

Unfolding with Singular Value Decomposition

V. Kartvelishvili

Lancaster University, United Kingdom

Abstract

An overview is given of the SVD approach to unfolding, including basic principles, error propagation and fitting applications.

1 Introduction

We will assume that an *initial* high statistics Monte Carlo sample was used to create \hat{A}_{ij} , the matrix simulating detector response — i.e. the probability for an event generated in the *true* bin j to be found in the *measured* bin i :

$$\sum_j \hat{A}_{ij} x_j^{\text{ini}} = b_i^{\text{ini}}, \quad (1)$$

which, by construction, is satisfied exactly. Here b_i^{ini} is the vector (histogram) of MC “measured” values, while x_j^{ini} is the vector (histogram) of MC true values.

Our aim is to determine the underlying real distribution x from a real measured distribution b , given \hat{A}_{ij} and some additional information on expected properties of x , i.e. find a meaningful way of solving the simultaneous system of equations

$$\hat{A}x = b. \quad (2)$$

The problem is twofold. Firstly, the right-hand-side b is known with some precision: e.g., for purely statistical errors, the covariance matrix $B = \text{diag}\{b\}$. Secondly, our knowledge of \hat{A} is not perfect either, due to finite MC statistics, as well as imperfections in detector simulation. Even worse, \hat{A} is almost always very close to being degenerate, so an attempt to solve the problem directly and “exactly” is unlikely to be useful.

2 Singular Value Decomposition

For a detailed description of the unfolding algorithm based on the Singular Value Decomposition of the response matrix, see Ref. [1] and references therein. The original development of the algorithm was based on the method presented in Ref. [2].

A Singular Value Decomposition (SVD) of a real $m \times n$ matrix A is its factorization of the form (see Ref. [1] and references therein)

$$A = U S V^T, \quad (3)$$

where U is an $m \times m$ orthogonal matrix ($U U^T = U^T U = I$), V is an $n \times n$ orthogonal matrix ($V V^T = V^T V = I$), while S is an $m \times n$ diagonal matrix with non-negative diagonal elements:

$$S_{ij} = 0 \text{ for } i \neq j, \quad S_{ii} \equiv s_i \geq 0. \quad (4)$$

The numbers s_i are called *singular values* of the matrix A . For example, if A itself is orthogonal, all $s_i = 1$, while if A is degenerate, at least one s_i is equal to zero. By swapping rows of U (and similarly for V), s_i can be ordered from largest to smallest.

Columns of U and V are called the left and right *singular vectors*. They define convenient orthonormal bases in their respective spaces.

With SVD, the linear system $Ax = b$ can be easily diagonalised by introducing rotated vectors z and d , and finding the exact solution looks deceptively simple:

$$\begin{aligned}
 USV^T x &= b \quad \Rightarrow \quad z \equiv V^T x, \quad d \equiv U^T b, \\
 s_i z_i &= d_i \quad \Rightarrow \quad z_i = \frac{d_i}{s_i} \quad \Rightarrow \quad x = Vz.
 \end{aligned}
 \tag{5}$$

However, there are at least two reasons why this determination of z_i can go horribly wrong: firstly, due to errors in b , some d_i may be poorly known, or not significant at all; secondly, some singular values s_i may be small (or even zero), thus exaggerating the contributions of poorly known coefficients.

Indeed, Fig. 1 shows an example taken from Ref. [2], and also used in Ref. [1]. It is clear that if the r.h.s. b contains random fluctuations, the exact solution x (the right plot) is essentially useless, as these fluctuations get greatly magnified by the small singular values.

