OPAL Technical Note TN361
March 8, 1996

The $\mathcal{RUN}$ manual

Regularized Unfolding for High-Energy Physics Experiments

Volker Blobel

Abstract

Distributions $f(x)$ measured in high-energy physics experiments are often distorted or transformed by limited acceptance and finite resolution of the detectors. The unfolding of measured distributions is an important, but due to inherent instabilities a very difficult problem. This manual is about the general unfolding program $\mathcal{RUN}$, written in FORTRAN77, which needs as input $n$tuples for data and Monte Carlo quantities. The Monte Carlo $n$tuples implicitly define the resolution function of the experiment. The unfolded distribution $f(x)$ is reconstructed in bins by a regularization method, which includes the determination of the covariance matrix of the result by error propagation from the data statistics and the properties of the resolution function.

Contents

Non quia difficilia sunt non audemus,
sed quia non audemus difficilia sunt.
Seneca

1 Introduction

1.1 The measurement process and unfolding

The measurement of *differential cross sections* or distributions is a standard task of high energy physics experiments. The cross section differential in the kinematical variable x in a scattering experiment is given by a distribution $f(x)$:

$$\frac{d\sigma}{dx} \equiv f(x) \qquad \text{with} \quad \sigma_{\text{tot}} = \int \frac{d\sigma}{dx}\,\mathrm{d}x = \int f(x)\,\mathrm{d}x \; . \tag{1}$$

The probability for a scattering in a single event with a value of the kinematical variable x in the interval $[x, x+dx]$ is proportional to the differential cross section or the distribution $f(x)$. The integral over the differential cross section is called the *total cross section*, σ_{tot}, and the event rate is proportional to the total cross section. In this respect the properties of the distribution $f(x)$ differ from the properties of the probability density function defined in the statistic, which by definition is *normalized* (integral equal to one). In a scattering experiment the proportionality constant relating the event rate dN/dt and the total cross section is the Luminosity $\mathcal{L}$ of the experiment:

$$\frac{dN}{dt} = \mathcal{L} \cdot \sigma_{\text{tot}} \qquad \text{with} \quad N = \int_{\text{time}} \mathcal{L}\,\mathrm{d}t \cdot \sigma_{\text{tot}} \; . \tag{2}$$

The total number of scattering events N is obtained by integration over the total time. This number N is in an experiment equal to the product of the total cross section and the integrated Luminosity

$$L_{\text{exp}} = \int_{\text{time}} \mathcal{L}\,\mathrm{d}t \; , \tag{3}$$

which for simplicity is called L_{exp} in the following. In practice the (integrated) Luminosity L_{exp} is often determined by the simultaneous measurement of the rate of a *known* reaction cross section.

Assume that a distribution $f(x)$ is to be determined by the measurement of particle reactions in a scattering experiment. With an ideal detector one could measure the quantity x in every event and could obtain $f(x)$ by a simple histogram of the quantity x, using as normalizing factor the (known) Luminosity L_{exp}. With *real* detectors the determination of $f(x)$ is complicated by three effects:

- **Limited acceptance**
  The probability to observe a given event, the *detector acceptance* P, is less than 1, resulting in a certain loss of recorded events. The acceptance can depend on the kinematical variable x, thus $P \equiv P(x)$.
- **Transformation**
  Instead of the quantity x a different, but related quantity y or several quantities y_1, y_2 ... are measured in the detector, which can be combined to a vector $\mathbf{y}$ of measured quantities. The relation between x and $\mathbf{y}$ is given by some transformation. The transformation can for example be caused by the non-linear response of a detector component.
- **Finite resolution**
  The measured quantities $\mathbf{y}$ are smeared out due to the finite resolution (or limited measurement accuracy) of the detector. Thus there is only a *statistical* relation between the true kinematical variable x and the vector $\mathbf{y}$ of measured quantities.

The case of limited acceptance alone (i.e. assuming no effects from transformation or finite resolution) is in principle simple. If the probability to observe a given event is $P(x) < 1$, this can be taken into account in the calculation of the cross section by simply weighting the event with the factor $1/P(x)$. Of course this correction could already be problematic, for example if $P(x) \ll 1$ or if $P(x)$ is rapidly varying with x. Also the correction of a *transformation* effect alone in the date like a non-linear detector response is in principle not difficult, once the parametrization of the transformation is known. However if a non-linearity is combined with a limited resolution the resolution is further reduced.

The really difficult effect in the data correction for experimental effects is the **finite resolution**, causing a *smearing* of the measured quantities. Mathematically the relation between the distribution $f(x)$ of the true variable x, to be determined in an experiment, and the measured distribution $g(\mathbf{y})$ of the vector $\mathbf{y}$ of measured quantities is given by the integral equation,

$$g(\mathbf{y}) = \int A(\mathbf{y}, x) f(x)\, dx + b(\mathbf{y})\,, \tag{4}$$

called a Fredholm integral equation of the first kind. In this equation the distribution $b(\mathbf{y})$ has been added to the right-hand side; it represents a *background* contribution to the measured distribution $g(\mathbf{y})$. This has in practice to be taken into account; the background contribution can be either independently measured or calculated and thus can be assumed to be known. The resolution function $A(\mathbf{y}, x)$ represents the effect of the detector. For a given value $x = x_0$ the resolution function $A(\mathbf{y}, x_0)$ describes the response of the detector in the variable(s) $\mathbf{y}$ for that fixed value x_0. The *difficult* problem to determine the distribution $f(x)$ from measured distributions $g(\mathbf{y})$ (and $b(\mathbf{y})$) is called *unfolding.* Unfolding of course requires the knowledge of the resolution function $A(\mathbf{y}, x)$, i.e. all the effects of limited acceptance, transformation and finite resolution.

The scope of the unfolding problem can be even wider. While the relation between the true value x and the measured quantity $\mathbf{y}$ mentioned above are due to imperfections of the detector, there may be other effects between x and $\mathbf{y}$, which are *outside of the experimental control*, even with an ideal detector. One example are radiative effects, which in experiments are often corrected afterwards (*radiative corrections*), but are in their effect similar to detector effects. Another effect is the *Fermi motion* and has to do with the dynamics within the substructure of the target or beam particles. If the true kinematical quantity is defined at the *parton level*, further effects from the fragmentation process of partons to the (observable) hadrons influence the measured quantity $\mathbf{y}$. All these effects are of statistical nature, and can only be calculated under model assumptions.

Due to the complexity of the whole measurement process the response function $A(\mathbf{y}, x)$ is in general *not* known as a function in analytical or empirical form. Instead the whole measurement process is described in a model, which allows to *simulate* the measurement process of single events in Monte Carlo calculations. Input to the Monte Carlo calculations is a distribution function $f_0(x)$, used to generate single x-values. The simulation can be described by the formula

$$g_0(\mathbf{y}) = \int A(\mathbf{y}, x) f_0(x)\, dx\,, \qquad \text{(Monte Carlo simulation)} \tag{5}$$

and generates events with a distribution $g_0(\mathbf{y})$, as response to the input distribution function $f_0(x)$. The basic requirement is, that the response function, assumed implicitly in the simulation, is *close* to the reality. By construction of the unfolding method in $\mathcal{RUN}$ the result should *not* depend on this input function $f_0(x)$, thus the input function $f_0(x)$ may differ from the true function. As result of the Monte Carlo calculations there is a set of $n$tuples $\{x_{\mathrm{MC}}, w, \mathbf{y}_{\mathrm{MC}}\}$ (where w is a weight assigned to the $n$tuple), which is input to the unfolding program, and contains implicitly the response function $A(\mathbf{y}, x)$. From the set of data $n$tuples $\{\mathbf{y}\}$ the unfolded data distribution is determined.

