Version 0.999 - 17. 10. 2000

Linear Least Squares Fits

with a Large Number of Parameters

1. Update

1. Problems with a large number of parameters
2. Least squares with global and local parameters
3. The program package MILLEPEDE

Volker Blobel

II. INSTITUT FÜR EXPERIMENTALPHYSIK
UNIVERSITÄT HAMBURG

2000

Abstract

In certain least squares fit problems with a very large number of parameters the set of parameters can be divided into two classes, global and local parameters. Local parameters are those parameters which are present only in subsets of the data. Detector alignment and calibration is one of the problems, where the interest is only in optimal values of the global parameters. A method to solve the linear least square problem, irrespectively of the number of local parameters, is derived in the paper. A practical limit for the number of global parameters is between one thousand and ten thousand.

The paper includes a description of a general program package Millepede for the efficient solution.

Contents

When all else fails,
read the instructions.

1 Problems with a large number of parameters

The most important general method for the determination of parameters from measured data is the linear least squares model. It is usually stable, fast and accurate, requires no initial values of parameters and is able to take into account all the correlations between different parameters, even if there are many parameters.
There is of course a practical limit in the number of parameters in simultaneous fits, because of space and time limitations.

Alignment problems for large detectors in particle physics often require the determination of a large number of alignment parameters, typically of the order of 100 to 1000. Alignment parameters for example define the accurate space coordinates and orientation of detector components. In the alignment usually special alignment measurements are combined with data of particle reaction, typically tracks from physics interactions and from cosmics. In this paper the alignment parameters are called *global* parameters; they contribute to all the data. Parameters valid for a single track are called *local* parameters, examples are track slopes and curvatures.

One approximate alignment method is to perform least squares fits on the data *ignoring* initially the alignment parameters. The *residuals*, the *deviations* between the fitted and measured data, are then used to determine the alignment parameters afterwards. Single fits depend only on the small number of local parameters like slope or curvature of the track, and are easy to solve. This approximate method however is not a correct method, because the (local) fits depend on a wrong model (namely: ignoring the global parameters), and the result are biased fits. The adjustment of parameters based on the observed (biased) residuals will then result in biased alignment parameters. If these alignment parameters are applied as corrections in repeated fits, the remaining residuals will be reduced, as desired. However, the fitted parameters will still be biased, even more, if an iterative procedure is applied.

A better method would be an overall least squares fit, with all the global parameters and local parameters, perhaps from thousands or millions of events, determined simultaneously. The size of the vectors and matrices in the least squares solution will be large, of the order of 10^4 or 10^6. Due to the special structure of the matrices with one set of global parameters and many sets of local parameters the problem can be reduced to a solvable size, without making any approximations.

1.1 Linear models

In a linear model the measured quantity z_k has an expectation value which is a *linear* function of the parameters $\boldsymbol{a}$. The difference between the measured value and the expectation value is the residuum r_k:

$$z_k = \boldsymbol{a}^T \cdot \boldsymbol{d}_k + r_k \tag{1}$$

The measured data are assumed to be *independent*; the result of a measurement z_k does not dependent on any other measurement. Thus the covariance matrix of the measured data is a diagonal matrix, which simplifies the treatment. The accuracy of the measurement z_k is given by the variance σ_k^2 or the standard deviation σ_k. It is assumed, that the precision is at least approximately known.
The solution of the method of least squares[1] is determined by the minimum of the weighted sum

[1]The least squares formalism is for example derived in: V. Blobel und E. Lohrmann , Statistische und numerische Methoden der Datenanalyse, Teubner (1998). In this book the reduction method discussed in this paper is mentioned too.

of the square of the residuals, usually called χ^2:

$$\chi^2 = \sum_k \frac{\left(z_k - \boldsymbol{a}^T \cdot \boldsymbol{d}_k\right)^2}{\sigma_k^2} \tag{2}$$

where the sum (Index k) is over all measurements. This χ^2-sum is a quadratic function of the parameters $\boldsymbol{a}$, and the solution is given by a set of linear equations (so-called normal equations of least squares), which can be written in matrix form

$$\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b} \tag{3}$$

with a symmetric n-by-n matrix $\boldsymbol{C}$ and a right-hand-side n-vector $\boldsymbol{b}$, which are sums with, because the measured data are independent, *independent* contributions from each measurement:

$$\boldsymbol{C} = \sum_k w_k \left(\boldsymbol{d}_k \cdot \boldsymbol{d}_k^T\right) \tag{4}$$

and

$$\boldsymbol{b} = \sum_k w_k z_k \boldsymbol{d}_k \ . \tag{5}$$

There is no limitation on the number of measured data; it is not necesary to keep all input data in memory, because the complete information from a single measurement is contained in the contributions to $\boldsymbol{C}$ and $\boldsymbol{b}$. It should be remarked, that further techniques are possible, if the complete input is available in matrix form in memory; this would allow to apply the method of singular value decomposition (SVD), which allows to control the solution in an optimal way.

The solution vector is written in the form

$$\boldsymbol{a} = \boldsymbol{C}^{-1}\boldsymbol{b} \tag{6}$$

with the inverse of the matrix $\boldsymbol{C}$, which is, by error propagation, the covariance matrix of the parameters $\boldsymbol{a}$.

What are the limits for the number n of parameters, if the matrix equation is solved? The symmetry property of the matrix $\boldsymbol{C}$ may be used to store the n^2 elements in $(n^2 + n)/2$ words and also to reduce the computer time to almost half the time for a non-symmetric matrix. Matrix inversion is an explicit process and requires a CPU time proportional to n^3 for a n-by-n matrix. Different algorithms are available for the matrix inversion. A particular method, suitable for very large matrices, is explained below, which transforms the original matrix (in place) in n steps to the inverse matrix, keeping the symmetry property at every step. For a 1000-by-1000 symmetric matrix in double precision (about 4 Mbyte) a CPU time of a few minutes (on a Pentium 266 MHz CPU) is required. The time ($\propto n^3$) and space ($\propto n^2$) requirements imply at present a practical limit[2] of the size of the matrix (and number of parameters) between $n =$ 1 000 and $n =$ 10 000.

The accuracy of the computation of the inverse matrix is a critical point, and computation in double precision should be made. The accuracy is generally low for a bad condition number of the matrix (the condition number can be defined as the ration of the largest to the smallest eigenvalue of the matrix). From the interpretation of the inverse matrix as covariance matrix it is clear, that the condition number will be large if the correlation between parameters is

[2]For $n =$ 10 000 the required space for the matrix is about 400 Mbyte and the CPU time for inversion is a few days.

high. The matrix is singular (one or more eigenvalues are zero) in case of a complete correlation between parameters. A singular value decomposition would allow to recognize the cases of zero or small eigenvalues, which would *destroy* the full solution. The standard method in case of zero or very small eigenvalues is to set their inverse to zero; this means essentially to remove parameters or linear combinations of parameters from the solution. In the inversion method described below (subroutine `SPMINV`) a similar, but simpler strategy is used to avoid the effects of a bad condition number. This strategy also works in cases, where due to missing input data[3] one or more parameters are not defined; the parameter will become zero in this case. The strategy allows in addition to remove certain parameters from the least squares fit without the necessity to reorganize the numbering. Note that a standard matrix inversion program will fail in all these cases.

