Unfolding in High Energy Physics

Stefan Schmitt, DESY
Daniel Britzger, DESY

Terascale statistics school 2014

Outline

- What is unfolding?
- Methods commonly used in HEP
  - Bin-by-bin
  - Matrix methods
- Problems specific to HEP
  - Multidimensional problems
  - Phase-space boundaries [only one slide]
  - Multiplicity measurement [not discussed in this talk]

Exercises

- Lecture is interleaved by exercises
  - Solutions are discussed during the lecture
- Instructions: run installation script after logging in

```
source
/afs/desy.de/user/s/school06/public/install.sh
```

- This creates and changes to a directory unfold_exercises
- Run exercises in that directory. Test:

```
root
.x exercise0.C++
```

- After logout, simply change dir and initialize root:

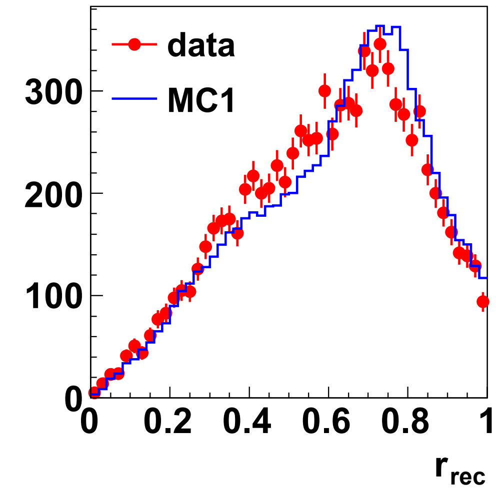
```
cd unfold_exercises; module load root
```

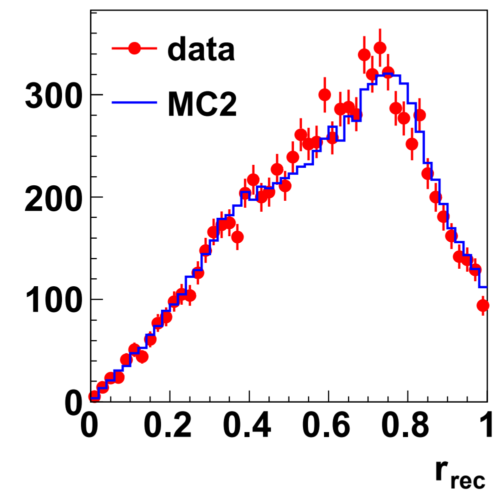
Check installation (exercise0.C)

- Run exercises in that directory. Test:

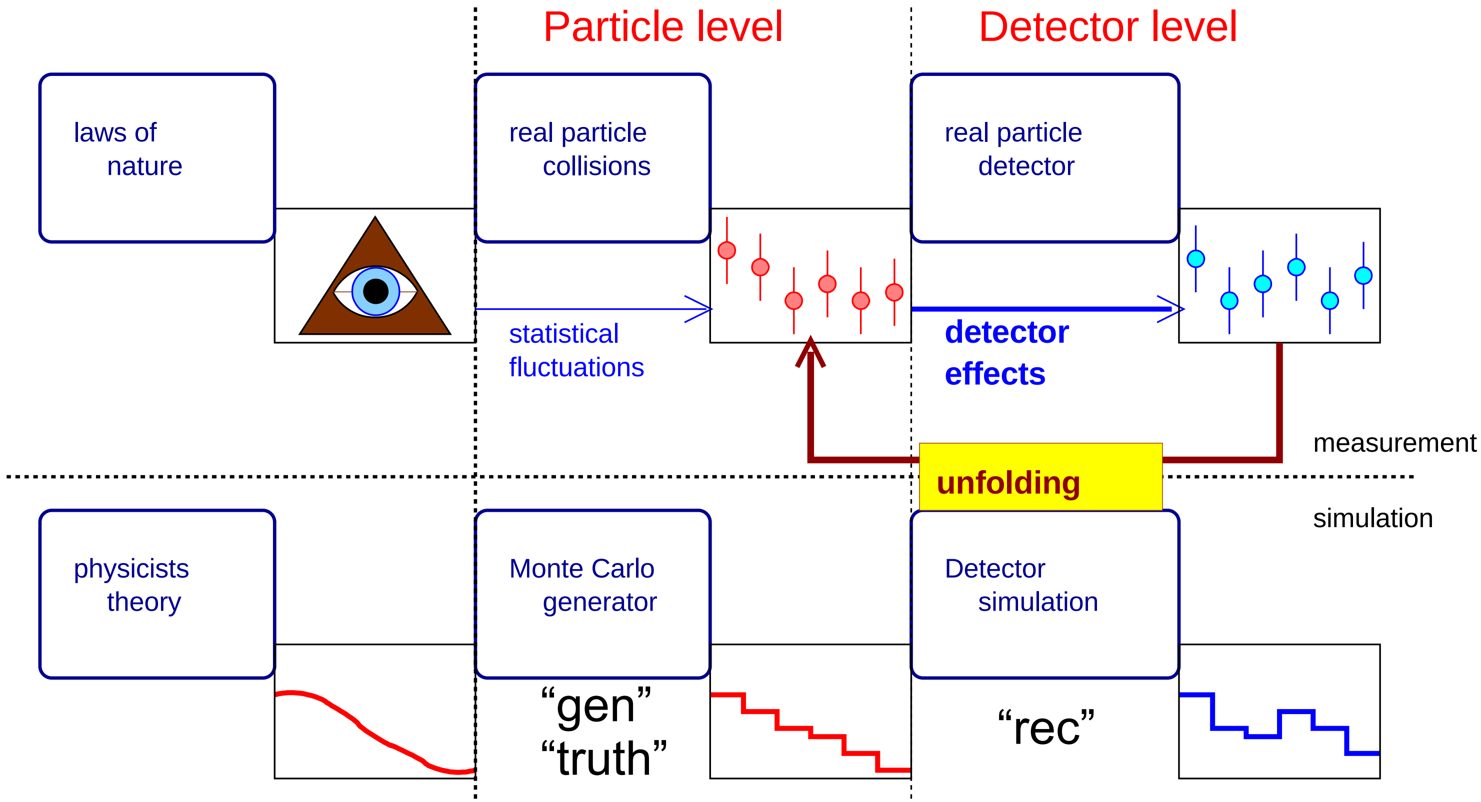
```
root
.x exercise0.C++
```

- Resulting plot should look like depicted to the right
- This macro is the starting point for the exercises
- Solutions are in the directory “solution”
- Assumption: you are familiar with root





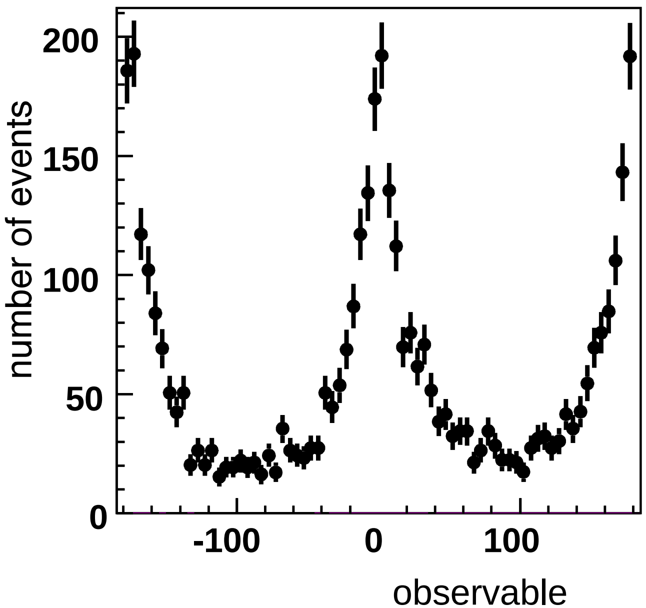
What is unfolding?

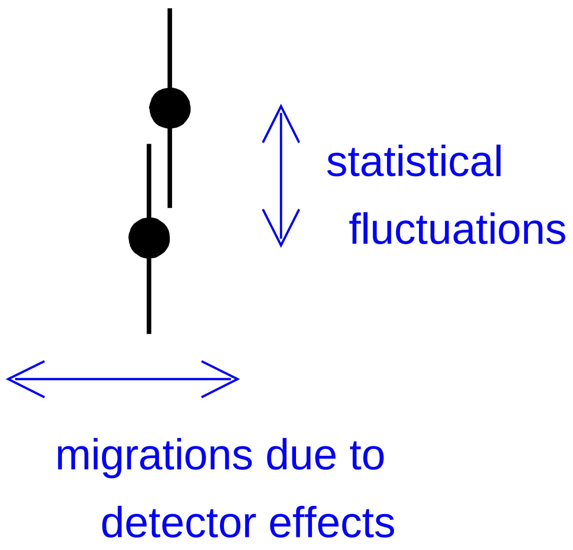


- Obtain measurements independent of detector effects, using the simulation
- Propagate statistical uncertainties back to particle level
- Require results to be independent of theory assumptions

Migrations and stat. fluctuations

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





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

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

Formal definitions

- This talk: measurements are given by event (jet, track, ...) counts, grouped in bins x_i^{rec}
- Bins are defined by regions in phase-space (observables)
- Not covered in this talk: unbinned methods
- Detector effects: detector response matrix A

  A_{ij}: probability that event from truth bin j is found in rec bin i

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

- Statistical fluctuations: Poisson distribution or Gaussian approximation

  $$P\left(Y_i = y_i^{\text{rec}}\right) = \frac{\exp\left[-\mu_i\right]\left(\mu_i\right)^{y_i}}{y_i!}$$

Exercises: data and MC sets

- In total, four root files, each file with events in a TTree
- Data “data.root” ~300k events
- Two Monte Carlo “mc1.root” and “mc2.root” ~3000k events
  - Different model but identical detector simulation
  - See influence of the model on the unfolding results (unfold data using MC1 or MC2 detector response)
  - Check whether MC2 can be recovered using MC1 detector response (unfold MC2 using MC1)
- Data truth (in your real analysis this is not present): “datatruth.root” ~300k events
  - Validate whether the unfolding worked

Exercises: observables

- Exercise provides only two observables:
  - Variable rRec (rGen) which ranges from 0 to ~1
  - Variable pRec (pGen) which ranges from -π to π
- (rGen,pGen): truth. (rRec,pRec): after detector simulation
- Flag (isTrig==1) means that rRec and pRec are valid
- Events are signal or background
  - Data: no discrimination possible
  - MC: flag (isBgr==1) indicates that this is background. Background events have no valid "truth" information [rGen and pGen can not be used in that case]

Exercises: details

- Data “data.root”
  - TTree “rec” with four variables
  - Event weight w
  - Observables rRec and pRec are valid if (isTrig==1)

```
TTree rec
    float rRec
    float pRec
    int isTrig
    float w
```

- Two different Monte Carlo “mc1.root” and “mc2.root”
  - TTree “recgen” with seven variables
  - Observables as in data
  - Truth variables rGen and pGen are valid only if(isBgr==0)
  - if(isBgr==1), then the event is background

```
TTree recgen
    float rRec
    float pRec
    int isTrig
    float w
    int isBgr
    float rGen
    float pGen
```

Exercises: data “truth” set

- Data truth “datatruth.root”
  - TTree “gen” with four variables
  - Event weight w
  - Observables rGen and pGen are valid if (isBgr==0)
  - Generator level informtion for our “data”
  - This is to evaluate how well our unfolding worked

    (for a real analysis, this has to be estimated by comparing different MC models)

```
TTree gen
    float rGen
    float pGen
    int isBgr
    float w
```

Exercises: unfolding of one variable

- Most Exercises: unfolding rGen from rRec (single-differential) and pRec or pGen are not used.
  - In this case we use only 1/30 of the events:

```
  int nSubset=30;
  int iSubset=0;
...
  int firstEvent=iSubset*tree->GetEntriesFast() / nSubset;
  int lastEvent=(iSubset+1)*tree->GetEntriesFast() / nSubset;
  int nEvent=lastEvent-firstEvent;
...
  tree->Draw("rRec","w","",nEvent,firstEvent);
```