The response function contains all effects of the measurement, including the limited acceptance and transformation effects. For a discussion of the meaning of the resolution function it is easier to consider the discrete version of the basic unfolding equation (4). The discrete version, connecting the distributions $f(x)$, $g(x)$ and $b(\mathbf{y})$, is the matrix equation

$$\mathbf{g} = A\mathbf{f} + \mathbf{b} \ . \tag{6}$$

Here the vectors $\mathbf{g}$ and $\mathbf{b}$ represent histograms of the measured value $\mathbf{y}$, and the histogram $\mathbf{f}$ represents the function $f(x)$. The resolution matrix A determines the transformation from the space of the variable x to the space of the measured variables $\mathbf{y}$. The elements A_{ij} can be interpreted as probabilities, and they can take on only *positive* values (or zero), with a maximum value of 1. The element A_{ij} is the probability, to observe an entry in histogram bin i in the histogram $\mathbf{g}$, if the true value x is from histogram bin j of the histogram $\mathbf{f}$ of true values. In other words: for a given index j the column j of matrix A (the elements A_{ij} with $i = 1, 2, \ldots$) represent the response distribution of the detector to the (true) entries in the bin j. The sum of the elements in this column is one only if the measurement process has full (100 %) acceptance for these true values. Usually the acceptance $P_j = \sum_{i=1}^{n} A_{ij}$ is below one. The values in different columns of matrix A are *independent.* The matrix equation above includes a term for a background contribution; the elements b_i represent the expected absolute background contribution in bin i of histogram $\mathbf{g}$.

In the special case of a *square* matrix A the simplest solution of the matrix equation (ignoring the statistical precision of the measurement) is by matrix inversion according to

$$\mathbf{f} = A^{-1} (\mathbf{g} - \mathbf{b}) \ . \tag{7}$$

The inverse matrix A^{-1} will have positive *and* negative values, larger in magnitude than 1; the elements of the inverse matrix A^{-1} can *not* be interpreted as probabilities. The resulting distribution $\mathbf{f}$ of such or similar calculation usually shows a strongly oscillating behaviour, unless the dimension parameters of the matrix is very small and the diagonal elements of matrix A are dominant. This oscillating behaviour is due to an *inherent instability* of solutions of the problem. Due to the finite resolution there is a large negative correlation between different bins. Essentially this instability has the effect that only a limited number of parameters can be extracted from the data, which can be either binned values or coefficients of the expansion of the result in terms of orthogonal contributions. An acceptable solution requires to control the instability by a method called *regularization.* The unfolding algorithm used in the unfolding program $\mathcal{RUN}$ is described in detail in the following reports:
V. Blobel, Unfolding Methods in high-energy physics, DESY 84-118, 1984
V. Blobel, Proceedings of the 1984 CERN Scool of Computing, Aiguablava, Spain, 9 - 22 September 1984, CERN 85-02.

Due to the limited resolution there is a natural limitation of the number of data points, which can be reconstructed from the data. This natural limitation, which can not be overcome by any clever unfolding method, means that the bin width, over which a data point represents the average, can not be smaller than 1.5 to 2 standard deviations of the measurement. If the bin width is choosen as two standard deviations, i.e. $\Delta x = 2\sigma$, then the factor, by which the error is larger than the pure statistical error, is about 3 to 4; for $\Delta x = 1.5\sigma$ the factor is already about 10 and for $\Delta x = \sigma$ it would already be above 100. These numbers may be useful in estimating the number of data points, to be reconstructed by the unfolding program.

In many experiments so called *unsmearing* is done on the basis of the ratio between Monte Carlo generated input and output distributions, using a reasonable guess of the result ($f(x)$). It is difficult to write down the mathematics of this method in terms of vectors and matrices. Often results of such *unsmearing* is shown in a large number of data points (with width $\Delta x < \sigma$), typically with very small internal fluctuations. Error estimates are difficult, since there is no real mathematical or statistical justification for this method. Usually it is distinguished between (pure) statistical errors (often rather small), and additional systematic errors, often rather large. In fact the plot of those results, many data points with little fluctations and large (systematic) error bars, shows immediately large *positive* correlations between

the data points, which make any comparison with other data or curves difficult. These properties shows, that the algorithm implicitly contains a lot of smoothing to the data. Since the *fine structure* of the result, corresponding to higher order components of an expansion of the result, can not be extracted from the data, the algorithm simply copies those properties from the input to the result, thus giving a smooth behaviour of the result, without being accurate in reality.

For practical applications the algorithm used in $\mathcal{RUN}$ can be described to work as follows. Internal to the program a weighting function $f_{\text{mult}}(x)$ is defined, which is used to calculate for each Monte Carlo $n$tuple an individual weight-factor, depending only on the true x value. The *reweighted* Monte Carlo $n$tuples should describe as good as possible the measured distribution $g(\mathbf{y})$. Thus the weighting function is $f_{\text{mult}}(x)$ is determined by a fit to the measured distribution $g(\mathbf{y})$, using histograms of the measured data. The histograms may be one- or two- or three-dimensional. The final unfolding result is then given by

$$f(x) = f_{\text{mult}}(x) \cdot f_0(x) \; , \tag{8}$$

(apart from an overall normalzation factor, depending on the data and Monte Carlo luminosities). The weighting function $f_{\text{mult}}(x)$ is represented by a smooth function, parametrized by B-splines. Finally, by integration over bins in x, binned data are determined together with a complete covariance matrix of the binned data. The bins may be defined by the user, but the program includes an algorithm to determine a set of bins, which should lead to small correlations between different bins by taking into account the different statistical precision and resolution as a function of x.
The weighting function $f_{\text{mult}}(x)$ is internally represented as an expansion

$$f_{\text{mult}}(x) = \sum_{k=0}^{n_k} a_k \; p_k(x) \; , \tag{9}$$

where the functions $p_k(x)$ form a *normalized orthogonal* function system, taking into account the experimental resolution function on the basis of the Monte Carlo input. Function $p_0(x)$ is usually close to a constant, function $p_1(x)$ *linear* in x, $p_2(x)$ *quadratic* in x and so on. As following from the properties of the functions $p_k(x)$, the different coefficients a_k of the result are *independent* and have a variance of one. In general only the first few coefficients are *significantly* different from zero, and for k above a certain index k_{cut} all coefficients up to the largest index n_k are compatible with zero. The value of the index k_{cut} depends mainly on the experimental resolution, to a smaller extend also on the statistics of the experiment. Contributions from the higher coefficients can thus be ignored, since they do not add any information to the result; if included, these contributions would add oscillating terms to the result. The essential point in unfolding is to recognize the index k_{cut} accurately. Actually there is no hard cut-off used in the program, but a smooth transition to zero for higher indices.

