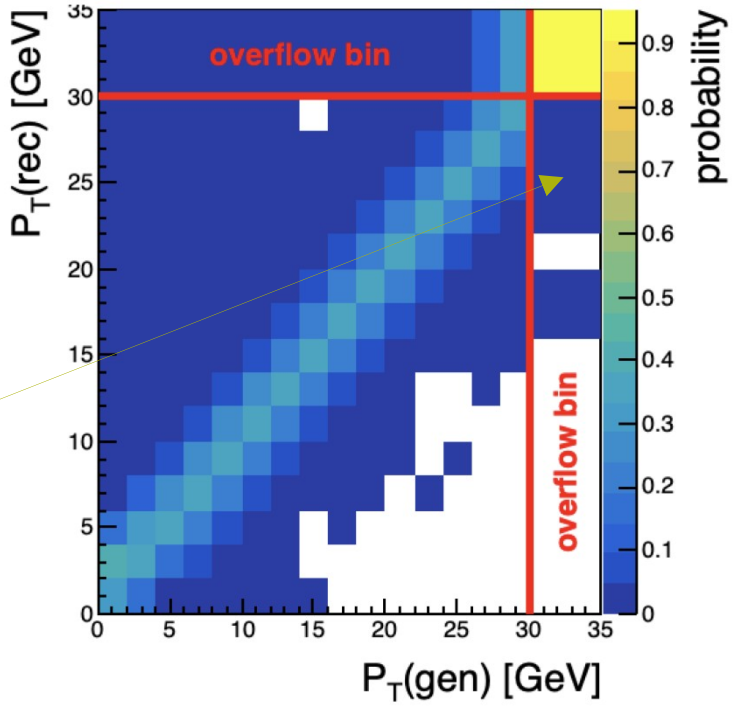
count events in $i : y_i^{\mathrm{MC}}$

count events in both i and $j : N_{ij}^{\mathrm{MC}}$

matrix of probabilities: $A_{ij} = \frac{N_{ij}^{\mathrm{MC}}}{x_j^{\mathrm{MC}}}$

Common pitfalls with constructing the matrix

- The matrix **A** and the background **b** together describe the expected event count
- Make sure all events are counted "somewhere"
- A frequent mistake:
  - Underflow and overflow bins on truth level are often neglected. But they may migrate into the observed sample
  - Always make sure the underflow and overflow are included OR are accounted for with the background ("fakes")



Unfolding and phase-space boundaries

- Many analyses have complicated phase-space cuts
  - Example: jet minimum transverse momentum and angular range, number of jets, etc
- Each phase-space cut is connected with migrations
- Truth events below these cuts may migrate into the sample (fakes)

- Two options
  - Fixed prediction for fakes: subtract as background.

    Introduces a model dependence
  - Use extra normalisation bins to determine fakes from data

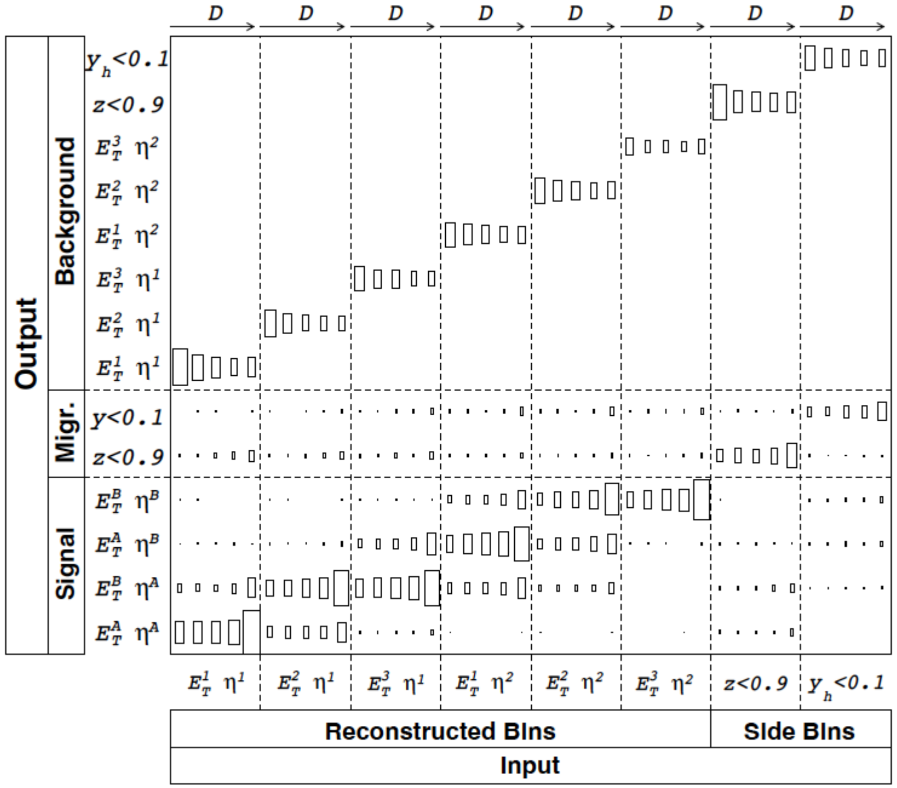
    Requires extra “reco” bins (control regions) to be included in the unfolding

