DRAFT

Millepede II

Linear Least Squares Fits with a Large Number of Parameters

Volker Blobel

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 based on track fits is one of the problems, where the interest is only in optimal values of the global parameters, the alignment parameters. The method, called Millepede, to solve the linear least squares problem with a simultaneous fit of all global and local parameters, irrespectively of the number of local parameters, is described.

The paper describes the *second* version of the general program package Millepede for the efficient solution, even for a number of global parameters above several ten thousands, in a reasonable time on a standard PC. Different solution methods can be selected.

Institut für Experimentalphysik
Universität Hamburg
2007

Pdf-file prepared 25th May 2007

Linear Least Squares Fits with a Large Number of Parameters

Contents

Preface **5**

I Mathematical Methods **7**

1 Large problems with global and local parameters **7**

2 Optimization without constraints **8**
2.1 The quadratic model . . . 8
2.2 Partitioning of matrices . . . 8
2.3 Local parameters . . . 9
2.4 Global parameters . . . 10
2.5 The simultaneous fit of global and local parameters . . . 10
2.6 Reduction of matrix size . . . 11
2.7 Nonlinear least squares . . . 11
2.8 Outliers . . . 12

3 Optimization with linear constraints **13**
3.1 The Lagrange multiplier method . . . 13
3.2 Feasible parameters . . . 14

II The Manual **15**

4 The programm package Millepede II **15**
4.1 Program code and makefile . . . 15
4.2 Data collection in the user program with subroutine Mille . . . 16
4.3 Solution with the stand-alone program Pede . . . 16

5 Measurements and parameters **17**
5.1 Measurements and local parameters . . . 17
5.2 Global parameters . . . 18
5.3 Writing data files with Mille . . . 19
5.4 Non-linearities . . . 20
5.5 Application: Alignment of tracking detectors . . . 20

6 Text files **21**
6.1 General data format of text files . . . 21
6.2 File information . . . 22
6.3 Parameter information . . . 22
6.4 Constraint information . . . 23
6.5 Global parameter measurements . . . 23
6.6 Solution method selection . . . 23
6.7 Solution methods . . . 25
6.8 Outlier treatment and debugging . . . 26
6.9 Further options . . . 27

7 Computation of the global fit **28**
7.1 Memory managament . . . 28

7.2 Initialization 28
7.3 First data loop 28
7.4 Second data loop 29
7.5 Solution method overview 29
7.6 Data file loops during iterations 29
7.7 Global fit 32
7.8 Output text files 34

8 Using MILLEPEDE II for track based alignment **34**
8.1 Global parameters 34
8.2 Constraints 34
8.3 Linear transformations in 3D 36

Preface

Linear least squares problems. The most important general method for the determination of parameters from measured data is the linear least squares method. It is usually stable 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.

Global and local parameters. The parameters of a least squares problem can sometimes be distinguished as *global* and *local* parameters. The mathematical model underlying the measurement depends on both types of parameters. The interested is in the determination of the global parameters, which appear in all the measurements. There are many sets of local parameters, where each set appears only in one, sometimes small, subset of measurements.

Alignment of large detectors in particle physics. 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 1 000, but sometimes above 10 000. 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. Parameters of a single track like track slopes and curvatures are called *local* parameters.

One approximate alignment method is to perform least squares fits on the data e.g. of single tracks assuming fixed alignment parameters. The *residuals*, the *deviations* between the fitted and measured data, are then used to estimate 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, they ignore 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. It practice this procedure is often applied iteratively; it is however not clear wether the procedure is converging.

A more efficient and faster method is an overall least squares fit, with all the global parameters and local parameters, perhaps from thousands or millions of events, determined simultaneously. The Millepede algorithm makes use of the special structure of the least squares matrices in such a simultaneous fit: the global parameters can be determined from a matrix equation, where the matrix dimension is given by the number of global parameters only, irrespective of the total number of local parameters, and without any approximation. If n is the number of global parameters, then an equation with a symmetric n-by-n matrix has to be solved. The Millepede programm has been used for up to $n \approx 5000$ parameters in the past. Solution of the matrix equation was done with a fast program for the solution of matrix equations with inversion of the symmetric matrix, which on a standard PC takes a time $t \approx 2 \times 10^{-8}\,\mathrm{sec} \times n^3$. Even for $n = 5000$ the computing time is, with $t = 2 \times 10^{-8}\,\mathrm{sec} \times 5000^3 = 40$ minutes, still acceptable.

The next-generation detectors in particle physics however require up to about 100 000 global parameters and with the previous solution method the space- and time-consumption is too large for a standard PC. Memory space is essential needed for the full symmetric matrix, corresponding to $1/2n^2 \times 8$ bytes for double precision, if the symmetry of the matrix is observed, and if solution is done *in-space*. This means memory space of 400 Mbyte and 40 Gbyte for $n = 10\,000$ and $n = 100\,000$, and computing times of about 6 hours and almost a year, respectively.

Millepede I and II. The second version of the Millepede programm, described here, allows to use several different methods for the solution of the matrix equation for global parameters, while keeping the algorithm, which decouples the determination of the local parameters, i.e. still taking into account all correlations between all global parameters. The matrix can be stored as a sparse matrix, if a large fraction of off-diagonal elements has zero content. A complete matrix inversion of a large

sparse matrix would result in a full matrix, and would take an unacceptable time. Different methods can be used in MILLEPEDE II for a sparse matrix. The symmetric matrix can be accumulated in a special sparse storage scheme. Special solution methods can be used, which require only products of the symmetric sparse matrix with different vectors. Depending on the fraction of vanishing matrix elements, solutions for up to a number of 100 000 global parameters should be possible on a standard PC.

The structure of MILLEPEDE II is different from the MILLEPEDE I version. The accumulation of data and the solution is splitted, with the solution in a stand-alone program. Furthermore the global parameters are now characterized by a label (any positive integer) instead of a (continuous) index. Those features should simplify the application of the program for alignment problems even for large number of global parameters in the range of 10^5.

Part I
Mathematical Methods

1 Large problems with global and local parameters

The solution of optimization problems requires the determination of certain parameters. In certain optimization problems the parameters can be classified as either *global* or *local*. This classification can be used, when the input data of the optimization problem consist out of a potentially large group of data sets, where the description of each single set of data requires certain *local* parameters, which appear only in the description of one data set. The *global* parameter may appear in all data sets, and the interest is in the determination of these global parameters. The local parameters can be called nuisance parameters.
An example for a optimization problem with global and local parameters from experimental high energy physics is the alignment of track detectors using a large number of track data. The track data are position measurements of a charged particle track, which can be fitted for example by a helix with five parameters, if the track detector is in a homogeneous magnetic field. The position measurement, from several detector planes of the track detector, depend on the position and orientation of certain sensors, and the corresponding coordinates and angles are global parameters. In an alignment the values of the global parameters are improved by minimizing the deviations between measurement and parametrization of a large number of tracks. Numbers of tracks in the order of one Million, with of the order of 10 data poinst per track, and of 1000 to $100,000$ global parameters are typical for modern track detectors in high energy physics.
Optimal values of the global parameters require a simultaneous fit of all parameters with all data sets. A straight-forward ansatz for such a fit seems to require to fit Millions of parameters, which is impossible. An alternative is to perform each local fit separately, fitting the local parameters only. From the residuals of the local fits one can then try to fit the values of the global parameters. This approach neglects the correlations between the global and local parameters. Nevertheless the method can be applied iteratively with the hope, that the convergence is not to slow and does not require to many iterations.
The optimization problem is complicated by the fact, that often not all global parameters are defined by the ansatz. In the alignment example the degrees of freedom describing a translation and rotation of the whole detector are not fixed. The solution of the optimization problem requires therefore certain equality constraints[1], for example zero overall translation and rotation of the detector; these constraints are described by a linear combination of global parameters. Such an equality constraint can only be satisfied in an overall fit, not with separated local and global fits.
Global parameters are denoted by the vector $\boldsymbol{p}$ and local parameters for the data set j by the vector $\boldsymbol{q}_j$. The objective function to be minimized in the global parameter optimization is the sum of the local objective Function, depending on the global parameters $\boldsymbol{p}$ and the vectors $\boldsymbol{q}_j$:

$$F(\boldsymbol{p},\boldsymbol{q}) = \sum_j F_j(\boldsymbol{p},\boldsymbol{q}_j)$$

In the least squares method the objective function[2] to be minimized is a sum of the squares of residuals z_i between the measured values and the parametrization, weighted by the inverse variance of the measured value:

$$F_j(\boldsymbol{p},\boldsymbol{q}_j) = \tfrac{1}{2}\sum_i \frac{z_i^2}{\sigma_i^2}$$

[1] Constraints in optimization are either equality or inequality constraints, which have exactly to be taken into account in the solution. The term *constraint* is often used in a loose sense, like: "the parameters are constrained by our measurement".

[2] In experimental high energy physics the objective function is usually called a χ^2-function. In statistics there is a χ^2-distribution with a well-defined meaning. The minimum value $\times 2$ of the objective function follows, under certain conditions, the χ^2-distribution.

with the residual $z_i = y_i - f_i(\boldsymbol{p}, \boldsymbol{q}_j)$, where y_i is the measured value and $f_i(\boldsymbol{p}, \boldsymbol{q}_j)$ is the corresponding parametrization. This form of the objective function assumes independent measurements, with a single variance value σ_i assigned.

2 Optimization without constraints

2.1 The quadratic model

The standard optimization method for smooth objective functions is the NEWTON method. The objective function $F(\boldsymbol{p})$, depending on a parameter vector $\boldsymbol{p}$, is approximated by a quadratic model $\tilde{F}(\boldsymbol{p})$

$$\tilde{F}_k\left(\boldsymbol{p}_k + \boldsymbol{d}\right) = F_k + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} + \tfrac{1}{2}\boldsymbol{d}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{d} \tag{1}$$

where $\boldsymbol{p}_k$ is the vector $\boldsymbol{p}$ in the k-th iteration and where the vector $\boldsymbol{g}$ is the gradient of the objective function; the matrix $\boldsymbol{C}$ is the Hessian (second derivative matrix) of the objective function $F(\boldsymbol{p})$ or an approximation of the Hessian. The minimum of the quadratic approximation requires the gradient to be equal to zero. A step $\boldsymbol{d}$ in parameter space is calculated by the solution of the matrix equation, obtained from the derivative of the quadratic model:

$$\boldsymbol{C}\,\boldsymbol{d} = -\boldsymbol{g}\,. \tag{2}$$

With the correction vector $\boldsymbol{d}$ the new value $\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{d}$ for the next iteration is obtained. The matrix $\boldsymbol{C}$ is a constant in a *linear least squares* problem and the minimum is determined in a single step (no iteration necessary).

2.2 Partitioning of matrices

The special structure of the matrix $\boldsymbol{C}$ in a matrix equation $\boldsymbol{C}\boldsymbol{d} = -\boldsymbol{g}$ may allow a significant simplification of the solution. Below the symmetric matrix $\boldsymbol{C}$ is partitioned into submatrices, and the vectors $\boldsymbol{d}$ and $\boldsymbol{g}$ are partitioned into two subvectors; then the matrix equation can be written in the form

$$\left(\begin{array}{c|c} \boldsymbol{C}_{11} & \boldsymbol{C}_{21}^{\mathrm{T}} \\ \hline \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{array}\right) \left(\begin{array}{c} \boldsymbol{d_1} \\ \hline \boldsymbol{d_2} \end{array}\right) = - \left(\begin{array}{c} \boldsymbol{g_1} \\ \hline \boldsymbol{g_2} \end{array}\right)\,, \tag{3}$$

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 $p+q = n$, and $\boldsymbol{C}_{21}$ is a q-by-p matrix. Now it is assumed that the inverse of the q-by-q sub-matrix $\boldsymbol{C}_{22}$ is available. In certain problems this may be easily calculated, for example if $\boldsymbol{C}_{22}$ is diagonal.