1.2 Normalisation of the unfolding result

For the correct normalization of the unfolded distribution function the (time integrated) Luminosities L_{exp} of the experiment and L_{MC} of the Monte carlo input have to be known. The normalization of the resulting distribution function $f(x)$ is proportional to the ratio $L_{\text{MC}}/L_{\text{exp}}$. If the experimental Luminosity is not specified by the user, it is assumed to be 1; this allows to convert the resulting function values to any data Luminosity.
The unfolding is done *relative* to the simulation. In order to get the distribution function $f(x)$ correct in normalization and shape, also the input Monte Carlo distribution function $f_0(x)$ has to be known from the Monte Carlo simulation. The acceptance probability of the detector is taken into account, i.e. the resulting distribution function $f(x)$ will correspond to all events, visible or invisible in the detector. With N_{MC} equal to the number of all generated Monte Carlo events (either visible or invisible) the

following relation is valid:

$$N_{\mathrm{MC}} = L_{\mathrm{MC}} \cdot \int f_0(x)\, dx \; . \tag{10}$$

It is possible for the user to specify the distribution $f_0(x)$ in the FORTRAN function XFUNCT *or* the Monte Carlo Luminosity L_{MC} *or* both. The specification of both the distribution $f_0(x)$ *and* of the Monte Carlo Luminosity L_{MC} is recommended (case 4 below), at least for the final calculation. Four different cases are distinguished and discussed below. In first attempts to apply the unfolding program the use of the simplest case 1 is possible. Many different conditions in the simulation are possible, for example a weighted Monte Carlo or several types of Monte Carlo calculations may be used. In the following first discussion of the four cases for simplicity it is assumed, that

- there is only one type of Data and one type of Monte Carlo $n$tuples,
- all weights of the Monte Carlo $n$tuples have the values of $|w_i| = 1$, and
- the Monte Carlo $n$tuple file contains *all* generated $n$tuples, i.e. those *inside and outside* of the detector acceptance. The convention is to give a *negative* weight to events outside the acceptance of the detector or removed from the accepted events by cuts, but it is recommended to include those events still in the $n$tuples; this allows a determination of the input function $f_0(x)$ or a check of the acceptance from the $n$tuples.

The following table gives an overview over the four cases discussed below.

case	$f_0(x) \equiv$ XFUNCT	L_{MC}	outside MC events
1	not supplied	undefined	included in n-tuple
2	not supplied	defined	included in n-tuple
3	supplied	undefined	included in n-tuple
4	supplied	defined	–

Case 1: Neither Monte Carlo function nor Luminosity specified

The shape of the distribution function $f_0(x)$ is determined from the MC data, the normalization is to an integral of one. According to the definition, the Luminosity is set to $N_{\mathrm{MC}} = \sum |w_i|$.

Case 2: Monte Carlo Luminosity specified

The shape of the distribution function $f_0(x)$ is determined from the MC data as in case 1, but the normalization is taken from the from the Monte Carlo sample, according to

$$f_0(x) := \frac{1}{L_{\mathrm{MC}}} \left(\frac{dN_{\mathrm{MC}}}{dx} \right) \; , \tag{11}$$

where (dN_{MC}/dx) is the histogram accumulated from the $n$tuples.

The distribution function $f_0(x)$ as determined from the Monte Carlo $n$tuples in cases 1 and 2 will show statistical fluctuations due to the limited Monte Carlo sample. In order to reduce these fluctuations a smooth spline function is fitted to the histogram of the $n$tuples and then used in the unfolding. The fluctuations within the bins of the histogram are reduced by a reweighting of the Monte Carlo $n$tuples.

Case 3: Distribution function specified

In this case (and the next case) the user has included the FORTRAN function XFUNCT(X), which returns, for a given argument x, the value of the distribution function $f_0(x)$ (shape and normalization) used in the simulation. From the integral of the function over the total x-range and the number of simulated $n$tuples the integrated Luminosity L_{MC} is calculated by

$$L_{\mathrm{MC}} := \frac{N_{\mathrm{MC}}}{\int f_0(x)\, dx} \; , \tag{12}$$

where $f_0(x)$ is the distribution as defined by the user-supplied function.

Case 4: Luminosity and distribution function specified

In this case all quantities in the luminosity expression are specified. For a full acceptance ($P \equiv 1$) (or if also the *outside*-events are in the file) the product of the integrated distribution function $f_0(x)$ and the luminosity has to agree with the number of simulated n-tuples. In the program the value of an *acceptance* P is determined as ratio of the number of n-tuples and the above product. A value $P > 1$ of the acceptance would point to an error in the specifications.

Case of weighted Monte Carlo $n$tuples. Sometimes the Monte Carlo generated events carry a weight, for example to generate more events in a kinematical region, where the cross section is small. For weighted Monte Carlo $n$tuples the number N_{MC} in the formulas above has to be replaced by the sum of the weights: $N_{\mathrm{MC}} := \sum_i |w_i|$; i.e. all $n$tuples, also those which are *outside* the acceptance of the detector, are taken in the sum for the definition of the MC input distribution $f_0(x)$ or the MC Luminosity L_{MC}. For the unfolding fit of course only the MC $n$tuples *inside* the acceptance (weight positive) are used, taking into account the weight factors w_i.
The presence of weights pose no principal problem, but the user should be aware of the potential problem of the limited statistical accuracy of weighted events. For a set of n $n$tuples the sum S of weights and the statistical error of the sum S are given by the formulas:

$$S = \sum_{i=1}^{n} w_i \quad \text{(positive weights)} \qquad\qquad \sigma_S = \left(\sum_{i=1}^{n} w_i^2 \right)^{1/2} . \tag{13}$$

If all the weigths have the same value, then their relative error is as small as $\sigma_S/S = 1/\sqrt{n}$. This shows the typical $1/\sqrt{n}$ behaviour of Monte Carlo accuracy. However in the case of varying weight the error depends critically on the *distribution* of the weight; in the case of one dominating weight the relative error σ_S/S can be even close to 1 (100 % error). It may be useful to calculate the inverse square of the relative error, which can be interpreted as the *effective* number of generated events

$$n_{\mathrm{eff}} = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{\left(\sum_{i=1}^{n} w_i^2\right)} . \tag{14}$$

The value n_{eff} can be any number between n (all weights identical) and 1 (one extremely large weight). If n_{eff} is very small compared to the true number n of generated events, then the weight distribution is bad and the Monte Carlo simulation inefficient.

Case of several types of Monte Carlo $n$tuples. If more than one type of Monte Carlo simulation is to be used in unfolding, these different simulations have to be distinguished by different labels of the $n$tuples, and a separate value of the Monte Carlo Luminosity $L_{\mathrm{MC}j}$ has to be assigned to each type. The *mixing* of different types can be achieved by assigning a relative weightfactor $\mathcal{W}_j$ to each $n$tuple type, with $\sum_j \mathcal{W}_j = 1$.

Case of several types of Data $n$tuples. If more than one type of Data $n$tuples is specified, the Luminosities are simple added together and the data sample are combined to one sample. It is not forseen to assign weights (different from 1) to data $n$tuples.

Missing invisible data. If the simulated events which are outside of the acceptance of the detector are *not* contained in the $n$tuple file, then neither the true distribution function $f_0(x)$ used in the Monte Carlo simulation nor the Luminosity can be determined correctly by the program. Thus Cases 1 to 3 above are not applicable, and only case 4 applies. This means that both the Monte Carlo input distribution and the corresponding Luminosity have to be specified by the user. Any mistake in these items will directly propagate into the unfolding result $f(x)$.

1.3 The structure of the unfolding program $\mathcal{RUN}$

Input to the unfolding program $\mathcal{RUN}$ are:

- files containing $n$tuples,
- the function XFUNCT (at least a dummy version), and
- parameters to steer the program.

