Alignment Algorithms

V. Blobel
Institut für Experimentalphysik, Hamburg, Germany

Abstract
Tracking detectors in high energy physics experiments require an accurate alignment in order to allow a precise reconstruction of tracks and vertices, according to the physics goals. Due to the large number of alignment parameters in a track-based alignment and in order to avoid potential distortions, advanced mathematical methods have to be used. The properties of several different methods used in HEP are discussed.

1 Introduction

The alignment and calibration of a track detector requires to *understand* the measurement process in detail. Ten thousands of parameters have perhaps to be optimized for the LHC track detectors. A good alignment is essential for important aspects of physics analysis with large accurate vertex detectors with a potential precision of a few μm. A nominal alignment is defined from design data and an initial optical survey, with corrections for electronics and mechanical effects. Aim of a track based alignment, supplemented by survey data and perhaps by Laser alignment data, is to improve the nominal alignment and to reduce the χ^2 of the track fits, in order to improve track and vertex recognition, to increase the precision of reconstructed tracks and vertices, and to improve the mass and vertex resolution, eliminating or reducing bias in detector data.

Calibration of instruments is usually done by measuring *reference standards* covering the range of interest. In a track based alignment however no *reference standards with known values* exist and certain degrees of freedom are only weakly defined or even undefined.

1.1 Classification of algorithms

Track-based alignment requires to minimize the sum of squared track residuals for a large number of tracks. A least squares objective function $F(\boldsymbol{\Delta p},\, \boldsymbol{q})$ to be minimized is defined, which depends on the alignment corrections $\boldsymbol{\Delta p}$, and the track parameters $\boldsymbol{q}$ of all tracks:

$$F(\boldsymbol{\Delta p},\, \boldsymbol{q}) = \sum_{\text{data sets}} \left(\sum_{\text{events}} \left(\sum_{\text{tracks}} \left(\sum_{\text{hits}} \Delta_i^2/\sigma_i^2 \right) \right) \right) + G(\boldsymbol{\Delta p}) \ . \tag{1}$$

The residual Δ_i is the difference between the fitted and the measured track position of the i-th measured value, and it is divided by the standard deviation σ_i of the measurement. The expression (1) is valid for uncorrelated measurements, and this assumption is usually made. In track data there are of course effects like multiple scattering, which introduce correlations between track points and require a corresponding track fit. Low-momentum tracks suffer from multiple scattering and those tracks are of little use for alignment. As indicated in equation (1) by $G(\boldsymbol{\Delta p})$, additional information like survey data may contribute significantly to the objective function.

A general overview on software alignment algorithms for tracking detectors is given in reference [1], and several details are not repeated here. Most alignment algorithms are based on the objective function (1), but they differ in the methods of minimization, as dicussed in section 2. However there are two rather different and interesting algorithms reported at this workshop, the "Track-based alignment using a Kalman Filter technique" [2], which is an extension of the Kalman Filter track fit technique, to

the alignment, and updates the alignment parameters sequentially during track processing, and the alignment method used for the SLD detector [3], where fits are made of functional forms to various residual types, and the alignment parameters are extracted from the fitted functional forms.

For several reasons the minimization of the objective function (1) is difficult. The number of alignment parameters is high because of the fine segmentation; for the LHC track detectors the number of parameters is of the order of 10^4 or even 10^5. The alignment precision depends on the amount of data used in the alignment and it may be necessary to use millions of tracks. The amount of data *and* the number of parameters, with weakly defined or undefined degrees of freedom, represents a problem even for todays computers.