If the sub-vector $\boldsymbol{d}_1$ would not exist, the solution for the sub-vector $\boldsymbol{d}_2$ would be defined by the matrix equation $\boldsymbol{C}_{22}\,\boldsymbol{d}_2^* = -\boldsymbol{g}_2$, where the star indicates the special character of this solution, which is

$$\boldsymbol{d_2}^* = -\boldsymbol{C}_{22}^{-1}\,\boldsymbol{g_2}\,. \tag{4}$$

Now, having the inverse sub-matrix $\boldsymbol{C}_{22}^{-1}$, the submatrix of the complete inverse matrix $\boldsymbol{C}$ correponding to the upper left part $\boldsymbol{C}_{11}$ is the inverse of the symmetric p-by-p matrix

$$\boldsymbol{S} = \boldsymbol{C}_{11} - \boldsymbol{C}_{21}^{\mathrm{T}}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{21}\,, \tag{5}$$

the so-called Schur complement. With this matrix $\boldsymbol{S}$ the solution of the whole matrix equation can be written in the form

$$\left(\begin{array}{c} \boldsymbol{d_1} \\ \hline \boldsymbol{d_2} \end{array}\right) = - \left(\begin{array}{c|c} \boldsymbol{S}^{-1} & -\boldsymbol{S}^{-1}\boldsymbol{C}_{21}^{\mathrm{T}}\boldsymbol{C}_{22}^{-1} \\ \hline -\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{21}\boldsymbol{S}^{-1} & \boldsymbol{C}_{22}^{-1} - \boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{21}\boldsymbol{S}^{-1}\boldsymbol{C}_{21}^{\mathrm{T}}\boldsymbol{C}_{22}^{-1} \end{array}\right) \left(\begin{array}{c} \boldsymbol{g_1} \\ \hline \boldsymbol{g_2} \end{array}\right)\,. \tag{6}$$

The sub-vector $\boldsymbol{d_1}$ can be obtained from the solution of the matrix equation

$$\left(\quad \boldsymbol{C}_{11} - \boldsymbol{C}_{21}^{\mathrm{T}} \boldsymbol{C}_{22}^{-1} \boldsymbol{C}_{21} \quad \right) \left(\quad \boldsymbol{d_1} \quad \right) = - \left(\quad \boldsymbol{g_1} - \boldsymbol{C}_{21}^{\mathrm{T}} \boldsymbol{C}_{22}^{-1} \boldsymbol{g}_2 \quad \right) = - \left(\quad \boldsymbol{g_1} - \boldsymbol{C}_{21}^{\mathrm{T}} \boldsymbol{d_2}^* \quad \right) \tag{7}$$

using the known right-hand-side of the equation with the special solution $\boldsymbol{d_2}^*$.
In a similar way the vector $\boldsymbol{d_2}$ could be calculated. However, if the interest is the determination of this sub-vector $\boldsymbol{d_1}$ only, while the sub-vector $\boldsymbol{d_2}$ is not needed, then only the equation (7) has to be solved after calculation of the special solution $\boldsymbol{d_2}^*$ (equation (4)) and the Schur complement $\boldsymbol{S}$ (equation (5)). 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[3]. The method can be applied repeatedly, and therefore may simplify the solution of problems with a large number of parameters. The method is not an approximation, but is exact and it takes into account all the correlations introduced by the removed parameters.

2.3 Local parameters

A set of local measured data y_i is considered. The local data y_i are assumed to be described by a linear or non-linear function $f(x_i, \boldsymbol{q})$, depending on a (small) number of local parameters $\boldsymbol{q}$.

$$y_i = f(x_i, \boldsymbol{q}) + \varepsilon_i \ . \tag{8}$$

The parameters $\boldsymbol{q}$ are called the local parameters, valid for the specific group of measurements (local-fit object). The quantity ε is the measurement error, with standard deviation σ_i. The quantity x_i is assumed to be the coordinate of the measured value y_i, and is one argument of the function $f(x_i, \boldsymbol{q})$. The local parameters are determined in a least squares fit. If the function $f(x_i, \boldsymbol{q})$ depends *non-linearly* on the local parameters $\boldsymbol{q}$, an iterative procedure is used, where the function is linearized, i.e. the *first* derivatives of the function $f(x_i, \boldsymbol{q})$ with respect to the local parameters $\boldsymbol{q}$ are calculated. The function is thus expressed as a linear function of local parameter corrections $\boldsymbol{\Delta q}$ at some reference value $\boldsymbol{q}_k$:

$$f(x_i, \boldsymbol{q}_k + \boldsymbol{\Delta q}) = f(x_i, \boldsymbol{q}_k) + \frac{\partial f}{\partial q_1} \Delta q_1 + \frac{\partial f}{\partial q_2} \Delta q_2 + \ldots \ , \tag{9}$$

where the derivatives are calculated for $\boldsymbol{q} \equiv \boldsymbol{q}_k$. For each single measured value, the residual measurement z_i

$$z_i \equiv y_i - f(x_i, \boldsymbol{q}_k)$$

is calculated. For each iteration a linear system of equations (normal equations of least squares) has to be solved for the parameter corrections $\boldsymbol{\Delta q}$ with a matrix $\boldsymbol{\Gamma}$ and a gradient vector $\boldsymbol{g}$ with elements

$$\Gamma_{jk} = \sum_i \left(\frac{\partial f_i}{\partial q_j} \right) \left(\frac{\partial f_i}{\partial q_k} \right) \frac{1}{\sigma_i^2} \qquad\qquad \beta_j = \sum_i \left(\frac{\partial f_i}{\partial q_j} \right) \frac{z_i}{\sigma_i^2} \ , \tag{10}$$

where the sum is over all measurements y_i of the local-fit object. Corrections $\boldsymbol{\Delta q}$ are determined by the solution of the matrix equation

$$\boldsymbol{\Gamma \Delta q} = -\boldsymbol{\beta} \ ,$$

and a new reference value is obtained by

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \boldsymbol{\Delta q}$$

and then, with k increased by 1, this is repeated until convergence is reached.

[3]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, Paragraph III: 429-433. J.B.Metzler, Stuttgart.

2.4 Global parameters

Now global parameters are considered, which contribute to all the measurements. The expectation function (9) is extended to include corrections for global parameters. Usually only few of the global parameters influence a local-fit object. A global parameter is identified by a label ℓ; assuming that labels ℓ from a set Ω contribute to a single measurement the extended equation becomes

$$z_i = y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p}) = \sum_{j=1}^{\nu} \left(\frac{\partial f}{\partial q_j} \right) \Delta q_j + \sum_{\ell \in \Omega} \left(\frac{\partial f}{\partial p_\ell} \right) \Delta p_\ell \, . \tag{11}$$

2.5 The simultaneous fit of global and local parameters

In the following it is assumed that there is a set of N local measurements. Each local measurement, with index j, depends on ν local parameters $\boldsymbol{q}_j$, 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 a system of n equations, for the global parameters only.

For a set of N local measurements one obtains a system of least squares normal equations with large dimensions, as is shown in equation (12) The matrix on the left side of equation (12) has, from each local measurement, three types of contributions. The first part is a contribution of a symmetric matrix $\boldsymbol{C}_{\mathbf{1}j}$, of dimension n (number of global parameters), and is calculated from the (global) derivatives $\partial f / \partial p_\ell$. All the matrices $\boldsymbol{C}_{\mathbf{1}j}$ are added up in the upper left corner of the big matrix of the normal equations. The second contribution is the symmetric matrix $\boldsymbol{\Gamma}_j$ (compare equation (10)), which gives a contribution to the big matrix on the diagonal and is depending only on the j-th local measurement and the (local) derivatives $\partial f / \partial q_j$. The third (mixed) contribution is a rectangular matrix $\boldsymbol{G}_j$, 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 (gradient), $\boldsymbol{g}_{\mathbf{1}j}$ for the global and $\boldsymbol{\beta}_j$ for the local parameters. The complete matrix equation is given by

$$\left(\begin{array}{c||c|c} \sum \boldsymbol{C}_{\mathbf{1}j} & \cdots & \boldsymbol{G}_j \quad \cdots \\ \hline\hline \vdots & \ddots & 0 \quad 0 \\ \hline \boldsymbol{G}_j^{\mathrm{T}} & 0 & \boldsymbol{\Gamma}_j \quad 0 \\ \hline \vdots & 0 & 0 \quad \ddots \end{array}\right) \cdot \left(\begin{array}{c} \boldsymbol{d} \\ \hline\hline \vdots \\ \hline \boldsymbol{\Delta q}_j \\ \hline \vdots \end{array}\right) = - \left(\begin{array}{c} \sum \boldsymbol{g}_{\mathbf{1}j} \\ \hline\hline \vdots \\ \hline \boldsymbol{\beta}_j \\ \hline \vdots \end{array}\right) \tag{12}$$

In this matrix equation the matrices $\boldsymbol{C}_{\mathbf{1}j}$, $\boldsymbol{\Gamma}_j$, $\boldsymbol{G}_j$ and the vectors $\boldsymbol{g}_{\mathbf{1}j}$ and $\boldsymbol{\beta}_j$ contain contributions from the j-th local measurement. Ignoring the global parameters (i.e. keeping them constant) one could solve the normal equations $\boldsymbol{\Gamma}_j \boldsymbol{\Delta q}_j^* = -\boldsymbol{\beta}_j$ for each local measurement separately by

$$\boldsymbol{\Delta q}_j^* = -\boldsymbol{\Gamma}_j^{-1} \boldsymbol{\beta}_j \, . \tag{13}$$

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 $\boldsymbol{G}_j$ und $\boldsymbol{C}_{\mathbf{1}j}$,

2.6 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 (12) 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 (12) 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

$$\begin{pmatrix} \\ \boldsymbol{C} \\ \\ \end{pmatrix} \begin{pmatrix} \\ \boldsymbol{d} \\ \\ \end{pmatrix} = - \begin{pmatrix} \\ \boldsymbol{g} \\ \\ \end{pmatrix} , \tag{14}$$

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

$$\boldsymbol{C} = \sum_j \boldsymbol{C_{1}}_j + \sum_j \boldsymbol{C_{2}}_j \qquad\qquad \boldsymbol{g} = \sum_j \boldsymbol{g_{1}}_j + \sum_j \boldsymbol{g_{2}}_j \tag{15}$$

with the following local contributions to $\boldsymbol{C}$ and $\boldsymbol{g}$ from the j-th local fit:

$$\boldsymbol{C_{2}}_j = -\boldsymbol{G}_j \boldsymbol{\Gamma}_j^{-1} \boldsymbol{G}_j^{\mathrm{T}} \qquad\qquad \boldsymbol{g_{2}}_j = -\boldsymbol{G}_j \left(\boldsymbol{\Gamma}_j^{-1} \boldsymbol{\beta}_j \right) = -\boldsymbol{G}_j \boldsymbol{\Delta q}_j^* \, . \tag{16}$$

The set of normal equations (14) 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 (16) represents the solution for the local parameters, ignoring the global parameters. The solution

$$\boldsymbol{d} = -\boldsymbol{C}^{-1} \boldsymbol{g} \tag{17}$$

represents the solution vector $\boldsymbol{d}$ with covariance matrix $\boldsymbol{C}^{-1}$. The solution is direct, no iterations or approximations are required. The dimension of the matrix to compute $\boldsymbol{d}$ from equation (2) is reduced from $(n + N \cdot \nu)$ to n. The vector $\boldsymbol{d}$ is the correction for the global parameter vector $\boldsymbol{p}$. 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, or which have to be down-weighted;
- the accuracy of the data is not known before, and has to be determined from the data (after the alignment).

For iterations the vector $\boldsymbol{d}$ is the correction for the global parameter vector $\boldsymbol{p}_k$ in iteration k to obtain the global parameter vector $\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{d}$ for the next iteration.

2.7 Nonlinear least squares

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.

2.8 Outliers

Cases with very large residuals within the data can distort the result of the method of least squares. In the method of M-estimates, the least squares method is modified to Maximum likelihood method. Basis is the residual between data and function value (expected data value), normalized by the standard deviation:

$$\zeta = \frac{y - f(x)}{\sigma} \tag{18}$$

In the method of M-estimates the objective function $F(.)$ to be minimized is defined in terms of a probability density function of the normalized residual

$$F(.) = \sum_i \rho(\zeta_i) \qquad \rho(\zeta) = \ln \mathrm{pdf}(\zeta) \tag{19}$$

From the probability density function $\mathrm{pdf}(\zeta)$ a *influence function* $\psi(\zeta)$ is defined

$$\text{influence function } \psi(\zeta) = \mathrm{d}\rho(\zeta)/\,\mathrm{d}\zeta \qquad \text{additional weight factor } \omega(\zeta) = \psi(\zeta)/\zeta \tag{20}$$

and a weight in addition to the normal least squares weight $w_i = 1/\sigma_i^2$ can be derived from the influence function for the calculation of the (modified) normal equations of least squares.

For the standard least squares method the function $\rho(\zeta)$ is simply $\rho(\zeta) = \zeta^2/2$ and it follows, that the influence function is $\psi(\zeta) = \zeta$ and the weight factor is $\omega(\zeta) = 1$ (i.e. no extra weight). The influence function value increases with the value of the normalized residual without limits, and thus outliers have a large and unlimited influence. In order to reduce the influence of outliers, the probability density function $\mathrm{pdf}(\zeta)$ has to be modified for large values of $|\zeta|$ to avoid the unlimited increase of the influence. Several functions $\mathrm{pdf}(\zeta)$ are proposed, for example the Huber function and the Cauchy function.

