

Розгортання (unfolding)

Олександр Зенаєв

Розгортання (unfolding): формулювання

- Unfolding (розгортання, відновлення, деконволюція) в експериментальній фізиці – математичний метод відновлення справжнього (істинного, true) розподілу фізичної величини на основі спостережуваних (вимірних, reconstructed, reco) даних
- Unfolding є ключовою проблемою при вимірюванні розподілів (наприклад, диференціальних перерізів народження частинок, $\frac{d\sigma}{dX}$), що зазнають суттєвого впливу через обмежену роздільну здатність детектора
- Експериментально вимірюють диференціальні перерізи, що усереднені за бінами

$$\frac{d\sigma}{dX} \sim \frac{\Delta\sigma}{\Delta X} \sim \frac{\Delta N}{\Delta X}$$

- Через обмежену роздільну здатність, події мігрують між бінами: події у справжньому біні ΔX_1 можуть бути реконструйовані в іншому біні ΔX_2

M. Kuusela, PHYSTAT Conference on Unfolding, 2024

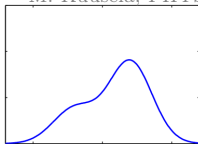


Figure: Smeared spectrum

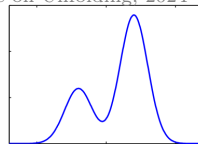
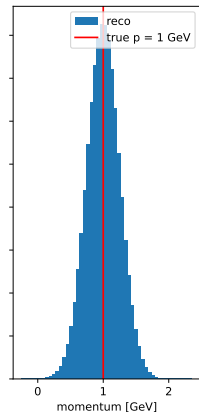


Figure: True spectrum



Розгортання (unfolding): розв'язок

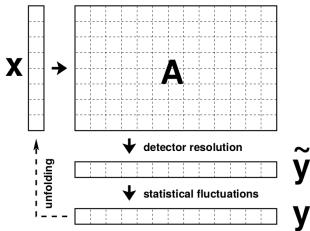
- Задача не має однозначного розв'язку (некоректна постановка задачі, ill-posed problem): нескінченна кількість параметрів
- Необхідна регуляризація:
 - ▶ обмеження кількості невідомих параметрів (біни), але через статистичні похибки все одне виникають швидко осцилюючі компоненти (high frequency components)
 - ▶ додавання додаткових обмежень на розв'язок (гладкість) для придушення швидко осцилюючих компонент
- Існує декілька бібліотек для розгортання, доступних через інтерфейс [RooUnfold](#).

Інструкція:

<https://gitlab.cern.ch/RooUnfold/RooUnfold/-/blob/master/README.md>

```
git clone https://gitlab.cern.ch/RooUnfold/RooUnfold.git
cd RooUnfold
mkdir build
cd build
cmake ..
make -j4
cd ..
source build/setup.sh
```

- Ми розглянемо метод матричного розгортання, реалізований в [TUnfold](#). Він дозволяє використовувати більшу кількість бінів на спостережуваному рівні, ніж на істинному рівні, що допомагає стабілізувати розв'язок.



- Unfolding problem: $\mu = \mathbf{Ax}$
- μ : detector expectation (M_y bins), \mathbf{V}_{yy} is its covariance
- \mathbf{A} : response matrix (taken from MC)
- \mathbf{x} : unknown truth, M_x bins ($M_x \leq M_y$)
- $\mathcal{L} = (\mathbf{y} - \mathbf{Ax})^T \mathbf{V}_{yy}^{-1} (\mathbf{y} - \mathbf{Ax}) + \tau^2 (\mathbf{x} - \mathbf{x}_0)^T (\mathbf{L}^T \mathbf{L}) (\mathbf{x} - \mathbf{x}_0)$
- $\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = 0 \Rightarrow \mathbf{x} = \mathbf{x}(\mathbf{y}, \mathbf{V}_{yy}, \mathbf{x}_0)$, $\mathbf{V}_{xx} = \mathbf{V}_{xx}(\mathbf{y}, \mathbf{V}_{yy}, \mathbf{x}_0)$
- \mathbf{x}_0 : bias vector for regularization (taken from MC)

- Регуляризація: накладання обмеження (шляхом додавання χ^2 penalty) на відмінність розв'язку \mathbf{x} від \mathbf{x}_0 (зазвичай МК симуляції), або (краще) його другої похідної (кривини)
- TUnfold підтримує багатовимірні розподіли
- Є декілька методів для підбору ступені регуляризації (regularisation strength)
- Бажано обирати $M_x < M_y$