1.2 Inversion of large matrices.

The algorithm of subroutine `SPMINV` works in n steps and transforms the symmetric matrix *in place* to the inverse. Each element C_{ij} of the matrix is modified in a step, but still called C_{ij} in the description of the algorithm. In a single step of the algorithm a so-called *pivot element* has to be selected on the diagonal; the pivot element is called A_{kk}. All elements of the matrix are then transformed according to the following three rules:

1. The elements C_{ij}, which are neither on the pivot row nor column, are transformed by

$$C_{ij} := C_{ij} - \frac{C_{kj}C_{ik}}{C_{kk}} = C_{ij} - \frac{C_{jk}C_{ik}}{C_{kk}} \,, \tag{7}$$

   where in the latter formula the symmetry property of the matrix has been used;
2. the elements C_{ik}, C_{kj} in the pivot row and column are divided by the pivot element C_{kk};
3. the pivot element is replaced by the negative inverse of the element.

After selecting the pivot in any order in the n different positions on the diagonal the negative of the inverse matrix is obtained, and the last operation is the sign inversion of all matrix elements. Because of the division by the pivot element in each step, the algorithm will fail for a *zero* pivot element, and may become inaccurate for a small value of the pivot element. The strategy therefore should be to select, among the remaining candidates, in each step the largest element as the pivot element. For a singular matrix the strategy will give a zero pivot element (or an almost zero element) in some step and the process has to stop; a full matrix inversion is impossible. Elements in the corresponding rows and columns may be set to zero in that case, the resulting matrix is called $\boldsymbol{C}^{'-1}$. In least squares applications the vector of unknowns $\boldsymbol{a}$ represents *corrections* and those parts of the matrix, which are already "inverted", may be meaningful, and can be used to calculate by $\boldsymbol{a} = \boldsymbol{C}^{'-1}\boldsymbol{b}$ a correction vector. Elements in this correction vector corresponding to the zero matrix elements will be zero – which make sense, since the non-zero elements are meaningful. It corresponds to the case where one or more parameters from the model are fixed at some value.

The full inverse matrix is the covariance matrix of the vector of parameters, and represents the accuracy *and* the correlation between the parameters. The case of a very high correlation between two parameters could lead to a matrix which is numerically singular. The strategy above corresponds to fixing one of the parameters (correction zero) and determining the correction of the other parameter with good precision.

In least squares applications the diagonal elements will be positive, and for a well-defined problem will be dominant over the off-diagonal elements. As can be seen from the inversion formula, each step reduces the positive values of the diagonal elements:

$$C_{ii} := C_{ii} - \frac{C_{ik}^2}{C_{kk}} \tag{8}$$

[3] Large detectors may have dead channels, which means missing data.

thus all candidates for pivot elements in future steps become smaller; the reduction is stronger for larger values of C_{ik}, i.e. for larger correlations between parameters k and i. A good strategy to check the significance of pivot candidates C_{ii} is to compare in the search the initial values C_{ii} of the diagonal elements and the current reduced values of the diagonal elements. An element can not be used as pivot element, if the current value is $\varepsilon \times$ the original value or less, where ε represents the relative machine precision (e.g. $\varepsilon \approx 10^{-7}$).

1.3 Inversion by partitioning.

The special structure of the matrix $\boldsymbol{C}$ to be inverted may allow to move the limit of solvable problems further. Below the matrix $\boldsymbol{C}$ is partitioned into submatrices and the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are partitioned into two subvectors; then the matrix equation becomes

$$\left(\begin{array}{c|c} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \hline \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{array}\right) \left(\begin{array}{c} \boldsymbol{a}_1 \\ \hline \boldsymbol{a}_2 \end{array}\right) = \left(\begin{array}{c} \boldsymbol{b}_1 \\ \hline \boldsymbol{b}_2 \end{array}\right) \tag{9}$$

where the submatrix $\boldsymbol{C}_{11}$ is a p-by-p square matrix and the submatrix $\boldsymbol{C}_{22}$ is a q-by-q square matrix, with $p + q = n$.

If the sub-vector $\boldsymbol{a}_1$ would not exist, the solution for the sub-vector $\boldsymbol{a}_2$ would be defined by the matrix equation

$$\boldsymbol{C}_{22}\, \boldsymbol{a}_2^* = \boldsymbol{b}_2 \tag{10}$$

where the star indicates the special character of this solution, which is

$$\boldsymbol{a}_2^* = \boldsymbol{C}_{22}^{-1}\, \boldsymbol{b}_2 \; . \tag{11}$$

The solution $\boldsymbol{a}_2^*$ is called the *local* solution. Only the q-by-q sub-matrix $\boldsymbol{C}_{22}$ has to be inverted. In certain problems this may be easy, if for example $\boldsymbol{C}_{22}$ is diagonal.
Now, having inverted the sub-matrix $\boldsymbol{C}_{22}$, the submatrix of the complete inverse matrix $\boldsymbol{C}$ correponding to the upper left part $\boldsymbol{C}_{11}$, called $\boldsymbol{B}$, can be obtained by

$$\boxed{\boldsymbol{B} = \left(\boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^T\right)^{-1}} \tag{12}$$

which requires in addition to some matrix products the calculation of the inverse of a p-by-p sub-matrix. With this matrix $\boldsymbol{B}$ the solution of the whole matrix equation can written in the form

$$\left(\begin{array}{c} \boldsymbol{a}_1 \\ \hline \boldsymbol{a}_2 \end{array}\right) = \left(\begin{array}{c|c} \boldsymbol{B} & -\boldsymbol{B}\boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1} \\ \hline -\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^T\boldsymbol{B} & \boldsymbol{C}_{22}^{-1} - \boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^T\boldsymbol{B}\boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1} \end{array}\right) \left(\begin{array}{c} \boldsymbol{b}_1 \\ \hline \boldsymbol{b}_2 \end{array}\right) \tag{13}$$

which can be used to obtain the complete result ($\boldsymbol{a}_1$ and $\boldsymbol{a}_1$). Note that to solve the matrix equation two smaller matrices, a symmetric p-by-p and a symmetric q-by-q matrix, have to be inverted.
The sub-vector $\boldsymbol{a}_1$ is obtained by the product

$$\boldsymbol{a}_1 = \boldsymbol{B}\boldsymbol{b}_1 - \boldsymbol{B}\boldsymbol{C}_{12}\,\underbrace{\boldsymbol{C}_{22}^{-1}\boldsymbol{b}_2}_{\boldsymbol{a}_2^*} \tag{14}$$

which can be simplified by using the special local solution $\boldsymbol{a}_2^*$ to

$$\boxed{\boldsymbol{a}_1 = \boldsymbol{B}\left(\boldsymbol{b}_1 - \boldsymbol{C}_{12}\boldsymbol{a}_2^*\right)} \,. \tag{15}$$

If the interest is the determination of this sub-vector $\boldsymbol{a}_1$ only, while the sub-vector $\boldsymbol{a}_2$ is not needed, then only the boxed equations have to be solved. Some computer time can be saved by this method, especially if the matrix $\boldsymbol{C}_{22}^{-1}$ is easily calculated or already known before (note that the matrix $\boldsymbol{C}_{22}$ does not appear directly in the solution, only the inverse $\boldsymbol{C}_{22}^{-1}$).