Huber function: A simple function is the Huber function, which is quadratic and thus identical to least squares for small $|\zeta|$, but linear for larger $|\zeta|$, where the influence function becomes a constant $C_\mathrm{H} \cdot \mathrm{sign}(\zeta)$:

$$\text{Huber function: pdf} \quad \rho(\zeta) = \begin{cases} \zeta^2/2 \\ C_\mathrm{H}\,(|\zeta| - C_\mathrm{H}/2) \end{cases} \qquad \text{factor} \quad \omega(\zeta) = \begin{cases} 1 & \text{if } |\zeta| \le C_\mathrm{H} \\ C_\mathrm{H}/|\zeta| & \text{if } |\zeta| > C_\mathrm{H} \end{cases} \tag{21}$$

The extra weight is $\omega(\zeta) = C_\mathrm{H}/|\zeta|$ for large $|\zeta|$. A standard value for C_H is $C_\mathrm{H} = 1.345$; for this value the efficiency for Gaussian data without outliers is still 95 %. For very large deviations the additional weight factor decreases with $1/|\zeta|$.

Cauchy function: For small deviation the Cauchy function is close to the least squares expression, but for large deviations it increases only logarithmically. For very large deviations the additional weight factor decreases with $1/\zeta^2$:

$$\text{Cauchy function: pdf} \quad \rho(\zeta) = \tfrac{1}{2} C_\mathrm{c} \ln\left(1 + (\zeta/C_\mathrm{c})^2\right) \qquad \text{factor} \quad \omega = 1/\left(1 + (\zeta/C_\mathrm{c})^2\right) \tag{22}$$

A standard value is $C_\mathrm{c} = 2.3849$; for this value the efficiency for Gaussian data without outliers is still 95 %.

3 Optimization with linear constraints

The minimization of an objective function $F(\boldsymbol{p})$ is often not sufficient. Several degrees of freedom may be undefined and require additional conditions, which can be expressed as equality constraints. For m linear equality constraints the problem is the minimization of a non-linear function $F(\boldsymbol{p})$ subject to a set of linear constraints:

$$\min F(\boldsymbol{p}) \qquad \text{subject to } \boldsymbol{A}\boldsymbol{p} = \boldsymbol{c}\,, \tag{23}$$

where $\boldsymbol{A}$ is a m-by-n matrix and $\boldsymbol{c}$ is a m-vector with $m \leq n$. In iterative methods the parameter vector $\boldsymbol{p}$ is expressed by $\boldsymbol{p} = \boldsymbol{p}_k + \boldsymbol{d}$ with the correction $\boldsymbol{d}$ to $\boldsymbol{p}_k$ in the k-th iteration, satisfying the equation

$$\boldsymbol{A}\,(\boldsymbol{p}_k + \boldsymbol{d}) = \boldsymbol{c}$$

with $\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{d}$.

There are two methods for linear constrainst:

- in the *Lagrange multiplier method* additional m parameters are introduced and the linear system of $n + m$ unknowns has to be solved;
- by *elimination* the minimization problem with constraints is transformed to an unconstrained problem with $n - m$ unknowns.

The Lagrange method is used in Millepede.

3.1 The Lagrange multiplier method

In the Lagrange multiplier method one additional parameter λ is introduced for each single constraint, resulting in an m-vector $\boldsymbol{\lambda}$ of Lagrange multiplier. A term depending on $\boldsymbol{\lambda}$ and the constraints is added to the function $F(\boldsymbol{p})$, resulting in the Lagrange function

$$\mathcal{L}(\boldsymbol{p}, \boldsymbol{\lambda}) = F(\boldsymbol{p}) + \boldsymbol{\lambda}\,(\boldsymbol{A}\boldsymbol{p} - \boldsymbol{c}) \tag{24}$$

Using as before a quadratic model for the function $F(\boldsymbol{p})$ and taking derivatives w.r.t. the parameters $\boldsymbol{p}$ and the Lagrange multipliers $\boldsymbol{\lambda}$, the two equations

$$\begin{aligned} \boldsymbol{C}\boldsymbol{d} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} &= -\boldsymbol{g} \\ \boldsymbol{A}\boldsymbol{d} &= \boldsymbol{c} - \boldsymbol{A}\boldsymbol{p}_k \end{aligned}$$

are obtained; the second of these equations is the constraint equation. This system of two equations can be combined into one matrix equation

$$\left(\begin{array}{c|c} \boldsymbol{C} & \boldsymbol{A}^{\mathrm{T}} \\ \hline \boldsymbol{A} & \boldsymbol{0} \end{array}\right) \left(\begin{array}{c} \boldsymbol{d} \\ \hline \lambda \end{array}\right) = \left(\begin{array}{c} -\boldsymbol{g} \\ \hline \boldsymbol{c} - \boldsymbol{A}\boldsymbol{p}_k \end{array}\right)\,. \tag{25}$$

The matrix on the left hand side is still symmetric. *Linear* least squares problems with *linear* constraints can be solved directly, without iterations and without the need for initial values of the parameters.

The matrix in equation (25) is indefinite, with positive and negative eigenvalues. A solution can be found even if the submatrix $\boldsymbol{S}$ is singular, if the matrix $\boldsymbol{A}$ of the constraints supplies sufficient information. Because of the different signs of the eigenvalues the stationary solution is not a minimum of the function $\mathcal{L}(\boldsymbol{p}, \boldsymbol{\lambda})$.

3.2 Feasible parameters

Feasible parameters. A particular value for the correction $\boldsymbol{d}$ can be calculated by

$$\boldsymbol{d} = \boldsymbol{A}^{\mathrm{T}} \left(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} \right)^{-1} (\boldsymbol{c} - \boldsymbol{A}\boldsymbol{p}_k) \ , \tag{26}$$

which is the *minimum-norm solution* of the constraint equation, that is, the solution of

$$\min \ \| \boldsymbol{A} \, (\boldsymbol{p}_k + \Delta \boldsymbol{p}) - \boldsymbol{c} \|_2 \ ,$$

which is zero here. The matrix $\boldsymbol{A}$ is a m-by-n matrix for m constraints and the product $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ is a square m-by-m matrix, which has to be inverted in equation (26), which allows to obtain a correction such that the linear constraints are satisfied. Parameter vectors $\boldsymbol{p}$, satisfying the linear constraint equations $\boldsymbol{A}\boldsymbol{p} = \boldsymbol{c}$, are called *feasible*. If the vector $\boldsymbol{p}_k$ in the k-th iteration already satisfies the linear constraint equations, then the correction $\boldsymbol{d}$ has to have the property $\boldsymbol{A}\,\boldsymbol{d} = \boldsymbol{0}$.

Part II
The Manual

4 The programm package MILLEPEDE II

The second version of the program package with the name MILLEPEDE (german: TAUSENDFÜSSLER) is based on the same mathematical principle as the first version; global parameters are determined in a simultaneous fit of global and local parameters. In the first version the solution of the matrix equation for the global parameter corrections was done by matrix inversion. This method is adequate with respect to memory space and execution time for a number of global parameters of the order of 1000. In actual applications e.g. for the alignment of track detectors at the LHC storage ring at CERN however the number of global parameters is much larger and may be above 10^5. Different solution methods are available in the MILLEPEDE II version, which should allow the solution of problems with a large number of global parameters with the memory space, available in standard PCs, and with an execution time of hours. The solution methods differ in the requirements of memory space and execution time.

The structure of the second version MILLEPEDE II can be visualized as the *decay* of the single program MILLEPEDE into two parts, a part MILLE and a part PEDE (Figure 1):

$$\text{MILLEPEDE} \Rightarrow \text{MILLE} + \text{PEDE}\,.$$

The first part, MILLE, is a short subroutine, which is called in user programs to write data files for MILLEPEDE II. The second part, PEDE, is a stand-alone program, which requires data files and text files for the steering of the solution. The result is written to text files.

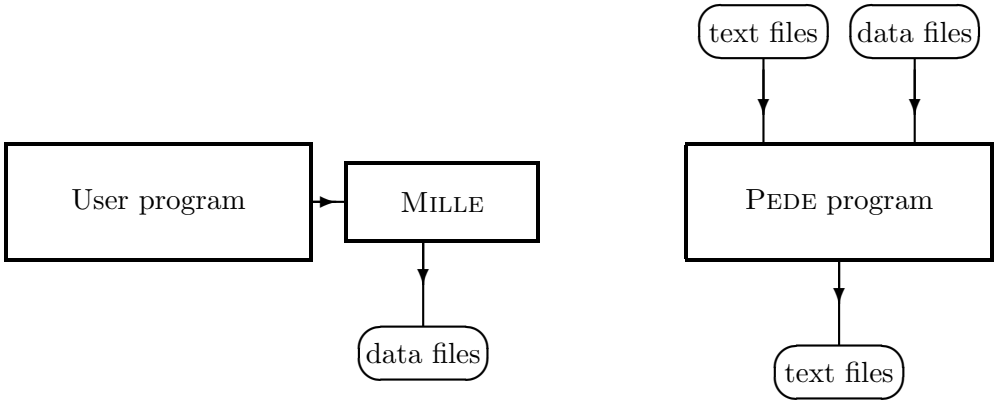

Figure 1: The subprogram MILLE (left), called inside the user program, and the stand-alone program PEDE (right), with the data flow from text and data files to text files.

4.1 Program code and makefile

The complete program is on a (`tar`)-file `Mptwo.tgz`, which is expanded by the command

```
tar -xzf Mptwo.tgz
```

into the actual directory. A `Makefile` for the program PEDE is included; it is invoked by the

```
make
```

command. There is a test mode of PEDE (see section 4.3), selected by

```
./pede -t
```

which can be used to test the program installation.

The computation of the solution for a large number of parameter requires large vector and matrix arrays. Most of the memory space is required in nearly all solution methods for a single matrix. The solution of the corresponding matrix equation is done *in-space* for almost all methods (no second matrix or matrix copy is needed). The total space of all data arrays, used in PEDE, is defined as a dimension parameter within the include file `dynal.inc` by the statement

```
PARAMETER (MEGA=100 000 000) !  100 mio words
```

corresponding to the memory space allowed by a 512 Mbyte memory. Larger memories allow an increase of the dimension parameter in this statement in file `dynal.inc`.
Memory space above 2 Gbyte (?) can not be used with 32-bit systems, but on 64-bit systems; a small change in the makefile to allow linking with a big static array (see file `dynal.inc`) may be necessary (see comment in makefile).

4.2 Data collection in the user program with subroutine MILLE

Data files are written within the user program by the subroutine `MILLE`, which is available in Fortran and in C. Data on the measurement and on derivatives with respect to local and global parameters are written to a binary file. The file or several files are the input to the stand-alone program PEDE, which performs the fits and determines the global parameters. The data required for MILLEPEDE and the data collection in the user program are discussed in detail in section 5.

4.3 Solution with the stand-alone program PEDE

The second part, PEDE, is a *stand-alone program*, which performs the fits and determines the global parameters. It is written in Fortran. Input to PEDE are the binary (unformatted) data files, written using subroutine MILLE, and text (formatted) files, which supply steering information and, optionally, data about initial values and status of global parameters. Different binary and text files can be combined.

Synopsis:

```
pede [options] [main steering text file name]
```

The following options are implemented:

-i	interactive mode
-t	test mode
-s	subito option: stop after one data loop

Option i. The interactive mode allows certain interactive steering, depending on the selected method.

Option t. In the test mode no user input files are required. This mode is recommended to learn about the properties of MILLEPEDE. Data files are generated by Monte Carlo simulation for a simple 200-parameter problem, which is subsequently solved. The file generated are: `mp2str.txt` (steering file), `mp2con.txt` (constraints) and `mp2tst.bin` (datafile). The latter file is always (re-)created, the two text files are created, if they do not exit. Thus one can edit these files after a first test job in order to test different methods.

Option s. In the *subito* mode, the PEDE program stops after the first data loop, irrespective of the options selected in the steering text files.

Text files. Text files should have either the characters `xt` or `tx` in the 3-character filename-extension. At least one text file is necessary (the default name `steer.txt` is assumed, if no filename is given in the command line), which specifies at least the names of data files. The text files are described in detail in section 6.

PEDE generates, depending on the selected method, several output text files:

```
millepede.log !  log-file for pede execution
```

```
mpgparm.txt !  global parameter results
mpdebug.txt !  debug output for selected events
mpeigen.txt !  selected eigenvectors
```

Existing files of the given names are renamed with a $\sim$ extension, existing files with the $\sim$ extension are removed.