2.2 Algorithm

The unfolding algorithm, as implemented in TUnfold, determines the stationary point of the “Lagrangian”

$$\mathcal{L}(x, \lambda) = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 \quad \text{where} \quad (3)$$

$$\mathcal{L}_1 = (\mathbf{y} - \mathbf{A}\mathbf{x})^\top \mathbf{V}_{\mathbf{yy}}^{-1} (\mathbf{y} - \mathbf{A}\mathbf{x}), \quad (4)$$

$$\mathcal{L}_2 = \tau^2 (\mathbf{x} - f_b \mathbf{x}_0)^\top (\mathbf{L}^\top \mathbf{L}) (\mathbf{x} - f_b \mathbf{x}_0), \quad (5)$$

$$\mathcal{L}_3 = \lambda (Y - \mathbf{e}^\top \mathbf{x}) \quad \text{and} \quad (6)$$

$$Y = \sum_i y_i, \quad (7)$$

$$e_j = \sum_i A_{ij}, \quad (8)$$

The term \mathcal{L}_1 is what one expects from a least square minimisation. The vector \mathbf{y} has n rows. The covariance matrix $\mathbf{V}_{\mathbf{yy}}$ of \mathbf{y} is diagonal in many cases, such that the diagonals hold the squares of the uncertainties. TUnfold also supports the use of non-diagonal $\mathbf{V}_{\mathbf{yy}}$. The vector \mathbf{x} corresponds to the unfolding result and has m rows. The elements A_{ij} of \mathbf{A} describe for each row j of \mathbf{x} the probabilities to migrate to bin i of \mathbf{y} . The matrix \mathbf{A} often is determined using Monte Carlo simulations.

The term \mathcal{L}_2 describes the regularisation, which damps fluctuations in \mathbf{x} . Such fluctuations originate from the statistical fluctuations of \mathbf{y} , which are amplified when determining the stationary point of equation 3. The parameter τ^2 gives the strength of the regularisation. It is considered as a constant while determining the stationary point of \mathcal{L} . The matrix \mathbf{L} has m columns and n_R rows, corresponding to n_R regularisation conditions. The bias vector $f_b \mathbf{x}_0$ is composed of a normalisation factor f_b and a vector \mathbf{x}_0 . In the simplest case, one has $f_b = 0$, $n_R = m$ and \mathbf{L} is the unity matrix. In that case, \mathcal{L}_2 simplifies to $\tau^2 \|\mathbf{x}\|^2$, effectively suppressing large deviations of \mathbf{x} from zero. If $f_b = 1$, deviations of \mathbf{x} from \mathbf{x}_0 are suppressed. Choices of the matrix \mathbf{L} different from the unity matrix are discussed in section 7.

- Регуляризація призначена для зменшення похибок на розв'язок (variance) ціною зміщення результату (викривлення, bias)
- Ключовою проблемою є знаходження балансу між похибками і зміщенням (bias-variance tradeoff). Це можна розуміти як баланс між статистичною і систематичною похибкою.
- Необхідною умовою є виконання покриття (coverage): наскільки добре розв'язок описує істинний розподіл в межах похибок
- Концепція покриття перевіряється через тести покриття (closure tests, cover tests): використовуючи МК симуляцію (коли відомий істинний розподіл), на штучних прикладах тестують розв'язок, наскільки добре він відтворює істинний розподіл. На основі цих тестів оптимізують розгортання (обирають метод та регуляризацію)
- Якщо можна, краще уникати регуляризації

- Згенерувати експоненційний розподіл (імпульси частинок)
- Викривити згенерований розподіл, накладаючи розмиття за нормальним розподілом із додатковим зсувом
- Виконати розгортання за допомогою бібліотеки TUnfold (через інтерфейс RooUnfold)
- Порівняти результати розгортання без регуляризації та з регуляризацією

[Python] <https://github.com/zenaiev/hep/blob/main/unfold/unfold.py>

[C++] <https://github.com/zenaiev/hep/blob/main/unfold/unfold.cpp>

Google Colab:

https://colab.research.google.com/github/zenaiev/hep/blob/main/unfold/unfold_cpp.ipynb