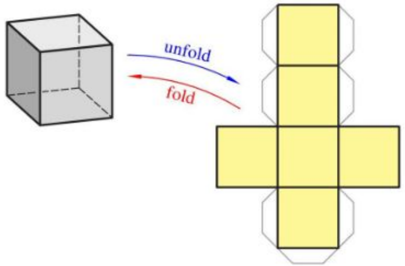
Unfolding in HEP

Hands on session 2

Arantza Oyanguren Campos, Olena Mezhenska, Dominik Babal, Rafał Staszewski

Introduction



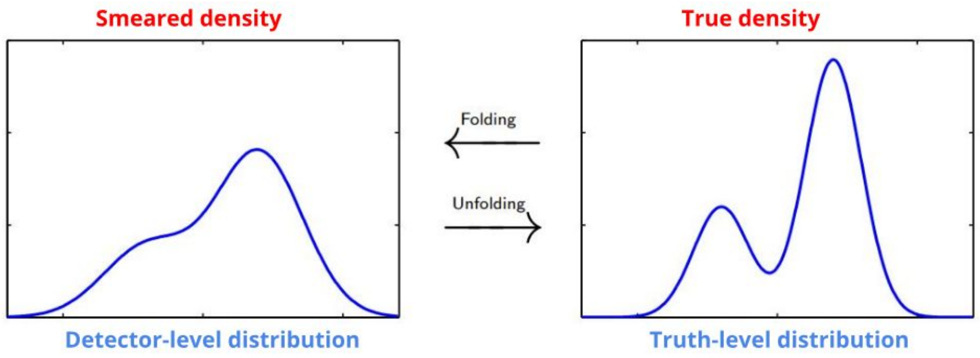
- **What you will learn in this Hands on:**
  - Introduction to **unfolding and its usage in HEP**
  - Practical **application of unfolding techniques** based on first Hands on exercise
  - Complete (but very simplified) **example of differential cross section measurement**

- **Useful resources to get started with unfolding:**
  - S. Schmitt - Data Unfolding Methods in High Energy Physics
  - G. D'Agostini - Improved iterative Bayesian unfolding
  - Slides from G. Cowan - Statistical Methods for Particle Physics - Lecture 4 unfolding
  - Slides from M. Kuusela - Introduction to Unfolding in High Energy Physics
  - S. Biondi - Experience with using unfolding procedures in ATLAS

What is unfolding in HEP ?

- Unfolding is the **procedure to remove effects of imperfect physical measurement** in order to extract "truth" distribution of the given quantity

- In HEP these effects can be caused by:
    - Finite detector resolution
    - Efficiency and acceptance corrections - some events are not reconstructed in the detector
    - Reconstruction of final state particles - techniques are not perfect, some particles can be wrongly reconstructed
    - Parton-to-particle evolution - QCD, QED radiative corrections, parton fragmentation, particle decay

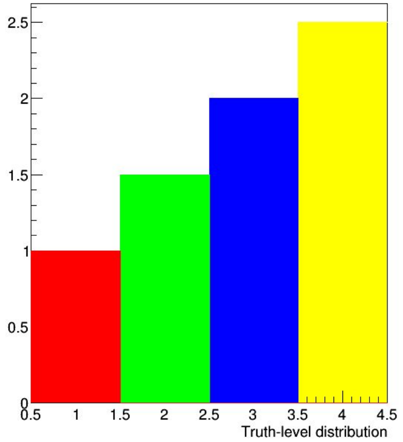


How to get from reconstructed to truth spectrum?