5 Measurements and parameters

5.1 Measurements and local parameters

Basic data elements are single measurements y_i; several single measurements belong to a group of measurements, which can be called a local-fit object. For example in a track-based alignment based on tracks a track is a local-fit object (see section 5.5 for a detailed discussion on the data for tracks in a track-based alignment). A local-fit object is described by a linear or non-linear function $f(x_i, \boldsymbol{q}, \boldsymbol{p})$, depending on a (small) number of local parameters $\boldsymbol{q}$ and in addition on global parameters $\boldsymbol{p}$:

$$y_i = f(x_i, \boldsymbol{q}, \boldsymbol{p}) + \varepsilon_i \ . \tag{27}$$

The parameters $\boldsymbol{q}$ are called the local parameters, valid for the specific group of measurements (local-fit object). The quantity ε is the measurement error, expected to have (for ideal parameter values) mean zero (i.e. the measurement is unbiased) and standard deviation σ_i; often ε will follow at least approximately the normal distribution $N(0, \sigma_i)$. The quantity x_i is assumed to be the coordinate of the measured value y_i, and is one of the arguments of the function $f(x_i, \boldsymbol{q}, \boldsymbol{p})$.
The global parameters needed to compute the expectation $f(x_i, \boldsymbol{q}, \boldsymbol{p})$ are usually already quite accurate and only small corrections habe to be determined. The convention is often to define the vector $\boldsymbol{p}$ to be a correction with initial values zero. In this case the final values of the global parameters will be small.

The fit of a local-fit object. The local parameters are usually determined in a least squares fit within the users code, assuming fixed global parameter values $\boldsymbol{p}$. If the function $f(x_i, \boldsymbol{q}, \boldsymbol{p})$ depends *non-linearly* on the local parameters $\boldsymbol{q}$, an iterative procedure is used, where the function is linearized, i.e. the *first* derivatives of the function $f(x_i, \boldsymbol{q}, \boldsymbol{p})$ with respect to the local parameters $\boldsymbol{q}$ are calculated. Then the function is expressed as a linear function of local parameter corrections $\boldsymbol{\Delta q}$ at some reference value $\boldsymbol{q}_k$:

$$f(x_i, \boldsymbol{q}_k + \boldsymbol{\Delta q}, \boldsymbol{p}) = f(x_i, \boldsymbol{q}_k, \boldsymbol{p}) + \frac{\partial f}{\partial q_1}\Delta q_1 + \frac{\partial f}{\partial q_2}\Delta q_2 + \dots \ ,$$

where the derivatives are calculated for $\boldsymbol{q} \equiv \boldsymbol{q}_k$. The corrections are determined by the linear least squares method, a new reference value is obtained by

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \boldsymbol{\Delta q}$$

and then, with k increased by 1, this is repeated until convergence is reached, i.e. until the corrections become essentially zero. For each iteration a linear system of equations (normal equations of least squares) has to be solved for the parameter corrections $\boldsymbol{\Delta q}$ with a right-hand side vector $\boldsymbol{b}$ with components

$$b_j = \sum_i \left(\frac{\partial f_i}{\partial q_j}\right) \left(y_i - f(x_i, \boldsymbol{q}_k, \boldsymbol{p}_k)\right) \frac{1}{\sigma_i^2} \ ,$$

where the sum is over all measurements y_i of the local-fit object. All components b_j become essential zero after convergence.
After convergence the equation (27) can be expressed with the fitted parameters $\boldsymbol{q}$ in the form

$$y_i = f(x_i, \boldsymbol{q}, \boldsymbol{p}) + \frac{\partial f}{\partial q_1}\Delta q_1 + \frac{\partial f}{\partial q_2}\Delta q_2 + \ldots + \varepsilon_i \ .$$

The difference $z_i = y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p})$ is called the residual and the equation can be expressed in terms of the residual measurement z_i as

$$z_i \equiv y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p}) = \left(\frac{\partial f}{\partial q_1}\right) \Delta q_1 + \left(\frac{\partial f}{\partial q_2}\right) \Delta q_2 + \ldots + \varepsilon_i \; . \tag{28}$$

After convergence of the local fit all corrections Δq_1, $\Delta q_2, \ldots$ are zero, and the residuals z_i are small and can be called the *measurement error*. Aa mentioned above the model function $f(x_i, \boldsymbol{q}, \boldsymbol{p}$ may be a linear or a non-linear function of the local parameters. The fit procedure in the users code may be a more advanced method than least squares. In track fits often Kalman filter algorithms are applied to treat effects like multiple scattering into account. But for any fit procedure the final result will be *small* residuals z_i, and it should also be possible to define the derivatives $\partial f/\partial q_j$ and $\partial f/\partial p_\ell$; these derivatives should express the *change* of the residual z_i, if the local parameter q_j or the global parameter p_ℓ is changed by Δq_j or Δp_ℓ. In a track fit with strong multiple scattering the derivatives will become rather small with increasing track length, because the information content is reduced due to the multiple scattering.

Local fits are later (in PEDE) repeated in a simplified form; these fits need as information the difference z_i and the derivatives, and correspond to the *last iteration* of the local fit, because only small parameter changes ares involved. Modified values of global parameters $\boldsymbol{p}$ during the global fit result in a change of the value of $f(x_i, \boldsymbol{q}, \boldsymbol{p})$ and therefore of z_i, which can be calculated from the derivatives. The derivatives with respect to the local parameters allow to repeat the local fit and to calculate corrections for the local parameters $\boldsymbol{q}$.

5.2 Global parameters

Now the global parameters $\boldsymbol{p}$ are considered. The calculation of the residuals $z_i = y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p})$ depends on the global parameters $\boldsymbol{p}$ (valid for *all* local-fit objects). This dependence can be considered in two ways. As expressed above, the *parametrization* $f(x_i, \boldsymbol{q}, \boldsymbol{p})$ depends on the global parameters $\boldsymbol{p}$ and on corrections $\boldsymbol{\Delta p}$ of the global parameters (this case is assumed below in the sign of the derivatives with respect to the global parameters). Technically equivalent is the case, where the *measured* value y_i is calculated from a raw measured value, using global parameter values; in this case the residual could be written in the form $z_i = y_i(\boldsymbol{p}) - f(x_i, \boldsymbol{q})$. It is assumed that reasonable values are already assigned to the global parameters $\boldsymbol{p}$ and the task is to find (small) corrections $\boldsymbol{\Delta p}$ to these initial values.
Equation (28) is extended to include corrections for global parameters. Usually only few of the global parameters influence a local-fit object. A global parameter carries a label ℓ, which is used to identify the global parameter uniquely, and is an arbitrary positive integer. Assuming that labels ℓ from a set Ω contribute to a single measurement the extended equation becomes

$$z_i = y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p}) = \sum_{j=1}^{\nu} \left(\frac{\partial f}{\partial q_j}\right) \Delta q_j + \sum_{\ell \in \Omega} \left(\frac{\partial f}{\partial p_\ell}\right) \Delta p_\ell \; . \tag{29}$$

MILLEPEDE essentially performs a fit to all single measurements for all groups of measurements simultaneously for all sets of local parameters and for all global parameters. The result of this simultaneous fit are optimal corrections $\boldsymbol{\Delta p}$ for all global parameters; there are no limitations in the number of local parameters sets. Within this global fit the local fits (or better: the last iteration of the local fit) has to be repeated for each local-fit object.

Global parameter labels. The label ℓ, carried by a global parameter, is an arbitrary positive (31-bit) integer (most-significant bit is zero), and is used to identify the global parameter uniquely. Arbitrary gaps between the numerical values of labels values are allowed. It is recommended to design the label in a way which allows to reconstruct the meaning of the global parameter from the numerical value of the label ℓ.

5.3 Writing data files with MILLE

The user has to provide the following data for each single measurement:

n_{lc} = number of local parameters array : $\left(\frac{\partial f}{\partial q_j}\right)$

n_{gl} = number of global parameters array : $\left(\frac{\partial f}{\partial p_\ell}\right)$; label-array ℓ

z = residual $(\equiv y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p}))$ σ = standard deviation of the measurement

These data are sufficient to compute the least squares normal equations for the local and global fits. Using calls of `MILLE` the measured data and the derivatives with respect to the local and global parameters are specified; they are collected in a buffer and written as one record when one local-fit object is finished in the user reconstruction code.

Calls for Fortran version: Data files should be written with Fortran on the system used for PEDE; otherwise the internal file format could be different.

```
CALL MILLE(NLC,DERLC,NGL,DERGL,LABEL,RMEAS,SIGMA)
```

where

`NLC` = number of local parameters `DERLC`
`DERLC` = array `DERLC(NLC)` of derivatives
`NGL` = number of global derivatives in this call
`DERGL` = array `DERLG(NGL)` of derivatives
`LABEL` = array `LABEL(NGL)` of labels
`RMEAS` = measured value
`SIGMA` = error of measured value (standard deviation)

After transmitting the data for all measured points the record containing the data for one local-fit object is written. following call

```
CALL ENDLE
```

The buffer content is written to a file with the record.

Alternatively the collected data for the local-fit object have to be discarded, if some reason for *not using* this local-fit object is found.

```
CALL KILLE
```

The content of the buffer is reset, i.e. the data from preceeding `MILLE` *calls are removed.*

Additional floating-point and integer data (special data) can be added to a local fit object by the call

```
CALL MILLSP(NSPEC,FSPEC,ISPEC)
```

where

`NSPEC` = number of special data `DERLC`
`FSPEC` = array `FSPEC(NSPEC)` of floating point data
`ISPEC` = array `ISPEC(NSPEC)` of integer data

The floating-point and integer arrays have the same length. These special data are not yet used, but may be used in future options.

Calls for C version:

The C++-class `Mille` can be used to write C-binary files. The constructor

```
Mille(const char *outFileName, bool asBinary = true, bool writeZero =
false);
```

takes at least one argument, defining the name of the output file. For debugging purposes it is possible to give two further arguments: If `asBinary` is false, a text file (not readable by pede) is written instead of a binary input. If `writeZero` is true, derivatives that are zero are not suppressed in the output as usual.

The member functions

```
void mille(int NLC, const float *derLc, int NGL, const float *derGl,
           const int *label, float rMeas, float sigma);
void end();
void kill();
```

have to be called equivalently like the Fortran subroutines `MILLE`, `ENDLE` and `KILLE`. To properly close the output file, the `Mille` object should be deleted (or go out of scope) after the last processed record.

Root version: A `root` version is perhaps added in the future.

5.4 Non-linearities

Experience has shown that non-linearities in the function $f(x_i, \boldsymbol{q}, \boldsymbol{p})$ are usually small or at least not-dominating. As mentioned before, the convention is often to define the start values of the global parameters as zero and to expect only *small* final values. Derivatives with respect to the local and global parameters can be assumed to be constants within the range of corrections in the determination of the global parameter corrections. Then a single pass through the data, writing data files and calculating global parameter corrections with MILLEPEDE is sufficient. If however corrections are large, they may require to re-calculate the derivatives and another pass or several passes through the data with re-writing data files may be necessary. As a check of the whole procedure a second pass is recommended.

5.5 Application: Alignment of tracking detectors

In the alignment of tracking detectors the group of measurement, the local-fit object, may be the set of hits belonging to a single track. Two to five parameters $\boldsymbol{q}$ may be needed to describe the dependence of the measured values on the coordinate x_i.

In a real detector the trajectory of a charged particle does not follow a simple parametrization because of effects like multiple scattering due to the detector material. In practice often a Kalman filter method is used without an explicit parametrization; instead the calculation proceeds along the track taking into account e.g. multiple scattering effects. Derivatives with respect to the (local) track parameters are not directly available. Nevertheless derivatives as required in equation (29) are still well-defined and can be calculated. Assuming that the (local) track parameters refer to the starting point of the track, for each point along the track the derivative $(\partial f/\partial q_j)$ has to be equal to $\Delta z_i/\Delta q_j$, if the local parameter q_j if changed by Δq_j, and the corresponding change of the residual z is Δz_i. This derivative could be calculated numerically.
For low-momentum tracks the multiple-scattering effects are large. Thus the derivative above will tend to small values for measured points with a lot of material between the measured point and the starting point of the track. This shows that the contribution of low-momentum tracks with large multiple-scattering effects to the precision of an alignment is small.

6 Text files

In text files the steering information for the program execution and further information on parameters and constraints is given. The main text file is mandatory. Its default name is `steer.txt`; if the name is different it has to be given in the command line of PEDE. In the main text file the file names of data files and optionally of further text files are given. Text files should have an extension which contains the character pairs `xt` or `tx`. In this section all options for the global fits and their keywords are explained. The computation of the global fit is described in detail in the next section 7; it may be necessary to read this section 7 to get an understanding of the various options.

6.1 General data format of text files

Text files contain file names, numerical data, keywords (text) and comment in free format, but there are certain rules.

File names for all text and steering file should be the leading information in the main steering file, with one file name per line, with correct characters, since the file names are used to open the files.

Numerical data can be given with or without decimal point, optionally with exponent field. The numerical data

```
13234     13234.0     13.234E+3
```

are all identical.

Comments can be given in every line, preceeded by the ! sign; the text after the ! is ignored. Lines with * or ! in the first columm are considered as comment lines. Blank lines are ignored.

Keywords are necessary for certain data; they can be given in upper or lower case characters. Keywords with few typo errors may also be recognized correctly. The program will stop if a keyword is not recognized.