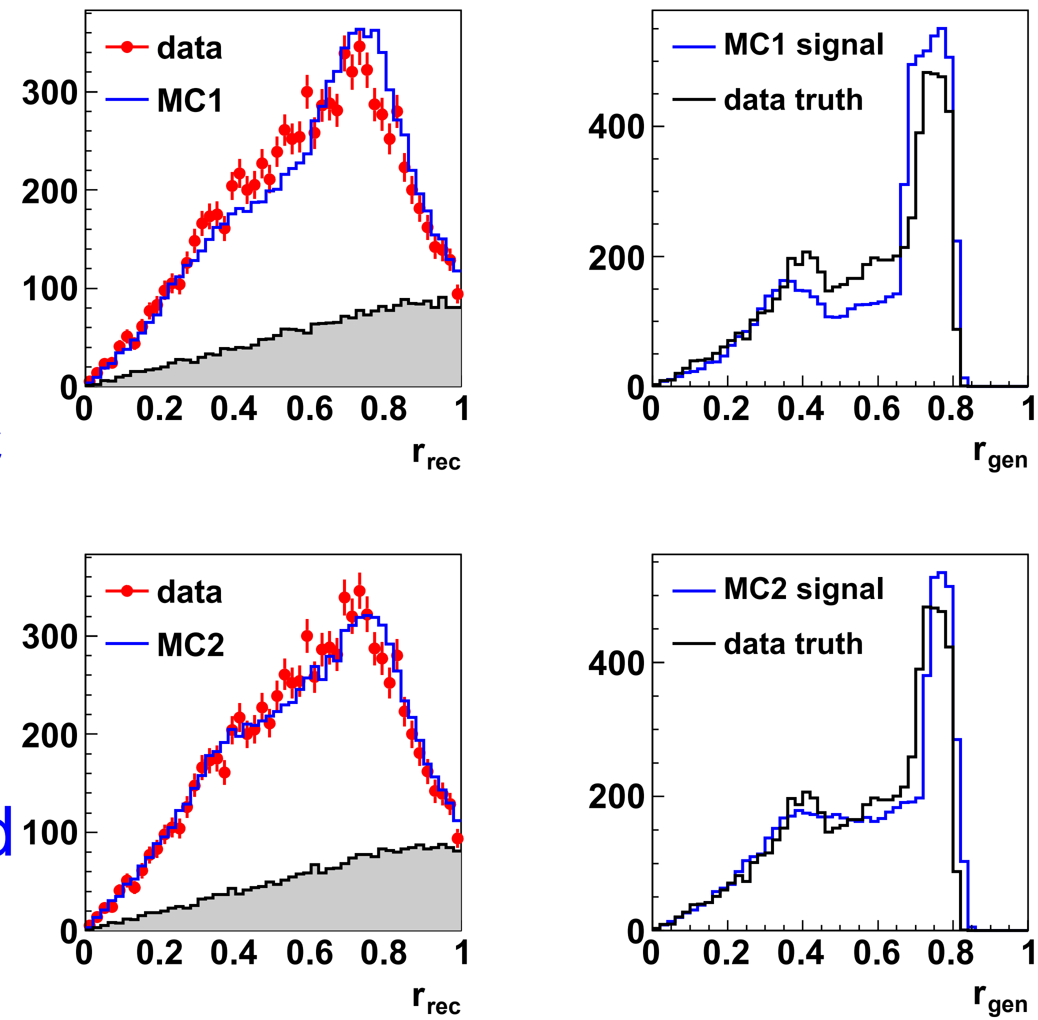
  - Data: use ~10k events, MC: use ~100k events
  - Full samples are needed for unfolding of double-differential distributions

Exercise 1

- Start with exercise0.C:
  - Data to MC comparison for the variable “rRec” with 0<rRec<1 using 50 bins
- Superimpose “rRec” for the background taken from MC
  - Hint: use the MC event weight: w*isTrig*isBgr
- Make a separate plot for the distribution of “rGen” from MC and superimpose data “truth”
  - Hint: use the MC event weight: w*(1-isBgr)
- Repeat plots using MC1 and MC2
- Discuss the results

Exercise 1 discussion

- Reconstructed and generated variables are quite different
- Large background
- Description of data by MC is not perfect
- Large differences on truth level
- Question: why are rec and gen so different?



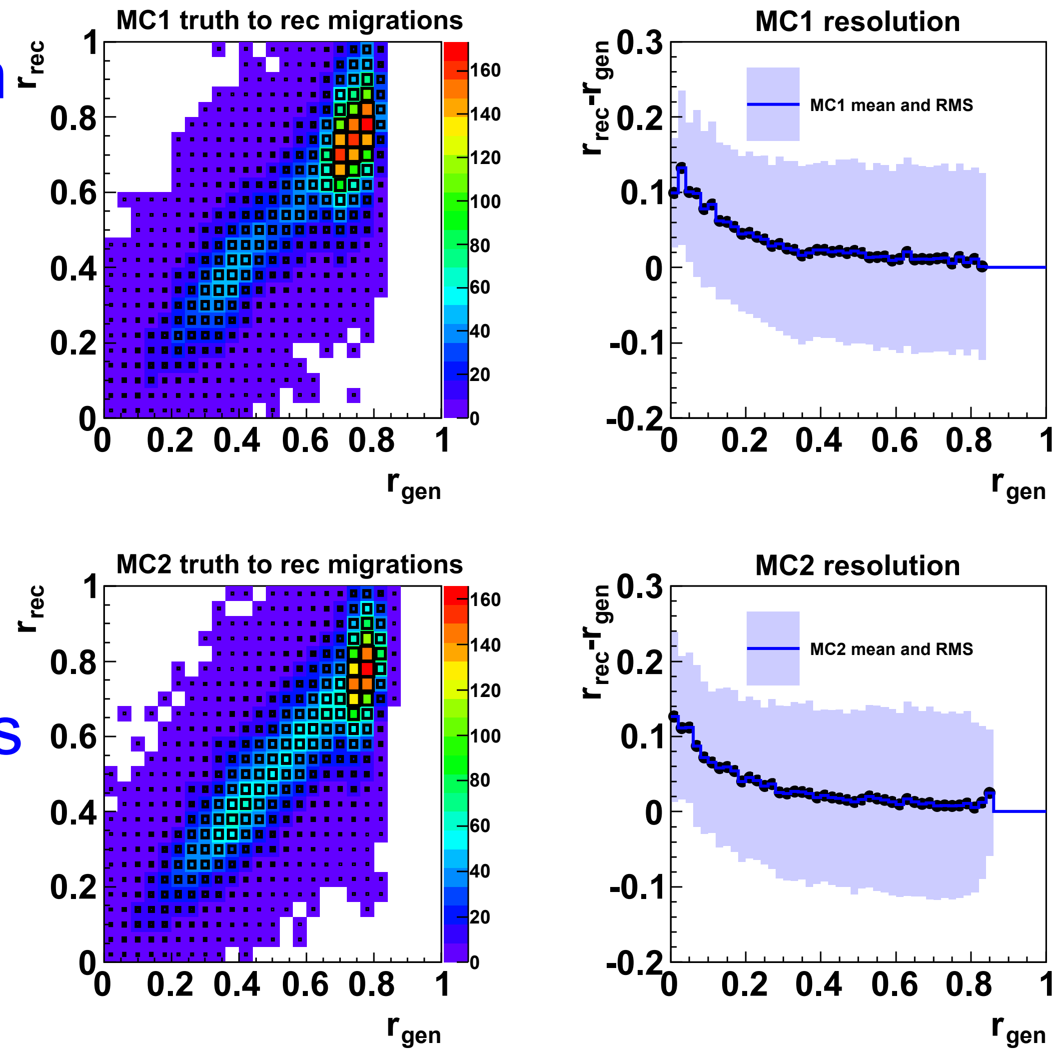
Exercise 2

- Investigate the correlation of rRec and rGen
  - Use a TH2D with 25x25 bins
  - The event weight for reconstructed signal is w*isTrig*(1-isBgr)
- [if there is time]: quantify the resolution rRec-rGen as a function of rGen
  - Use a TProfile with options “s” and fill with option “profs”
- Compare MC1 and MC2, discuss the results

Exercise 2 discussion

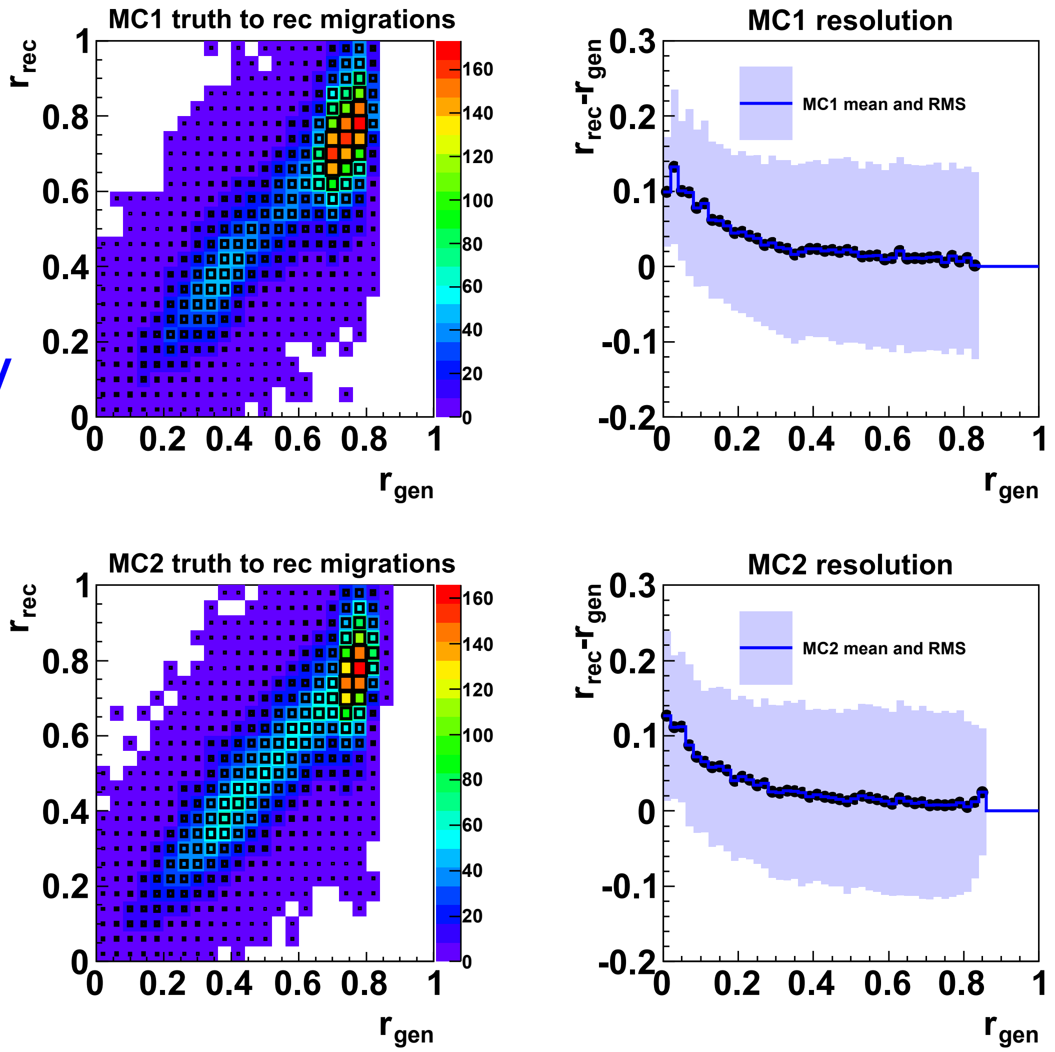
- Correlation of rec and gen visible
- At low r: mean rec-gen is different from zero (bias)
- Resolution (RMS) is of order 0.1
- Similar resolution is observed for different MCs

Resolution depends only on detector simulation, is independent of MC model



Choice of bin width

- Typically, choose bin width ~resolution
  - choose Δr=0.1
- Range of r is approximately 0..0.9
  - unfold 9 bins in rGen



Unfolding methods

- Unfolding methods discussed in this talk
  - Bin-by-bin ”correction”
  - Matrix methods:
    - Matrix inversion
    - “Bayesian” [D'Agostini 1995]
    - “Iterative” [D'Agostini 1995 iterated]
    - Fraction fit: TUnfold [no exercise: TFractionFitter]
    - Regularised unfolding: TUnfold [no exercise: TSVDUnfold]
  - Detailed references can be found on page 61

“Bin-by-bin”

- Assumption: migration effects are “small” so they can be “corrected” using a multiplicative factor
- The factor is determined from MC
- Two variants:
  - Simple bin-by-bin
  - Bin-by-bin with background subtraction

Simple “bin-by-bin”

- Data in a given bin: $y_i^{\mathrm{data}} \pm \delta y_i^{\mathrm{data}}$
- Reconstructed MC in a given bin: y_i^{rec}
- Generated MC truth in a given bin: x_i^{gen}

- Define correction factor: $f_i = \frac{x_i^{\mathrm{gen}}}{y_i^{\mathrm{rec}}}$

Note: sometimes $1/f_i$ is called “generalized efficiency”

- Corrected data: $x_i^{\mathrm{BBB}} = \frac{x_i^{\mathrm{gen}}}{y_i^{\mathrm{rec}}} y_i^{\mathrm{data}}$
- Corrected data uncertainty: $\delta x_i^{\mathrm{BBB}} = \frac{x_i^{\mathrm{gen}}}{y_i^{\mathrm{rec}}} \delta y_i^{\mathrm{data}}$

“bin-by-bin” with backg. subtraction

- Data in a given bin: $y_i^{\text{data}} \pm \delta y_i^{\text{data}}$
- Reconstructed MC in a given bin: $y_i^{\text{rec,sig}} + y_i^{\text{rec,bgr}}$
  - Includes background contribution: $y_i^{\text{rec,bgr}}$
- Generated signal truth in a given bin: x_i^{gen}

- Define correction factor: $f_i = \frac{x_i^{\text{gen}}}{y_i^{\text{rec,sig}}}$