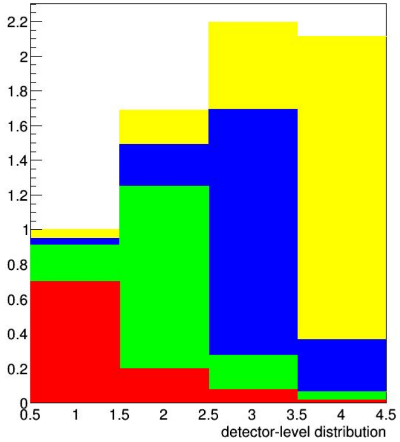
- When working with **binned distributions**, reco (detector) spectrum ***d*** and truth spectrum ***t*** are represented by vectors
- We need to find some **transfer matrix** so that:

$$d_i = \sum_j M_{ij} t_j$$

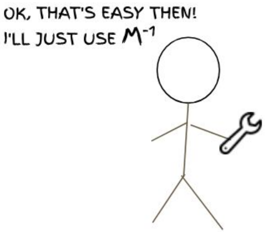
- Matrix is usually called **Migration matrix** and is obtained from MC samples where we know both reco and truth spectrum
- It represents **probability of the event which originate from truth bin *j* to be reconstructed in bin *i***



$$\times \begin{pmatrix} 0.7 & 0.2 & 0.08 & 0.02 \\ 0.14 & 0.7 & 0.13 & 0.03 \\ 0.02 & 0.12 & 0.71 & 0.15 \\ 0.02 & 0.08 & 0.20 & 0.7 \end{pmatrix} =$$

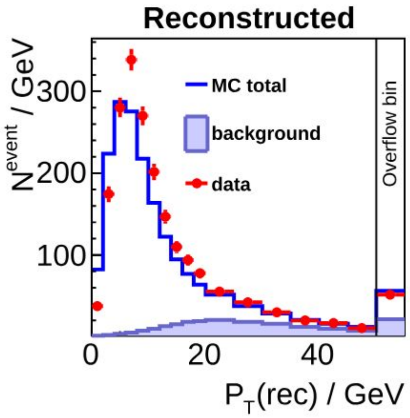


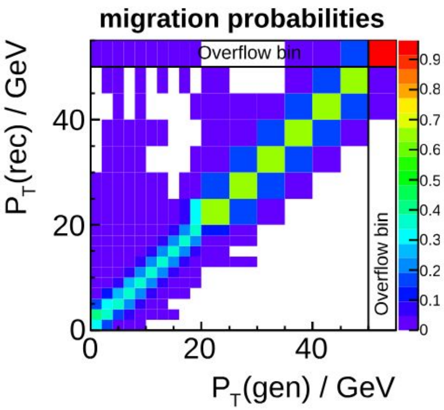
Naive solution: Matrix inversion

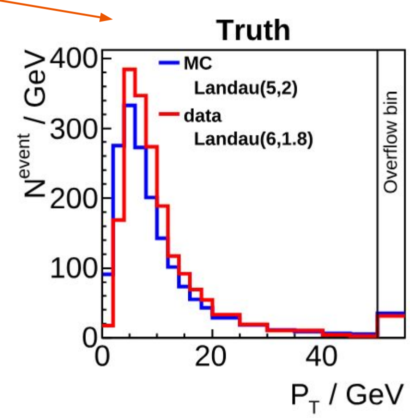


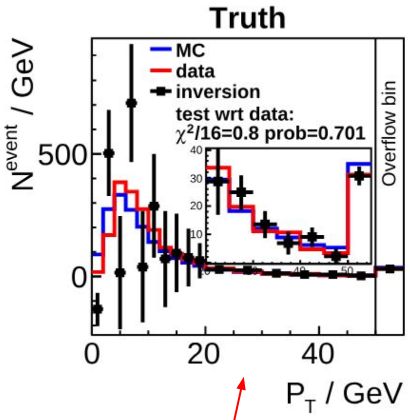
- Why not to simply solve it by **inverting migration matrix**:

$$t_j = \sum_i M^{-1}_{ij}\, d_i$$









Unfolded distribution

- Matrix inversion can in some cases produce **very large oscillations between neighboring bins** - statistical fluctuations are amplified
- Problem is related to **large off-diagonal elements in the migration matrix**

How to mitigate oscillations?