This method of removing unnecessary parameters was already known in the nineteenth century[4]. The method can be applied repeatedly, and therefore problems with a huge number of parameters can be solved, if the interest is a subset of 1 000 to 10 000 parameters. The method is not an approximation, but is exact and it takes into account all the correlations introduced by the removed parameters. This type of application is discussed in more detail in the next chapter.

1.4 Constraints

If there are explicit relations between the parameters of a least squares fit, then they should be explicitly taken into account in the solution. The case of a single linear constraint between parameters of the form

$$\boldsymbol{a}^T \cdot \boldsymbol{f} = f_0 \tag{16}$$

is assumed. The Lagrange multiplier method is the standard method to include constraints in a least squares solution; one additional parameter λ is introduced for each constraint. The equation (3) is modified by the additional parameters to the form

$$\left(\begin{array}{c|c} \boldsymbol{C} & \boldsymbol{f} \\ \hline \boldsymbol{f}^T & \boldsymbol{0} \end{array}\right) \left(\begin{array}{c} \boldsymbol{a} \\ \hline \lambda \end{array}\right) = \left(\begin{array}{c} \boldsymbol{b} \\ \hline f_0 \end{array}\right) , \tag{17}$$

where the matrix on the left hand side is still symmetric. This shows that *linear* least squares problems with *linear* constraints can be solved direcly, without iterations and without the need for initial values of the parameters.

[4]One example is: Schreiber, O. (1877): Rechnungsvorschriften für die trigonometrische Abteilung der Landesaufnahme, Ausgleichung und Berechnung der Triangulation zweiter Ordnung. Handwriten notes. Mentioned in W. Jordan (1910): Handbuch der Vermessungskunde, Sechste erw. Auflage, Band I, §III: 429-433. J.B.Metzler, Stuttgart.

2 Least squares with global and local parameters

2.1 Local parameters

Local parameters are denoted by α_j, $j = 1, 2, \ldots \nu$. A set of measurements z is considered. A single value z from the set is ideally given by a linear expression $z = \alpha_1 \cdot \delta_1 + \alpha_2 \cdot \delta_2 + \ldots \alpha_\nu \cdot \delta_\nu$, depending on the parameters α_j and on (known) constant factors δ_j. More complete, the dependence can be written as

$$z_k = \alpha_1 \cdot \delta_{1k} + \alpha_2 \cdot \delta_{2k} + \ldots \alpha_\nu \cdot \delta_{\nu k} = \sum_{j=1}^{\nu} \alpha_j \cdot \delta_{jk} \ , \tag{18}$$

with an additional index k indicating the k-th measured value.
As an example the measurement of a straight particle track is considered, where each z_k is a space measurement at plane k. The local parameters are intercept (α_1) and slope (α_2) of the particle track. If the distance of the plane is denoted by S_k, the linear track model can be written as

$$z_k = \alpha_1 \cdot 1 + \alpha_2 \cdot S_k \ .$$

From a least squares fit of the model to the set of measured z-values the local parameters intercept and slope can be determined. The formalism of least squares allows to take into account the different precision of the different measurements, by assigning a weight to each measurement:

$$\text{weight (determined from the standard deviation } \sigma) \qquad\qquad w_k = \frac{1}{\sigma_k^2}$$

in the least squares formalism.
The least squares solution, minimizing the weighted sum of squared residuals, is defined by the so-called normal equations, which can be written in terms of matrices

$$\boldsymbol{\Gamma} \boldsymbol{\alpha} = \boldsymbol{\beta} \,, \qquad \text{solved by} \quad \boldsymbol{\alpha} = \boldsymbol{\Gamma}^{-1} \boldsymbol{\beta} \,. \tag{19}$$

The elements of the matrix $\boldsymbol{\Gamma}$ and the vector $\boldsymbol{\beta}$ are formed by sums

$$\Gamma_{ij} = \sum_k w_k \cdot \delta_{ik} \delta_{jk} \qquad\qquad \beta_j = \sum_k w_k \cdot z_k \delta_{jk} \ . \tag{20}$$

These formulas are valid in the case of *uncorrelated* z-measurements. Uncorrelated measurement are assumed thoughout this paper. The inverse matrix $\boldsymbol{\Gamma}^{-1}$ is the covariance matrix of the parameter vector $\boldsymbol{\alpha}$.

2.2 Global parameters

Now global parameters are considered, which contribute to all the measurements. They are assumed to be *alignment parameters* here, although the formalism is general and can also be applied to other problems. Aligment parameters are defined to represent *corrections* to ideal (design) values; this parametrization adds further terms to the equations (18)

$$z = \underbrace{a_1 \cdot d_1 + a_2 \cdot d_2 + \ldots a_n \cdot d_n}_{\text{global parameters}} + \underbrace{\alpha_1 \cdot \delta_1 + \alpha_2 \cdot \delta_2 + \ldots \alpha_\nu \cdot \delta_\nu}_{\text{local parameters}} = \sum_{i=1}^{n} a_j \cdot d_j + \sum_{j=1}^{\nu} \alpha_j \cdot \delta_j \ , \tag{21}$$

Usually only few terms containing global parameters are nonzero for a single measurement.
In the following it is assumed that, as written in equation (21), the dependence on the global parameters is *linear*. A non-linear relationship for the global parameters would require iterations, starting with reasonable initial values for the parameters, and assuming a linearized expression in each iteration; these are standard methods for nonlinear least squares problems.

2.3 The simultaneous fit of global and local parameters

In the following it is assumed that there is a set of N partial measurements. Each partial measurement, with index i, depends on local parameters $\boldsymbol{\alpha}_i$, and all of them depend on the global parameters. In a simultaneous fit of all global parameters plus local parameters from N subsets of the data there are in total $(n + N \cdot \nu)$ parameters, and the standard solution requires the solution of $(n + N \cdot \nu)$ equations with a computation proportional to $(n + N \cdot \nu)^3$. In the next chapter it is shown, that the problem can be reduced to n equations with a computation proportional n^3.