The files with events have to be prepared before in a special format. The unfolding is done in three passes:

Pass 1: In this pass the user has access to every event and may modify the data, for example to change the energy calibration, if energy is measured with a calorimeter. In this step the events are stored in an intermediate file, to be used in the later steps. During pass 1 limits of the variables are determined and a statistic of the events is collected.

Pass 2: The histograms for all types of data are collected and the unfolding fit is performed.

Pass 3: This step is optional and may be used to generate histograms, which compare the measured variables with the sum of the Monte Carlo events after weighting with the result of unfolding and the background events.

This scheme allows maximum flexibility in the detailed data analysis. The execution time for one unfolding is very short (usually it is much shorter than the reconstruction time of a single event, or the time of a detailed simulation of one event). The time to setup the unfolding job and to get the first meaningful unfolding result may be a few hours only. However the program allows to study the data and the simulation in great detail, which may require to repeat the unfolding with different conditions many times (for example one could study the effect of calibrations constants on the result). Any systematic error in the simulation may cause systematic effects in the simulation. If effects of this kind are detected during unfolding then very time-consuming studies may become necessary to understand the detector response and to improve the simulation methods.

2 Input data

The unfolding program requires as input the following information:

- ***N*tuple data**
  The measurement data from the experiment, from the Monte Carlo simulation of the measurement process, and for the background subtraction are required in the form of $n$tuple data. Two $n$tuple files A and B are used within the program. The file A has to be filled by the user by calling subroutine UEVENT and may contain more types of $n$tuples than used in the unfolding. The file B is used internally.
- **Function $f_0(x)$**
  The internal unfolding result is defined *relative* to the x-dependence $f_0(x)$, assumed in the generation of the Monte Carlo $n$tuples. The user has to supply a FORTRAN function with the name XFUNCT(X), which has to return the function value $f_0(x)$ for the value X given in argument. At least a dummy version of this function is necessary, if the dependence $f_0(x)$ is to be determined by the unfolding program.
- **Definition of parameters and selection of options**
  Some parameters of the unfolding program have to be defined by the user. Other parameters can either be determined automatically by the program or can be defined by the user. Text strings are used to define parameters and to select special options of the program.

2.1 *N*tuples

Each single set of measurement data is represented by the following set of values:

LABEL	=	string of 4 characters
NY	=	number of Y-values
WT	=	weight of the event
X	=	value of the unfolding variable (true variable)
Y(1)...Y(NY)	=	array of measured or simulated data

Each set of measurement data is called a *n*tuple and has to be stored on the *n*tuple file A.
On the file each *n*tuple carries a LABEL, which is a string of four characters, which characterizes the type of the *n*tuple (data, Monte Carlo, type of background). The NY values Y(1) ... represent the measured data. The value X $\equiv$ Y(0) is the true value of the unfolding variable, as used in the Monte Carlo generation of the *n*tuple (in data and background *n*tuples the value of X is of course not defined and may be e.g. zero in the call). The variable WT is the weight of the *n*tuple. The weight of data *n*tuples should be always equal to 1. The weight for Monte Carlo events may differ from event to event. The weight of background events is a relative weight; it is usually changed for unfolding in order to get the correct normalization of the background contribution.
The number NY of measurement variables Y(j), which can be stored on the *n*tuple files, is limited to 100; the variables used directly in the unfolding fit have to be among the first 32 variables ($1 \le j \le 32$). In principle only those variables directly related to the unfolding variable X are necessary. However it may be very useful for detailed checks of the whole measurement, simulation and unfolding procedure, to include further variables in the *n*tuples. A weight factor is calculated after the unfolding for each simulated *n*tuple, which allows to generate histograms of all measurement variables corresponding to a *reweighted* simulation. The comparison of these histograms from the experiment and from the simulation allows sensitive checks of the simulation.
In the preparation of the files, one *n*tuple is stored on the standard file A by the call:

```
CALL UEVENT(LABEL,NY,WT,X,Y)
```

Any order is allowed for the different *n*tuples. Since the writing to the file is buffered, there has to be a final call to write the buffer, with NY=–1 (all other arguments are arbitrary):

```
CALL UEVENT(LABEL,–1,WT,X,Y)
```

By default the unit numbers 51 and 52 are used for the *n*tuple files A and B. Different unit numbers may be assigned by the user using keyword NTUPLEUNITS (see section on Unfolding parameters). The programs opens the two files as scratch files, unless they are already opened by the user. The capacity of the two files has to be sufficient for

N*4*(4+NY) bytes

for N *n*tuples. File A may be a permanent file, created in an independent job. File B is used within the unfolding program to store intermediate information.
If the *n*tuples are already stored in LOOK vectors, then the vectors to be used can be specified as parameters (see next subsection) and the standard subroutine FRLOOK can be used for the format conversion from LOOK *n*tuples to RUN *n*tuples.

2.2 Function $f_0(x)$

The dummy version of the function XFUNCT(X) has to return the value 0.0 for any value of the argument X and looks as follows:

```
      FUNCTION XFUNCT(X)
*     dummy version returning 0.0
      XFUNCT=0.0
      END
```

The user version of this function has to return the value of the Monte Carlo input function $f_0(x)$ for the value x of the argument.

2.3 Unfolding parameters

The unfolding procedure requires a number of parameters to be defined and of options to be selected by the user. These parameters are defined by text strings, which can be either read in from a text file (CALL RDCAIM(lun)) or can be specified in calls of a subroutine (CALL CAIM(text)). Below is an overview over all options and parameters. Each text string carries a keyword, which is shown in upper case characters below, followed by constants, which are shown by a name in lower case characters below.

PRINTFLAG	pflag	optional
TITLE	title	optional
DATA	label lumi	mandatory
MONTECARLO	label lumi weight	mandatory
BACKGROUND	label evtnr	mandatory
VARIABLE	j nbin ! description	mandatory
XLIMITS	xlow xhigh	mandatory
XBINS	x1 x2 x3 ... xl	
KNOTS	knots	recommended
NRDF	nrdf	recommended
CONSTRAINT	x fx	
LINWEIGHT	wlin	
ZEROLEFT		
ZERORIGHT		
FXPOSITIVE		
USEFRACTION	frac	
SMOOTHX		
LENGD	lengd	
NTUPLEUNITS	luna lunb	

2.3.1 Title for the problem

Some text characterizing the problem is specified with the TITLE keyword.

TITLE	title	optional

The text is printed within the general printout of the program.

2.3.2 Selection of data

By the following textstrings the $n$tuples to be used in the unfolding are selected in three categories. The $n$tuples from the experiment are called DATA $n$tuples and are selected with the keyword DATA. The Monte Carlo $n$tuples from simulation defining the resolution are selected with the keyword MONTECARLO and the keyword BACKGROUND is used to select the background. Each type of $n$tuple is selected by specification of the label, which is a four-character string. Up to 32 different labels may be specified.

DATA	label lumi	mandatory
MONTECARLO	label lumi weight	mandatory
BACKGROUND	label evtnr	mandatory

For data $n$tuples a luminosity of 1 is assumed, if the luminosity is not specified. In the case of several types of data $n$tuples all data are combined to one data sample and the luminosities are added together. For Monte Carlo $n$tuples the numerical parameters are luminosity and (optional) a weight factor, assumed to be 1.0 if not specified. If only one type of Montecarlo $n$tuple is specified, then there should be no reason to specify the weight factor. The case of missing specification of the Montecarlo luminosity is discussed earlier. If more than one type of Montecarlo $n$tuples is specified, then the Luminosity of each type should be specified. In addition the weight factors $\mathcal{W}_j$ should be specified such that there sum is 1.0: $\sum_j \mathcal{W}_j = 1$. This allows to *mix* different Montecarlo simulations together in order to get the optimal description of the measurement.
For background contributions, the constant evtnr is the number of entries, to which the background contribution has to be normalized. This number should refer to the *total* data sample, which is selected for use in the unfolding. If within the user program further cuts are applied to the data, which reduce the total number of data and background events, the constant **evtnr** can remain unchanged.
Additional parameters are required, if the $n$tuples are already stored as LOOK vectors (VEC type) with a certain figure number and second number ifg and INR.