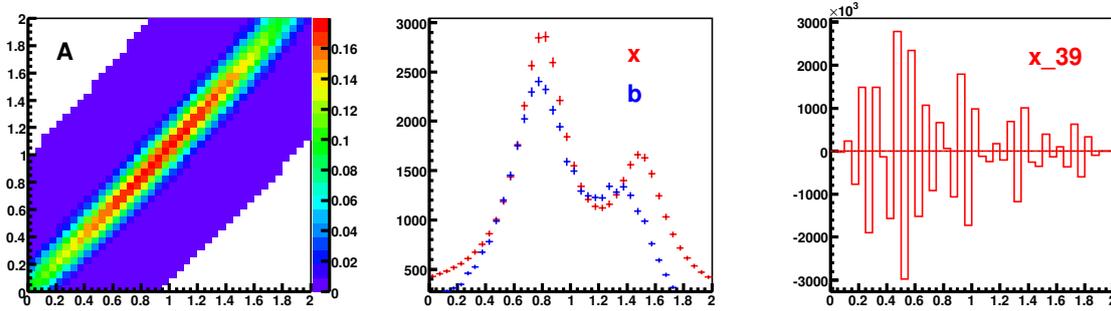


Fig. 1: An example of the response matrix A (left), the “measured” and “true” distribution (b and x , respectively, centre) and the exact solution x of the system $Ax = b$ (right), once the r.h.s. b contains random statistical fluctuations.

So, since the orthogonal matrices U and V are totally harmless, SVD allows the problem to be narrowed down to individual s_i and/or d_i .

3 Rescaling variables and equations

In general, different equations in the simultaneous system $Ax = b$ should contribute to any meaningful solution with different weights, in accordance with the precision of the respective coefficients and the r.h.s. Hence, in order to make the equations and the unknowns more uniform, some rescaling is required.

Rescaling of unknowns. If the Monte Carlo and the data are fast-varying functions ranging over many orders of magnitude, it makes sense to try and find x relative to the true distribution x^{ini} (i.e. relative to the MC used for creating the matrix A). This is achieved by multiplying each column of A_{ij} by x_j^{ini} and simultaneously defining new unknowns $w_j = x_j/x_j^{\text{ini}}$. This transformation has a “side-effect” of changing A from “probability to “number-of-events” matrix. The latter is arguably more useful, as bins with higher statistics, and hence more significance, are now enhanced.

Rescaling of equations. In general, if one equation is multiplied by a factor, SVD of the matrix changes. One of the ideas of the GURU algorithm, described in Ref. [1], was to make sure that all the equations are “made equal” by checking that the error in the r.h.s. is always ± 1 . In the simplest case of uncorrelated errors in the r.h.s., this is achieved by dividing each equation (i.e. each row of A_{ij} as well as b_i) by the error Δb_i .

After these manipulations, the original system $Ax = b$ has transformed into

$$\sum_j \tilde{A}_{ij} w_j = \tilde{b}_i, \quad (6)$$

where, by construction, the covariance matrix of the r.h.s. is equal to the unit matrix, and the new unknowns w_i are defined relative to the input MC distribution. I.e., $w_i = 1$ would mean that the “unfolded” x is the same as the “truth” MC used to generate the response matrix.

4 Solving equations

Now let's use SVD to solve the rescaled system:

$$\begin{aligned} \tilde{A}w = \tilde{b} &\Rightarrow \tilde{A} = U S V^T \Rightarrow U S V^T w = \tilde{b} \Rightarrow z \equiv V^T w, \quad d \equiv U^T \tilde{b}, \quad (7) \\ s_i z_i = d_i &\Rightarrow z_i = \frac{d_i}{s_i} \Rightarrow w = V z. \end{aligned}$$

By construction, all d_i are independent and have errors ± 1 . Assuming all $s_i \neq 0$, z_i form the *exact* solution of the initial system, which is still highly unlikely to be useful.