Generalizing the formalism of equations (18) to the complete measurement, starting from equations of the type of (21), one obtains a system of normal equations with large dimensions, as is shown in equation (22).
The matrix on the left side of equation (22) has, from each partial measurement, three types of contributions. The first part is a contribution of a symmetric matrix $\boldsymbol{C}_i$, of dimension n (number of global parameters). All the matrices $\boldsymbol{C}_i$ are added up in the upper left corner of the big matrix of the normal equations. The second contribution is the symmetric matrix $\boldsymbol{\Gamma}_i$ (compare equation (20)), which gives a contribution to the big matrix on the diagonal and is depending only on the i-th partial measurement. The third contribution is a rectangular matrix $\boldsymbol{G}_i$, with a row number of n (global) and a column number of ν (local).
There are two contributions to the vector of the normal equations, $\boldsymbol{b}_i$ for the global and $\boldsymbol{\beta}_i$ for the local parameters.
The complete matrix equation is given by

$$\left(\begin{array}{c||c|c|c} \sum \boldsymbol{C}_i & \cdots & \boldsymbol{G}_i & \cdots \\ \hline\hline \vdots & \ddots & 0 & 0 \\ \hline \boldsymbol{G}_i^T & 0 & \boldsymbol{\Gamma}_i & 0 \\ \hline \vdots & 0 & 0 & \ddots \end{array} \right) \cdot \left(\begin{array}{c} \boldsymbol{a} \\ \hline\hline \vdots \\ \hline \boldsymbol{\alpha}_i \\ \hline \vdots \end{array} \right) = \cdot \left(\begin{array}{c} \sum \boldsymbol{b}_i \\ \hline\hline \vdots \\ \hline \boldsymbol{\beta}_i \\ \hline \vdots \end{array} \right) \qquad (22)$$

In this matrix equation the matrices $\boldsymbol{C}_i$, $\boldsymbol{\Gamma}_i$, $\boldsymbol{G}_i$ and the vectors $\boldsymbol{b}_i$ and $\boldsymbol{\beta}_i$ contain contributions from the i-th partial measurement. Ignoring the global parameters one could solve the normal equations $\boldsymbol{\Gamma}_i \boldsymbol{\alpha}_i = \boldsymbol{\beta}_i$ for each partial measurement separately by

$$\boldsymbol{\alpha}_i = \boldsymbol{\Gamma}_i^{-1} \boldsymbol{\beta}_i \, . \qquad (23)$$

The complete system of normal equations has a special structure, with many vanishing sub-matrices. The only connection between the local parameters of different partial measurements is given by the sub-matrices $\boldsymbol{G}_i$ und $\boldsymbol{C}_i$,

2.4 Reduction of matrix size

The aim of the fit is solely to determine the global parameters; final best parameters of the local parameters are not needed. The matrix of equation (22) is written in a partitioned form. The general solution can also be written in partitioned form. Many of the sub-matrices of the huge matrix in equation (22) are zero and this has the effect, that the formulas for the sub-matrices of the inverse matrix are very simple.

By this procedure the n normal equations

$$\begin{pmatrix} & & \\ & \boldsymbol{C}' & \\ & & \end{pmatrix} \begin{pmatrix} \\ \boldsymbol{a} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \boldsymbol{b}' \\ \\ \end{pmatrix} , \tag{24}$$

are obtained, which only contain the global parameters, with a modified matrix $\boldsymbol{C}'$ and a modified vector $\boldsymbol{b}'$,

$$\boldsymbol{C}' = \sum_i \boldsymbol{C}_i - \sum_i \boldsymbol{G}_i \boldsymbol{\Gamma}_i^{-1} \boldsymbol{G}_i^T \qquad \boldsymbol{b}' = \sum_i \boldsymbol{b}_i - \sum_i \boldsymbol{G}_i \left(\boldsymbol{\Gamma}_i^{-1} \boldsymbol{\beta}_i \right) . \tag{25}$$

This set of normal equations contains explicitly only the global parameters; implicitly it contains, through the correction matrices, the complete information from the local parameters, influencing the fit of the global parameters. The parentheses in equation (25) represents the solution for the local parameters, ignoring the global parameters.
The solution

$$\boldsymbol{a} = \boldsymbol{C}'^{-1} \boldsymbol{b}' \tag{26}$$

represents the solution vector $\boldsymbol{a}$ with covariance matrix $\boldsymbol{C}'^{-1}$. The solution is direct, no iterations are required.

Iterations may be necessary for other reasons, namely

- the equations depend *non-linearly* on the global parameters; the equations have to be linearized;
- the data contain outlier, which have to be removed in a sequence of cuts, becoming narrower during the iteration;
- the accuracy of the data is not known before, and has to be determined from the data (after the alignment).

2.5 Application

A very important point in problems discussed above is the definition of the set of global parameters. It is essential to define them in a way, which is sensitive to the data used to determine them. Computing the inverse of a huge (symmetric) matrix can be a delicate numerical problem. The matrix will be singular or close to a singular matrix, if one or more of the global parameters are not really determined (and constrained) by the data, and this will destroy the whole determination. This will happen in case of a strong correlation between different global parameters.
It is wise to start from few global parameters and to observe the correlations between them, before adding more global parameters. For example if too many global parameters are used in a detector alignment, the whole detector may *rotate* freely in the fit. Usually it is necessary to fix certain global parameters and to use also data, that relate different parts of the detector. In this respect for example high-momentum cosmic muons may be useful, which traverse the whole detector and thus relate different parts of the detector.

3 The programm package MILLEPEDE

The program package with the name MILLEPEDES (german: TAUSENDFÜSSLER) consists of about 800 statements of user callable routines plus about 400 statements for matrix routines (these are: SPMINV for matrix inversion, SPAVAT for a special matrix product, SPAX for the product of a general matrix and a vector) and other utility subprograms. All arguments of the user routines are in *single* precision, while internal matrices and the matrix routines are in double precision.

3.1 Initialization

Internally the maximum number of global and local parameters is limited to values, which are defined in PARAMETER statements. Eventually the package has to be recompiled with modified values of the parameters. The standard parameter statement is:

```
PARAMETER (MGLOBL=1000,MLOCAL=10,NSTORE=10000,MCS=10)
MGLOBL = maximum number of global parameters
MLOCAL = maximum number of local parameters
NSTORE = maximum number of items in a partial data set
MCS    = maximum number of constraint equations
```

(the parameter NSTORE is explained on page 12). The actual dimension parameters for the application of the package have to specified once in a call, which initializes the package.

```
PARAMETER (NAGB=...)    ! number of global parameters
PARAMETER (NALC=...)    ! max. number of local  parameters

CALL INITGL(NAGB,NALC,NSTDEV,IPRLIM)

where NSTDEV = cut parameter for local fit (values 0, 1, 2 or 3)
      IPRLIM = print flag
```

It may be neccessary to exclude data with a bad χ^2 in a local fit. The parameter NSTDEV means *number of standard deviations*; a good value is 3, which means that only 0.3 % of correct data are rejected due to large deviations; for this value the χ^2-limit is 9 in the case of one degree of freedom. The cut is made actually on the quantity χ^2/n_{df}, and the limit is calculated for all values of n_{df} for the *same* probability as for one degree of freedom. The value 0 means no cut; other allowed values are 1 and 2.
By default no iterations are done. Iterations may be necessary to remove efficiently outliers in the data; in this case the option INITUN should be used, which allows to make less strict cuts during a few iterations.
The print flag IPRLIM = -1 means no printout. For zero or positive values some distributions are printed, which may be useful especially for the development. For IPRLIM > 0 the value gives the number of local fits which are printed in detail (for debugging).