- Several ways how to **eliminate amplification of statistical fluctuations**:
    - **Ignore off-diagonal elements in migration matrix**
        - Method called **bin-by-bin unfolding** - use correction factors obtained only from the diagonal elements - bin-by-bin factors M_{ii} for each reco bin i
        - Requires same subdivision of bins on reco and truth + can't take into account large migrations
    - **Reduce off-diagonal elements in migration matrix**
        - Improve detector resolution
        - Optimise binning of the observable - compromise between diagonality, statistical uncertainty and measurement sensitivity
    - **Add more information**
        - Maximum-likelihood unfolding with more reco bins than truth bins
    - **Eliminate oscillations through introducing regularisation**

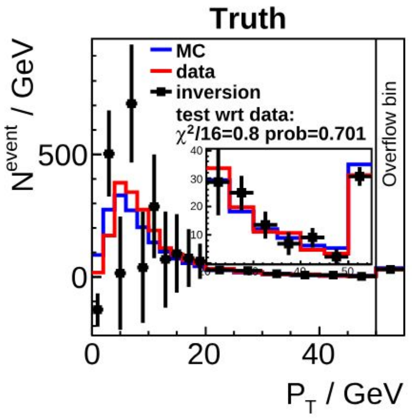
- Note: Only unbiased method is matrix inversion - no regularisation, no external constraint (bias) by e.g. prior distribution

Regularised unfolding

- <u>Regularisation can be different forms:</u>
    - Matrix inversion is numerically equivalent to **maximum-likelihood fit**

$$L(t_j) \sim \exp[-\sum_i (d_i - \sum_{ij} M_{ij} t_j)^2]$$

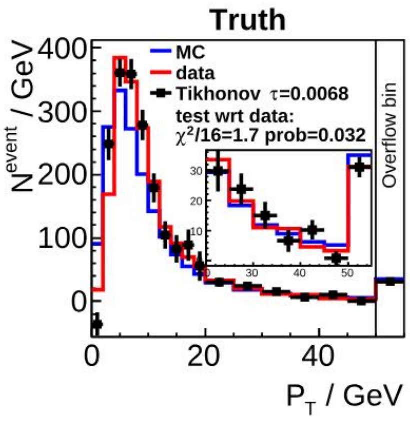
    - where t_j is **free-floating** and its best-fit value represent unfolded distribution
    - Regularisation can be introduced by **penalty term** added to the exponent (**Tikhonov regularisation**)



$$\chi_R^2 = (t - t^0)^T R^T R (t - t^0)$$

regularization condition matrix
(more often used **2nd derivative constraint**)

bias vector (truth MC distribution)



Iterative Bayesian Unfolding

- Different regularisation technique with **iterative approach**
- Proposed by D'Agostini in 1995 (improved in 2010)
- Robust method that **does not require matrix inversion**
- Based on Bayes theorem:

Likelihood - probability that if event is reconstructed in bin ***j*** than it originates from truth bin ***i*** (Migration matrix) - **from MC**

Prior - probability of the truth spectrum - **from MC**

$$P(T_i|R_j) = \frac{P(R_j|T_i)P_0(T_i)}{\sum_{l=1}^{n_T} P(R_j|T_l)P_0(T_l)},$$

Posterior - conditional prob. to observe event originating from truth bin ***i*** in reco bin ***j***

Normalisation term - to ensure that posterior is normalized to 1

How can Bayes give me truth?

- Estimator of the truth (unfolded) distribution is then:

Posterior - from Bayes

Truth $$\hat{n}(T_i) = \frac{1}{f_i}\sum_{j=1}^{n_R} n(R_j)P(T_i|R_j),$$

Reco

- where $0 \le f_i \equiv \sum_{j=1}^{n_E} P(R_j|T_i) \le 1$ is **efficiency term** (not every event need to be reconstructed)
- It may seem that posterior **is dependent on prior truth MC**… **But:**
  - As a by product of the method we get also **estimator of the total number of truth events**:

$$\hat{N}_{true} = \sum_{i=1}^{n_T} \hat{n}(T_i).$$

This suggests to proceed iteratively!!!