Looking at the plots of $|d|$ and s (Fig. 2), it's easy to understand what's happening: many d_i are

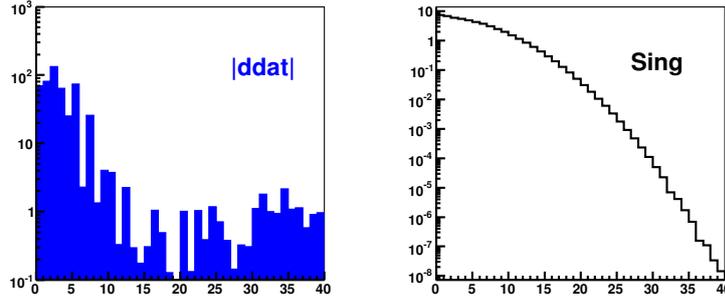


Fig. 2: The distribution of $|d_i|$ (left) and the singular values s_i (right), for the example described above. All components d_i for $i \gtrsim 10$ are essentially insignificant, as by construction all d_i have (independent) errors of ± 1 .

insignificant (compatible with zero), while corresponding s_i are small. Respective z_i will be huge, but nonetheless essentially random.

Note that the exact solution of the system, Eq. (7), is equivalent to the minimisation of the residual χ^2 :

$$\chi^2 \equiv (\tilde{A}w - \tilde{b})^T (\tilde{A}w - \tilde{b}) = \min. \quad (8)$$

where (ignoring machine precision and assuming all $s_i \neq 0$) $\chi_{\min}^2 = 0$. A simple way of regularisation would be the truncation of the (diagonalised) system at some $i = k$, which would make $\chi^2 = \sum_{i>k} d_i^2$.

While simple truncation is better than keeping all i , this is not a good solution, as biases are hard to control. One of the usual choices, used in high energy physics, is regularisation by adding an *a priori* requirement that the regularised solution is *smooth*. Technically, this requirement is introduced into the χ^2 minimisation condition by adding an extra term [1, 2]:

$$\chi^2 \equiv (\tilde{A}w - \tilde{b})^T (\tilde{A}w - \tilde{b}) + \tau (Cw)^T Cw = \min. \quad (9)$$

By choosing Cw to be the second finite difference of w , so that

$$(Cw)^T Cw = \sum [(w_{i+1} - w_i) - (w_i - w_{i-1})]^2, \quad (10)$$

we find a minimum of χ^2 that keeps the “integrated square of the second derivative” of w constrained. The parameter τ essentially plays the role of the Lagrange multiplier in the new conditional minimisation problem, given by Eq. (9).

This new minimisation problem gives rise to a new, overconstrained linear system¹ :

$$\begin{bmatrix} \tilde{A} C^{-1} \\ \sqrt{\tau} \cdot I \end{bmatrix} C w = \begin{bmatrix} \tilde{b} \\ 0 \end{bmatrix}. \quad (11)$$

A fairly straightforward method allows the reduction of the new problem to the old one.

For $\tau = 0$ the new system, Eq. (11), is identical to the old system (6), and can be solved using SVD. However, it’s convenient to replace \tilde{A} with $\tilde{A}C^{-1}$, and w with Cw . Then, once the exact solution for $\tau = 0$ is found, the solution for $\tau \neq 0$ is obtained immediately [1]:

$$z_i^{(\tau)} = \frac{d_i}{s_i} \cdot \frac{s_i^2}{s_i^2 + \tau}. \quad (12)$$

For large $s_i \gg \tau$, the suppression factor $s_i^2/(s_i^2 + \tau)$ is close to 1, but for smaller s_i it works as a low-pass filter. So, for a smooth way of eliminating the wildly oscillating contributions (which correspond to the values of i with non-significant d_i and small s_i) one should choose $\tau \simeq s_k^2$ where k is the index of the last significant d .

The solution for the regularised version of the above example is presented in Fig. 3, taken from Ref. [1].