3.2 The fit of local parameters

The data for a local fit (e.g. a track) are transmitted data point by data point using EQULOC; after transmitting all data for a local fit the fit is performed using FITLOC, and then the data are used later in the global fit. If the user has transmitted already data points, but decides that the data should not be used, the call KILLOC removes the data from the current local fit.
The expectation for a data point is given by a linear expression of the type of equation (21), that is

$$z = \underbrace{a_1 \cdot d_1 + a_2 \cdot d_2 + \ldots a_n \cdot d_n}_{\text{global parameters}} + \underbrace{\alpha_1 \cdot \delta_1 + \alpha_2 \cdot \delta_2 + \ldots \alpha_\nu \cdot \delta_\nu}_{\text{local parameters}} = \sum_{i=1}^{n} a_i \cdot d_i + \sum_{j=1}^{\nu} \alpha_j \cdot \delta_j \; .$$

The factors d_j and δ_j are the derivatives of the expectation value w.r.t. to the coefficients a_j and α_j. The weight of the equation is defined as $1/\sigma^2$, where σ is the standard deviation of the measured value. The derivatives and the measured value and its standard deviation have to be transmitted in the EQULOC-call by the arguments:

$$\texttt{DERGB(.)} \equiv \boldsymbol{d} \qquad \texttt{RMEAS} \equiv z$$
$$\texttt{DERLC(.)} \equiv \boldsymbol{\delta} \qquad \texttt{SIGMA} \equiv \sigma \quad .$$

Usually many of the derivatives will be zero. In subroutine EQULOC all derivatives are set to zero at return. Initially the user has to set all elements of the derivate arrays to zero; this could be done by a call to ZERLOC.
Transmitting the data points could be done in a loop like:

```
REAL DERGB(NAGB)          ! derivatives w.r.t. global variables
REAL DERLC(NALC)          ! derivatives w.r.t. local  variables

CALL ZERLOC(DERGB,DERLC)  ! set derivatives to zero
DO ...
 CALL EQULOC(DERGB,DERLC,RMEAS,SIGMA)    ! one equation
END DO

CALL FITLOC  or CALL KILLOG              ! end of equations (local)
```

If a nonzero value of NSTDEV is given in the initializing call, the ratio χ^2 to degrees of freedom n_{df} is evaluated for the local fit alone and compared with the limit; the data will be used in the determination of the global parameters only if the ratio is below the limit. This strategy can be meaningful *only* if the global parameters represent only a *small* correction to the fit, and a cut based on residuals of the local fits is meaningful. In the case of iterations (option INITUN) the same data may however be used in later iterations, depending on the actual value of χ^2.
The possibility to reject data with this cut has some consequences for the program code: all data transmitted by EQULOC-calls are stored internally in an auxiliary array. The contribution to the matrices according to equations (25) are added (in subroutine FITLOC) only if the data set is accepted. The value of the dimension parameter NSTORE in the parameter statement corresponds to the total number of measured values, weights and non-zero derivatives in all EQULOC-calls for a partial data set. A small histogram of the used space will be printed; in case of insufficient space the program has to rerun with a larger value of NSTORE.
Often the true standard deviation of the data is not known initially, but only *after* the fit procedure. In this case one has to start with a good guess on the standard deviation, and rerun the procedure with improved values afterwards. The best test of the size of the standard deviations of the measurement is made by histograms of pulls, not by histograms of residuums r, which is

$$\text{residuum } r = \text{measured value} - \text{fitted value} \ . \tag{27}$$

The pull takes into account the standard deviation σ_r of the residuum, which is given by

$$\sigma_r^2 = \sigma^2 - \sigma_f^2 \ , \tag{28}$$

where σ is the standard deviation of the measurement and σ_f is the standard deviation of the corresponding fitted value, to be calculated by error propagation. The pull is a dimensionless quantity defined as

$$\text{pull } p = \frac{r}{\sigma_r} = \frac{r}{\sqrt{\sigma^2 - \sigma_f^2}} \ . \tag{29}$$

The expected distribution of the pull p is according to $N(0,1)$ (standardized normal distribution with mean 0 and standard deviation of 1). This is especially important, when local fits depend on measured data with different precision and a few precise measured data dominate the solution. One could prefer instead of the pull a *scaled* residuum

$$\text{scaled residuum} = \left(\frac{\sigma}{\sigma_r}\right) r \; , \tag{30}$$

where the scale factor removes the effect from the fact, that the measured value was used in the fit; the quantity has the same dimension as the measured value.

3.3 The final determination of global parameters

The final fit for the determination of optimal values of the global parameters is performed by calling FITGLO.

```
CALL FITGLO(PARGB)       ! final fit for global parameters
where PARGB(NAGB) is the array of the final global parameters.
```

In this call the potentially most time-consuming part of the fit is performed, the inversion of the $n \times n$ symmetric matrix $\boldsymbol{C}'$. A special algorithm (subroutine SPMINV) is applied, which makes use of the symmetry of the matrix. An attempt is made to recognize a *singular* matrix. If during inversion no acceptable pivot element is found, the remaining parameters are returned as zero. This strategy is often successful, to eliminate highly correlated and unnecessary parameters. The strategy has another helpful side-effect: a global parameter can be eliminated from the fit by simple setting its derivative always to zero, without the necessity to change the application code otherwise.
The resulting parameters, their statistical errors and their global correlation are printed on standard output, unless there are very many parameters; in that case only few parameters are printed by default. Printing is also done on unit LUNP (ascii file) with a call

```
CALL PRTGLO(LUNP)
```

which can be inspected by a text editor. In the printout for each parameter the global correlation coefficient is shown, which is defined as the largest correlation of a parameter to all possible linear combinations of the other parameters, and can have values between zero and one, where one would mean a complete correlation to other parameters.
The errors of the final parameters may be obtained by a function call

```
ERROR = ERRPAR(I) ! error of parameter I
```

after the global fit. The correlations between different global parameters are not printed, but the correlation coefficient may be obtained by a function call[5]:

```
CORR = CORPAR(I,J) ! correlation coefficient between par I and J
```

(with a result between -1 and $+1$).

3.4 Further options of the program package

Several further options are possible within the program package. All these options require only a minimal change of the application program. Each option is selected by calls to subroutines described below, and these calls may be in any order after the initializing call CALL INITGL(...), and before the transmission of the first data for local parameters.

[5] Printing of all correlations in a 1000-parameter problem would require more than 1000 pages, each with 60 lines of 8 numbers.

Iterations

The model used for global and local parameters is linear, and therefore in principle an iterative improvement of parameters is not neccessary; the solution is found in a single step. However, the problem of outlier rejection may require iterations and repeated use of the same data. In order to allow iterations in a simple way, the program can store all data for local fits in an intermediate file, which is internally read again for each iteration.

```
CALL INITUN(LUN,CFACTR)       ! define unit for file and a factor
where LUN is the unit number for the data file
```

If a file is already opened by the user for the given unit, this file will be used for writing and reading. If no file is opened for the given unit, a scratch file will be opened. The factor (> 1) will be used to enlarge the cutvalue for χ^2/n_{df}.
There is no unique general method for outlier rejection. Bad data within the data for a local fit will usually result in a large χ^2-value, and can thus be removed by a cut on the local χ^2-value, or better: χ^2/n_{df} (n_{df} = number of degrees of freedom). But an initially bad alignment also has a large χ^2-value as a result, even for otherwise good data.
The strategy followed in the program package is the following. The global parameters are determined in iterations. The cut value `CFACTR` is used during the first iteration, and then reduced to the square root for each iteration; a value below 1.2 is replaced by 1.0 and this value is used in the last two iterations.