  - And thus we can estimate **new prior probability** as:

$$\hat{P}(T_i) = \frac{\hat{n}(T_i)}{\hat{N}_{true}}.$$

Iterative approach

- **Just follow these steps:**

1. In the first iteration choose initial prior $P_0(T)$ - either truth distribution or it can be started also from uniform distribution $P_0(T_i) = 1/n_T$
2. Calculate estimator of the truth spectrum $\hat{n}(T)$ and the new prior $\hat{P}(T)$
3. Make a χ^2 test between $\hat{n}(T)$ and $n_0(T)$ $(n_0(T_i) = P_0(T_i)N_{obs})$ in case of first iteration, or between previous and current estimators of the truth spectrum
4. If χ^2 is not reasonable small, replace $P_0(T)$ by $\hat{P}(T)$ and $n_0(T)$ by $\hat{n}(T)$ and start the process again with next iteration

- **Important:** **Number of iterations plays the role of regularisation parameter** and needs to be optimised
  Too many iterations lead to amplification of statistical fluctuations - same as for matrix inversion
  When $N_{iter} \to \infty$, unfolding becomes unregularised (equivalent to M inversion)

Other unfolding methods

- Singular Value Decomposition (SVD)
    - Mitigate problem with migration matrix inversion by **decomposing** it to USV^T where three new matrices are: ,
      $U = m \times m$ $V = n \times n$ $S = m \times n$
- Fully Bayesian Unfolding (FBU)
    - Similar to IBU but **does not use iterations** and convergence to truth
    - Answer is not estimator of truth spectra, but posterior **probability density defined in the space of possible spectra**
    - Regularisation done by choosing prior which favors certain characteristics (e.g. smoothness)
- Profile Likelihood Unfolding (PLU)
    - Based on the **profile likelihood fit -** each bin of the reco spectrum represented by the **free-floating normalisation factor** that is fitted in maximum-likelihood fit

        - **The binned profile-likelihood**:

data

Poisson

Gaussian
(or other pdf...)

$$L(\vec{n} \mid \vec{\theta}, \vec{k}) = \prod_i P(n_i \mid \underbrace{S_i(\vec{\theta}, \vec{k}) + B_i(\vec{\theta}, \vec{k})}) \times \prod_j \underbrace{G(\theta_j)}$$

constrained parameters:
nuisance parameters (**NPs**)
associated to systematic
uncertainties

data events
in bin *i*

prediction in bin *i*
(*signal+background*)

constraint term
for nuisance
parameter *j*

unconstrained parameters:
parameter of interest (**POI** or “μ”) + unconstrained nuisance
parameters (e.g. background normalization parameters)

    - Naturally takes into account **systematic uncertainties** (constrained by priors)
    - **Preferred method used nowadays**

Unfolding Exercises

Exercise 1: Preparation for the unfolding

- Get the code from https://rafal.web.cern.ch/teshep2025_unfolding.tar.gz
- First you need to get **RooUnfold package** - extension of ROOT containing methods for unfolding
  - Clone repository from https://gitlab.cern.ch/RooUnfold/RooUnfold
  - Follow instructions under “Building the Library”
  - Try that your compilation was successful by opening python interpreter and try to “import RooUnfold”
- Now you are ready to run **preparation step** for the unfolding:
  - Inputs need to be processed by running ./process.py for each sample(signal, bckg and data)
  - Then run ./unfold.py histos_data.root histos_sig.root histos_bkg.root unfolding.pdf --prepare
  - Investigate the output saved to .pdf and .root
  - **Questions:**
    - How is the composition of signal and background?
    - Which selection do we use for the unfolding?
    - Is migration matrix diagonal?
  - **Action items:**
    - Try changing selection in the input ntuples - try “all”, “iso”, and “SR” - Which do you think provide best inputs for unfolding? Why?
    - Try changing the number of bins in reco and truth M_{ll} - does it change migrations?
    - How do we interpret efficiency and acceptance corrections?