5 Error propagation

The components of the exact solution, $z_i = z_i^{(\tau=0)} = \frac{d_i}{s_i}$, were uncorrelated, and hence had a diagonal covariance matrix. The covariance matrix of the regularized $z_i^{(\tau)}$,

$$Z_{ik}^{(\tau)} = \frac{s_i^4}{(s_i^2 + \tau)^2} \cdot \delta_{ik},$$

is still diagonal, but the errors corresponding to insignificant d_i are suppressed. Propagating these errors back to covariance matrices of w and x , we have:

$$\begin{aligned} W^{(\tau)} &= C^{-1} V Z^{(\tau)} V^T C^{T-1}, \\ X_{ik}^{(\tau)} &= x_i^{\text{ini}} W_{ik}^{(\tau)} x_k^{\text{ini}}. \end{aligned} \quad (13)$$

Inevitably, for any non-zero τ , the covariance matrices $W^{(\tau)}$ and hence $X^{(\tau)}$ are not diagonal, and the larger the value of τ , the larger are the bin-to-bin correlations in errors.

This behaviour is not surprising: this type of unfolding is nothing else but a fit, and hence behaves as such. In an extreme case of only one significant d_i , all one can say is that the data is *proportional* to the Monte Carlo, with the error essentially reflecting the overall data statistics — but this error is fully correlated between all bins. Unfolding cannot help here — you need more statistics (and/or a better detector, with a “more diagonal” response matrix) to make more d_i values significant!

We, however, have to admit that a non-diagonal covariance matrix creates a presentational problem: showing just $x_i \pm \sqrt{X_{ii}}$ is, in most cases, misleading.

¹Note that the matrix C , defined according to Eq. (10), is itself singular and needs to be regularised before the inverse, C^{-1} , can be calculated. One can regularise C by including the first finite differences at the end-points, and by adding a small term proportional to the unit matrix (see Ref. [1] for details).

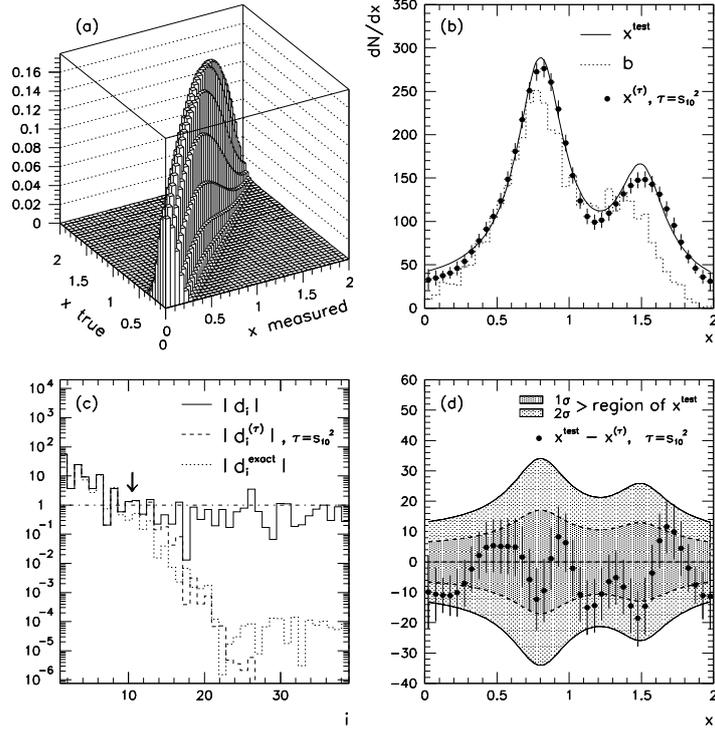


Fig. 3: (a) The response matrix matrix \hat{A} . (b) The true distribution x^{test} compared to the measured histogram b and the unfolded distribution $x^{(\tau)}$ for $\tau = s_{10}^2$. (c) The absolute values of d_i (solid line) compared to the regularized r.h.s. (dashed line) and the one unaffected by the statistical fluctuations (dotted line). The horizontal line shows statistical errors in d_i , while the arrow indicates the boundary between the significant and non-significant equations. (d) The deviation of the unfolded distribution from the true one, compared to 1σ and 2σ error bands [1].