DATA	label lumi LOOK ifg inr	mandatory
MONTECARLO	label lumi weight LOOK ifg inr	mandatory
BACKGROUND	label evtnr LOOK ifg inr	mandatory
FRLOOK	wtindex xindex	mandatory

With the FRLOOK keyword the indices wtindex and xindex are specified, which elements of the LOOK vector are the weight WT and the true variable X.
With these additional parameters the internal $n$tuple-files of $\mathcal{RUN}$ are created by the call of subroutine FRLOOK within the MAIN program.

2.3.3 Variables

Variables are numbered 0, 1, ... The unfolding variable is X = Y(0). The variables Y(1) ... are the measured variables, and only variables Y(1) to Y(32) can be used directly as fit variables in the unfolding fit (the $n$tuple file may contain up to 100 measured variables). If a variable Y(j) with j between 1 and 32, is to be used in the unfolding fit, the user has to specify the number of bins nbin for the variable. Up to three variables can be used in the unfolding fit. Note that the total number of bins in the distribution is the *product* of all (non-zero) nbin values. A value nbin = 0 or a missing parameter nbin indicates, that the variable is not used in the unfolding fit. The text given after an exclamation mark (!) is a description of the variable and is printed in control plots with the distributions.

VARIABLE	j nbin !	for fit variables
XLIMITS	xlow xhigh	mandatory
XBINS	x1 x2 x3 ... xl	

From all fit variables the one- or two- or three-dimensional histogram is accumulated for the unfolding fit using the given bin numbers. By default the limits of the histogram are determined automatically, to include *all* n-tuples. Sometimes it may be neccessary to exclude n-tuples with certain extreme values. The user may exclude $n$tuples with values outside certain limits in pass 1 by a cut, with specification of a negative weight for the rejected $n$tuples. The user may also specify the boundaries of the histogram in the VARIABLE statement:

VARIABLE	j nbin lowlimit highlimit ! description	
		mandatory for fitting variables

Here lowlimit and highlimit are the lower and upper limit of the accepted range of the variable; $n$tuples with values outside these limits are automatically excluded by specification of a negative weight. For variables which are not fitting variables, only a description should be given:

VARIABLE	j ! description	optional

To allow further checks of the unfolding result the histograms of the measured variables are accumulated automatically for the first 32 variables using the fixed number of 120 bins. Bin limits for all these variables are determined automatically to include *all* accepted $n$tuples.
For the unfolding variable x it is mandatory to define limits with the keyword XLIMITS. Usually the limits will just be the total physical range of the variable x. It is not meaningful to make the range larger. Due to potential migration effects it may be problematic to make the range smaller, and a warning is printed in this case in the program (see discussion below).
Bin limits for the unfolding solution are determined automatically with a certain algorithm. The user may in addition define new bin limits with keyword XBINS; in that case the result is printed for both definition of bins.

2.3.4 Unfolding parameters

The variable nrdf is the number of degrees of freedom. This is a very important parameter, since it defines effectively the regularization of the result: the regularization parameter is calculated such that nrdf will be the effective number of degrees of freedom. If the value of nrdf is too small, then the result may be distorted, because already significant amplitudes are damped. There is a recommendation for a lower limit of nrdf printed with the result, but this can only be an estimate. The number of data points should be roughly equal to nrdf. If the number of data points is much larger than nrdf, they are necessarily positively correlated to each other.
If nrdf is zero or not specified, the program determines automatically a number by a certain algorithm, but it is recommended to fix this number.
The variable knots is the number of knots of the spline function, used in the unfolding. It should be about twice the number nrdf, for example nknots = 2*nrdf + 3; the exact choice should be uncritical. A value of knots which is too large for the given resolution has the effect, that some of the linear combinations of B-splines are not sensitive to the data. These linear combinations are automatically exclude in the calculation.

KNOTS	knots	recommended
NRDF	nrdf	recommended
CONSTRAINT	x fx	
LINWEIGHT	wlin	
ZEROLEFT		
ZERORIGHT		
FXPOSITIVE		

The resulting unfolded distribution can (perhaps always) by definition take on only positive values, and normally the result of the unfolding is positive without any further action. Sometimes there could be some problem with very small or even negative function values during the fit, especially in regions of small function values and in case of bad statistics (or systematically wrong simulation). By selecting the option FXPOSITIVE the solution is forced to be positive in the whole region of x by adding internally certain constraints to the solution.
The user may in addition require a solution, where the function value $f(x)$ at a certain x is fixed. This may be selected by the option CONSTRAINT (up to 10 pairs). This strategy may be necessary, if the

whole physical region is *not* accesssible to the measurement (see comments below), to take migration into account.
The unfolding solution is regularized using the square of the second derivatives. The square of the first derivative may be included in the regularization with a weight given in the LINWEIGHT option. If the distribution $f(x)$ is necessarily zero at the left or right limit of its range, then this condition may be included in the calculation as an additional constraint. If the solution has to become zero at the left and/or the right end of the x region, the user may specify the options ZEROLEFT or ZERORIGHT.

2.3.5 Further options

In addition there are the following options:

PRINTFLAG	pflag	
USEFRACTION	frac	
SMOOTHX		
NTUPLEUNITS	luna	lunb
LENGD	lengd	

The amount of printout during unfolding is controlled by the printflag pflag, which has a default value of 3 (full printout). The meaning of the print flag is as follows:

pflag	meaning
0	no printout
1	only errors and warnings
2	short - important parameters only
3	normal printout (default)

For tests it is possible to speed up the program by the option USEFRACTION by using only a fraction frac of the data. The program automaticaly determine the Montecarlo input function $f_0(x)$ from a histogram of the Montecarlo $n$tuples, if this function is not provided by the user. For the option SMOOTHX the histogram is smoothed before used as the input function, with an algorithm which tries to follow small structres in the histogram. If the function is known to be very smooth, then the option SMOOTHX should not be specified; in this case the histogram is fitted to a spline function with the number of knots as specified with the KNOTS keyword.
Default unit numbers for the two $n$tuple files are 51 and 52. Using the keyword NTUPLEUNITS these two unit numbers may be redefined. The keyword LENGD allows to specify the length of the common UNCOMD, which has a default length of LENGD=50000 WORDS.

3 The USER program

In this section a model MAIN program is listed, which shows the general structure of the unfolding program. Unfolding is done in three passes, as is shown. The flexible structure allows the user to add his own code to the program, to interface the program to the user data and software.