Not discussed in this talk

- Corrected data: $x_i^{\text{BBB}} = \frac{x_i^{\text{gen}}}{y_i^{\text{rec,sig}}} \left(y_i^{\text{data}} - y_i^{\text{rec,bgr}} \right)$
- Corrected data uncertainty: $\delta x_i^{\text{BBB}} = \frac{x_i^{\text{gen}}}{y_i^{\text{rec,sig}}} \delta y_i^{\text{data}}$

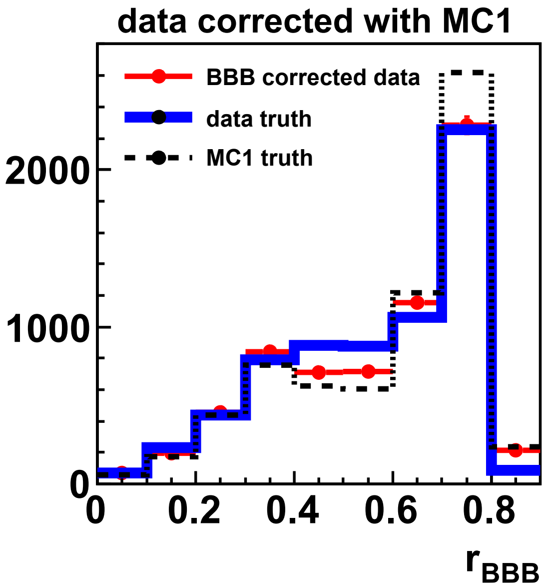
Exercise 3: bin-by-bin unfolding

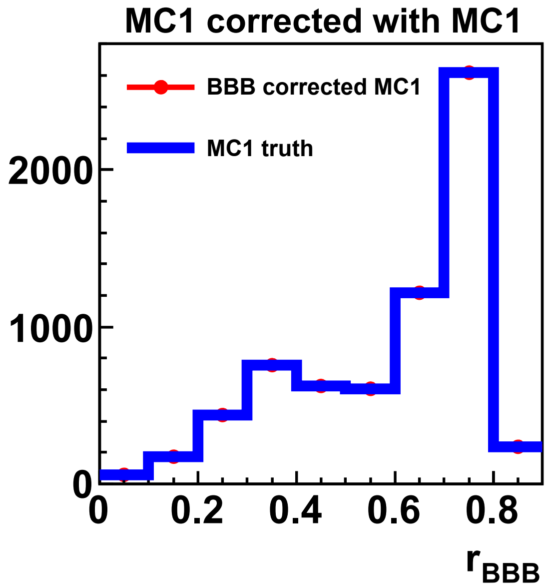
- Correct data with MC1, compare to data truth and MC1 truth
  - Use 9 bins, $0<r<0.9$
- Correct MC1 with MC1, compare to MC1 truth
- Correct MC2 with MC1, compare to MC2 truth and MC2 truth

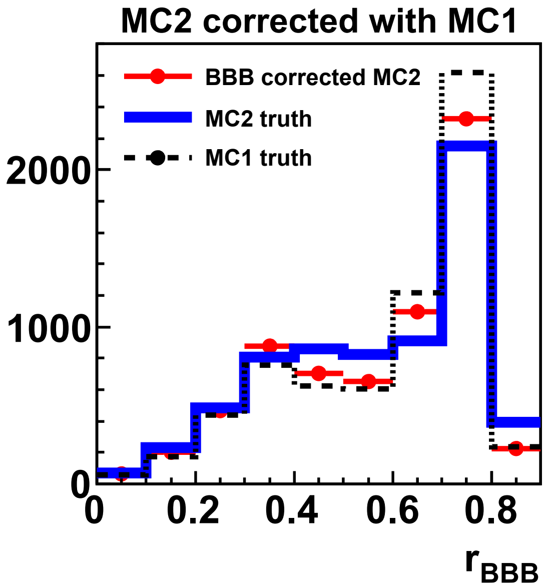
- Discuss the results

Exercise 3 discussion

Bin-by-bin basically returns the MC truth decorated with statistical fluctuations taken from data!





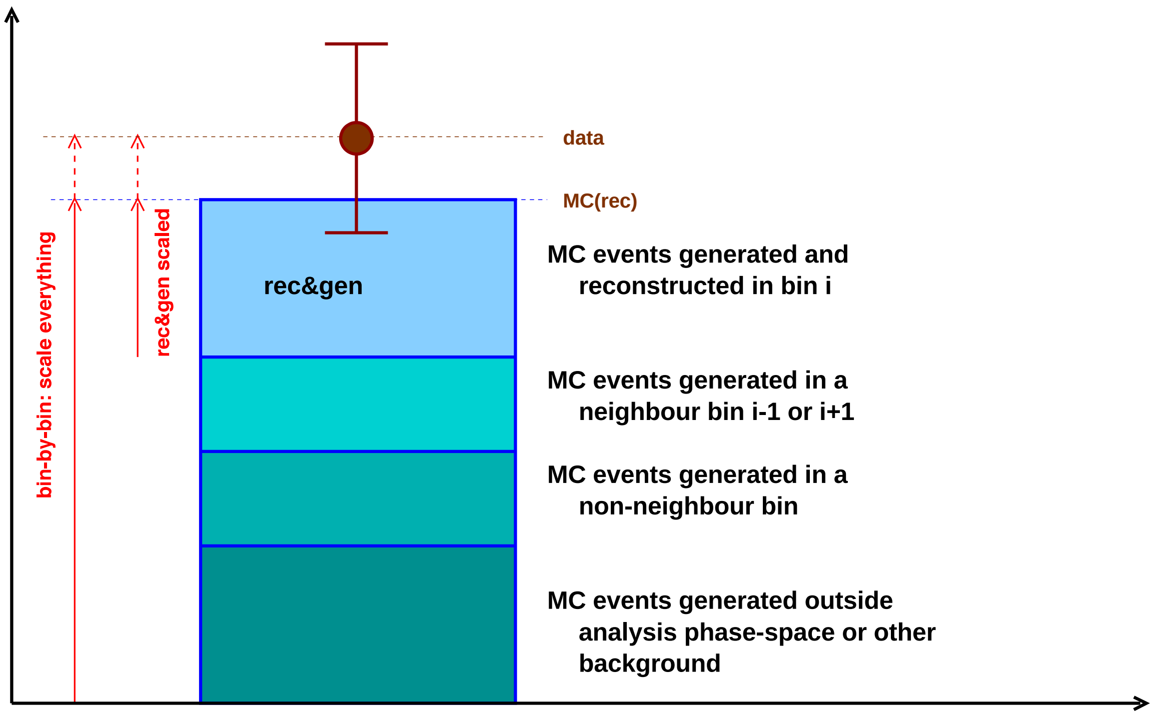


Never use this method for your data analysis!!!

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:

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

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

- Relevant quantity: purity

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

Exercise 4: purity and migrations

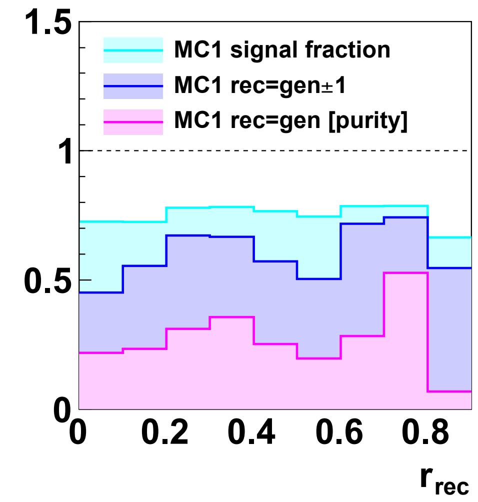
- Plot bin purity
- Plot fraction of events migrating into a bin from a neighbour bin in the range i-1 … i+1
  - Hint: fill a 2D histogram rec vs gen to get number of events where iRec=iRen or iRec=iGen±1
- Plot fraction of signal events in a bin
- Show everything on one plot
- Repeat this for MC1 and MC2

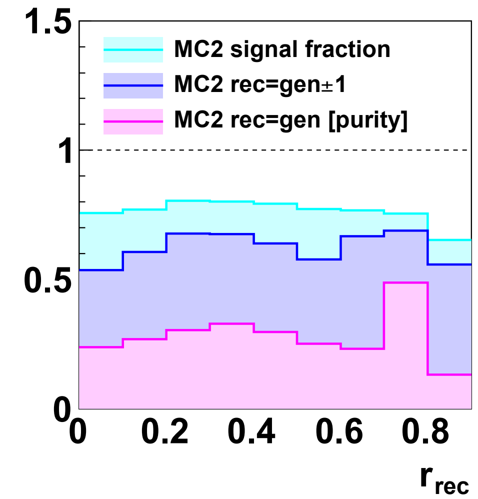
No time to do as exercise today.
We only discuss the results

Exercise 4 discussion

- In the example, the purity is low, of order 20-30%
- Purities and migrations into the "rec" bins are different for MC1 and MC2

→ Looking for quantities which do not depend so much on the model!





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.

Exercise 5

- Compare the matrix of probabilities for MC1 and MC2
  - Fill matrix nRec vs nGen (Exercise 2)
  - Normalize each nGen,nRec bin to the total nGen
- Quantitative comparison: fill a histogram of the difference in probability

$$A_{ij}^{\mathrm{MC1}} - A_{ij}^{\mathrm{MC2}}$$

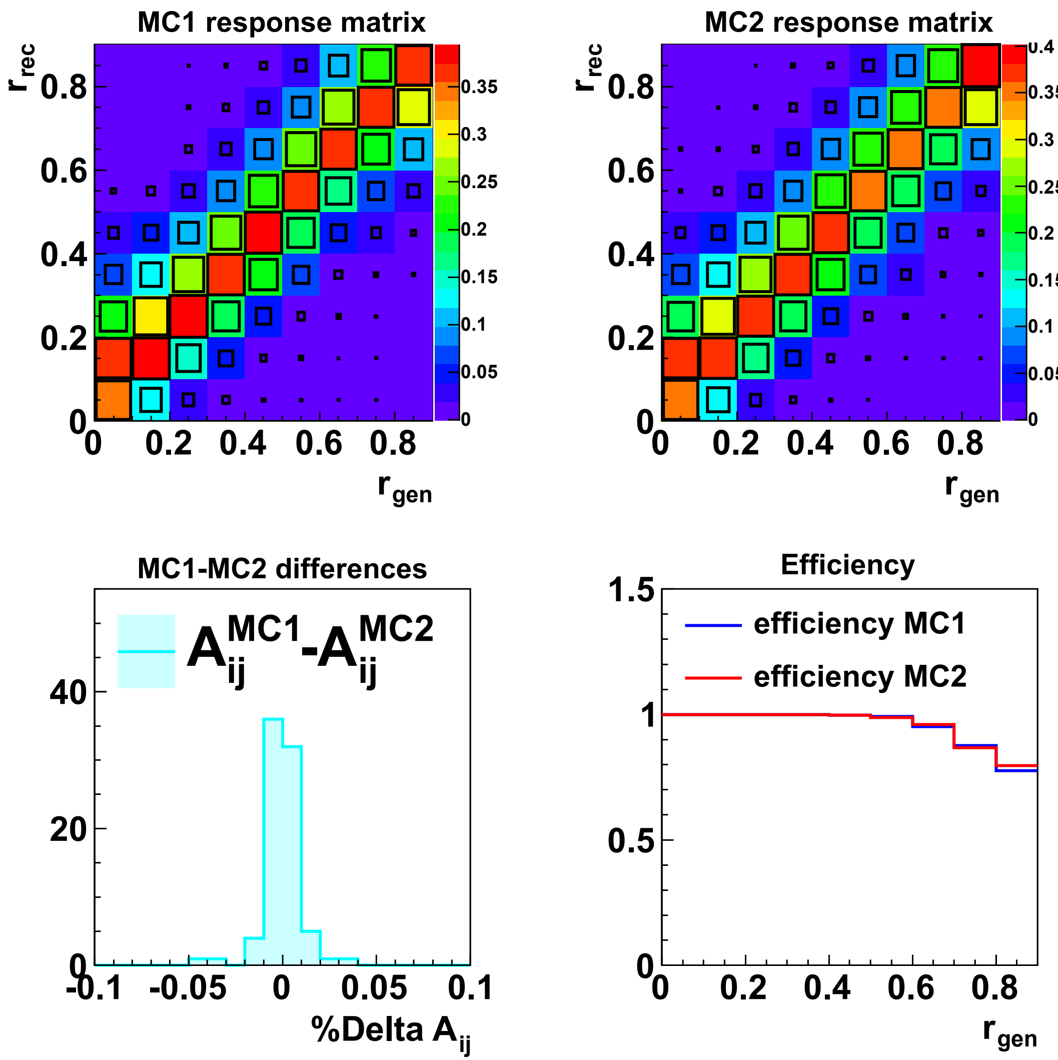
- Also calculate the efficiencies for MC1 and MC2

$$\text{Matrix } A_{ij} = \frac{y_{ij}^{\mathrm{rec,gen}}}{y_j^{\mathrm{gen}}}$$

$$\text{Reconstruction efficiencies } \varepsilon_j = \sum_i A_{ij}$$

Discussion exercise 5

- Probabilities are very similar between MC1/MC2. Differences are up to 5%.
- Efficiencies are very similar
- Expect to have little model dependence if unfolding algorithm uses only A_{ij} and not x_i^{gen}



Background in matrix methods

- Basic formula

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

- There is no background foreseen!