The standard minimization method to obtain corrections $\boldsymbol{\Delta p}$ to alignment parameters $\boldsymbol{p}$ according to function (1) is based on derivatives and requires the solution of a system of linear equations:

$$\boldsymbol{C}\ \boldsymbol{\Delta p} = \boldsymbol{b}\ , \tag{2}$$

or of a sequence of such systems of linear equations in case of non-linearities; the symmetric n-by-n matrix $\boldsymbol{C}$ is an approximation of the second derivative matrix and the right-hand-side vector $\boldsymbol{b}$ the (negative) first derivative ∇F of the function (1). While mathematicians do not recommend matrix inversion for the solution of a matrix equation, most (not all) physicists think, that a a matrix inversion is unavoidable. Physicists also think that the inversion of a large matrix is impossible (from a talk: "The solution of 4200 equations in 4200 unknowns is computationally infeasible. Even worse, non-linear fit won't converge."). The possibility of an accurate solution of (2) depends on the condition number κ of the matrix (κ = ratio of largest to smallest eigenvalue) and a very high condition number, caused by large correlations between parameters, indicates an almost singular matrix. Large correlations between parameters and undefined degrees of freedom are of course a problem. If the origin of these problems is identified, one can try to improve the condition. Matrix inversion has the advantage of providing the covariance matrix and can be used for $n \approx 1\ 000$, but is problematic for larger value of n. Already for $n = 16\ 000$ the matrix requires a Gbyte of memory and a direct solution of equation (2) by matrix inversion (requiring a computing time $\propto n^3$) would take more than a day on a standard PC. There are several modern methods [4, 5] which are less demanding with respect to memore space and execution time. An overview is given in section 3. Iterative methods for the solution of the matrix equation (2) may be fast, and they can be applied to sparse or approximate matrices with a reduced space consumption. There are further efficient methods for large scale minimization without the need to store a matrix.

Non-linearities of the alignment parameterization and the track fit corrections are expected to be not large, at least not for routine alignment; they may require to go back to the original reconstruction program and that will take a lot of cpu time. Depending on the method the solution of equation (2) may be time consuming. For an iterative algorithm that converges only *linearly* to the solution $\boldsymbol{x}^*$ with a sequence $\{\boldsymbol{x}_k\}$ the inequality

$$\text{linear convergence} \quad \frac{||\boldsymbol{x}_{k+1} - \boldsymbol{x}^*||}{||\boldsymbol{x}_k - \boldsymbol{x}^*||} \leq r \qquad r \in (0,1)$$

is valid for all k sufficiently large. The speed of convergence depends on the eigenvalue spectrum and is slow for small eigenvalues. Iteration numbers of 100 or 1000 not uncommon, and an algorithm with r close to 1, i.e., $(1-r) \ll 1$ is in practice considered as *not converging at all*. For a quadratic function the constant is given by $r = (\kappa - 1)/(\kappa + 1)$. Convergence-recognition may be difficult; because of

$$||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k|| \approx (1-r)||\boldsymbol{x}_k - \boldsymbol{x}^*|| \qquad \text{with } (1-r) \ll 1$$

a small value of $||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k||$ does not yet mean small distance to the solution. In addition the outlier rejection, if necessary, will introduce non-linear effects and require iterations for the minimization of function (1).

1.2 Alignment parameters

Already before the alignment procedure a certain quality of the parameter value is required, which has to be sufficient to recognize tracks with a good efficiency. Initial parameter values $\boldsymbol{p}$ are based on nominal and design values, perhaps improved by survey data. The track-based alignment procedure should produce corrections $\Delta\boldsymbol{p}$ to the nominal values.

For planar sensors (silicon pixel or strip detectors) the local (sensor) coordinates (u, v, w) and global detector coordinates $\boldsymbol{r} = (x, y, z)$ are related by the equation $(u, v, w) = \boldsymbol{R}(\boldsymbol{r} - \boldsymbol{r}_0)$, where $\boldsymbol{R}$ is the nominal rotation matrix and $\boldsymbol{r}_0$ is the nominal position. Up to six corrections are required for each individual detector element or larger detector component, to improve the position and orientation, a translation $(\Delta u,\, \Delta v,\, \Delta w)$ with three values and three (small) angles $(\alpha,\, \beta,\, \gamma)$, which define a correction matrix, usually written in the form

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

where the approximation has sufficient accuracy for small angles. The precision of the six parameters is of course very different and one could reduce the total number of parameters by fixing e.g., three of the six parameters. For drift chambers there are additional parameters like Lorentz-angle, T_0, drift velocity v_{drift}, global values and values per plane and/or per layer, and e.g., parameters for corrections dependent on the angle between track and drift direction. In general the definition and parametrization related to the alignment corrections requires a detailed knowledge of the detector. Finally also external parameters like vertex position and beam direction can be included in the alignment procedure.