3.1 The Model MAIN program

The model MAIN program is listed below. Each part of the MAIN program is explained and discussed in the next subsection.

```
*        Model MAIN program
         COMMON/UNCOMA/IDMB,NY,WT,X,Y(100)
```

```
          PARAMETER (LENGD=20000)
          COMMON/UNCOMD/H(LENGD)
          CHARACTER*4 LABEL
*          ...
*
*         Definition of steering parameters ...
*          ... by calls        or      via text file
          CALL CAIM('text')      CALL RDCAIM(lun)
           ...
          CALL CAIM('text')
*
*         Filling of Ntuple-file A (see remark)
*          ...
          CALL UEVENT ...
          CALL FRLOOK       ! alternative for LOOK vectors
*
*         ———————— Pass 1 of unfolding
*
10        CALL RPASS1(LABEL)
          IF(LABEL.NE.'    ') THEN
           ... (user modification of data in COMMON/UNCOMA/)
            GOTO 10
          END IF
*
*         ———————— Pass 2 of unfolding
*
          CALL RPASS2
*
*         ———————— Pass 3 of unfolding
*
30        CALL RPASS3(LABEL)
          IF(LABEL.NE.'    ') THEN
           ... (user plots from data in COMMON/UNCOMA/)
            GOTO 30
          END IF
*
*         LOOK plots (optional)
*
          CALL RULOOK
*
*         Printer plots (optional)
*
          CALL RUNPRT
          STOP
          END
```

3.2 Parts of the MAIN program

3.2.1 Common dimension parameter

Internally several histograms are accumulated from the *n*tuples for the unfolding fit. These histograms are stored in the common UNCOMD, which has a default length of LENGD = 20000 words. If two- or three-dimensional histograms are used in the unfolding with a large number of bins, a larger common length may be necessary. The required length of the common will be printed during the unfolding und the program will stop in case of insufficient common length. To increase the length of the common the parameter LENGD in the parameter statement has to be modified, in addition the text string

    LENGD   lengd

has to be added to the definition of steering parameters.

3.2.2 Definition of steering parameters

Steering parameters are defined via text strings. The text strings may be either in the argument of subroutine CAIM or may be in a text file, read by subroutine RDCAIM(lun) from unit lun until the text string

    END

is found or until the end-of-file.

3.2.3 *N*tuple file A

As mentioned before all *n*tuples have to be stored in the *n*tuple file A before the unfolding. Usually the *n*tuples will exist in some files of a standard histogram package like LOOK, and they have to be copied to file A using the call of subroutine UEVENT. If the file A already exists or if the user wants to use a permanent file, it has to be opened by a statement like

```
OPEN(UNIT=...,FORM='UNFORMATTED',ACCESS='SEQUENTIAL',FILE=filename)
```

Default unit numbers for the files A and B are 51 and 52. If different numbers have to be used, the user can either define these numbers in the NTUPLEUNITS control statement or by a call:

```
CALL ABUNIT(luna, lunb)
```

Scratch units are opended automatically within the program.

3.2.4 Pass 1 of unfolding

In the pass 1 the unfolding is prepared. One *n*tuple is read at a time into the common array

    COMMON/UNCOMA/IDMB,NY,WT,X,Y(100)

and is available to the user in the MAIN program; in addition the label of the current *n*tuple is available as argument LABEL of type CHARACTER*4 of the subroutine RPASS1.

LABEL	=	string of 4 characters
IDMB	=	1 for Data
	=	2 for Monte Carlo
	=	3 for background
NY	=	number of Y-values
WT	=	weight of the event
X	=	value of the unfolding variable (true variable)
Y(1)...Y(NY)	=	array of measured or simulated data

At this point the user has the freedom to reorganize or modify the data in the common. Such a reorganization or modification of the data may be necessary to apply a certain correction to the data or to study systematic effects, which may affect the unfolding.

Example. Assume that one of the measured values is an energy, measured with a calorimeter. One of the uncertainties may be the calibration factor. In order to study the effect of this calibration factor on the unfolding result one can repeat the unfolding several times with different values of the calibration factor, using the identical $n$tuple repeatedly.

A rejection of certain $n$tuples, which should not be used in the unfolding, is also possible at this point. The recommended action to reject a $n$tuple is to set the weight WT to its negative value, with a statement

```
WT = - IABS(WT)
```

within the RPASS1 loop. Ntuples with negative weight will not be used in the unfolding, however the previous (positive) weight is still known and counted for the normalization, for example of the background $n$tuples to the total number of events; thus the number evtnr, given with the keyword BACKGROUND can remain unchanged, if another cut is added within pass 1. For background $n$tuples the weights WT are accumulated in pass 1; after pass 1 the weight for background $n$tuples is changed automatically according to the number evtnr. For Montecarlo $n$tuples the weight may be modified for example to mix different types of $n$tuples with different weight.

Significant for the unfolding is the $n$tuple *after* possible modifications in the MAIN program. At the next entry of the subroutine RPASS1 (after the statement GOTO 10, the actual content of the common UNCOMA is written to the second n-tuple file B, which is internal and used for the unfolding.

Furthermore the following is done:

The events of the different types are counted, the sums of the weights etc. are accumulated separately for each type. For the measurement values Y(j) the minimum and maximum values from $n$tuples of positive weight are determined from the combined data and Montecarlo $n$tuples. Furthermore a histogram of the *true* values X is accumulated for the Montecarlo $n$tuples. For printflag $>$ 2 the accumulated data are printed at the end of pass 1. The minimum and maximum values of the measurement values will be used later in the definition of the limits of the histograms of the variables.

3.2.5 Pass 2 of unfolding

The second pass of the unfolding is executed with the call of RPASS2 and runs complete automatic. It uses the $n$tuples from $n$tuple-file B and fills all the histograms necessary for the unfolding fit. Then the solution is determined by a fit of the Montecarlo histograms to the data histograms, using the parameter values as specified before.

The data from $n$tuple-file B are actually read twice. First the complete one- or two- or three-dimensional histogram is accumulated and all non-zero bins are determined. Then only the non-zero bins are stored for the histograms, which are now accumulated separately for each knot of the B-spline. Thus in total knots (see keyword KNOTS) histograms are accumulated from the Montecarlo $n$tuples.

3.2.6 Pass 3 of unfolding

This pass is optional and not necessary to obtain the unfolding result. However it is highly recommended to execute this step, which allows a sensitive test of all the steps in the analysis from the measurement, the simulation up to the final determination of the unfolding result.

During pass 3 the $n$tuple data are available, one at a time, exactly in the same way as they were in Pass 1 (after the modifications by the user), with the only exceptions, that the weights now take into account the result of the unfolding. This means that for the Montecarlo $n$tuples the weight is modified and corresponds to the unfolding result. For the background $n$tuples the weight is already adjusted to the normalization, as requested by the steering parameters. This allows to accumulate histograms

Group	Explanation
1	Result $f(x)$ of unfolding and related x dependent plots
2	Orthogonal functions of x
3	Miscellaneous: coefficients, probability distribution
4	MC histogram and B-spline decomposition from MC storage in /UNCOMD/H
5	Data and MC after unfolding, constant bin width in HIST(120,3,32), collected in RPASS3
6	Data and MC after unfolding, variable bin width
7	Difference between data and MC description, constant bin width
8	Difference between data and MC description, variable bin width

Groups of standard plots generated as graphical output by $\mathcal{RUN}$.

with any histogram package, separately for the data, and for the Montecarlo and background $n$tuples. The latter ones should provide a good description, including the normalization, to the data histograms. Any discrepancies can point to a potential problem in the understanding of the data. Here one has to distinguish the histograms of those variables, which were used in the unfolding fit, and those that were not. On average the histograms used in the unfolding fit should agree; any systematic effect and difference between the data from the experiment will distort the fit result and thus the unfolding result, because the fit forces some agreement. This is different for the histogram of variables, which were not used directly in the fit. All deviations in these histograms have to be checked. They *can* point to a wrong understanding (or simulation) of the detector. It is a general experience that the most time-consuming part in data analysis is the test and improvement of the understanding of the detector response. Histograms accumulated in pass 3 of the unfolding can be very helpful in this process.