Initial global parameters

By default initial values of 0.0 are assumed for the global parameters. When good starting values of the global parameters `PAR` are known, they can and should be used. They are transmitted by the following call:

```
CALL PARGLO(PAR)       ! define initial values for global parameters
where PAR(NAGB)  is the array of initail global parameters
```

Good starting values of global parameters allow to make tighter initial cuts on the data.

Fixed or tough parameters

The accuracy of the determination of the global parameters has to be analysed in detail, perhaps by trying out severel modifications of the parametrization. The global correlation coefficient in the final printout can help in this analysis. A global correlation close to 1.0 indicates a large correlation to other parameters; in this case one could fix the parameter in order to increase the stability of the whole system. The I'th parameter can simply be fixed (at the initial value) by a call of `PARSIG` with `SIG=0.0`. This removes the parameter from the fit.

```
CALL PARSIG(I,SIG)       ! define value SIG for the I'th global parameter
where SIG is 0.0 or positive (see explanation below)
```

The subprogram `PARSIG` can also be called with a positive value of `SIG`. This value is then used to add a term

$$\Delta\chi^2 = \frac{(a_i - \bar{a}_i)^2}{\sigma_i^2}$$

to the overall χ^2, where σ_i =`SIG`, and $\bar{a}_i$ is the previous value of the parameter a_i. This has the effect of allowing only a small change in the parameter, corresponding to the "standard deviation" σ_i, and improves the stability of the solution. The magnitude of the value `SIG` should be of the order of the statistical error. If used in one iteration, it will bias the resulting parameter towards the previous value, but the bias will be reduced with increasing iteration number.

Constraints

Constraints among the parameter values will be essential for many applications in order to get stable and meaningful results. Assume that the global parameters represent shifts of the coordinates of certain track chambers. Without constraints the overall position of the whole detector may not be fixed by the data and it can move around in an arbitrary way. In this case one should fix the overall detector by the requirement, that the sum of all shifts is zero, mathematically

$$f(\boldsymbol{a}) = \sum_i a_i = 0 \; .$$

In general a constraint equation is a linear equation of the type

$$f(\boldsymbol{a}) = \sum_i a_i \cdot t_i - r = 0 \; .$$

with constant factors t_i and a constant r (only linear constraint equations are possible). A constraint equation is defined by:

```
      CALL CONSTF(DERCS,RHS)       ! define linear constraint function
      where DERCS(NAGB) is the array of factors and RHS is a constant.
```

Up to ten constraint equations of this type may be defined.

3.5 Example

Below an example for the application of the program package is shown. In this example a detector with ten planes is simulated by Monte Carlo methods. Straight tracks hit all the ten planes and allow to measure one coordinate z per plane, transverse to the axis of the detector. Each plane may be displaced perpendicular to the detector axis. The parametrization of the track in terms of the local parameters is

$$z = \alpha_1 + \alpha_2 \, x \tag{31}$$

The Table 1 shows the x-coordinates of the planes as well as their properties: standard deviation of the measurement, and the bias or shift Δ. 1000 tracks are generated and the simulated measurements are used determine the bias or shift Δ with different methods. The problem has in total $9 + 2 \times 1000 = 2009$ parameters, while the interest is in the first 9 parameters.
The code of the program is shown below.

```
      PROGRAM TEST
      REAL DERGB(10),DERLC(2),PAR(10)
      PARAMETER (NPLAN=10)
      REAL X(NPLAN),Y(NPLAN),SIGMA(NPLAN),BIAS(NPLAN),HEFF(NPLAN)
      REAL FROT(NPLAN)
      DATA X/5.0,9.0,20.0,25.0,30.0,35.0,40.0,45.0,50.0,55.0/
      DATA SIGMA/2*0.0020,8*0.0300/        ! standard deviation
      DATA BIAS                            ! bias of planes
     + /0.00,-0.04,0.15,0.03,-0.075,0.045,0.035,-0.08,0.09,-0.05/
      DATA HEFF/4*0.9,0.50,5*0.9/          ! hit efficiency
*     ...
      CALL INITGL(10,2,3,1)     ! define dimension parameters etc.
      CALL PARSIG(1,0.0)        ! parameter is fixed
*A    DO I=3,10                            ! alternative 2
*A     CALL PARSIG(I,0.03)                 ! alternative 2
*A    END DO                               ! alternative 2
      CALL INITUN(11,100.0)
      FROT(1)=0.0
```

```
      DO I=2,10
         FROT(I)=1.0/(X(I)-X(1))
      END DO
      CALL CONSTF(FROT,0.0)                 ! alternative 1
      CALL ZERLOC(DERGB,DERLC)  ! zero initialization
*     initialization end --- initialization end --- initialization end
*     data loop --- data loop --- data loop --- data loop --- data loop
      NCASES=1000
      DO NC=1,NCASES
       CALL GENER(Y,A,B,X,SIGMA,BIAS,HEFF) ! generate hits
       CALL SRLINE(NPLAN,X,Y,SIGMA,A,B)
       DO I=1,NPLAN
        IF(Y(I).NE.0.0) THEN
           DERLC(1)= 1.0        ! calibrate  Y(I) = A + B * X(I)
           DERLC(2)= X(I)
           DERGB(I)= 1.0
           CALL EQULOC(DERGB,DERLC,Y(I),SIGMA(I)) ! single measurement
        END IF
       END DO
       CALL FITLOC              ! local fit
      END DO !  end loop cases
*     parameter computation -------------------- parameter computation
      CALL FITGLO(PAR)          ! fit of global parameters
      END

      SUBROUTINE GENER(Y,A,B,X,SIGMA,BIAS,HEFF)  !  generate Y values
      REAL Y(*)
      PARAMETER (NPLAN=10)
      REAL X(NPLAN),SIGMA(NPLAN),BIAS(NPLAN),HEFF(NPLAN)
*     ...
*     generate straight line parameters
      A=20.0*ZRAND()-10.0        ! A =   -10 ...  +10   uniform
      B= 0.3*ZNORM()             ! B =  -0.3 ...  +0.3  gaussian
      DO I=1,NPLAN               ! planes
        IF(ZRAND().LT.HEFF(I)) THEN
           Y(I)=A+B*X(I) + BIAS(I)+SIGMA(I)*ZNORM()
        ELSE
           Y(I)=0.0              ! unmeasured
        END IF
      END DO
      END
```

i	x [cm]	σ [cm]	Δ [cm]	Δ_f	error	Δ_n	error	Δ_r
1	5	0.0020	0.0000	0.0000	0.0000	0.0000	0.0000	−0.0274
2	9	0.0020	−0.0400	−0.0399	0.0001	−0.0407	0.0012	−0.0636
3	20	0.0300	0.1500	0.1495	0.0009	0.1465	0.0047	0.1269
4	25	0.0300	0.0300	0.0293	0.0009	0.0252	0.0062	0.0099
5	30	0.0300	−0.0750	−0.0755	0.0012	−0.0806	0.0078	−0.0921
6	35	0.0300	0.0450	0.0483	0.0010	0.0423	0.0093	0.0346
7	40	0.0300	0.0350	0.0342	0.0010	0.0271	0.0108	0.0216
8	45	0.0300	−0.0800	−0.0814	0.0010	−0.0895	0.0123	−0.0954
9	50	0.0300	0.0900	0.0902	0.0011	0.0811	0.0139	0.0840
10	55	0.0300	−0.0500	−0.0494	0.0011	−0.0595	0.0154	−0.0513

Table 1: The parameters of the planes, used in the simulation, are shown on the left. On the right the results of different methods to reconstruct the bias Δ from 1000 tracks are shown. The column Δ_f is the result obtained with the Millepede package with constraints.