Below is an example for a steering file:

```
Fortranfiles
!/home/albert/filealign/lhcrun1.11    ! data from first test run
 /home/albert/filealign/lhcrun2.11    ! data from second run
 /home/albert/filealign/cosmics.bin   ! cosmics
 /home/albert/detalign/mydetector.txt ! steering file from previous log file
 /home/albert/detalign/myconstr.txt   ! test constraints

constraint 0.14       ! numerical value of r
713 1.0               ! pair of parameter label and numerical factor
719 0.5  720 0.5      ! two pairs

CONSTRAINTS 1.2
 112 1.0
 113 -1.0
 114 0.5
 116 -0.5

Parameter
201 0.0   -1.0
202 1.0   -1.0
204 1.23 0.020

method inversion 5 0.1
end
```

6.2 File information

Names of data and text files are given in single text lines. The file-name extension is used to distinguish text files (extension containing `tx` or `xt`) and binary data files. Data files may be Fortran or C files; by default C files are assumed. Fortran files have to be preceded by the textline

```
Fortranfiles
```

and C files may be preceded by the textline

```
Cfiles
```

Mixing of C- and Fortran-files is allowed.

6.3 Parameter information

By default all global parameters appearing in data files with their labels are assumed to be variable parameters with initial value zero, and used in the fit. Initial values different from zero and the so-called presigma (see below) may be assigned to global parameters, identified by the parameter label. The optional parameter information has to be provided in the form below, starting with a line with the keyword `Parameter`:

```
Parameter
label     initial_value     presigma
...
label     initial_value     presigma
```

(italics are numerical values). The three numbers *label* *initial_value* *presigma* in a textline may be followed by further numbers, which are ignored[4]. All parameters not given in this file, but present in the data files are assumed to be variable, with initial value of zero.

Pre-sigma. The pre-sigma s_ℓ defines the status of the global parameter:

$s_\ell > 0$ The parameter is *variable* with the given initial value. A term $1/s_\ell^2$ is added to the diagonal matrix element of the global parameter to stabilize a perhaps poorly defined parameter. This addition should *not* bias the fitted parameter value, if a sufficient numner of iterations is performed; it may however bias the calculated error of the parameter (this is available only for the matrix inversion method).

$s_\ell = 0$ The pre-sigma is zero; the parameter is *variable* with the given initial value.

$s_\ell < 0$ The pre-sigma is negative. The parameter is defined as **fixed**; the initial value of the parameter is used in the fits.

The lines below show examples, with the numerical value of the label, the initial value, and the pre-sigma:

```
11 0.01232  0.0    ! parameter variable, non-zero initial value
12 0.0      0.0    ! parameter variable, initial value zero
20 0.00232  0.0300 ! parameter variable with initial value 0.00232
30 0.00111  -1.0   ! parameter fixed, but non-zero parameter value
```

Result file. At the end of the Pede program a result file `mpgparm.txt` with the result is written. In the first three columns it carries the label, the fitted parameter value and the pre-sigma (default is zero). This file can, perhaps after editing, be used for a repeated Pede job.

[4]Note that the result file has the same format and may be used as input file, if a program execution should be continued.

6.4 Constraint information

Equality constraints allow to fix certain undefined or weakly defined linear combinations of global parameters by the addition of equations of the form

$$c = \sum_{\ell \in \Omega} f_\ell \cdot p_\ell$$

The constraint value c is usually zero. The format is:

`Constraint` *value*
label *factor*
...
label *factor*

where *value*, *label* and *factor* are numerical values. Note that no error of the value is given. The equality constraints are, within the numerical accuracy, exactly observed in a global fit. Each constraint adds another Lagrange multiplier to the problem, and needs the corresponding memory space in the matrix.

Instead of the keyword `Constraint` the keyword `Wconstraint` can be given (`this option is not yet implemented!`). In this case the factor for each global parameter given in the text file is, in addition, is multiplied by the weight W_ℓ of the corresponding global parameter in the complete fit:

$$c = \sum_{\ell \in \Omega} f_\ell \cdot W_\ell \cdot p_\ell$$

Thus global parameters with a large weight in the fit get also a large weight in the constraint. Mathematically the effect of a constraint does not change, if the constraint equation is multiplied by an arbitrary number. Because of round-off errors however it is recommended to have the constraint equation on a similar accuracy level, perhaps by a scale factor for constraint equations.

6.5 Global parameter measurements

Measurements with measurement value and measurement error for linear combinations of global parameters in the form

$$y = \sum_{\ell \in \Omega} f_\ell \cdot p_\ell + \epsilon$$

can be given, where ϵ is the measurement error, assumed to follow a distribution $N(0, \sigma^2)$, i.e. zero bias and standard deviation σ. The format is:

`Measurement` *value* *sigma*
label *factor*
...
label *factor*

where *value*, *sigma*, *label* and *factor* are numerical values. Each measurement *value* $\pm$ *sigma* (standard deviation) given in the text files contributes to the overall objective function in the global fit. At present no outlier tests are made.

The global parameter measurement and the constraint information from the previous section differ, in the definition, only in the additional information on the standard deviation given for the measurement. They differ however in the mathematical method: constraints add another parameter (Lagrange multiplier); measurements contribute to the parameter part of the matrix, and may require no extra space; however for a sparse matrix may become less *sparse*, if matrix elements are added, which are empty otherwise.

6.6 Solution method selection

Methods. One out of several solution methods can be selected. The methods have different execution time and memory space requirement, and may also have a slighly different accuracy. The statement

to select a method are:

`method inversion` *number1 number2*
`method diagonalization` *number1 number2*
`method fullGMRES` *number1 number2*
`method sparseGMRES` *number1 number2*
`method cholesky` *number1 number2*
`method bandcholesky` *number1 number2*
`method HIP` *number1 number2*

The two numbers are:

number1 = number of iterations,
number2 = limit for ΔF (convergence recognition).

For preconditioning in the GMRES methods a band matrix is used. The width of the variable-band matrix is defined by

`bandwidth` *number*

where *number* is the numerical value of the semi-bandwidth of a band matrix. The method called `bandcholesky` uses only the band matrix (reduced memory space requirement), and needs more iterations. The different methods are described below.

Iterations and convergence. The mathematical method in principle allows to solve the problem in a single step. However due to potential inaccuracies in the solution of the large linear system and due to a required outlier treatment certain internal iterations may be necessary. Thus the methods work in iterations, and each iteration requires one or several loops with evaluation of the objective function and the first-derivative vector of the objective function (gradient), which requires reading all data and local fits of the data. The first loop is called iteration 0, and (only) in this loop in addition the second-derivative matrix is evaluated, which takes more cpu time. This first loop usually brings a large reduction of the value of the objective function, and all following loops will bring small additional improvements. In all following loops the same second-derivative matrix is used (in order to reduce cpu time), which should a good approximation.
A single loop may be sufficient, if there are no outlier problems. Using the command line option $-\mathbf{s}$ (*subito*) the program executes only a single loop, irrepective of the options required in the text files
The treatment of outliers turns a linear problem into a non-linear problem and this requires iterations. Also precision problems, with rounding errors in the case of a large number of global parameters, may require iterations. Each iteration 1, 2, ... is a line search along the calculated search direction, using the *strong Wolfe conditions*, which force a *sufficient* decrease of the function value and the gradient. Often an iteration requires only one loop. If, in the outlier treatment, a cut is changed, or if the precision of the constraints is insufficient, one additional loop may be necessary.
At present the iterations end, when the number specified with the method is reached. For each loop, the expected objective-function decrease and the actual decrease are compared. If both values are below the limit for ΔF specified with the method, the program will end earlier.

Memory space requirements. The most important consumer of space is the symmetric matrix of the normal equations. This matrix has a parameter part, which requires for example for full storage $n(n+1)/2$ words, and for sparse storage $n + qn(n-1)/2$ words, for n = number of global parameters and q = fraction of non-zero off-diagonal elements. In addition the matrix has a constraint part, corresponding to the Lagrange multipliers. Formulae to calculate the number of (double precision) elements of the matrix are given in table 1.
The GMRES method can be used with a full or with a sparse matrix. Without preconditioning the GMRES method may be slow or may even fail to get an acceptable solution. Preconditioning is recommended and usually speeds up the GMRES method considerably. The band matrix required for preconditioning also takes memory space. The parameter part requires (only) $n \cdot m$ words, where m = semibandwidth specified with the method. A small value like $m = 6$ is usually sufficient. The constraint part of the band matrix however requires up to $n \cdot n_C + n_C(n_C + 1)$ words, and this can be a large number for large number n_C of constraint equations.
Another matrix with $n_C(n_C + 1)/2$ words is required for the constraints, and this matrix is used to

method	memory space requirement
inversion, fullGMRES, cholesky	$(n^2+n)/2 + nn_C + (n_C^2+n_C)/2$
diagonalization	$n + n(n-1)/2 + n^2$
sparseGMRES	$n + qn(n-1)/2n_{cf}$
bandcholesky, preconditioning	$nm + nn_C + (n_C^2+n_C)/2$
HIP (no constraints)	nm

Table 1: Matrix space requirements for the different methods, with: q = fraction of non-zero off-diagonal elements, m = bandwidth of variable band matrix, n_C = number of constraints, n_{cf} = number of constraint-factors.

improve the precision of the constraints. Several vectors of length n and n_C are used in addition, but this space is only proportional to n and n_c.
The diagonalization method requires more space: in addition to the symmetric matrix another matrix with $(n+n_C)^2$ words is used for the eigenvectors; since diagonalization is also slower, the method can only be used for smaller problems.

Comparison of the methods. The methods have a different computing-time dependence on the number of parameters n. The methods `diagonalization`, `inversion` and `cholesky` have a computing time proportional to the third power of n, and can therefore be used only up to $n = 1\,000$ or perhaps up to $n = 5\,000$. Especially the diagonalization is slow, but provides a detailed information on weakly- and well-defined linear combinations of global parameters.
The GMRES methods has a weaker dependence on n, and allows much larger n-values; larger values of n of course require a lot of space and the sparse matrix mode should be used. The bandcholesky method can be used for preconditioning the GMRES method (recommended), but also as a stand-alone method; its cpu time depends only linearly on n, but the matrix is only an approximation and usually many iterations are required. For all methods the constraints increase the time and space consumption; this remains modest if the number of constraints n_C is modest, for example $n_C \leq 100$, but may be large for n_C = several thousand.

6.7 Solution methods

Inversion. The computing time for inversion is roughly $2 \times 10^{-8} \cdot n^3$ seconds (on a standard PC). For $n \approx 1\,000$ the inversion time of ≈ 20 seconds is still acceptable, but for $n \approx 10\,000$ the inversion time would be already more than 5 hours. The advantage of the inversion is the availability of *all* parameter errors and global correlations.

Cholesky decomposition. The Cholesky decomposition (under test) is an alternative to inversion; eventually this method is faster and/or numerically more stable than inversion. At present parameter errors and global correlations are not available.

Diagonalization. The computing time for diagonalization is roughly a factor 10 larger than for inversion. Space for the square (non-symmetric) transformation matrix of eigenvectors is necessary. Parameter errors and global correlations are calculated for all parameters. The main advantage of the diagonalization method is the availability of the eigenvalues. Small positive eigenvalues of the matrix correspond to linear combinations of parameters which are only weakly defined and it will become possible to interactively remove the weakly defined linear combinations. This removal could be an alternative to certain constraints. Constraints are also possible and they should correspond to *negative* eigenvalues.

Generalized minimization of residuals (GMRES). The GMRES method is a fast iterative solution method for full and sparse matrices. Especially for large dimensions with $n \gg 1\,000$ it should be much faster than inversion, but of similar accuracy. Although the solution is fast, the building of the sparse matrix may require a larger cpu time. At present there is a limit for the number of internal

GMRES-iterations of 2000. For a matrix with a bad condition number this number of 2000 internal GMRES-iterations will often be reached and this is often also an indication of a bad and probably inaccurate solution. An improvement is possible by preconditioning, selected if GMRES is selected together with a bandwidth parameter for a band matrix; an approximate solution is determined by solving the matrix equation with a band matrix within the GMRES method, which improves the eigenvalue spectrum of the matrix and often speeds up the method considerably. Experience shows that a small bandwidth of e.g. 6 is sufficient. In interactive mode parameter errors for selected parameters can be determined.

Variable band matrix. Systems of equations with a band matrix can be solved by the Cholesky decomposition. Often the matrix element around the diagonal are the essential matrix elements and in these cases the matrix can be approximated by a band matrix of small width, with a small memory requirement. The computing time is also small; it is linear with the number of parameters. However because the band matrix is only the approximate matrix often many iterations are necessary to get a good solution. The definition of the band width is necessary. The band width is variable and thus the constraints equations can be treated completely. This means that the constraints are observed even in this approximate solution.