3.2.7 Graphical output

Graphical data including histograms are accumulated internally during the passes 1, 2 and 3 of the unfolding. Interfaces to the packages LOOK and HBOOK/PAW exist.
With a call of subroutine RULOOK, which is the more complete interface, these data are transferred to the LOOK histogram system; there are eight groups of standard plots in this interface, as described below. The subroutine RHBOOK creates only a subset of these data (the additional subroutine THBOOK creates some plot form the example program).
For other histogram packages no subroutine is supplied, but probably it is easy to code such subroutine according to the existing subroutine RULOOK. A look at the plots generated by $\mathcal{RUN}$ is considered as an important part of the check of the unfolding. An alternative to a graphical output is the call of subroutine RUNPRT, which generates some graphical output on a normal printer.
An overview over the plots from the eight groups is given in a table.

number	explanation
	group 1:
101	data points as result from unfolding (x)
102	result of unfolding: data points and curve
103	(n+1)-orthogonal function
104	User function (x)
105	input x histogram and fitted/smoothed histogram (x)
106	weight distribution (x)
	group 2:
201	1.st orthogonal functions
200+j	j.th orthogonal function
200+nknot	nknot.th orthogonal functions
	group 3:
301	coefficients and damping of coefficients (index)
302	probability histogram of correlation test (Prob)
	group 4
411	Histogram of first fit variable and B-splines 1, 5, 9 ...
412	Histogram of first fit variable and B-splines 2, 6, 10 ...
413	Histogram of first fit variable and B-splines 3, 7, 11 ...
414	Histogram of first fit variable and B-splines 4, 8, 12 ...
421	Histogram of second fit variable and B-spline 1, 5, 9 ...
422	Histogram of second fit variable and B-spline 2, 6, 10 ...
423	Histogram of second fit variable and B-spline 3, 7, 11 ...
424	Histogram of second fit variable and B-splines 4, 8, 12 ...
431	Histogram of third fit variable and B-spline 1, 5, 9 ...
432	Histogram of third fit variable and B-spline 2, 6, 10 ...
433	Histogram of third fit variable and B-spline 3, 7, 11 ...
434	Histogram of third fit variable and B-splines 4, 8, 12 ...
	Group 5, with narrow bins of equal size:
501	histogram of variable 1, Monte Carlo and background
502	histogram of variable 2, Monte Carlo and background
...	
532	histogram of variable 32, Monte Carlo and background
	Group 6, same with variable bins:
601	histogram of variable 1, Monte Carlo and background
602	histogram of variable 2, Monte Carlo and background
...	
632	histogram of variable 32, Monte Carlo and background
	Group 7, histograms with equal bin size:
701	Difference between data and fit for variable 1
702	Difference between data and fit for variable 2
...	
732	Difference between data and fit for variable 32
	Group 8, variable bins:
801	Difference between data and fit for variable 1
802	Difference between data and fit for variable 2
...	
832	Difference between data and fit for variable 32

3.2.8 Storage of the result

After pass 2 the result of the unfolding is stored in a compact form in array XW in the common /RUNRES/XW(527). All values are in floating-point representation. This is the result as requested by the user defined bins in x under keyword XBINS.

index	word content	number of words
1	number of bins (NB)	1
2	first bin lower x-value	NB+1
3	second bins lower x-value	
...	...	
NB+2	last bin upper x-value	
NB+3	content first bin	NB
NB+4	content second bin	
...	...	
NB+NB+2	content last bin	
NB+NB+3	first element of covariance matrix	(NB*NB+NB)/2
NB+NB+4	second element ...	
...	...	

The total number of words is 2+NB+NB+(NB*NB+NB)/2. The array is printed using the format (2X,5E14.6) and can be read in again for further calculations using this format.

There is an utility subroutine, which allows to calculate the result for any bins.

```
CALL  RURSLT(NB,BINLIM,IPRINT,WW)

Input:
           NB =  0  automatic bin limits
              <  0  equidistant bins
              >  0  bins limits in BINLIM
           BINLIM(1) ... BINLIM(NB+1) = bin limits in x
                          in ascending order (only for NB > 0)

       IPRINT =  0  result calculation only
              =  1  compact printout of result array
              =  2  printout of result
              =  3  printer plot of result
              =  3  analysis of covariance matrix and printout
 Result:
        WW(.) =     array with the same definition as above for XW
                       dimension 2+NB+NB+(NB*NB+NB)/2
```

4 Discussion of practical aspects

4.1 The unfolding variable X and related parameters

The unfolding variable x or X is the *true* variable, defined only in the Montecarlo $n$tuples. The range of this variable is defined by

$$\mathsf{X}\mathit{LOW} \leq \mathsf{X} \leq \mathsf{XHIG}\ .$$

This range should be the total range of the unfolding variable. Internally B-splines are used to parametrize the solution. The B-splines are defined by $n = \mathsf{nknots}$ equidistant knots, with a distance of $(\mathsf{XHIG} - \mathsf{XLOW})/(\mathsf{nknots} - 3)$ between knots.

$$x_2 = \mathsf{XLOW} \qquad\qquad x_{n-1} = \mathsf{XHIG}\ .$$

4.2 Fit variables

The variables selected by the keyword VARIABLE are directly used in the unfolding fit and are called here *fit* variables. The number of bins defined with this keyword defines the binning made for the unfolding fit, since the fit itself works with *binned* distributions of the variables. The binnned distribution, or histogram, may be one-dimensional, two-dimensional or three-dimensional. The total number of bins may become rather large if more than one dimension is used.

Binning means a potential loss of resolution and in order to reduce this unwanted effect the width of the bins should be not much larger than the resolution, measured for example by the standard deviation. On the other hand the limited statistics of simulated events may require a larger bin width. Since the bins have a constant width, a variable with a constant resolution is preferred. Often a certain transformation can be made to achieve a constant resolution or at least a shape of the distribution more suited for the unfolding.

The following two examples show, how variables with certain properties of their resolution are transformed to new variables with a resolution, which is independent of the new variable. The standard deviation of the measurement of a *particle energy* E by calorimetric methods is often given by

$$\frac{\sigma_E}{E} = \frac{\text{const}}{\sqrt{E}}$$

(the relative error decreases with increasing energy). By the *square root* the variable E can be transformed to a new variable h with a constant resolution:

$$h = \sqrt{E} \quad \text{with} \quad \sigma_E = \text{const}\,\sqrt{E} \quad \Rightarrow \quad \sigma_h = \frac{1}{2}\,\text{const}\ .$$

The standard deviation of the measured *momentum* p of a charged particle (charge $\pm$e =elementary charge) in a magnetic detector is often given by

$$\frac{\sigma_p}{p} = \text{const} \cdot p$$

(the relative error is proportional to the momentum). The inverse value q of the momentum will have a constant resolution:

$$q = 1/p \quad \text{with} \quad \sigma_p = \text{const} \cdot p^2 \quad \Rightarrow \quad \sigma_q = \text{const}\ .$$

The new variable q may carry the sign of the charge of the particle. The region around zero in the variable q is meaningful and describes the possibility to measure the wrong sign of the particle charge.