  (a): subtract background from data, then proceed

  $$y_i^{\text{data}} - y_i^{\text{bgr}} \underset{\text{stat.fluct.}}{\sim} \mu_i = \sum_{j=1,nGen} A_{ij} x_j^{\text{truth}}$$

  (b): include background normalization as one “truth” bin

  $$y_i^{\text{data}} \underset{\text{stat.fluct}}{\sim} \mu_i = \sum_{j=1,nGen+1} A_{ij} x_j^{\text{truth}}$$ where

  x_{nGen+1} is the background normalization

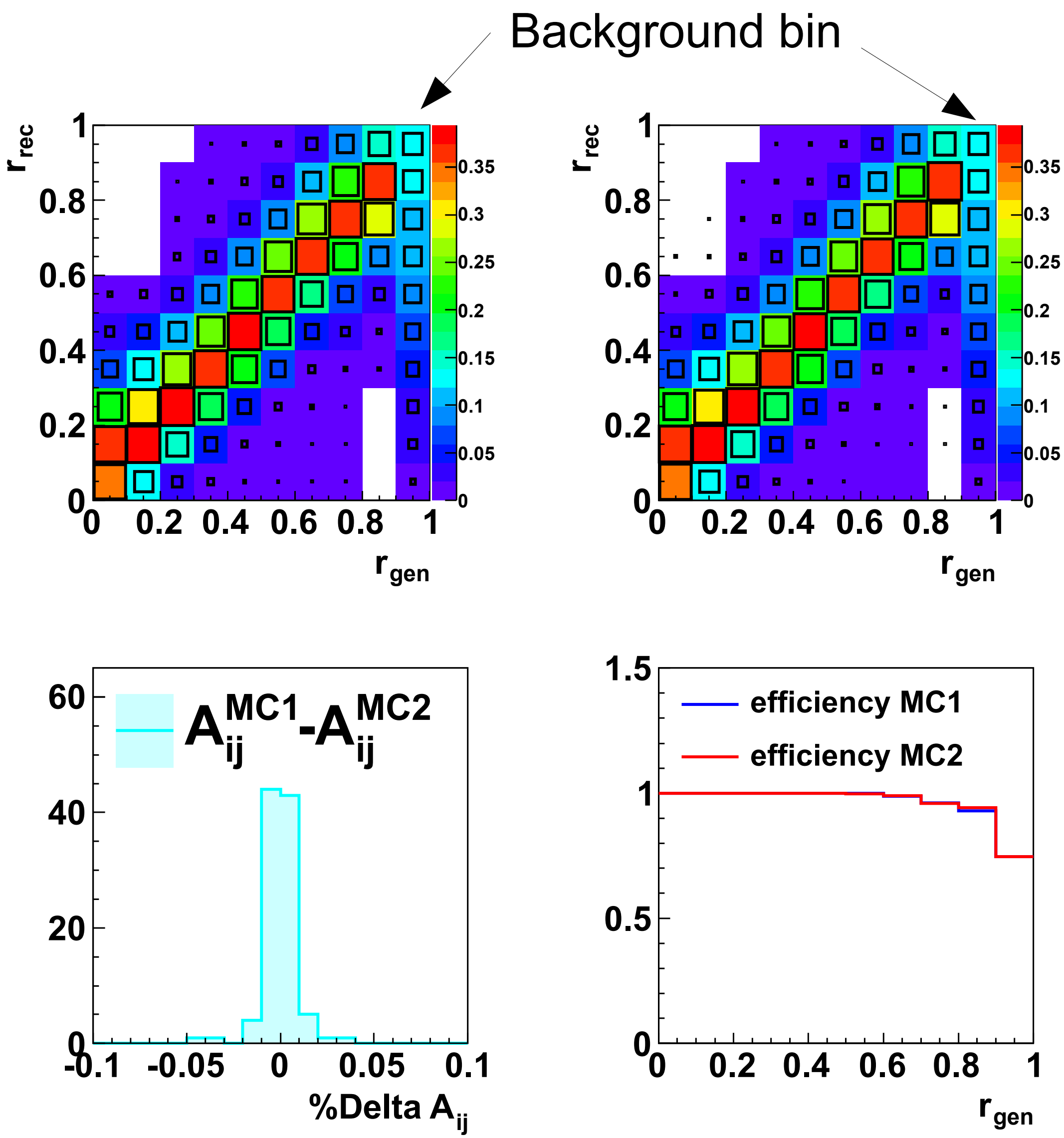
  $$A_{i,nGen+1} = \frac{y_i^{\text{bgr}}}{\sum_k y_k^{\text{bgr}}}$$ is the background shape

Exercise 6

- Extend the resolution plots by one bin
- Use gen-bin #10 to store the background
  - nBin=10, r_0=0.0, r_1=1.0
  - When filling generator quantity:
    - if(iBgr==0) fill rGen
    - if(iBgr==1) fill 0.95 [into extra gen bin 0.9..1.0]
- Produce plots as in exercise 5. Discuss
- Note: for all following exercises the background normalisation is included in the unfolding procedure. If this has not become clear, please ask now!

Exercise 6 discussion

- Background shape is quite similar for the two MC
- Background “efficiency” is low because it is normalized to all bgr events, including those with rRec>1



Matrix inversion method

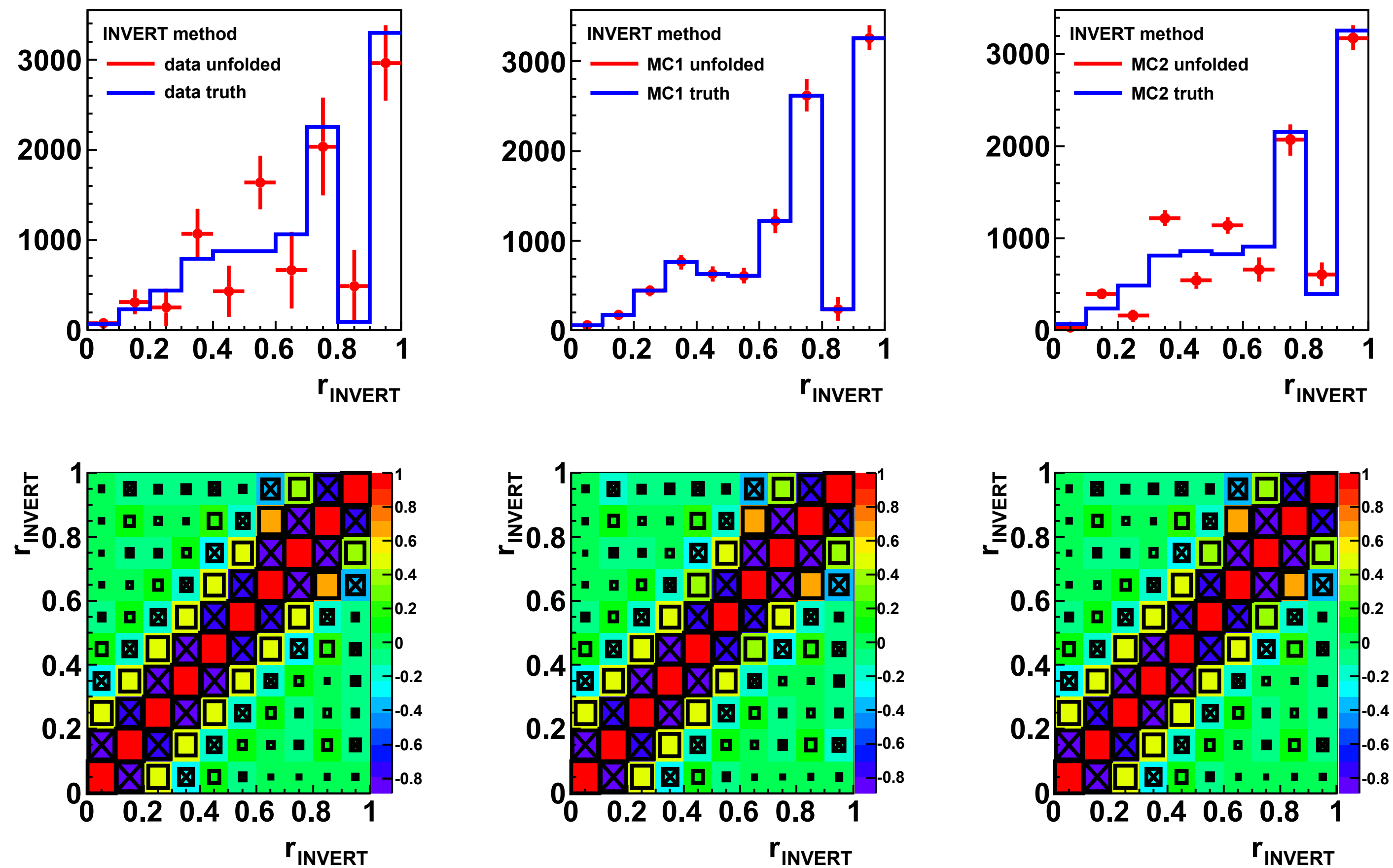
- Simplest method for matrix unfolding: invert response matrix $$y_i^{\text{data}} \underset{\text{stat.fluct}}{\sim} \mu_i = \sum_{j=1,nGen} A_{ij} x_j^{\text{truth}}$$ $$x_j^{\text{INVERT}} = \sum_i \left(A^{-1}\right)_{ji} y_i^{\text{data}}$$
- Covariance matrix (“uncertainties”): $$Cov\left(x_j^{\text{INVERT}}, x_k^{\text{INVERT}}\right) = V_{jk} = \sum_i \left(A^{-1}\right)_{ji} \left(\delta y_i^{\text{data}}\right)^2 \left(A^{-1}\right)_{ki}$$
- General feature of matrix unfolding methods: covariance has off-diagonal elements
- Diagonal elements: uncertainties $\delta x_j^{\text{INVERT}} = \sigma_j = \sqrt{V_{jj}}$
- Size of off-diagonals is often quantified using correlation coefficients $$\rho_{jk} = \frac{V_{jk}}{\sigma_j \sigma_k}$$

Exercise 7: matrix inversion

- Invert the matrix of probabilities for MC1
  - Class TMatrixD, method Invert()
  - Caution: first matrix index=0, first histogram bin=1
- Unfold data, MC1, MC2 using the matrix of probabilities from MC1
- Calculate the covariance matrices and uncertainties
- Compare the unfolded results with the truth distributions
- Also calculate and show the correlation coefficients
- Discussion

Exercise 7 discussion

- Unfolded data show large point-to-point oscillations
- Consistent with truth within errors
- Oscillations are “allowed” by large negative correlation coefficients for adjacent bins
- Goal: damp fluctuations: regularisation

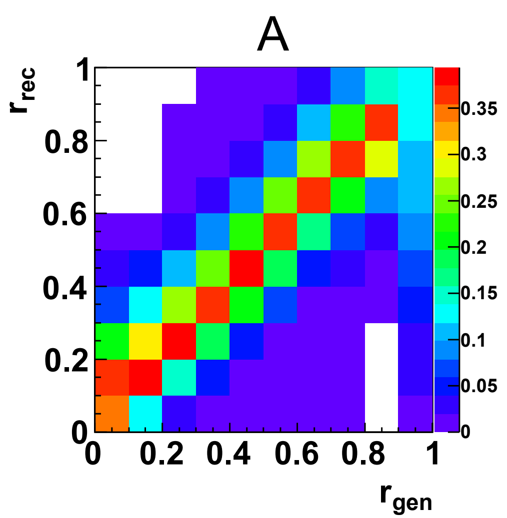


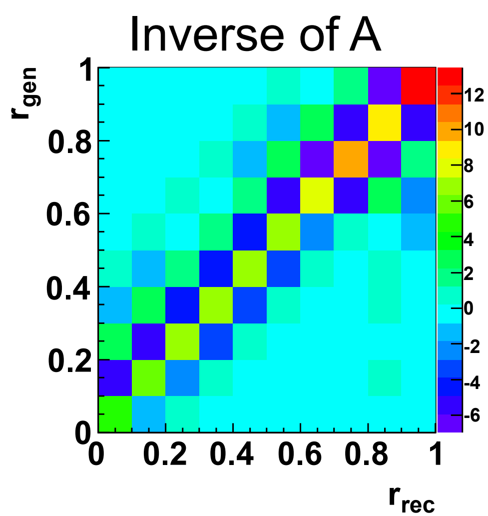
Correlation coefficients are important for fits, integrals etc.
Example: uncertainty of x_1+x_2:

$$\Delta(x_1+x_2)=\sqrt{\sigma_1^2+2\sigma_1\sigma_2\rho_{12}+\sigma_2^2}$$

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





"Bayesian" (D'Agostini)

- Motivated by Bayesian statistics: D'Agostini (1995)
- Here, we simply use the prescription and test it
- See 2010 paper by D'Agostini for detailed discussion
- Prescription:

D'Agostini

$$x_j^{\mathrm{AGO}} = x_j^{\mathrm{gen}} \sum_i \frac{A_{ij}}{\varepsilon_j} \frac{y_i^{\mathrm{data}}}{y_i^{\mathrm{rec}}}$$

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

Compare to Bin-by-bin:

Bin-by-Bin

$$x_j^{\mathrm{BBB}} = x_j^{\mathrm{gen}} \left(\frac{y_i^{\mathrm{data}}}{y_i^{\mathrm{rec}}} \right)_{i=j}$$

- Covariance matrix:

$$V_{jk}^{\mathrm{AGO}} = x_j^{\mathrm{gen}} x_k^{\mathrm{gen}} \sum_i \frac{A_{ij}}{\varepsilon_j} \frac{A_{ik}}{\varepsilon_k} \left(\frac{\delta y_i^{\mathrm{data}}}{y_i^{\mathrm{rec}}} \right)^2$$

Main criticism:
(1) from non-Bayesians: result depends on "prior" MC model nGen
(2) data statistical fluctuations are not treated with Bayesian methods
[see D'Agostini 2010 paper]

Iterative “Bayesian”

- Idea: minimize model-dependence by iterating D'Agostini method

$$x_j^{\mathrm{AGO}} = x_j^{\mathrm{gen}} \sum_i \frac{A_{ij}}{\varepsilon_j} \frac{y_i^{\mathrm{data}}}{y_i^{\mathrm{rec}}} \text{ Iterate: replace } x_j^{\mathrm{gen}} \text{ by } x_j^{\mathrm{AGO}} \text{ and } y_i^{\mathrm{rec}} \text{ by } \sum_j A_{ij} x_j^{\mathrm{AGO}}$$

- Resulting formula for ITER→ITER+1

$$x_j^{\mathrm{ITER+1}} = x_j^{\mathrm{ITER}} \sum_i \frac{A_{ij}}{\varepsilon_j} \frac{y_i^{\mathrm{data}}}{\sum_k A_{ik} x_k^{\mathrm{ITER}}}$$

- Covariance matrix: usually determined from MC toy experiments, using N times the equivalent data statistics (error propagation is difficult when iterating)

Fit-based matrix methods

- Recall basic equation

$$y_i^{\text{data}} \underset{\text{stat.fluct}}{\sim} \mu_i = \sum_{j=1,nGen} A_{ij} x_j^{\text{truth}}$$

- When unfolding, the x^{truth} are unknowns and the y^{data} are measured
- Formulate the problem as a maximum-likelihood fit. For example, if the data are Gaussian distributed, minimize:

$$\chi^2(x^{\text{fit}}) = \sum_{i=1,nRec} \left(\frac{y_i^{\text{data}} - \sum_{j=1,nGen} A_{ij} x_j^{\text{fit}}}{\delta y_i} \right)^2$$

- If nRec=nGen: equivalent to matrix inversion method
- Interesting case: nRec>nGen. Typical: nRec ~ 2*nGen

Fit based methods in Root

- In root, there are at least two such methods available:
  - TFractionFitter
    - Uses Poisson statistics (log-likelihood fit)
  - TUnfold or TUnfoldSys
    - Uses Gaussian statistics (linear fit)
- For simplicity, we use only TUnfold for the exercises

TUnfold (with tau=0)

- Minimizes the matrix equation

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

x : vector of unknowns
y : vector of measurements
A : matrix of probabilities
V_{yy} : covariance matrix of y (uncertainties squared)

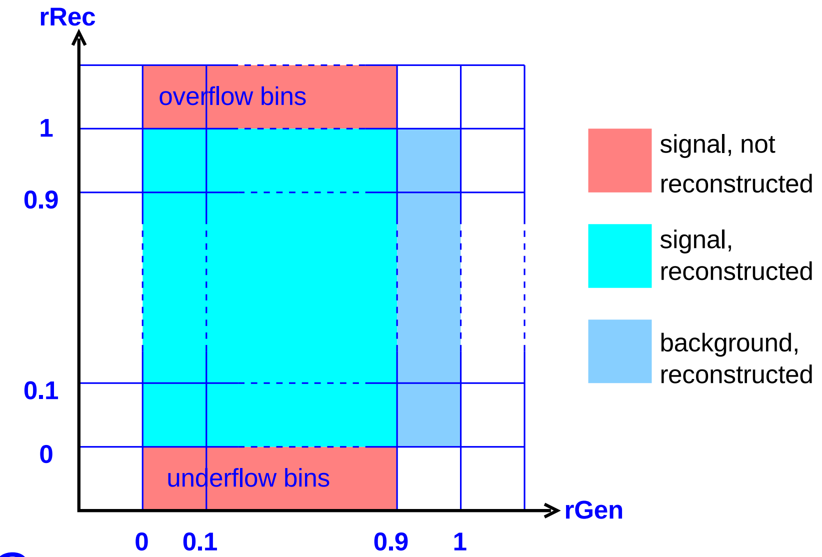
- When setting up Tunfold, the matrix A is calculated from the Monte Carlo histogram of event counts (gen vs rec)

→ normalisation is handled inside Tunfold.

.

TUnfold: setting up the matrix

- When histogram rec.vs.gen is filled as usual, how to account for inefficiencies (events which are not reconstructed)?
- In TUnfold: “rec” underflow/overflow bins hold events which are not reconstructed
- In our example:
  - If (iBgr==0)&&(isTrig==1) fill (rGen,rRec)
  - If (iBgr==1)&&(isTrig==1) fill (0.95,rRec)
  - If (iBgr==0)&&(isTrig==0) fill (rGen,-1)
  - If (iBgr==1)&&(isTrig==0) fill (0.95,-1)



- Check: sum over all rec bins for a given iGen (including underflow and overflow) should be equal to nGen(iGen)

Exercises 8-10

- Test and compare the following methods
  - Exercise 3: bin-by-bin
  - Exercise 7: matrix inversion
  - Exercise 8: D'Agostini
  - Exercise 9: Iterative methods
  - Exercise 10: Fraction fit (TUnfold with τ=0)
- Compare central values to truth and compare correlation coefficients
- Start now / continue after today's lectures

Exercise 8: D'Agostini method

- Unfold data, MC1, MC2 using the detector response matrix from MC1 and the D'Agostini method
- Calculate and display correlation coefficients
- Compare to unfolding results to data,MC1,MC2 truth
- Discussion

Exercise 9: Iterative method

- Iterative method (100 cycles) to unfold data using MC1
- Show results after 1,10,100 iterations (=D'Agostini)
- Calculate Covariances using data replicas

```
// produce data replicas with extra stat. Fluctuations
// original data histogram:  hist_data_rec[0]
// replicas: hist_data_rec[1..NREPLICA]
TRandom3 rnd(0);
for(int iRec=1;iRec<=nBin;iRec++) {
  double n0=hist_data_rec[0]->GetBinContent(iRec);
  for(int iReplica=1;iReplica<NREPLICA;iReplica++) {
    double ni=rnd.Poisson(n0);
    hist_data_rec[iReplica]->SetBinContent(iRec,ni);
    hist_data_rec[iReplica]->SetBinError
          (iRec,TMath::Sqrt(ni));
  }
}
```

```
// given bin jGen kGen
// and the result for the replicas
// in histograms h[1..NREPLICA]
double s[2][2]={{0.,0.},{0.,0.}};
for(int iReplica=1;iReplica<NREPLICA;iReplica++) {
  double yj=h[iReplica]->GetBinContent(jGen);
  double yk=h[iReplica]->GetBinContent(kGen);
  s[0][0]+=1.;
  s[1][0]+=yj;
  s[0][1]+=yk;
  s[1][1]+=yj*yk;
}
double meanJ=s[1][0]/s[0][0];
double meanK=s[0][1]/s[0][0];
double rmsJK=s[1][1]/s[0][0]-meanJ*meanK;
```

Exercise 10

- Use TUnfold to unfold the data using the MC1 detector response. Use 10,20,100 reconstructed bins and compare the results
- Code example

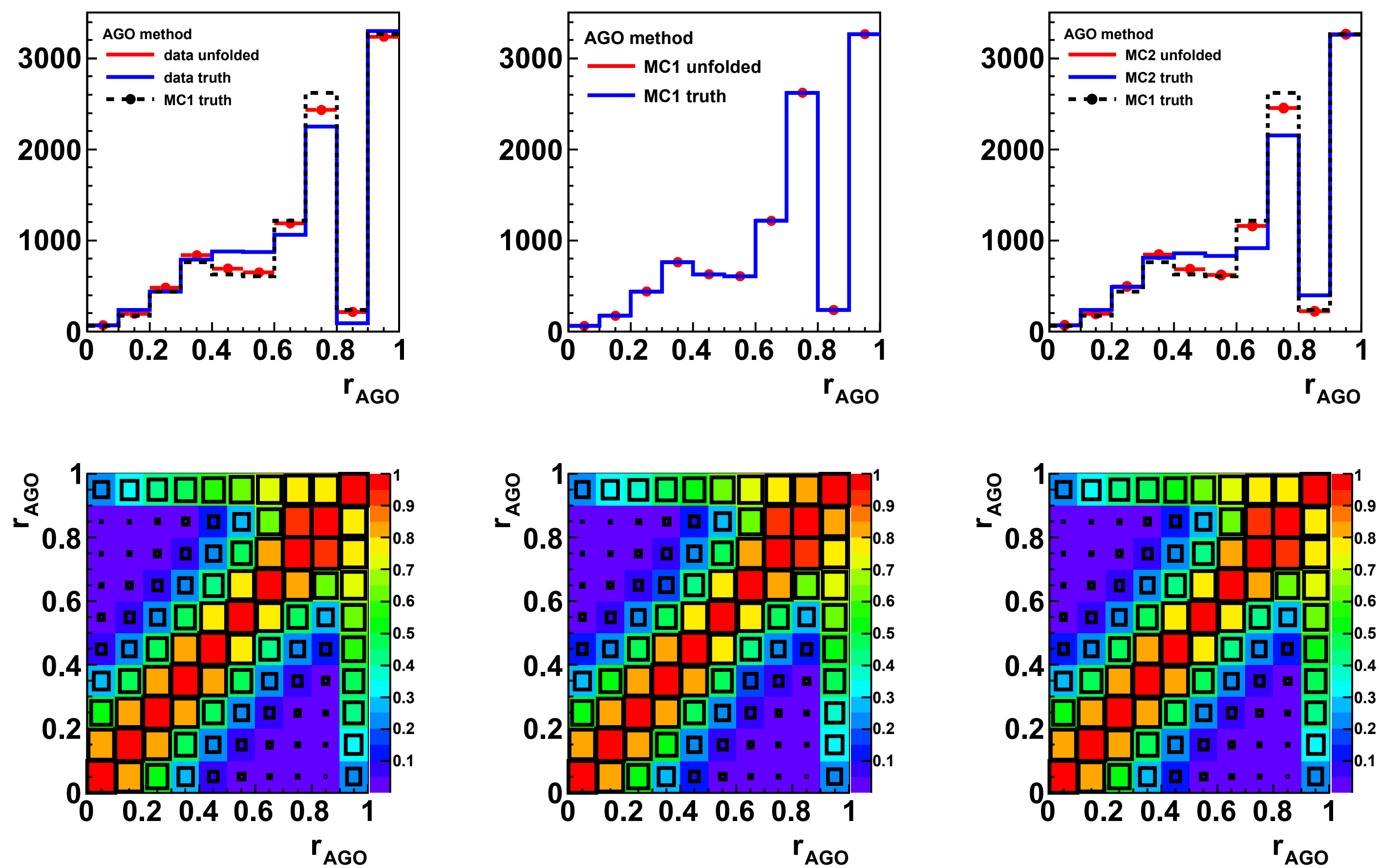
```
TUnfold unfold(hist_mc1_recgen,         // event counts from MC 2D histogram
          Tunfold::kHistMapOutputHoriz,   // gen events on x-axis
          Tunfold::kRegModeSize,          // explained later
          Tunfold::kEConstraintNone);     // not discussed in this talk
  unfold.SetInput(hist_data_rec);         // observed data 1D histogram
  Double tau=0.0;                         // explained later
  unfold.DoUnfold(tau);                   // run the unfolding
  unfold.GetOutput(hist_data_result); // get histogram with unfolded data
  unfold.GetRhoIJ(hist_data_rho);     // get 2D histogram of correlation coeff
```

Exercise 8 discussion

D'Agostini result is almost identical to the MC1 truth, independent of the data

Covariance: large positive correlations (rather expect negative correlations)

D'Agostini is the MC truth decorated with statistical fluctuations taken from data!