6.8 Outlier treatment and debugging

Cases with very large residuals within the data can distort the result of the least squares fit. Reason for outliers may be selection mistakes or statistical fluctuations. A good outlier treatment may be neccessary to get accurate results. The problem of outlier treatment is complicated by the fact that initially the global parameters may be far from optimal and therefore large deviation may occur even for correct data before the global parameter determination. There are options for the complete removal of bad cases and for the down-weighting of bad data. Cases with a **huge χ^2 are automatically removed** in every iteration. The options to remove large χ^2 cases and down-weighting are *not* done in the first iteration. The options are:

`chisqcut`	*number1 number2*
`outlierdownweighting`	*number*
`dwfractioncut`	*number*
`printrecord`	*number1 number2*

Chisquare cut. With the keyword `chisqcut` two numbers can be given. Basis of χ^2 rejection is the χ^2-value corresponding to 3 standard deviations (and to a probability of 0.27%). For one degree of freedom this χ^2-value is 9.0, and for 10 degrees of freedom the χ^2-value is 26.9. The first number given with the keyword `chisqcut` is a factor for the χ^2-cut value, to be used in the first iteration, and the second number is the factor for the second iteration. In subsequent iterations the factor is reduced by the square root, with values below 1.5 replaced by 1. For example the statement

```
chisqcut 5.0 2.5
```

means: the cut factor is $5 \times \chi^2$-value corresponding to three standard deviations, $2.5 \times \chi^2$-value for the second iteration, $1.58 \times \chi^2$-value for the third iteration and $1 \times \chi^2$-value for subsequent iterations. Thus in the first iteration the cut is $5 \times 26.9 = 134.5$ for 10 degrees of freedom, and 26.9 after the third iteration.

Outlier downweighting. Outlier down-weighting for single data values requires repeated local fits with > 1 iterations in the local fit. The number given with the keyword `outlierdownweighting` is the number of iterations. In down-weighting the weight of the data point, which by default is $1/\sigma^2$, is reduced by an extra factor $\omega_i < 1$ according to the residual (see below). For n data points the total sum $S_f = \sum_i \omega_i$ of the extra factors is $\leq n$. The ratio $(n - S_f)/n$ is called the down-weight fraction, and a histogram of this ratio is accumulated in the second function evaluation. The first iteration of the local fit is done *without* down-weighting, iterations 2 and 3 use the M-estimation method with the Huber function. Subsequent iterations, if requested, use the M-estimation method with the Cauchy function for down-weighting, where the influence of very large deviations is further reduced. For example the statement

```
outlierdownweighting 4
```
means: iteration 1 of the local fit without down-weighting, iterations 2 and 3 with Huber function down-weighting and iteration 4 with Cauchy function down-weighting.

Downweighting fraction cut. Cases with a very large fraction of down-weighted measurements are very likely wrong data and should be rejected. The cut value should be determined from the histogram showing the down-weight fraction. Typical values are 0.1 for weights from the Huber function (`outlierdownweighting` ≤ 3) and 0.2 for weights from the Cauchy function (`outlierdownweighting` > 3). For example the statement
```
dwfractioncut 0.2
```
means to reject all cases with a down-weight fraction ≥ 0.2.

Record printout. For debugging many details of single records and the local fits are printed. Two record number *number1 number2* can be given with the keyword `printrecord`; printout is done in iteration 1 and 3. For numbers given as negative, the records with extreme deviations are selected in iteration 2, and printed in iteration 3. If the first number is given as -1, the record with the largest single residual is selected. If the second number is given as -1, the record with the largest value of χ^2/N_{df} is selected. For example the statement
```
printrecord 4321 -1
```
means: the record 4321 is printed in iterations 1 and 3, and the record with the largest value of χ^2/N_{df} is selected in iteration 2 and printed in iteration 3.

6.9 Further options

There are miscellaneous further options which can be selected within the text files:
```
subito
entries          number
nofeasiblestart
wolfe            C1 C2
histprint
end
```

The option `subito` is also selected by `-s` in the command line. The program will end regularly, with normal output of the result, already after the first function evaluation (iteration 0).

The number, given with the keyword `entries`, is the minimum number of data for a global parameter. Global parameters which have a number of entries = number of measured points connected to it in the local fit-objects, smaller then the given number are ignored. This option is used to suppress global parameters with only few data, which are therefore rather inaccurate, and would spoil the condition of the linear system.

By default the initial global parameter values are corrected to follow all the constraints. A check for the validity of the constraints is repeated in every iteration, and eventually another correction is made at the end of an iteration. With the keyword `nofeasiblestart` the parameters are *not* made feasible (respecting the constraints) at the start.

The constants *C1* and *C2* are the line search constants for the strong Wolfe conditions; default values are the standard values $C_1 = 10^{-4}$ and $C_2 = 0.9$.

Histograms accumulated during the program are written to the textfile `millepede.his` and can be read after the job execution. If selected by the keyword `histprint` the histograms accumulated during the program are also printed.

The reading of a textfile ends, when the keyword `end` is encountered. Textlines after the line with `end` are not read.

7 Computation of the global fit

7.1 Memory managament

The solution of the optimization problem for a large number of parameters requires one large memory with many arrays, required to store vectors and matrices of large size corresponding to the number of parameters and constraints. MILLEPEDE II uses a large array in a common. This large array is dynamically divided into so-called subarrays, which are created, enlarged and moved, and removed according to the requirements. The space limitation is thus given for almost all subproblems only by the total space requirement. The largest space is required for the symmetric matrix of the normal least squares equations. This matrix can be a full matrix or for certain problems a sparse matrix. As an approximation a band matrix with a small band width can be used, if there is not enough space for the full or sparse matrix.

7.2 Initialization

After initialization of the memory management the command line options are read and analysed. Then the main text file is analysed, the names of other text files and the data files are recognized and stored in a table. Default value are assigned to the various variables; then all text files are read and the requested options are interpreted. The data for the definition for parameters, constraints and measurement are read and stored in subarrays.

7.3 First data loop

In subroutine `LOOP1` tables for the variable and fixed global parameters and translation tables are defined. The table of global parameter labels is first filled with the global parameters, appearing in the text files. All data files are read and all global parameter labels from the records are included in the list.
There are three integers to characterize a global parameter.

global parameter label = `ITGBL`: this is a positive integer, from the full range of 32-bit integers. $1 \ldots 2\,147\,483\,647 = 2^{31} - 1$.

global parameter index = `ITGBL`: A translation table is constructed to translate the global parameter label `ITGBL` to a global parameter index `ITGBI`, with a range $1 \ldots$ `NTGB`, where `NTGB` is the total number of global parameters (variable and fixed parameters).

variable-parameter index = `IVGBI`: the global parameters which are *variable* carry a variable-parameter index, with a range $1 \ldots$ `NVGB`, where `NVGB` is the number of variable global parameters. Parameter vectors in the mathematical computation are vectors containing only the variable parameters.

Function calls are used to translate the global parameter label `ITGBL`:

global parameter index ← global parameter label	`ITGBI = INONE(ITGBL)`
variable parameter index ← global parameter label	`IVGBI = INSEC(ITGBL)` .

The translation is done with a hash-index table. The translation from one parameter index to another one and back to the parameter label is done by the following statement functions:

global parameter label ← global parameter index	`ITGBL = JTGBL(ITGBI)`
variable-parameter index ← global parameter index	`IVGBI = JVGBI(ITGBI)`
global parameter index ← variable-parameter index	`ITGBI = JTGBI(IVGBI)` ,

which use simple fixed-length tables.

7.4 Second data loop

In subroutine `LOOP2` the subarray dimensions of the various vectors and matrices are determined. All data files are read and for each local-fit object the number of local and of global parameters is determined; the maximum values of the numbers are used to define the subarray dimensions for the arrays to be used in local fits, and for the contributions to the global fit.

The sparse matrix storage requires to establish the pointer structure of the sparse matrix. During reading the data files the list of global parameters used in each local-fit object is determined; an entry is made in a table for each pair of global parameters, which later corresponds to an off-diagonal matrix element. A hash-index table is used for the search operation. For sparse and full matrix storage the requirements of the constraint equations and Lagrange multipliers has to be included. For the sparse matrix a pointer structure is build, which allows a fast matrix×vector product. At the end of the subroutine all subarrays required for the determination of the solution are prepared.

7.5 Solution method overview

The section gives an short overview over the solution method; more detailed explanation is given in the subsequent sections.
In principle the solution can be determined in a single step; this mode can be selected by the keyword `subito` or option `-s`. There are however two reasons to perform iterations, which improve the result:

- Due to the large size or the method the one-step solution may be affected by rounding errors and may not be precise. Experience shows that the overall value objective-function value can be reduce using more than one step, although the decrease is sometimes rather small;
- Outliers may be important; they add a *non-linear* component to the otherwise linear problem and this requires iterations and repeated evaluation of the objective function; each function evaluation requires to read again all data files and to repeat the local fits. In a comparison of Millepede I and II results initially certain differences were observed; only after a careful outlier treatment these differences became small or disappeared.

The solution determined in iterations is explained. The starting iteration, with iteration number 0, is the most important and time-consuming one. The data files are read, and for each case a local fit is made. The matrix of the normal equations is formed only in this starting iteration. Depending on the selected method the matrix is inverted, diagonalized or decomposed, and the resulting matrix and decomposition is used in all later data-file loops, which take less time compared to the first data-file loop. Thus the data-file loop in iteration number 0 may be rather time consuming, especially for the case of a sparse matrix, where the index determination takes some time. The right-hand-side vector is formed in each data-file loop. A correction step in global parameter space is calculated at the end of the data-file loop in iteration number 0. Often the precision is already sufficient and no further data-file loops are necessary. In the subito mode (keyword `subito` or option `-s`) the correction is added to the global parameter values and the program stops.
Iterations with more data-file loops are recommended, to check the result, eventually to improve the result and to treat outliers. In each iterations a so-called line search is done, where the overall value of the objective functions is optimized along the correction-step direction *sufficiently*, using the strong Wolfe criterion. Often a single step is already sufficient for an iteration. The sample of local-fit objects may change during the iterations becuse of changing cut values; this may require extra function and derivative calculations.

7.6 Data file loops during iterations

In subroutine `LOOPN` all data files are read. For each local-fit object the local fit is performed, based on the actual global parameter corrections, and eventually including downweighting of outliers. Using the results of the local fit the value of the objective function F, the vector of first derivatives (gradient)

and, in the first data file loop, the matrix of second derivatives is calculated. For a linear fit the matrix of second derivatives will not change during the iterations; the outlier treatment will change the matrix, but usually the changes are small und the matrix collected in the first data loop is at least a good and sufficiently accurate approximation. Thus data loops after the first one are faster; depending on the method the solution of the matrix equation will be faster after the first data loop.

The local fit

The number of local parameters of a local-fit object is usually small; typical values are between 2 and 5. Since the problem is linear, a single step is sufficient in a local fit with minimization of the sum

$$\tfrac{1}{2}\sum_i \left(\frac{y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p})}{\sigma_i}\right)^2 ,$$

unless outlier downweighting is required, which may require a few iterations. In a loop over the local measurements the matrix and the right-hand side of the least squares normal equations are summed. For each single measured value, the residual measurement z_i

$$z_i \equiv y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p})$$

(see equation (28)) has to be corrected for the actual global parameters corrections $\boldsymbol{\Delta p}$ using the first global parameter derivatives (see equation (29)). The corrected residual z_i' is then used in the accumulation of the matrix and vector:

$$\Gamma_{jk} = \sum_i \left(\frac{\partial f_i}{\partial q_j}\right)\left(\frac{\partial f_i}{\partial q_k}\right)\frac{1}{\sigma_i^2} \qquad\qquad \beta_j = \sum_i \left(\frac{\partial f_i}{\partial q_j}\right)\frac{z_i'}{\sigma_i^2} .$$

Corrections $\boldsymbol{\Delta q}$ are determined by the solution of the matrix equation

$$\boldsymbol{\Gamma \Delta q} = -\boldsymbol{\beta} ,$$

which is determined using matrix inversion $\boldsymbol{\Delta q} = -\boldsymbol{\Gamma}^{-1}\boldsymbol{b}$, because the inverse matrix $\boldsymbol{\Gamma}^{-1}$ (the covariance matrix) is necessary for the contribution to the matrix of the global normal equations. The residual z_i' are then corrected for the local parameter corrections:

$$z_i'' = z_i' - \sum_j \frac{\partial f_i}{\partial q_j}\Delta q_j$$

and the new residuals z_i'' are used to calculate the χ^2 value S of the local fit

$$S = \sum_i \left(\frac{z_i''}{\sigma_i}\right)^2 ,$$

which should follow a χ^2 distribution with the given number of degrees of freedom n_{df} = number of measurements minus number of parameters. A fraction of 0.27 % or one out of 370 of the *correct* cases should have a deviation which corresponds to 3 standard devations in the $n_{\mathrm{df}} = 1$ case.
Very badly fitting cases should be rejected before using them for the global parameter fit. Basis of the rejection is the comparison of the sum S and the 0.27 %-value χ^2_{cut} of the $\chi^2_{n_{\mathrm{df}}}$ distribution. If the value S exceeds χ^2_{cut} by more than a factor of 50, then the sum is called *huge* and the local-fit object is rejected. Cases with $n_{\mathrm{df}} = 0$ are rejected too, because they do no allow any check. These rejections are always made, even if the user has not requested any outlier rejection.