The shape of the distribution of the transformed quantities may allow to see some effects, which are invisible in the original variables due to limited statistics. Such a property of a transformation may be of advantage irrespective of the question of resolution. Another example for this is from charged lepton-nucleon deep inelastic scattering, where the variable Q^2 follows essentially a $1/Q^4$ distribution, which means a distribution in the variable Q^2 above a fixed minimum value of Q_0^2 with a long tail of decreasing statistics. The variable $k = Q_0^2/Q^2$ will approximately follow a uniform distribution between $k = 1$ and $k = 0$ (which corresponds to $Q^2 \to \infty$).

A large number of bins would be possible, if the Montecarlo statistic is much larger than the data statistic. However this is usually not the case in practice and this requires to use few bins. A analysis of the statistical precision of the data and Montecarlo bins is printed in the standard output.

4.3 Vanishing acceptance in certain regions

Unfolding of a distribution $f(x)$ requires in principle a non-vanishing acceptance in the complete region, where x is defined; the migration effects at a certain value $x = x_c$, where $f(x_c)$ is required, can only be taken into account if $f(x)$ is also measured for all x-values, which could influence the data at $x = x_c$.

Theoretically $f(x)$ could even be measured in a region of vanishing acceptance, if by migration effects this region contributes to the region with non-zero acceptance, but in practice the statistical and/or systematic uncertainties will make this impossible. In practice it may be necessary to make some assumption about $f(x)$ in the region of vanishing acceptance, in order to calculate the migration to the region on non-vanishing acceptance.

The program $\mathcal{RUN}$ supports the analysis of those cases. The keyword CONSTRAINT allows the user to define up to twelve pairs of values $\{x, f(x)\}$ in regions of vanishing or very low acceptance. These constraints force a certain degree of migration to the region of higher acceptance. Unfolded values heavily influenced by these given pairs of values $\{x, f(x)\}$ of course do not represent a *measurement* and cannot be presented as such, but they help in understanding the really measured part of the range of the variable x. The sensitivity of the result can be determined by different assumptions on the pairs of values, and finally their effect can be taken into account by some *systematic* errors calculated for the measured values.

4.4 Discrete distribution

In special cases the variable x may be a *discrete* variable, as well as the measured variables. An example is the measurement of the distribution of the multiplicity of charged particles in particle reactions. The method used in the program $\mathcal{RUN}$ is derived on the assumption of continuous variables, especially the *smoothness* of the resulting distribution $f(x)$ plays an important role in the algorithm. This *concept* may not be applicable for certain problems with discrete variables. *Smoothness* may still have a meaning for the distribution of discrete variables, for example in many particle reactions one would expect a *smooth* distribution of the particle multiplicity, and in those type of problems the program may still be useable. However technically there may be some problems, eventually not immediately visible to the user.

First the problem of discrete *measured* values is discussed. For simplicity it is assumed that the discrete values are integer values, with a spacing of 1. The bin boundaries should be defined as non-integer values, in order to avoid wrong bin entries due to rounding errors in the variables (note that the measured values are of type floating point). Furthermore in this case the bin width used in all internal histograms should be 1, in order not to loose any resolution. In case of a large range of the variable the bin width could be a multiple of 1, in order to have a *controlled* addition of bins. This would also avoid a discontinous shape of the histogram of an otherwise smooth distribution, which could make a comparison of data and expected distribtion difficult (although the method does not require smoothness in the measured variable). The number of bins as well as the bin limits can be specified by the user with the keyword VARIABLE. An example of a definition with a bin width of 1 is

```
VARIABLE   1   27   1.5   28.5
```

for the (integer) range 2 to 28.

Smoothness is *essential* for the distribution of the unfolding variable x, and several parameters are related to this distribution. The final result is only meaningful, if the *bins* in x are defined by the user with the XBINS keyword. A natural choice would be a binwidth of 1, and bin boundaries in the middle between integer values. This could mean a bin width which is too narrow with respect to the resolution, and an alternative choice would be a bin width of 2 for example, in order to avoid too large errors due to the negative correlation between adjacent integer values. The bin boundaries as determined automatically in the program are probably meaningless in this case.

Further parameters related to the distribution of the unfolding variable x are the limits of x (with keyword XLIMITS) and the number of knots (with keyword KNOTS), which have to result in a knot distance of exactly 1 (or a multiple of 1 in case of a very large range of the variable x). Otherwise a smooth distribution would be converted to a distribution with a strongly fluctuating behaviour, which does not allow regularization in the sense of the program. For the above example of a discrete variable between 2 and 28 this would mean steering by

```
XLIMITS   1.5   28.5
NRDF      27
```

(compare line above with keyword VARIABLE above). In this example the number of knots is equal to the number of bins *plus* one, because the number of knots is equal to the bins *boundaries*, which is equal to the number of bins *plus* one.
Also in the case of a discrete variable a maximum of available information from the measurement should be used. In the example of the measurement of the distribution of the multiplicity of charged particles n_{ch} in particle reaction, the pair (n_+, n_-) with $n_{\mathrm{ch}} = n_+ + n_-$ will be measured in each individual particle reaction. The unfolding variable is $x \equiv n_{\mathrm{ch}}$ (true). As the measured quantity one could define $y_1 \equiv n_+ + n_-$ (measured), but more information is used if two measured variables $y_1 \equiv n_+$ and $y_2 \equiv n_-$ are defined and the unfolding fit is performed in the (n_+, n_-)-plane (another possibility is to use the measured variables $y_1 \equiv n_+ + n_-$ and $y_2 \equiv n_+ - n_-$). This could in addition improve checks of the simulation. While the unfolding fit in one dimension will perhaps give a perfect agreement, even if assumptions in the simulation were wrong, this would not happen in the (n_+, n_-)-plane. Furthermore a background contribution in the measurement probably has an asymmetric distribution around the line of charge conservation (fixed value of $n_+ - n_-$), which should be visible.

4.5 The Montecarlo statistic

The number of n-tuples for the Montecarlo sample should in principle be high, much higher than the number of n-tuples of the data sample, because the Montecarlo n-tuple have to give the measured distribution for each bin of the true variable. In other words: the dimension of the Montecarlo n-tuple distribution is higher than the data n-tuple distribution by one. A factor of ten more Montecarlo n-tuples should be a minimum requirement. If properly organized this requirement should not be a problem. Experience shows however that under certain circumstances the requirement is difficult to fulfil: often about the same number of n-tuples is available in both samples, which is insufficient and at least adds some statistical error from the Montecarlo sample to the result.

5 Program source

The program source of the $\mathcal{RUN}$ program will become available from CERN via anonymous soon.

The following table gives an overview over the files.

File	Content
runtest.f	MAIN test program and dummy XFUNCT function
example.f	subprograms for example job
runpack.f	basic subprograms of the unfolding program
ntuples.f	ntuple IO subprograms
random.f	random number generators
splft.f	smoothing package
minstep.f	subprogram for minimization in one dimension
matrix.f	matrix operations
hiscomp.f	histogram utilities
fpl.f	printer plot subprograms
splines.f	cubic B-splines subprograms
fspline.f	cubic splines subprograms
uprint.f	print utilities
cardin.f	text handling subprograms
runlook.f	graphics output subprograms, using the LOOK system
runpaw.f	graphics output subprograms, using the PAW/HBOOK system
manual.ps	Manual in postscript
readme.txt	information on files

Included in the source code file is a standard test case, which can be run on a new installation for a test. It also allows to study the effect of different values of the steering parameters.