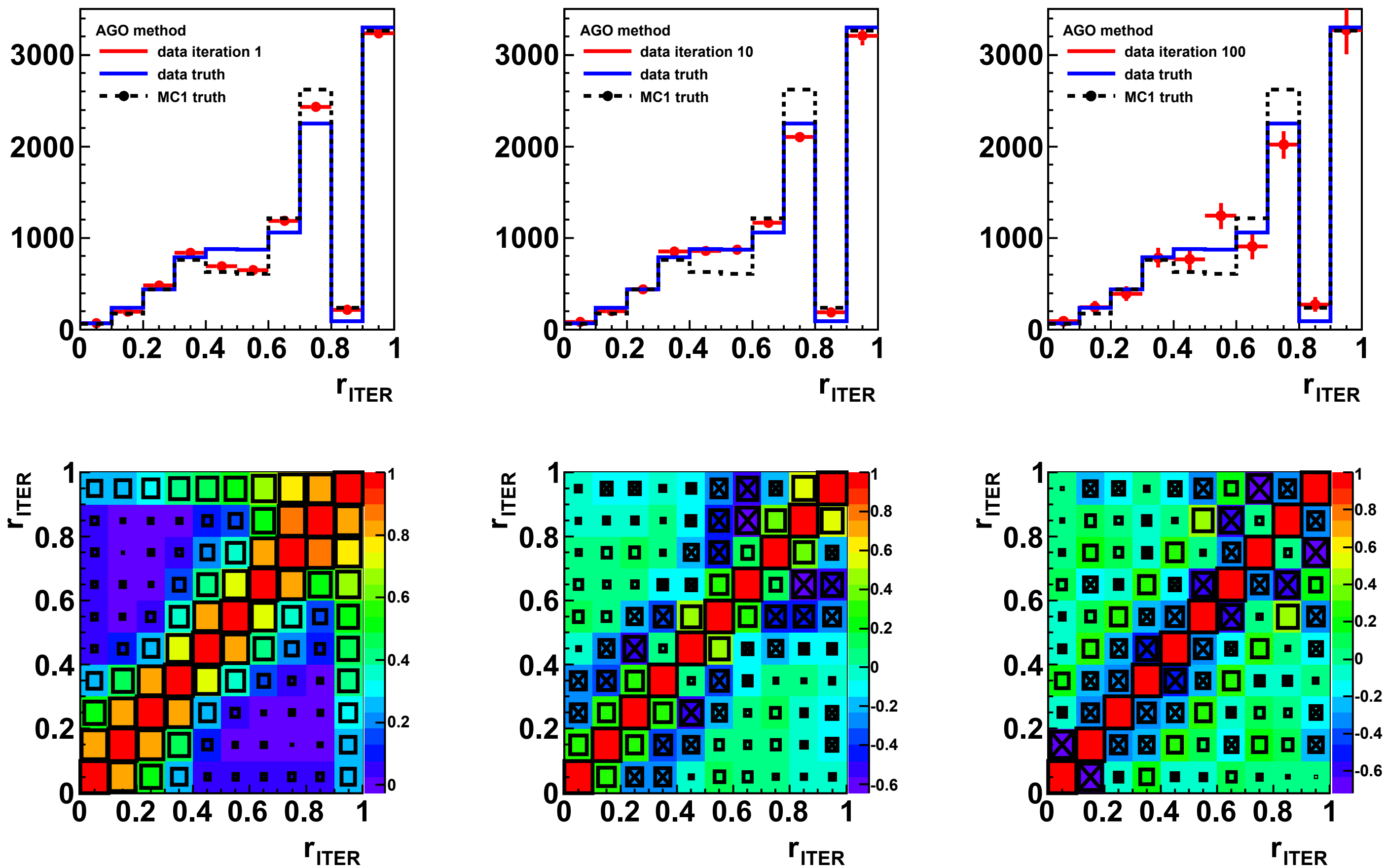
Do not use this method for your data analysis!!!

Exercise 9 discussion

After 10 iterations, result is “good”

After 100 iterations, results starts to oscillate → similar to inversion method

Unclear how many iterations are “good”. Danger to have a bias to MC which is difficult to quantify



Better not to use this for your data analysis

Iterative method: how often?

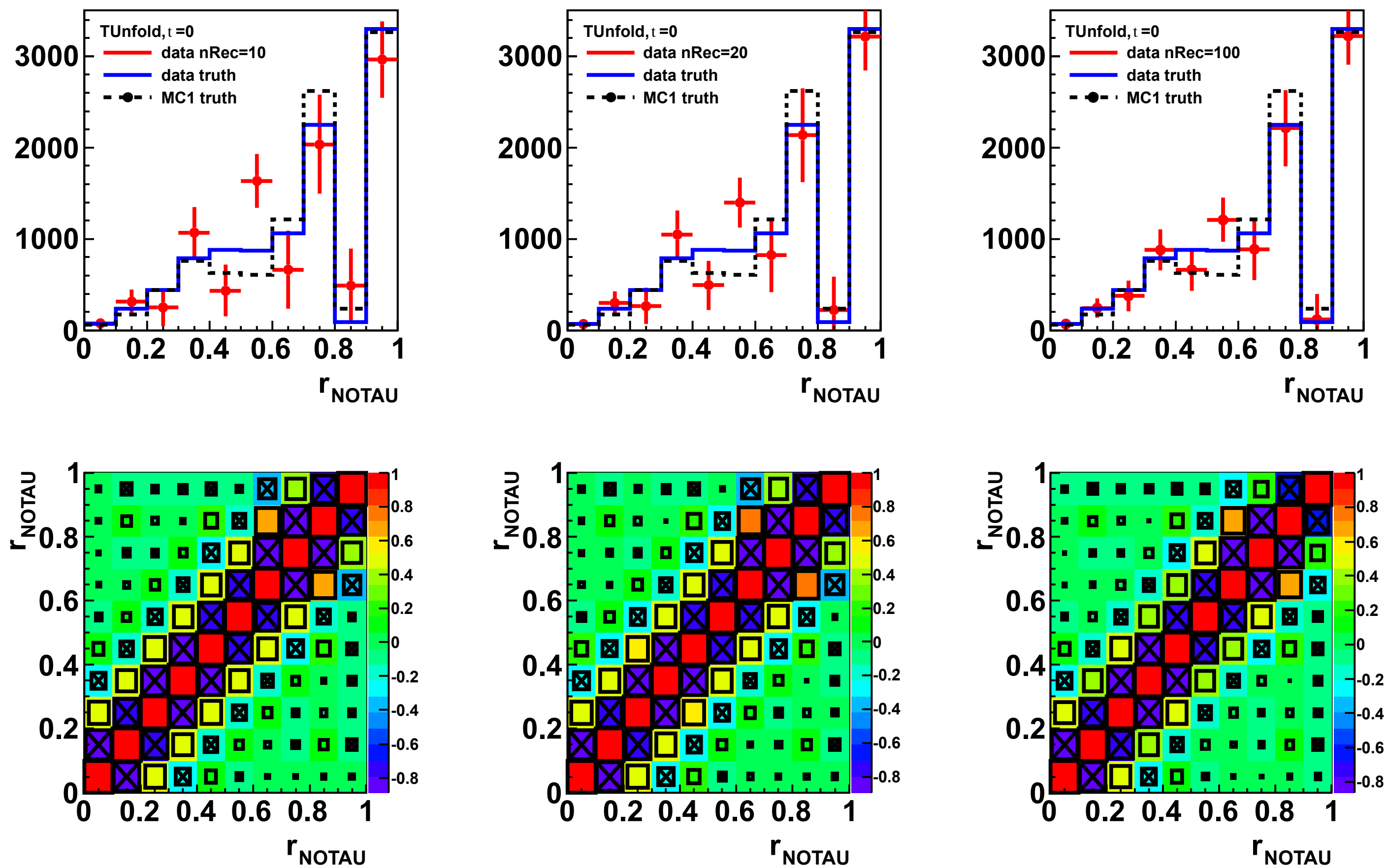
- Main question: when to stop the iteration?
- Also, does the iteration converge?
  - Answer: maybe. If it converges, one typically gets back the result from matrix inversion
- What people do:
  - Iterate until “it does not change anymore”
  - Or iterate until “the result becomes instable”
- Lack of definition of a good stopping condition

  → better do not use that method for your analysis

Discussion exercise 10

Results are simular to matrix inversion: oscillations and negative correlations

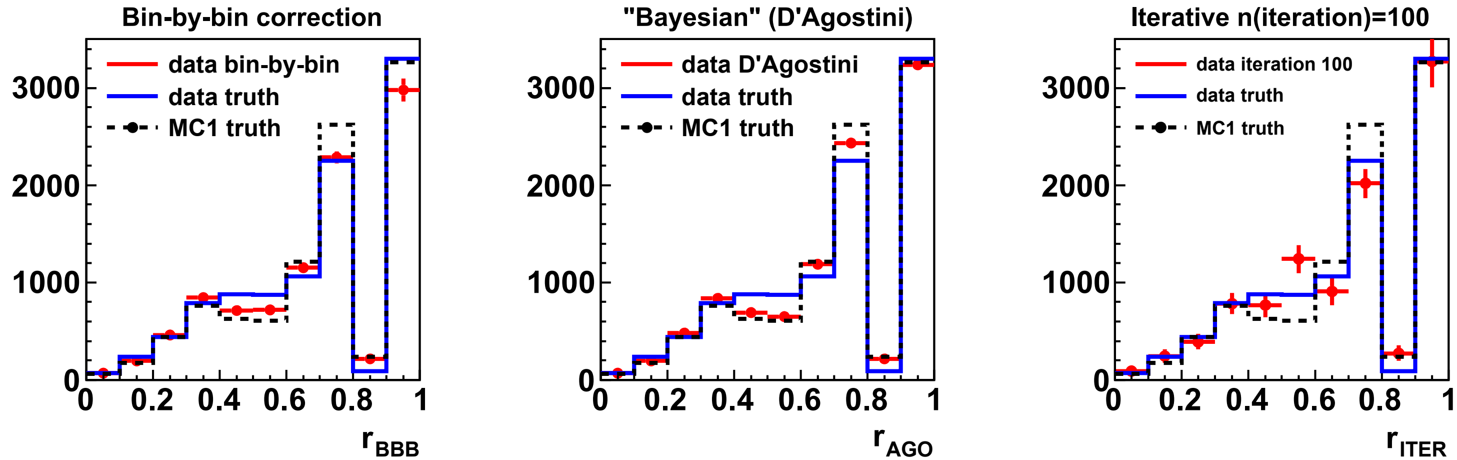
Some improvements are visible when using 20 or 100 bins

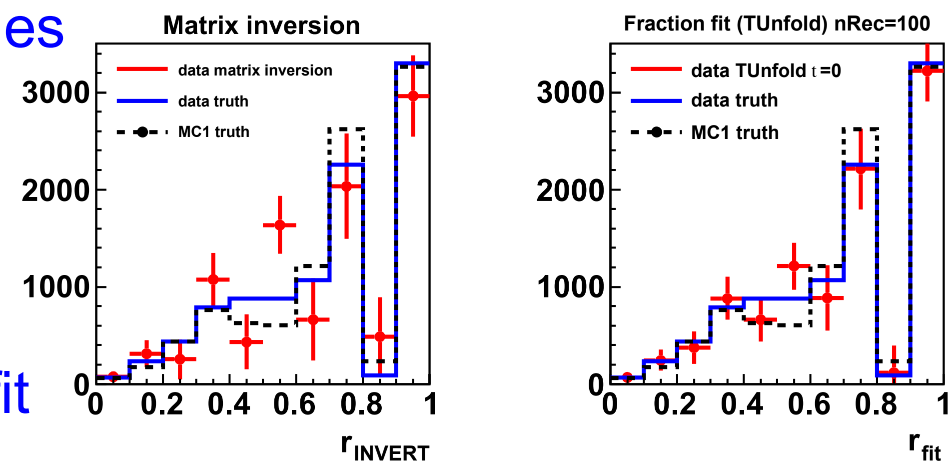


Visible improvements when using more bins, but still not satisfactory.