In the application of the MILLEPEDE package one shift parameter for each of the planes is introduced. It is easy to see, that this ansatz would not give a stable solution. Adding a constant term to all planes would not change the quality of the fit. Furthermore one could also rotate the set of all planes without a change of the quality of the fit. Thus two constraints are essential to get stable results. As first parameter the shift of the first plane is fixed to zero, by removing this parameter from the variable parameters (`PARSIG(1,0.0)`). In addition one constraint equation is added, which forces the overall rotation to be zero (`CALL CONSTF(FROT,0.0)`). The result of this fit is shown in the column Δ_f. The data in this column show, that the bias can be reconstructed with high precision, already with 1000 tracks. The deviation between the true and the reconstructed shifts are about 1 μm for the accurate plane 2, and about 10 μm for planes 3 ... 10, which have a standard deviation of 300 μm.
The other reconstructed shifts (Δ_n and Δ_r) are discussed later. Below the printout of the program is shown for the test case.

```
             *   o   o                        o
                 o   o                        o
  o ooooo    o   o   o    oo   ooo    oo     ooo   oo
   o  o  o   o   o   o   o  o  o  o  o  o  o  o  o  o
   o  o  o   o   o   o   oooo  o  o  oooo  o  o  oooo
   o  o  o   o   o   o   o     ooo   o     o  o  o
   o  o  o   o  oo  oo    oo   o      oo    ooo   oo  starting
                               o

 Number of global parameters       10
 Number of local parameters         2
    Number of standard deviations   3
    Number of test printouts        1
 Final cut corresponds to 3 standard deviations.
 The actual cuts are made in Chisquare/Ndf at:
                   Ndf =   1   Limit =    9.00
                   Ndf =   2   Limit =    5.91
                   Ndf =   5   Limit =    3.64
                   Ndf =  10   Limit =    2.69
                   Ndf =  20   Limit =    2.10
                   Ndf =  50   Limit =    1.62
                   Ndf = 100   Limit =    1.43
 Scratch file opened
 Initial cut factor is  10.
```

The printout below is test output for debugging, as reqested by the last argument of `INITGL`.

```
 ---------------------------------------------------
 Printout of local fit (FITLOC) with rank= 2
    Result of local fit: (Index/Parameter/error)
    1  -7.653      0.3630E-01     2  0.3722       0.000

 1. equation: measured value  -0.109208807 +-  0.0299999975
  number of derivates (global, local): 1, 2
   Global derivatives are: (index/derivative/parvalue)
    3   1.000      0.000
   Local derivatives are: (index/derivative)
    1   1.00000         2   20.0000
 Chi square contribution=   11.23       0.1006    = residuum

 2. equation: measured value   1.6283915 +-  0.0299999975
  number of derivates (global, local): 1, 2
   Global derivatives are: (index/derivative/parvalue)
    4   1.000      0.000
   Local derivatives are: (index/derivative)
    1   1.00000         2   25.0000
```

```
Chi square contribution=    0.57      -0.2260E-01= residuum

etc.

Entry number 1
Final chi square, degrees of freedom  48.5001851 6
Reject if Chisq/Ndf=  8.08336449 >   33.4360008
```

The two histograms below are printed after the first iteration. The first histogram shows, that the alignment is in fact bad, with a large residuum of the individual data in local fits, because no correction is done by global parameters. In many cases the residual ist outside 5 standard deviations.
The second histogram shows the use of the internal buffer, in which all data for one local fit are stored. In this test case only a very small fraction of the buffer is used.
After the global fit the histogram of χ^2/n_{df} is shown, which after calibration should peak at 1.0. In the histogram shown the χ^2/n_{df} is rather high, and many of the local fits are rejected.

```
  Initial residual histogram:  Total 8556 entries with 2807 overflows
                           X
                           X
                           X          XXXX X X
                           XX X XXX XXXXXXXXXX
                          XXXXXXXXXXXXXXXXXXXXX X    X
                        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
            XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    ----+----1----+----2----+----3----+----4----+----5
  -5.0      -3.0      -1.0       1.0       3.0       5.0

                       1111211111111111111111121111111
       12122334455766890124663746766679999798153323204
    34930454445394937044337337226981562592404433793786

Histogram of used local fit storage: total 1000 local fits with 0 overflows
    X
    X
    X
    X
    X
    X
    X
    X
    X
    X
    ----+----1----+----2----+----3----+----4----+----5
   0.0       2.0       4.0       6.0       8.0      10.0
  E 03      E 03      E 03      E 03      E 03      E 03

    1
    0
    0
    0

... making global fit ...
The rank defect of the symmetric 11-by- 11 matrix is  0 (should be zero).
```

```
  Total 1000 local fits, 631 rejected.
  Histogram of Chi**2/ndf:  Total 1000 entries with 871 overflows
                                               X
                                             X X   X
                                             X XX  X
                                     XX   X  X XX  X X
                                 X  XXXX  XX X XX XXXX
                             X   X  XXXXX XX X XX XXXX
                             XXX X XXXXXX XXXX XXXXXXX
               X   X  X    XXXXX XXXXXXXXXXXXX XXXXXXX
          X   XXX  X XX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
    ----+----1----+----2----+----3----+----4----+----5
   0.0       2.0       4.0       6.0       8.0      10.0

          1   121  2 12    224331523566542653819735856
```

The printout below shows for the next iterations the last histogram again. The χ^2 is very much reduced and all or nearly all local data can now be used.

```
Iteration  2 with cut factor=  3.16

... making global fit ...
The rank defect of the symmetric 11-by- 11 matrix is  0 (should be zero).

  Total 1000 local fits, 0 rejected.
  Histogram of Chi**2/ndf:  Total 1000 entries with 0 overflows
       X
       XX
       XXX
       XXXX
      XXXXX
      XXXXXX
     XXXXXXXX
     XXXXXXXX
     XXXXXXXXXX
     XXXXXXXXXXXX
    ----+----1----+----2----+----3----+----4----+----5
   0.0       2.0       4.0       6.0       8.0      10.0

       1111
    1795330764321
    3080616751399994221111 2

Iteration  3 with cut factor=  1.78
...
Iteration  4 with cut factor=  1.33
...
Iteration  5 with cut factor=  1.00
...
Iteration  6 with cut factor=  1.00
```