Unfolding of multi-dimensional distributions

- For unfolding multi-dimensional distributions, map the 2D bins on a 1D histogram
- In many cases the number of reco bins is different from truth bins

  → two mappings, for truth and for reco

  → mapping also may require extra bins to account for fakes and control regions
- A complex example is shown to the right



Common unfolding tools

- In this lecture: we have used our own unfolding tools
- Not so difficult after all – and we knew exactly what we are doing
- Otherwise : can use unfolding packages. With ROOT for example:
  - RooUnfold

    https://hepunx.rl.ac.uk/~adye/software/unfold/RooUnfold.html
  - TUnfold

    https://www.desy.de/~sschmitt/tunfold.html
- Make sure the package is doing what you expect it to do! If unsure, read the code!

Iterative method – background subtraction

- Background subtraction
  - EM iterative method: must not subtract background from data (instead add it in the denominator)

$$\hat{x}_j^{(N+1)} = \frac{\hat{x}_j^{(N)}}{\epsilon_j} \sum_i \frac{A_{ij}\, y_i}{\left(\sum_k A_{ik}\, \hat{x}_k^{(N)}\right) + b_i}$$

$$\xcancel{\hat{x}_j^{(N+1)} = \frac{\hat{x}_j^{(N)}}{\epsilon_j} \sum_i \frac{A_{ij}\left(y_i - b_i\right)}{\left(\sum_k A_{ik}\, \hat{x}_k^{(N)}\right)}}$$

- Not sure this is handled properly in RooUnfold…
- Always check your tools

Proper formula: converges to the Poisson maximum likelihood. All numbers are guaranteed to be positive.

The other formula has no proven properties. Small (y-b) could fool the algorithm about the statistical uncertainty in that bin

Iterative method – normalisation of fakes

- RooUnfold adds up the content of the event matrix and compares it to the predicted “reco” distribution
- These “fake” events are background
- If the fakes are >0, RooUnfold allocates an extra bin to determine their normalisation in the iterative method (this was suggested by D’Agostini (?))

- Possible effect: user has given N x N matrix
- But RooUnfold uses N x (N+1) matrix (+1 fake normalisation bin)
- So one unfolds N+1 bins from only N data bins - NOT GOOD
- And the fakes normalisation may be different from what one thinks it is (because RooUnfold adjusts it in the unfolding)

TUnfold difficulties

- TUnfold is now version 17.9 (and soon 17.10)
- But in Root6 there is version 17.6
- Make sure to use the latest version (bug fixes, SURE scan, etc)

- TUnfold has a strange concept to account for inefficiencies
  - Reco underflow and overflow are counted as “not reconstructed” (possibility to account for inefficiencies)
  - In contrast, truth underflow and overflow are unfolded (similar to “fakes” in RooUnfold iterative method)
  - Special care to be taken when setting up the bins (→TUnfoldBinning)

Summary

Unfolding methods (1)

- Unregularised unfolding: matrix inversion, maximum likelihood
  - Result is unbiased
  - But there are large statistical fluctuations and anti-correlated bins
- Bin-by-bin method
  - Very strong bias to the model
  - Do not use

Unfolding methods (2)

- Least square plus Tikhonov
  - Very flexible, regularisation pattern and regularisation strength
  - For s=0 obtain maximum likelihood result
- EM iterative method
  - Seemingly very robust and easy to use
  - Too small number of iterations results in large bias to the model
  - Convergence to the maximum likelihood is very slow

Setting regularisation parameters

- Tikhonov regularisation: three methods
  - Eigenvalue analysis: understand all the details of the data modes
  - L-curve scan: intuitive geometrical picture
  - SURE minimisation: statistician’s choice
- Iterative methods
  - SURE minimisation works well , but is not widely used in the HEP community
  - Physicists often use handwaving arguments (result does not change for more than 4 iterations … 4 is the default in RooUnfold ...) Be careful! The method converges very slowly!

Tools

- Frequently used with Root
  - RooUnfold
  - TUnfold
- Both are nice tools, but each with their own difficulties
  - RooUnfold: originally putting emphasis on easy comparison of methods
  - TUnfold : original intention to have “minimal” input: only the matrix of events. This lead to a confusing role of underflow+overflow bins

If in doubt, do not hesitate to contact the authors for help. I am glad to help with TUnfold

Not covered in this talk

- Many unfolding methods exist but are not covered here
  - SVD with curvature regularisation
  - Fully Bayesian unfolding, BAT

    … and many other tools
- Brand-new tool: machine learning unfolding (OMNIFOLD)
  - Unbinned unfolding: ML based reweighting of the Monte Carlo to look like data. Reweight bootstrap samples to get uncertainties
  - Can do arbitrary plots of the unfolded Monte Carlo truth parameters
  - First physics paper using that method published recently: PRL 128 (2022), 132002

Thank you for your attention

Please apologize for the improper preparation of the exercises (should have gone together with a root tutorial?)