With keyword `chisqcut` the user can require a further rejection. The first number given with the keyword is used during iteration 0; the local-fit object is rejected, if the value S is larger than this

number times $\chi^2_{\rm cut}$. Often this number has to rather high, e.g. 12.0; the initial value of S may be rather high before the first determination of global parameters, even for correct data, and, if possible, no correct data should be rejected by the cut. The second number given with the keyword is used during iteration 1 and can be smaller than the first number, because the first improvement of the local parameters is large. In further iterations the number is reduced by the sqrt-function. Values below 1.5 are replaced by 1, and thus the rejection is finally be done with a 0.27 %-rejection probability for correct data.

All not rejected local-fit objects contribute to the global fit. The function value F is the sum of the S-values for the local-fit objects:

$$F = \sum_l S_l \qquad \text{sum with index } l \text{ over all local fit-objects .}$$

Here even S-values from local-fit objects, which are rejected by cuts are included, with $S_l =$ cut value, in order to keep the sum F meaningful (otherwise the rejection of many local-fit object could simulate an improvement – which is not the case).

Contributions to the global parameter fit

After an accepted local fit the contributions to the normal least squares equations of the global fit have to be calculated. The first contribution is from the first order derivatives with respect to the global parameters:

$$(\boldsymbol{\Delta C}_1)_{jk} = \sum_i \left(\frac{\partial f_i}{\partial p_j}\right)\left(\frac{\partial f_i}{\partial p_k}\right)\frac{1}{\sigma_i^2} \qquad \Delta g_j = \sum_i \left(\frac{\partial f_i}{\partial p_j}\right)\frac{z_i''}{\sigma_i^2} \; . \tag{30}$$

Note that in the g_j-term the residual is already corrected for the local fit result. The vector $\boldsymbol{g}$ is the gradient of the global objective function.

The second contribution is, according to the MILLEPEDE principle, a mixed contribution from first order global and local derivatives. After the local fit the matrix $\boldsymbol{G}$ is formed, which has a number of columns equal to the number of local parameters, and a number of rows equal to the number of global parameters. The elements are

$$(\boldsymbol{G})_{jk} = \sum_i \left(\frac{\partial f_i}{\partial p_j}\right)\left(\frac{\partial f_i}{\partial q_k}\right)\frac{1}{\sigma_i^2} \; .$$

The second contribution to the global matrix $\boldsymbol{C}$ is then

$$\boldsymbol{\Delta C_2} = -\boldsymbol{G\Gamma}^{-1}\boldsymbol{G}^{\rm T} \tag{31}$$

with the matrices $\boldsymbol{G}$ and $\boldsymbol{\Gamma}$ from a local fit. Because of the dimension of matrices $\boldsymbol{G}$ and $\boldsymbol{\Delta C}$ the calculation would require a large number of operations. However only a small number of rows of matrix $\boldsymbol{G}$ is not equal to zero and the calculation can be restricted, with the help of pointers, to the non-zero part and in addition one can use the fact that the result is a symmetric matrix. With this code, which may appear somewhat complicated in comparison to the standard matrix multiplication, the computation time is small. Note that no extra contribution to the gradient vector $\boldsymbol{g}$ is necessary because the residual z_i'' used above already includes the local fit result.

The sum $\boldsymbol{C}$ of the two contributions, $\boldsymbol{\Delta C}_1$ from equation (30), and $\boldsymbol{\Delta C}_2$ from equation (31), summed over all local fit-objects, is the final matrix of the least squares normal equations. Element $(\boldsymbol{C})_{jk}$ is related to the global parameters with indices j and k. The matrix $\boldsymbol{C}$ corresponds to a simultaneous fit of the global *and* local parameters of all local-fit objects. The first term (30) alone corresponds to the fit, when the local parameters are assumed to be fixed; only those elements $(\boldsymbol{C})_{jk}$ of the matrix $\boldsymbol{C}$ are non-zero, where the two global parameters with indices j and k appear in the same measurement y_i. In contrast, in the second term (31) those elements $(\boldsymbol{C})_{jk}$ of the matrix $\boldsymbol{C}$ are non-zero, where the two global parameters with indices j and k appear in the same local fit.

Outlier downweighting in local fits

The influence of a single measurement y_i resp. z_i is proportional to the absolute value of the normalised residual $\zeta_i = z_i/\sigma_i$ in the pure least squares fit of local-fit objects. Thus measurements with a large deviation will in general distort the resulting fit. This effect can be avoided by downweighting those measurements. Measurements with large residuals are included in the method of M-estimates by assigning to them, instead of the ideal Gaussian distribution of least squares, a different distribution with larger tails. Technically this is done by multiplying the standard weight $1/\sigma_i^2$ of least squares by another factor, which depends on the actual value of $\zeta_i = z_i/\sigma_i$. This method requires an initial fit without downweighting, followed by iterative downweighting.
In MILLEPEDE II the local fits are improved, if necessary, by downweighting *after* the first iteration, with a number of iterations specified by the user with keyword `outlierdownweighting` it numberofiterations. In iterations 2 and 3 the downweighting is done with the Huber function and the Cauchy function is used, if more iterations are requested (see section 2.8 for an explanation of the two functions).

7.7 Global fit

Global fit without constraints

The aim is to find a step vector $\boldsymbol{\Delta p}$, which minimizes the objective function: $F(\boldsymbol{p}+\boldsymbol{\Delta p}) = \text{minimum}$. In the k-th iteration the approximate solution is $\boldsymbol{p}_k$. A quadrative model function $\widetilde{F}(\boldsymbol{p}+\boldsymbol{d})$

$$\widetilde{F}(\boldsymbol{p}_k+\boldsymbol{d}) = F_k + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} + \tfrac{1}{2}\boldsymbol{d}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{d} \tag{32}$$

is defined with a vector $\boldsymbol{d}$. The step vector $\boldsymbol{d}$ is determined as solution of the linear system

$$\boldsymbol{C}\boldsymbol{d} = -\boldsymbol{g}\ . \tag{33}$$

The step $\boldsymbol{d}$ will minimize the quadratic function $\widetilde{F}(\boldsymbol{p}+\boldsymbol{d})$ of equation (32), unless there is some inaccuracy due to rounding errors and other approximations of the matrix or the solution method. In order to get a *sufficient* decrease of the objective function $F(\boldsymbol{p}+\boldsymbol{\Delta p})$ a line search algorithm is used.

Line search

In the line search algorithm a search is made for the minimum of the function $F(\boldsymbol{p}+\boldsymbol{\Delta p})$ along a line $\boldsymbol{p}+\alpha\cdot\boldsymbol{d}$ in the parameter space, where α is a factor to be determined. A function $\Phi(\alpha)$ of this factor,

$$\Phi(\alpha) \equiv F(\boldsymbol{p}+\alpha\cdot\boldsymbol{d}) \tag{34}$$

is considered. Each function evaluation at a certain value of α requires one loop through the data with all local fits. The function value for $\alpha = 0$ is already calculated before (start value); the first value of α to be considered is $\alpha = 1$ and this value is often already a sufficiently accurate minimum. In the line search algorithm the so-called strong Wolfe conditions are used; these are

$$F(\boldsymbol{p}+\alpha\cdot\boldsymbol{d}) \le F(\boldsymbol{p}) + C_1\alpha\nabla F^{\mathrm{T}}\boldsymbol{d} \tag{35}$$

$$\left|\nabla F(\boldsymbol{p}+\alpha\cdot\boldsymbol{d})^{\mathrm{T}}\boldsymbol{d}\right| \le C_2\left|\nabla F^{\mathrm{T}}\boldsymbol{d}\right| \tag{36}$$

with $0 < C_1 < C_2 < 1$. The first condition requires a sufficient decrease of the function. The second condition, called curvature condition, ensures a slope reduction, and rules out unacceptable short steps. In practice the constant C_1 is chosen to small, for example $C_1 = 10^{-4}$. Typical values of the constant C_2 are 0.9 for a search vector by a Newton method and 0.1 for a search vector by the conjugate gradient method. Default values in MILLEPEDE II are $C_1 = 10^{-4}$, $C_2 = 0.9$. Often only one value of α, sometimes two to three values are evaluated. The last function value should never be higher than the start value.

Iterations

The global fit can be performed with a result $\boldsymbol{\Delta p}$ in one step, if the matrix equation is accurately solved and if there are no outliers. The result $\boldsymbol{\Delta p}$ calculated in the first data loop is usually already close to the final solution; it gives the largest reduction of the objective function. Outliers introduce some non-linearity into the problem and require iterations with repeated data loops. Another argument for repeating the steps is the non-neglectigible inaccuracy of the numerical solution of the step calculation due to rounding errors for the large number of parameters.

Iteration 0. The first data loop is called iteration 0. The function value, the first derivative vector $\boldsymbol{g}$ and the second derivative matrix $\boldsymbol{C}$ or an approximation to it are calculated. They allow to calculate the first step $\boldsymbol{\Delta p} = \boldsymbol{d}$.

Iteration ≥ 1. In all further iterations a line search is performed. Starting value is the point in parameter space, obtained in the previous iteration. Each function evaluation requires a data loop with local fits. The expected decrease ΔF in an iteration can be estimated by

$$\Delta F_{\text{estimate}} = -\boldsymbol{g}^{\mathrm{T}}\boldsymbol{d}\ .$$

This an be compared with the actual decrease

$$\Delta F_{\text{actual}} = F(\boldsymbol{p}) - F(\boldsymbol{p} + \alpha\boldsymbol{d})$$

at the end of the iteration. Both values should become smaller during the iterations; convergence can be assumed if both are of the order of ≈ 1.

Global fit with constraints

In the Lagrange multiplier method one additional parameter λ is introduced for each single constraint, resulting in an m-vector $\boldsymbol{\lambda}$ of Lagrange multiplier. A term depending on $\boldsymbol{\lambda}$ and the constraints is added to the objective function, resulting in the Lagrange function

$$\mathcal{L}(\boldsymbol{p}, \boldsymbol{\lambda}) = F_k + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} + \tfrac{1}{2}\boldsymbol{d}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{d} + \boldsymbol{\lambda}\left(\boldsymbol{A}\left(\boldsymbol{p}_k + \boldsymbol{d}\right) - \boldsymbol{c}\right) \tag{37}$$

Using as before a quadratic model for the function $F(\boldsymbol{p})$ and taking derivatives w.r.t. the parameters $\boldsymbol{p}$ and the Lagrange multipliers $\boldsymbol{\lambda}$, the two equations

$$\begin{aligned} \boldsymbol{C}\ \boldsymbol{d} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} &= -\boldsymbol{g} \\ \boldsymbol{A}\ \boldsymbol{d} \qquad\quad &= \boldsymbol{c} - \boldsymbol{A}\boldsymbol{p}_k \end{aligned}$$

are obtained; the second of these equations is the constraint equation. This system of two equations can be combined into one matrix equation

$$\left(\begin{array}{c|c} \boldsymbol{C} & \boldsymbol{A}^{\mathrm{T}} \\ \hline \boldsymbol{A} & \boldsymbol{0} \end{array}\right) \left(\begin{array}{c} \boldsymbol{d} \\ \hline \lambda \end{array}\right) = \left(\begin{array}{c} -\boldsymbol{g} \\ \hline \boldsymbol{c} - \boldsymbol{A}\boldsymbol{p}_k \end{array}\right)\ . \tag{38}$$

The matrix on the left hand side is still symmetric. *Linear* least squares problems with *linear* constraints can be solved directly, without iterations and without the need for initial values of the parameters. Insufficient precision of the equality constraints is corrected by the method discussed in section 3.2.

7.8 Output text files

Several output text file are generated in PEDE. All files have a fixed name. If files with this name are existing at the PEDE start, they are renamed by appending a $\sim$ to the file name; existing files with file name already carrying the $\sim$ are removed!

Log-file for PEDE. The file `millepede.log` summarizes the job execution. Especially information on the iterations is given.

Result file. The file `mpgparm.txt` contains one line per (variable or fixed) global parameter; the first three numbers have the same meaning as the lines following the `Parameter` keyword in the input text files. The first value is the label, the second is the fitted parameter value, and the third is the presigma, either copied from the input text file or zero (default value).

Debug file. The file `mpdebug.txt`, if requested by the `printrecord` keyword, contains detailed information of all data and fits for local-fit objects.