6 Calculating the matrix X^{-1}

In order to compare the measurement b with a theoretical prediction $f = f(\alpha)$, one could try *folding* the theory with the detector response matrix, and calculate

$$\chi^2 = (b - Af)^T B^{-1} (b - Af).$$

Alternatively, because $b = Ax$, one can replace b with Ax and have

$$\begin{aligned} \chi^2 &= (Ax - Af)^T B^{-1} (Ax - Af), \\ &= (x - f)^T A^T B^{-1} A (x - f), \\ &= (x - f)^T X^{-1} (x - f), \end{aligned} \quad (14)$$

where $X^{-1} = A^T B^{-1} A$, or, in our notation,

$$X_{jk}^{-1} = \frac{1}{x_j^{\text{ini}} x_k^{\text{ini}}} \sum_i \tilde{A}_{ij} \tilde{A}_{ik}. \quad (15)$$

Hence, for a problem with fully significant matrix, comparing folded theory with measured data or “raw” theory with unfolded data will give equivalent results. But it is also clear, that in order to compare theory with the unfolded experiment, one needs the *inverse* of the error matrix of the unfolded distribution, rather than the error matrix itself. And from the above it is obvious that the calculation of this inverse does *not* require any regularisation or unfolding.

By substituting $x^{(\tau)}$ for x , one only changes the χ^2 by a “small” amount of order of τ , which may or may not disturb the fit, depending on the model. Hence, if the aim is to fit the unfolded distribution to theory, one is generally better off by folding theory, rather than by unfolding the measurement. However if the effects of regularisation are “small”, the results will be equivalent.

If a theoretical parameter happens to be sensitive to an insignificant element of the data, this will show up in the fit one way or another. In any case, in our understanding, the regularised covariance matrix $X^{(\tau)}$, defined in Eq. (13), should only be used for calculating errors on the unfolded distribution. Calculating the *inverse* of $X_{ik}^{(\tau)}$ is neither helpful, nor useful. Instead, if necessary, use the exact X^{-1} from Eq. (15) above.

7 Advice to unfolders

- Three things contribute to the solution: the response matrix, data statistics, and binning.
- Choose binning wisely: large variations in data may be avoided by using variable bin widths. Size of bin-to-bin error correlations may also be affected.
- Rescaling is important: only unfold the equations once they have equal “weights”.
- The Monte Carlo sample(s) used for building the response matrix should have as high statistics, and should be as close to the real experiment, as possible.
- Monte Carlo samples used for testing should have the same statistics as the real data; use of larger or smaller samples can be misleading.
- To assess unfolding systematics, vary the matrix within its tolerances, and study the effects on the singular values.
- Even if you are using a different algorithm for unfolding your data, try applying SVD: it will help identify the bottlenecks, and assess any benefits of performing unfolding in the first place.
- It’s wise not to expect any miraculous solutions to unfolding problems.

8 Implementations

A number of implementations of the algorithm described in Ref. [1] exist. Some of these are listed below:

- The original Fortran package called GURU is still accessible (see Ref. [3]).
- A C++ wrapper for the above Fortran code was developed by G. Hesketh, and can be found at Ref. [4].
- A new C++ implementation of the algorithm, TSVDUnfold, now exists in ROOT (see Ref. [5], and the talk by K. Tackmann in these proceedings).

9 Acknowledgements

The author is grateful to the organisers for the opportunity to give this presentation, to V. Blobel for many helpful discussions, and to G. Hesketh, A. Hoecker and K. Tackmann for their efforts in re-implementing the algorithm.

References

- [1] A. Hoecker and V. Kartvelishvili, “SVD Approach to Data Unfolding,” Nucl. Instrum. Meth. A **372** (1996) 469 [arXiv:hep-ph/9509307].
- [2] V. Blobel, Unfolding methods in high-energy physics experiments, DESY 84-118 (1984).
- [3] <http://www.hep.lancs.ac.uk/internal/guru.tar.gz>.

[4] <http://www-d0.fnal.gov/~ghesketh/unfolding>.

[5] <http://www.root.cern.ch/root/html/TSVDUnfold.html>.