Exercise 2: Running Unfolding

- Run unfolding step with:
    - ./unfold.py histos_data.root histos_sig.root histos_bkg.root unfolding.pdf --prepare --unfold
    - Investigate the results:
        - Which method provides better results, matrix inversion or IBU?
        - Is matrix inversion stable? Why?
        - How many iterations do we use for IBU?
        - Compare the statistical uncertainties for both methods and Asimov vs. data
    - Action items:
        - Try to generate small (5k) signal and background samples, but apply larger smearing for pT, eta and phi in the detector simulation step.
            - Hint: Use script ./generate.py and change sigma of the random normal sampling for pT, eta and phi. For this you need to install pythia from https://pythia.org/
            - Then create data.root by mixing signal and background events, use ./mix_events.py
            - Finally process all new samples (signal, background and data) and run unfolding
        - Rerun the unfolding
        - How did the output change? What are the migrations now? Is matrix inversion still stable? How does it compare to IBU result?

Exercise 4: Differential cross section measurement

- **Differential cross section of variable X** in bin ***i*** can be calculated using following formula:

$$\frac{d\sigma}{dX^i} = \frac{1}{\mathscr{L} \cdot \mathscr{B} \cdot \Delta X^i \cdot f^i_{\rm acc}} \cdot \boxed{\sum_j R^{-1}_{ij} \cdot \epsilon^i_{\rm eff} \cdot \left(N^j_{\rm obs} - N^j_{\rm bkg} \right)}$$

luminosity | branching ratio | bin width | acceptance correction (signal MC) | response matrix | efficiency correction (signal MC) | data | background (MC)

$$f_{\rm acc} = \frac{N^{\rm reco \wedge truth}}{N^{\rm truth}}$$

$$\epsilon_{\rm eff} = \frac{N^{\rm reco \wedge truth}}{N^{\rm reco}}$$

Luminosity is stored in data sample

Branching ratio =1 in case of particle level

This is your unfolding result

Bin width of the unfolded distribution

Exercise 4: Differential cross section measurement

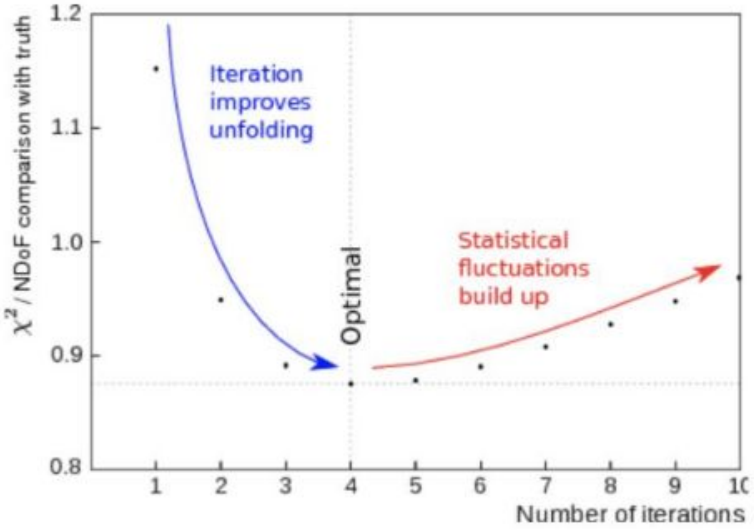
- **Action item:**
  - Go to unfold.py and implement calculation of the differential cross section from unfolded and truth distributions
  - You need to write the code here:

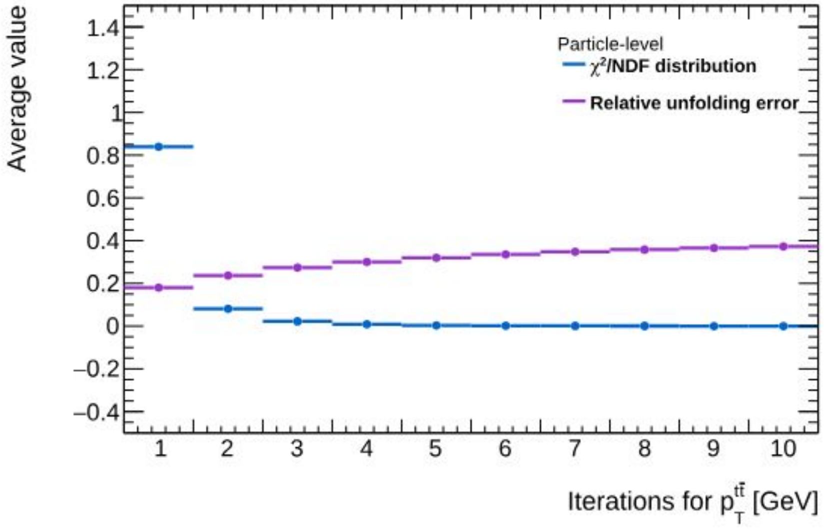
```
def calculate_cross_section(h_unfolded, h_truth, lumi, out_pdf, canvas, data = False):
    ###################################
    print("YOU NEED TO IMPLEMENT THE CODE FOR CROSS SECTION")
    sys.exit()
    # First Clone the unfolded and truth distributions (to not break them with following operations), use Clone() method from ROOT
    # Then scale down both histograms by luminosity, use Scale() method from ROOT
    # Finally divide both distributions by bin widths, you can do it manually, or use "divide_by_bin_width()" from this code
    # Don't forget to remove these two lines from above :)
    # print("YOU NEED TO IMPLEMENT THE CODE FOR CROSS SECTION")
    # sys.exit()
    ###################################

    # Draw plot of differential cross section
    draw_diff_xs(h_unf, h_truth, out_pdf, canvas, data)

    return h_unf
```

- Run the code as in the previous exercise but now add --xsec flag at the end
- Compare the total cross section obtained by summing bins of differential spectrum with predicted value from MC. Also compare results for data and Asimov

Exercise 5: Optimising number of iterations

- **Number of iterations** serves as **regularisation parameter of IBU**
- When $N_{iter} \to \infty$ **unfolding becomes equivalent to matrix inversion** and statistical fluctuations can be amplified - therefore it is important to choose optimal number of iterations





Exercise 5: Optimising number of iterations

- Action item:
  - Here we want to find compromise between number of iterations (good convergence) and statistical uncertainty of the unfolding
  - Go to ./unfold.py and try to implement iteration test according to the instructions:

```
def run_iteration_optimisation(data, response, acceptance, background, truth, output_pdf, canvas):
    # Define max number of iterations to test
    max_iter = 10
    chi2_values = []
    error_values = []

    # Create histograms
    pseudo_data = data.Clone()

    # Create pseudo data - smear each bin of the data distribution (MC) with Poisson
    random = ROOT.TRandom3(142857)
    for bin_x in range(1,pseudo_data.GetNbinsX()+1):
        smeared_value = random.Poisson(pseudo_data.GetBinContent(bin_x))
        pseudo_data.SetBinContent(bin_x, smeared_value)
        pseudo_data.SetBinError(bin_x, (pseudo_data.GetBinContent(bin_x))**0.5)

    # Unfold the pseudo data for particular number of iterations
    ##########################
    print("YOU NEED TO IMPLEMENT THE CODE FOR ITERATION TEST")
    sys.exit()
    # In a for loop going from 1 to max_iter:
    #     - unfold the pseudo data above (smeared asimov data) with particular number of iterations and
    #     - take only unfolded bayes distribution and calculate mean of the relative unfolding error over all bins of unfolded spectra
    #     - Hint: to do this use methods GetBinError(), GetBinContent() and calculate mean by summing relative errors
    #       (relative error is error in the bin divided by its content)
    #     - save mean error to list error_values defined above
    #     - calculate chi2 between unfolded spectrum with N iterations and N-1 iterations, in case of first iteration use truth spectrum
    #     - For calculating chi2 use method Chi2Test(compared_hist,"WWCHI2/NDF") from ROOT
    #     - save chi2 value to list chi2_values
    ##########################

    # Draw results
    draw_iterations_optimisation(chi2_values, error_values, output_pdf, canvas)
```

Exercise 5: Optimising number of iterations

- When done with the implementation of code:
    - Run the code as in the previous exercise but now add --iter flag at the end
- Questions:
    - What is the evolution of the relative unfolding error and chi2 values between unfolded distributions?
    - Do you see amplification of statistical fluctuations for large number of iterations?
    - What number of iterations seems to be optimal?