Eigenvector file. The file `mpeigen.txt` is written for the `method diagonalization`. It contains selected eigenvectors and their eigenvalues, and can be used e.g. for an analysis of weak modes.

8 Using MILLEPEDE II for track based alignment

8.1 Global parameters

Global parameters to be determined in a track-based alignment are mostly geometrical corrections like translation shifts and rotation angles. Usually only small corrections to the default position and angle information is necessary, and the assumption of *constant* first derivatives is sufficient (otherwise an iteration with creation of new data files would be necessary). In general an alignment should be done simultaneously of *all* relevant detectors, perhaps using structural constraints (see below).
In addition to the geometrical corrections there are several additional parameters, which determine trak-hit positions like Lorentz-angle and T_0-values, drift-velocities etc. These parameters should be included in the alignment process, since, due to the correlation between all pairs of global parameters, a separate determination of those parameters cannot be optimal. Also beam parameters like vertex position and beam direction angles should be included in the alignment, if those value are used with the tracks. Incorrectness of the model used e.g. to calculate the expected track hit positions, for example a wrong value of the Lorentz-angle, can introduce distortions in other parameters.

8.2 Constraints

Equality constraints are, in the mathematical sense, equations between global parameters, which have to be exact (within the computing accuracy). Only linear constraints are supported. Below two areas of constraints are discussed.

Removal of undefined degrees of freedom

A general linear transformation has 12 parameters. Three parameters of the translation and additional three parameters of a rotation are undefined, if only track residuals are minimized. At least these six parameters have to be fixed by constraining to zero the overall translation and rotation.

Structural constraints

A large track detector has usually a sub-detector structure: the track detector is built from smaller units, which are built by smaller sub-units etc. down to the level of a single sensor. The different levels can be visualized by a tree structure. Figure 2 shows the tree structure of the cms detector.
The tree structure can be reflected in the label structure and constraints. One set of global parameters is assigned to a unit. To each of the subunits another set of global parameters; the sum of the

transformations of the subunits is forced to zero by sum constraints, which remove the otherwise undefined degrees of freedom. For a unit with N subunits, with for example 6 degrees of freedom, the total number of degrees of freedom becomes

$$[6 + 6 \cdot N] \text{ (parameters)} - 6 \text{ (constraints)} = 6 \cdot N \ .$$

The parameters of the unit are more accurately defined than the parameters of the sub-units. Additional external measurement information from e.g. a survey can be assigned to the parameters of the unit. If all sub-unit parameters are fixed after an initial alignment, the parameters of the unit can still be variable and be improved in an alignment with a reduced total number of parameters.

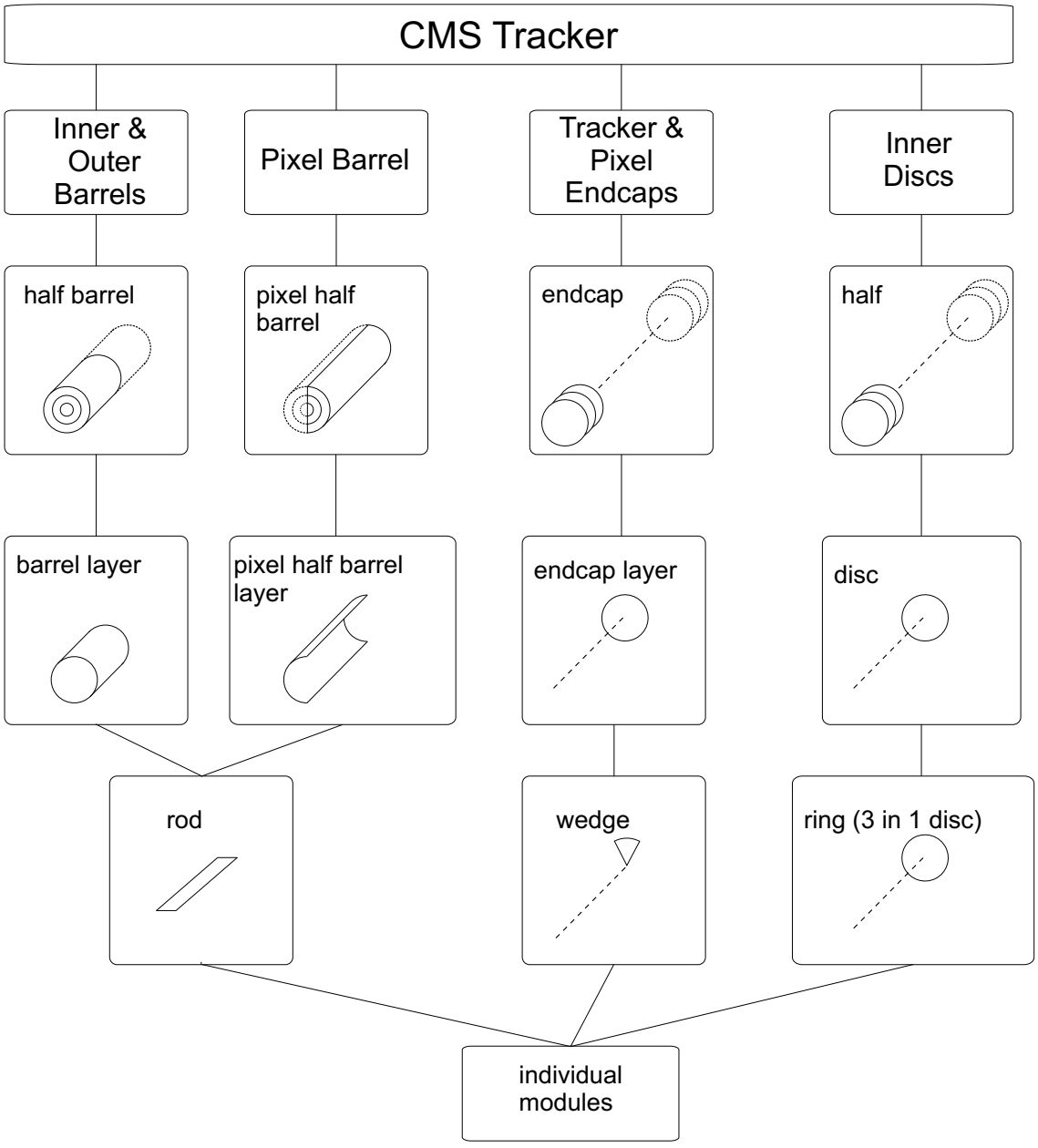

Figure 2: The tree of components of the cms inner track detectors.

8.3 Linear transformations in 3D

Nominal transformation. For track reconstruction the local coordinate system and the global $x\,y\,z$ coordinate system are used. The local system with coordinate $\boldsymbol{q}$ and components u, v and w is defined w.r.t. a sensor with origin at the center of the sensor; the u-axis is along the precise and the v-axis is along the coarse coordinate, and the w-axis is normal to the sensor. The transformation from the global to the local system is given by

$$\boldsymbol{q} = \boldsymbol{R}\,(\boldsymbol{r} - \boldsymbol{r}_0)$$

with the definitions

$$\boldsymbol{q} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \qquad \boldsymbol{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad \boldsymbol{r}_0 = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} .$$

The nominal position $\boldsymbol{r}_0$ and the rotation matrix $\boldsymbol{R}$ are determined by detector assembly and survey information. The convention is to use the three Euler angles ϑ, ψ and φ to determine the nominal rotation matrix $\boldsymbol{R}$:

$$\boldsymbol{R} = \begin{pmatrix} \cos\psi\,\cos\varphi - \cos\vartheta\,\sin\psi\,\sin\varphi & -\cos\psi\,\sin\varphi - \cos\vartheta\,\sin\psi\,\cos\varphi & \sin\vartheta\,\sin\psi \\ \sin\psi\,\cos\varphi + \cos\vartheta\,\cos\psi\,\sin\varphi & -\sin\psi\,\sin\varphi + \cos\vartheta\,\cos\psi\,\cos\varphi & -\sin\vartheta\,\cos\psi \\ \sin\vartheta\,\sin\varphi & \sin\vartheta\,\cos\varphi & \cos\vartheta \end{pmatrix}$$

or

$$\boldsymbol{R} = \begin{pmatrix} \cos\psi\,\cos\varphi - \cos\vartheta\,\sin\psi\,\sin\varphi & \sin\psi\,\cos\varphi + \cos\vartheta\,\cos\psi\,\sin\varphi & \sin\vartheta\,\sin\varphi \\ -\cos\psi\,\sin\varphi - \cos\vartheta\,\sin\psi\,\cos\varphi & -\sin\psi\,\sin\varphi + \cos\vartheta\,\cos\psi\,\cos\varphi & \sin\vartheta\,\cos\varphi \\ \sin\vartheta\,\sin\psi & -\sin\vartheta\,\cos\psi & \cos\vartheta \end{pmatrix}$$

The ranges of the angles are:$0 \le \vartheta < \pi$, $0 \le \psi < 2\pi$, and $0 \le \phi < 2\pi$.

A different convention is to parametrize the rotation by Euler angles φ, ϑ and ψ with the rotation $\boldsymbol{R} = \boldsymbol{R}_3(\varphi)\boldsymbol{R}_2(\vartheta)\boldsymbol{R}_3(\psi)$, where $R_i(\delta)$ is a rotation by an angle δ about the axis $\boldsymbol{n}_i$. The rotation is

$$\boldsymbol{R} = \begin{pmatrix} \cos\varphi\cos\vartheta\cos\psi - \sin\varphi\sin\psi & -\sin\varphi\cos\psi - \cos\varphi\cos\vartheta\sin\psi & \cos\varphi\sin\vartheta \\ \cos\varphi\sin\psi + \sin\varphi\cos\vartheta\cos\psi & \cos\varphi\cos\psi - \sin\varphi\cos\vartheta\sin\psi & \sin\varphi\sin\vartheta \\ -\sin\vartheta\cos\psi & \sin\vartheta\sin\psi & \cos\vartheta \end{pmatrix}$$

The ranges of the angles are: $0 \le \phi < 2\pi$, $0 \le \vartheta < \pi$ and $0 \le \psi < 2\pi$.

Alignment correction to the transformation. The alignment procedure determines a correction to the nominal transformation by an incremental rotation $\Delta\boldsymbol{R}$ and a translation $\Delta\boldsymbol{r}_0$. The combined translation and rotation becomes

$$\begin{aligned} \boldsymbol{r}_0 &\to \boldsymbol{r}_0 + \Delta\boldsymbol{r} \\ \boldsymbol{R} &\to \Delta\boldsymbol{R}\boldsymbol{R}\,. \end{aligned}$$

The correction matrix $\Delta\boldsymbol{R}$ is given by small rotations by $\Delta\alpha$, $\Delta\beta$ and $\Delta\gamma$ around the u-axis, the (new) v-axis and the (new) w-axis:

$$\Delta\boldsymbol{R} = \boldsymbol{R}_\alpha\boldsymbol{R}_\beta\boldsymbol{R}_\gamma \approx \begin{pmatrix} 1 & \Delta\gamma & -\Delta\beta \\ -\Delta\gamma & 1 & \Delta\alpha \\ \Delta\beta & -\Delta\alpha & 1 \end{pmatrix} ,$$

where the approximation has sufficient accuracy for small angles $\Delta\alpha$, $\Delta\beta$ and $\Delta\gamma$. The position correction $\Delta\boldsymbol{r}$ transforms to the local system as

$$\Delta\boldsymbol{q} = \Delta\boldsymbol{R}\,\boldsymbol{R}\,\Delta\boldsymbol{r} \qquad \text{with} \quad \Delta\boldsymbol{q} = \begin{pmatrix} \Delta u \\ \Delta v \\ \Delta w \end{pmatrix} \qquad \Delta\boldsymbol{r} = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}$$

These corrections define the corrected (aligned) transformation from global to local coordinates:

$$\boldsymbol{q}^{\text{aligned}} = \Delta\boldsymbol{R}\,\boldsymbol{R}\,(\boldsymbol{r} - \boldsymbol{r}_0) - \Delta\boldsymbol{q}\ .$$

The task of the alignment by tracks is to determine the correction angles $\Delta\alpha$, $\Delta\beta$ and $\Delta\gamma$ (and thus the rotation matrix $\Delta\boldsymbol{R}$) and the translation vector $\Delta\boldsymbol{r}$ or vector $\Delta\boldsymbol{q}$ for each individual detector element.

Acknowledgement

Markus Stoye has worked with different versions of MILLEPEDE II during the development phase in the last two years in alignment studies for the inner tracker of the cms experiment. I would like to thank him for the close collaboration, which was essential for the development of MILLEPEDE II.
I would like to thank also Gero Flucke, who contributed the C^{++} code, and Claus Kleinwort for the MILLEPEDE I/II comparison using data from the H1 experiment.