Comparison exercise 3,7,8-10

- None of these methods works satisfactory
- Bin-by-bin and d'Agostini have bias to MC truth
- Iterative properties are not well defined
- Matrix inversion and -to a lesser extent- fraction fit have oscillating solutions





Regularised unfolding

- Add extra term to the fit function "regularisation"
- The parameters x are constrained to be "similar" to x_b
- Strength of regularisation is given by parameter τ

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

x : vector of unknowns
y : vector of measurements
A : matrix of probabilities
V_{yy} : covariance matrix of y (uncertainties squared)
τ : regularisation strength
L : regularisation conditions
x_b : regularisation bias

Choice of Regularisation

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

τ : regularisation strength

L : regularisation conditions

x_b : regularisation bias

- Typical choice of L: unity matrix or “curvature” matrix
- Typical choice of x_b: “zero” or “MC truth”
- Typical choice of τ:
  - Eigenvalue analysis (TSVDunfold)
  - L-curve scan (TUnfold, TUnfoldDensity)
  - Minimize correlations (TUnfoldDensity)

TSVDunfold

This lecture: no exercise on TSVDUnfold
Use with the given example is not straight-forward

- Restrictions:
  - nRec=nGen
  - Regularisation always by curvature
- Some differences to TUnfold
  - e.g. definitions of L and τ
- Choice of regularisation
  - Parameter τ is calculated from Eigenvalue analysis
  - User has to define integer parameter kReg. See Höcker/Kartvelishvili (1995) for details

TUnfold: choices of L

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

τ : regularisation strength

L : regularisation conditions

x_b : regularisation bias

- Simplest choice: L=unity matrix, x-x_b is pulled to zero
- Curvature L=(-1,2,-1), derivative of (x-xb) is pulled to zero
- Effect: oscillations are damped. If x_b=0, pull x to zero (L=unity) or pull x to a straight line (L=curvature matrix)

$$\text{Curvature matrix: } L_{n\times n-2} = \begin{pmatrix} -1 & 2 & -1 & 0 & & \cdots & & 0 \\ 0 & -1 & 2 & -1 & & & & \vdots \\ \vdots & & & & \ddots & & & \\ 0 & & & & & -1 & 2 & -1 \end{pmatrix}$$

Choice of regularisation in TUnfold

- Three basic choices for matrix L
  - kRegmodeSize [L=unity matrix]
  - kRegmodeDerivative [L~(-1,1)]
  - kRegmodeCurvature [L~(-1,2,-1)]
- One basic method to determine τ
  - ScanLCurve()
- Note: new version of TUnfold (V17) provides another method to determine τ by minimizing correlation coefficients: ScanTau()

https://www.desy.de/~sschmitt/tunfold.html

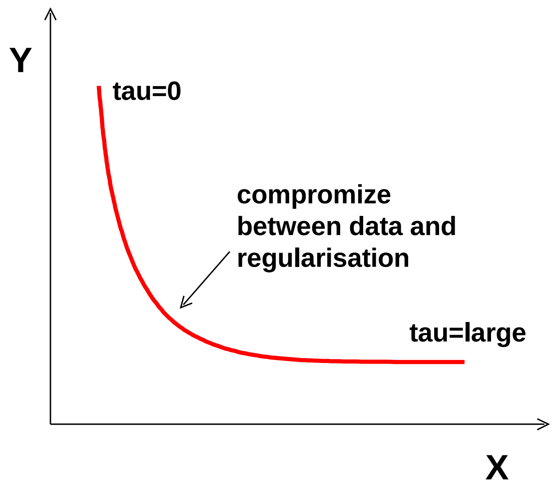
L Curve scan

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

- If τ is zero, χ^2_A is minimized and χ^2_L is large
- If τ is very large, χ^2_L is minimized and χ^2_A is large
- Parametric plot of

  $X=\log_{10}(\chi^2_A)$ vs $Y=\log_{10}(\chi^2_L)$ is L-shaped
- “Best” compromise:

  kink position (largest curvature)



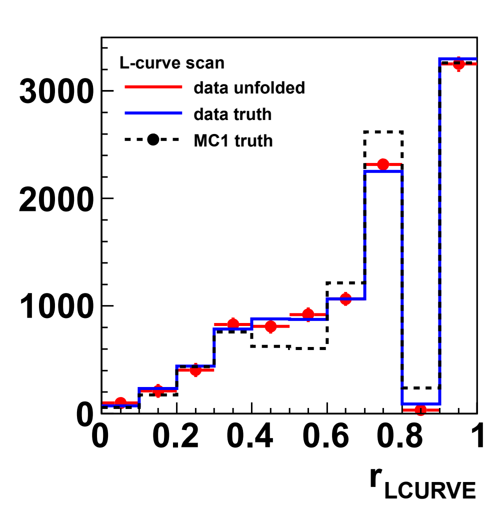
Exercise 11: L curve scan

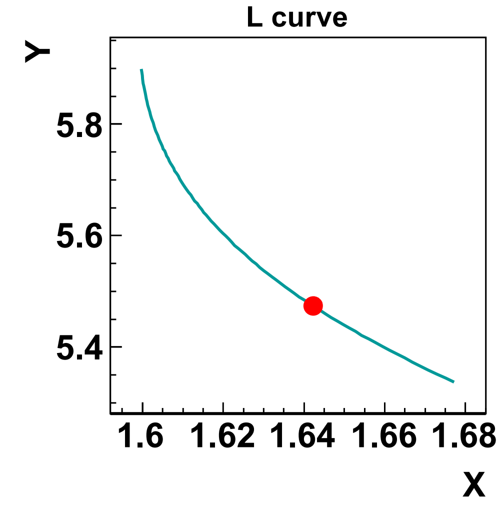
- Run TUnfold to unfold the data, using the response matrix from MC1 (nRec=20, nGen=10)
- L=1 and x_b=MC(truth)
- Compare the result to truth
- If there is time:
  - Show correlations
  - Show L-curve
- Discuss

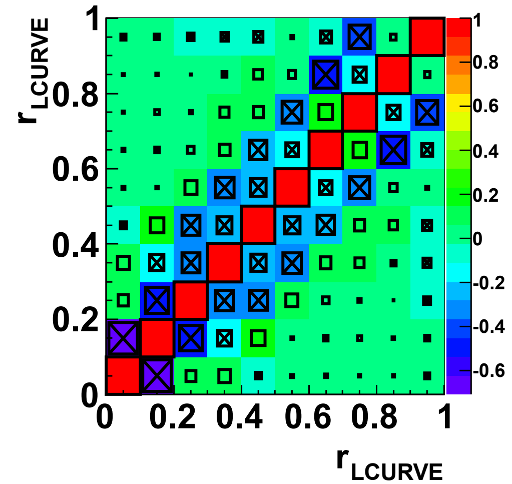
```
// run the unfolding and retreive results
TUnfold unfold(hist_mc1_recgen,
               TUnfold::kHistMapOutputHoriz,
               TUnfold::kRegModeSize,
               TUnfold::kEConstraintNone);
unfold.SetInput(hist_data_rec,1.0);
TGraph *lcurve=0;
TSpline *logTauX=0,*logTauY=0;
unfold.ScanLcurve(100,0.,0.,&lcurve,&logTauX,&logTauY);
unfold.GetOutput(hist_data_LCURVE);
unfold.GetRhoIJ(hist_data_LCURVErho);
double tau=unfold.GetTau();
double logTau=TMath::Log10(tau);
double lcurveX=logTauX->Eval(logTau);
double lcurveY=logTauY->Eval(logTau);
```

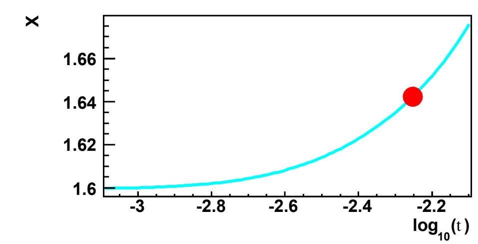
Exercise 11 discussion

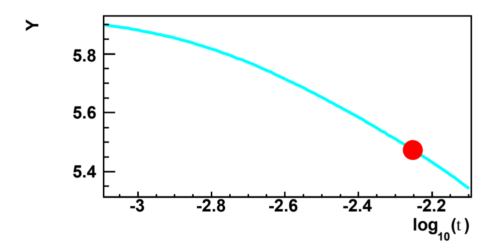
- Bias to MC is small
- Result is very good
- Uncertainties have reasonable size
- Covariance matrix with moderate correlations
- Choice of “best” point on L-curve is difficult to understand because range of X and Y axis is different









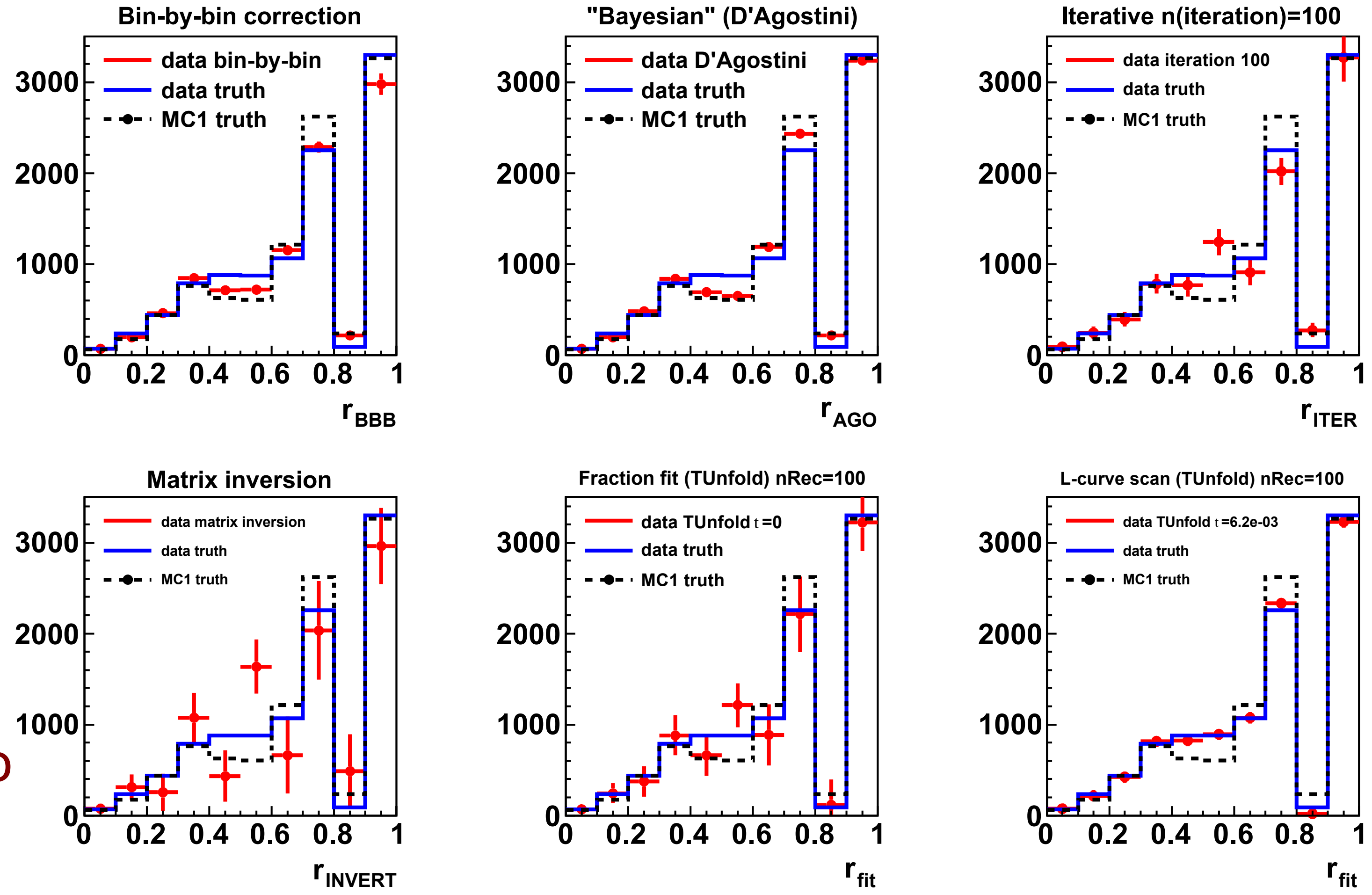


L curve scan: caveats

- Possible problems with $X=\log_{10}(\chi^2_A)$
  - If nRec=nGen, then χ^2_A is zero [if τ=0]
  - If MC(rec) is used as "data", then χ^2_A is zero because MC(truth) reproduced exactly the "data"
- Rules when using TUnfold with L-curve scan
  - Always use: nRec>nGen
  - Toy studies can not be done using the exact MC distribution which was used to build the response matrix $\rightarrow$ apply extra statistical fluctuation to the MC or use independent samples

Comparison of unfolding methods

- Bin-by-bin and D'Agostini do not work (bias)
- Iterative method: difficult to define end condition
- Matrix inversion and fraction fit: oscillations
- Regularized unfolding seems to work best!