The printout below shows the result for the shifts after the last iteration. It shows the initial values of the parameters, the final fit values, the difference, the correction in the last iteration (practically zero here), the statistical error and the global correlation coefficient. the statistical error shown here is obtained by error propagation from the input data, taking into acount the constraint. For the constraint the left and the right hand side is shown; the residual deviation is very small (`0.40690E-10`).
The histograms of $\chi^2/n_{\rm df}$ and of the residuals show a final good fit. Only few local fits are excluded because of large deviations, as expected.

```
... making global fit ...
```

```
The rank defect of the symmetric 11-by- 11 matrix is  0 (should be zero).

         Result of fit for global parameters
         ===================================
   I        initial       final      differ     lastcor    Error glcor
 ---     ----------- ----------- ----------- ----------- -------- -----
   1        0.00000     0.00000     0.00000     0.00000  0.00000 0.000
   2        0.00000    -0.03993    -0.03993     0.00000  0.00010 0.376
   3        0.00000     0.14951     0.14951     0.00000  0.00091 0.500
   4        0.00000     0.02928     0.02928     0.00000  0.00092 0.476
   5        0.00000    -0.07550    -0.07550     0.00000  0.00124 0.485
   6        0.00000     0.04834     0.04834     0.00000  0.00097 0.411
   7        0.00000     0.03421     0.03421     0.00000  0.00100 0.373
   8        0.00000    -0.08137    -0.08137     0.00000  0.00104 0.337
   9        0.00000     0.09019     0.09019     0.00000  0.00109 0.305
  10        0.00000    -0.04936    -0.04936     0.00000  0.00113 0.280

Constraint 1     Sum - RHS = 0.40690E-10 -  0.0000     =  0.40690E-10

  Total 1000 local fits, 2 rejected.
  Histogram of Chi**2/ndf:  Total 1000 entries with 0 overflows
       XX
      XXXX
      XXXX
     XXXXX
     XXXXXX
     XXXXXX
     XXXXXXXX
     XXXXXXXXX
    XXXXXXXXXXX
    ----+----1----+----2----+----3----+----4----+----5
   0.0       2.0       4.0       6.0       8.0      10.0

     11111
    203553856321
    453929162903274211

  Residual histogram:  Total 8556 entries with 0 overflows
                            XX
                           XXX
                           XXXX
                          XXXXXX
                         XXXXXXXX
                        XXXXXXXXXX
                       XXXXXXXXXXXX
                     XXXXXXXXXXXXXXX
                    XXXXXXXXXXXXXXXXXX
    ----+----1----+----2----+----3----+----4----+----5
  -5.0      -3.0      -1.0       1.0       3.0       5.0

                    112345678975543211
                126638630415911855517275211
             366020617400080029011267732930721  1


             *   o   o                        o
                 o   o                        o
  o ooooo    o   o   o    oo   ooo    oo    ooo   oo
   o  o  o   o   o   o   o  o  o  o  o  o  o  o  o  o
   o  o  o   o   o   o   oooo  o  o  oooo  o  o  oooo
   o  o  o   o   o   o   o     ooo   o     o  o  o
   o  o  o   o   oo  oo   oo   o      oo    ooo   oo   ending.
                               o
```

For comparison another result is shown in the Table 1: the column Δ_n is obtained by small modifications of the main program (shown in rows with comments alternative 1 and 2), with the constraint of the overall rotation removed. As explained above, the fit can not be stable without the constraint, and the parameter list of the final result would indicate this problem. The matrix inversion would detect a rank defect, and one shift would be returned (wrongly) as zero. The matrix is singular without the constraint. In the alternative code above stability is established again by marking the shifts of planes 3 ... 10 as tough by `PARSIG(I,0.03)`. The matrix in made non-singular by this change and the results are meaningful, although the precision is much lower: the standard deviations are a factor of about 10 higher.
The quantity Δ_r in the final column of Table 1 shows the result of the determination of the shift by residuals; this is perhaps a method used very often in practice. Here the local data are fitted without introducing the global shift parameters; the residuals for each plane is added to a histogram and the shifts are determined from the peak in each of the histograms. The deviations to the true shifts are up to a factor 100 higher compared to the first method. It should ne noted, that a repeated application of this method would make the results even worse, since the procedure can not be stable, due to missing constraints.

3.6 Nonlinear models and constraints

A method for *linear* least squares fits with a large number of parameters, perhaps with *linear* constraints, is discussed in this paper. Sometime of course the model is *nonlinear* and also constraints may be nonlinear. The standard method to treat these problems is linearization: the nonlinear equation is replaced by a linear equation for the correction of a parameter (Taylor expansion); this requires a good approximate value of the parameter. In principle this method requires an iterative improvement of the parameters, but sometimes even one iteration may be sufficient.
The treatment of nonlinear equations is not directly supported by the program package, but it will in general not be too difficult to organize a program application with nonlinear equations.

Addendum 17.10. 2000

Faster Execution. The program has been applied to problems with a large number of global parameters (about 5000). A modification of the matrix multiplication in equation (25), making use of the sparse character of the matrices involved, reduced the execution time (thanks to Th.Oest for discussion and tests).

Parallel execution. An extension to allow the use of the program under online conditions has been made. Under online conditions the program is executed in parallel on many processor, where each processor can see only a small fraction of the data. The aim is to combine all the data finally to obtain a single accurate solution. The proposed solution works only in the non-iterative mode, i.e. the intermediate file can not be used (no call of `INITUN`).
The strategy proposed for this application is as follows. On the many processors working in parallel on fractions of the data, the final call to fit the global parameters has to be removed; instead a call

```
CALL FITMUT(NN,VEC)        ! get vector
where VEC(VEC) is an array of matrix and vector elements
```

has to be added, where a data vector of `NN = (NAGB*NAGB+NAGB)/2+NAGB` elements is extracted. This vector contains the accumulated information of the data in compressed form.
Now the sum of all vectors from all processors has to be formed. This may be done by creating a histogram with `NN` words on each of the many processors and storing the content of the vector in the histogram bins; combining the same histogram from all the processors means in modern histogramming packages like `LOOK` *adding* all the histogram contents.

The final fit is then done on one processor using the sum vector, which has to be inserted in the program by

```
CALL FITMIN(NN,VEC)       ! put vector
where VEC(VEC) is the sum vector.
```

before the final global fit call `CALL FITGLO(PAR)`. The data loop is replaced by this code.
The two versions of the program, the first one on the parallel processors and the second one for the final computation of the global parameters are shown below for the case of the example, shown earlier.

```
      PROGRAM ...                        ... on many parallel processors
      REAL VEC(...)                        ! declare vector
      REAL DERGB(10),DERLC(2),PAR(10)
      ...
      CALL CONSTF(FROT,0.0)                 ! alternative 1
      CALL ZERLOC(DERGB,DERLC)  ! zero initialization
*     initialization end --- initialization end --- initialization end
*     data loop --- data loop --- data loop --- data loop --- data loop
      DO
       ...
       CALL FITLOC              ! local fit
      END DO !  end data loop
*     extract data vector
      CALL FITMUT(NN,VEC)
      ... write vector VEC(NN) to file or store in histogram ...
      END

      PROGRAM ...                        ... on the final processor
      REAL VEC(...)                        ! declare vector
      REAL DERGB(10),DERLC(2),PAR(10)
      ...
      CALL CONSTF(FROT,0.0)
      CALL ZERLOC(DERGB,DERLC)  ! zero initialization
*     initialization end --- initialization end --- initialization end
*     read and insert data vector
      ... read vector VEC(NN) ...
      CALL FITMIN(NN,VEC)
*     parameter computation -------------------- parameter computation
      CALL FITGLO(PAR)          ! fit of global parameters
      END
```