Summary of unfolding methods

- Strong bias to MC truth. Do not use
  - Bin-by-bin
  - Bayesian
- Unclear bias to MC truth. Better not to use
  - Iterative “Bayesian”
- No bias but oscillations and large anti-correlations
  - Matrix inversion, fraction fit
- Small bias, oscillations damped
  - Regularised unfolding with proper choice of τ

  $\rightarrow$ **TUnfold TSVDUnfold**

References for unfolding methods

- 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]
  - https://www.desy.de/~sschmitt/tunfold.html
- Collection of talks by V.Bobel
  - https://www.desy.de/~blobel/unfold.html

Unfolding problems specific to HEP

- Background
  - See exercise 7: extra bins to extract background normalisation
- Multidimensional histograms
  - exercise 12 and 13
- Phase-space boundaries [only one slide]
- Measurements of multiplicities [not covered in this talk]

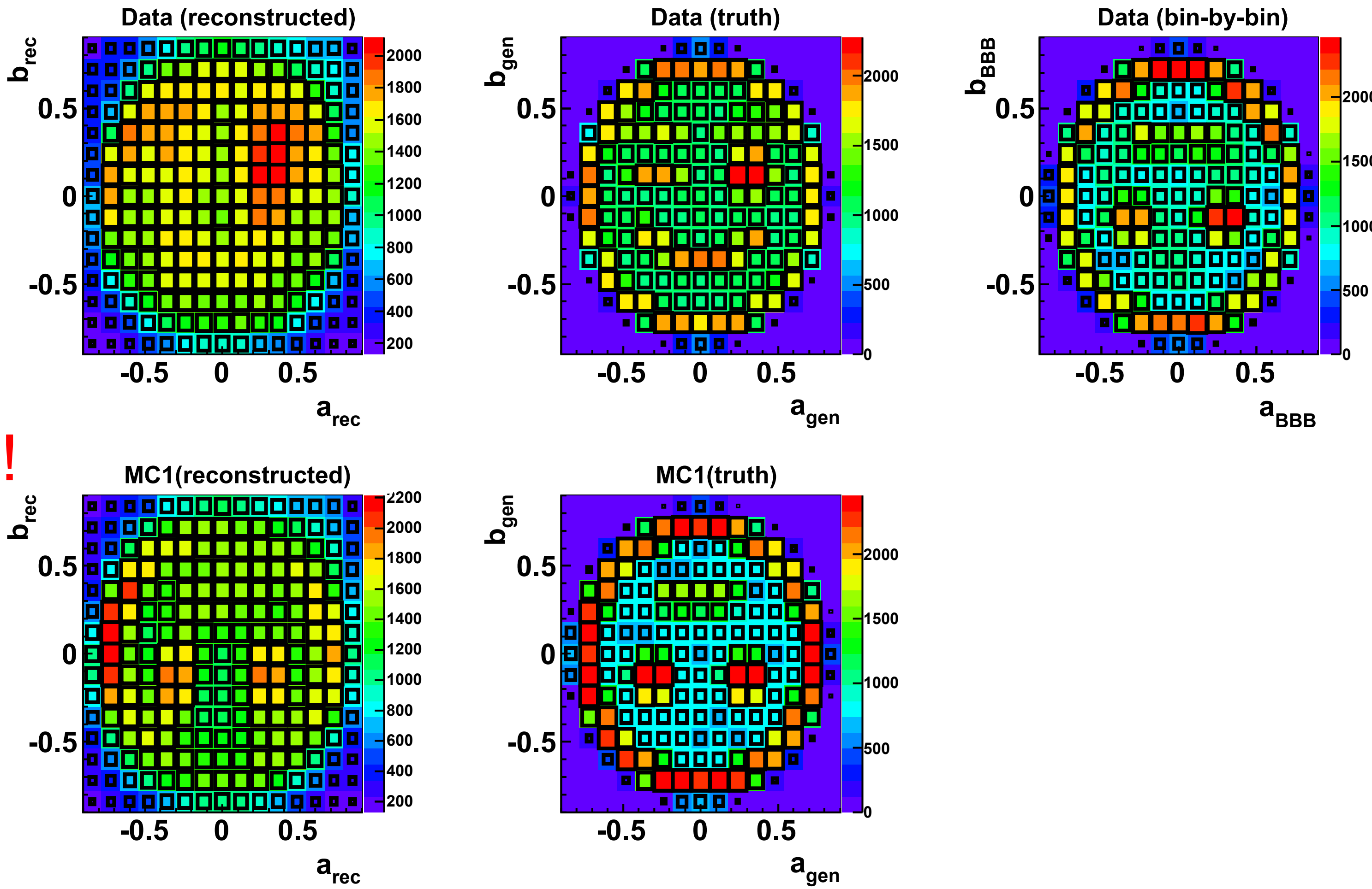
Exercise 12: 2D unfolding

- Make 2D plot of the variables (a,b) [use full TTree]
  - aRec=rRec*sin(pRec) and bRec=rRec*cos(pRec)
  - aGen=rGen*sin(pGen) and bGen=rGen*cos(pGen)
- $-0.9<a,b<0.9$
- 15x15 bins for aRec, bRec
- 15x15 bins for aGen,bGen
- Unfold the data bin-by-bin
- Compare the histograms

Exercise 12 discussion

Bin-by-bin unfolding fails completely: result is MC truth, has nothing to do with data truth!

What about regularized unfolding?



Multi-dimensional matrix unfolding

- Basic formula defines only one dimension for reconstructed bins and another dimension for generated bins

$$y_i^{\text{data}} \underset{\text{stat.fluct}}{\sim} \mu_i = \sum_{j=1,nGen} A_{ij} x_j^{\text{truth}}$$

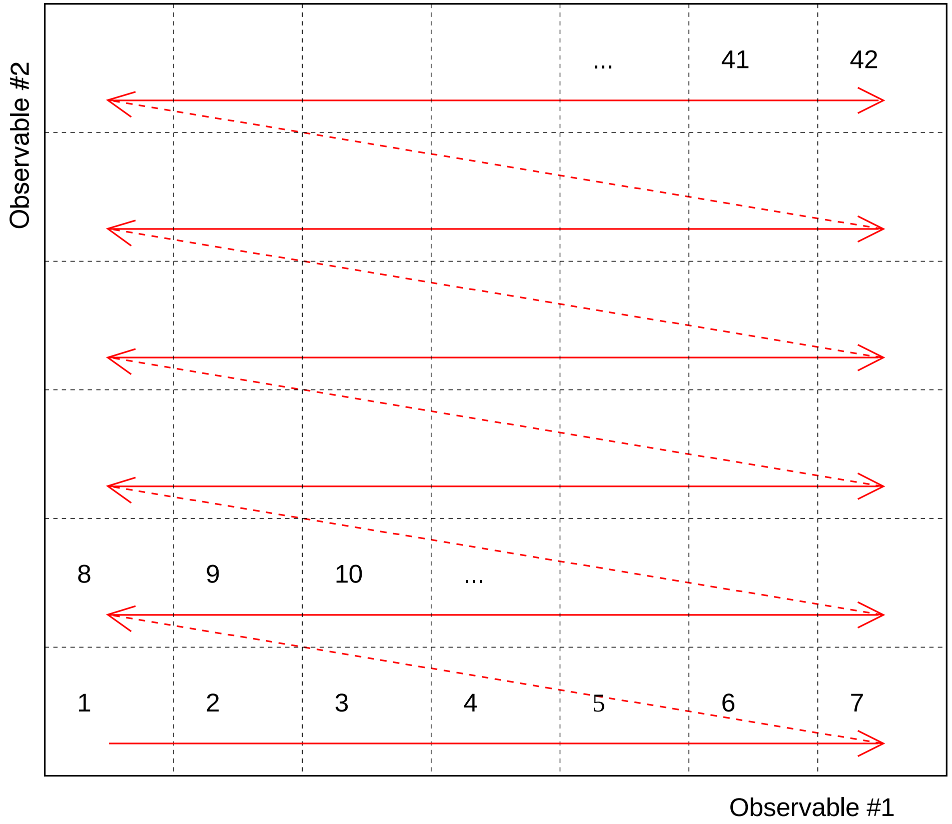
- The equation does not care how the bins are arranged
- Recall: we already added one extra bin for background normalisation

Multi-dimensional histograms

- No fundamental difference between unfolding one-dimensional or multi-dimensional histograms

$$\mu_j = \sum A_{ij} x_i^{\text{truth}}$$

- Sum runs over all bins i, no matter how they are arranged
- Example: order bins as shown to the right



Problem: algorithms which use curvature regularisation (TSVDUnfold) may calculate the wrong curvature (in this example between the bins 7,8 or 14,15 etc)

TUnfoldBinning

- 2D problems have to be mapped to 1D
- Mapping 2D to 1D is complicated ($\rightarrow$ error prone)
- V17.3 of TUnfold provides class TUnfoldBinning
- Binning scheme may be defined in xml language
- Event loop: get bin number from TUnfoldBinning, then fill histograms with TUnfold internal binning

- After unfolding: retreive results in histograms with user binning
- For running the unfolding, use the class TUnfoldDensity which is able to deal with binning schemes

Exercise 13

- Look at XML file “exercise13binning.xml”
- Look at stand-alone program “exercise13.C”
- Compile and run program exercise13:
  - make exercise13prog
  - ./exercise13prog
- Results are written to “exercise13.root”
- Extend macro from exercise12 to also plot the histogram: hist_data_unfold
- Compare bin-by-bin and regularized unfolding results

TUnfold V13 is not available in root 5.34

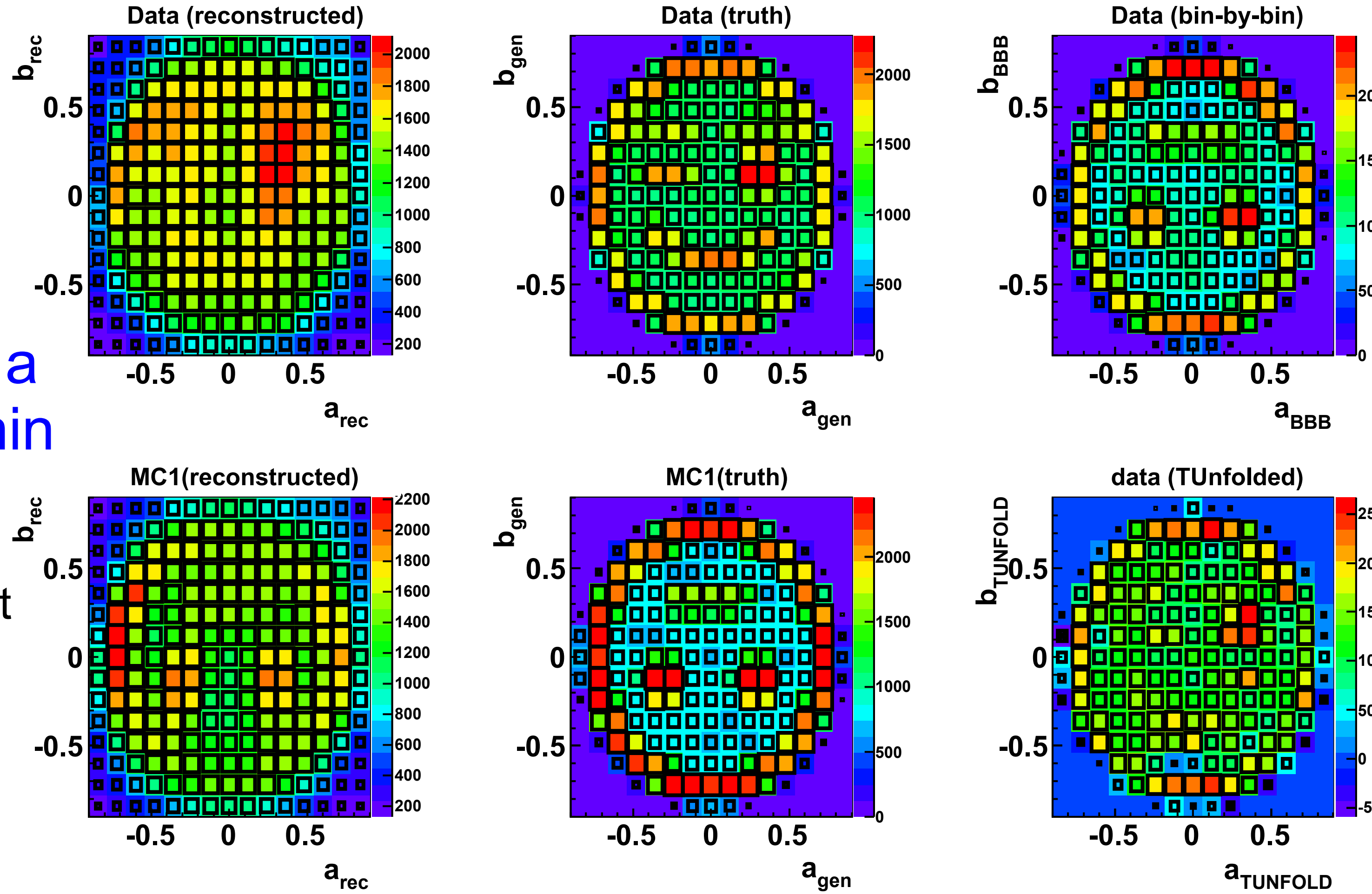
Can be linked with root libraries as shown in the example13prog

Exercise 13 discussion

2D unfolding: bin-by-bin fails (exercise 12)

Regularized unfolding gives a good result within uncertainties

(uncertainties are not shown here)



Phase space boundaries

- Analysis in HEP often have complicated phase-space definitions
- Measurement is differential in one variable [v], but there are also cuts in other variables [e.g. w0<w<w1]
- Two options
  - Subtract background from w<w0 and from w>w1 prior to unfolding TUnfoldSys::SubtractBackground()
  - Or use extra generator bins w<w0 and w>w1 to unfold background contributions from data → compare to our example of unfolding the background normalisation from the data

Summary and Conclusions (1)

- Unfolding: “correct” measurements for detector effects. Our daily business in HEP analysis
- Many different methods:
  - Strong bias to MC: bin-by-bin and “Bayesian” without iterating. Do not use
  - Unknown bias: iterative method. Better not use
  - Unbiased, large errors and large correlations: matrix inversion, fraction fits.
  - Small bias to damp oscillations: regularized fits
    - TSVDUnfold: eigenvalue analysis
    - TUnfold: L-curve scan

Summary and Conclusions (2)

- Problems specific to unfolding in HEP
  - Multidimensional distributions
  - Complicated phase-space definition
  - Measurement of multiplicities [not discussed in this talk]
  - Systematic uncertainties [not discussed in this talk]
- Most problems with unfolding in HEP are related to the choice of bins, inside and outside the phase-space
- For complex binning schemes, the class TUnfoldBinning may be useful (not available in root 5.34 → standalone program needed)