Methods with equality constraints can be used to reduce the number of parameters significantly. A certain part of the track detector can be considered as one unit, consisting of many sensors. One set of six parameters (translation, rotation) is assigned to the unit, and individual parameters sets are assigned to individual sensors. The transformation of a single sensor is thus determined by the individual parameters *and* the parameter set of the unit. Using constraint equations the overall translation and rotation of the sensors in the unit from the individual parameter sets is forced to be zero.

1.3 Undefined or weakly defined degrees of freedom

If the alignment of a track detector is based *only* on the minimization of residuals, then several degrees of freedom and certain parameters or linear combinations of parameters are undefined or only weakly defined and as a consequence the matrix of equation (2) is singular or almost singular.

There are some trivial undefined degrees of freedom. A general *linear* transformation of the whole detector with a translation and a 3×3 matrix $\boldsymbol{R}$ is determined by $3 + 9$ parameters and will not affect the χ^2 of the track fits. Those degrees of freedom can be defined by fixing detector planes. A more general method is the introduction of equality constraints[1] with Lagrange multipliers (see Section 3.1), for example to force the overall translation to zero. Additional (non-linear) distortions can be caused by coherently added corrections in the radius R, azimutal angle Φ and in the longitudinal coordinate z, leading to e.g., radial or z expansion and to effects called telescope, curl and bowing. These distortions are connected with large correlations between parameters. The use of external information like survey data or laser alignment data and a mixture of different track data will remove or reduce these effects. For example overlaps between adjacent wafers in the same layer and cosmic rays in the detector without magnetic field (straight tracks) are helpful to connect different areas of the detector and to avoid curvature distortions. Equality constraints can be applied in addition to control potential distortion; these methods are studied in a PhD thesis [6].

[1]The term *constraint* is often used to describe the effect of a certain measurement; here it is used for exact *equality* constraints.

2 Alignment algorithms

2.1 Biased algorithms

Almost all alignment methods tried in HEP experiments consider the dependence of the objective function on the alignment parameters only, and ignore the dependence on the track parameters $\boldsymbol{q}$, i.e., an objective function $F(\boldsymbol{\Delta p})$ instead of $F(\boldsymbol{\Delta p}, \boldsymbol{q})$ is considered. Optimal track parameters are determined independently in a track fit and the residuals of the measured points w.r.t. the optimal fit are calculated. In some methods called *robust* the mean values of the residuals are determined and used to improve the alignment. Otherwise the derivatives of the residuals w.r.t. the global parameter corrections are determined. For a single point measured on a track the linearization

$$\text{a single track point:} \qquad y = f(x; \boldsymbol{p}^{\text{global}}) + \left(\frac{\partial f(x)}{\partial \boldsymbol{p}}\right)^{\mathrm{T}} \Delta \boldsymbol{p}_k^{\text{global}} + \epsilon$$

is used, where y is the measured coordinate, $f(x; \boldsymbol{p}^{\text{global}})$ the track parametrization and ϵ is the measurement error, with a standard deviation σ. Each measured point contributes in the method of least squares to the matrix and vector of the normal equations; in detail the term $\partial f/\partial p_i \cdot \partial f/\partial p_j \,/\sigma^2$ is added to the element C_{ij} of the matrix. Only elements C_{ij}, where both parameters p_i and p_j appear *in a single measured point*, get a contribution. The result is a big matrix, where only 6-by-6 submatrices are non-zero, if six parameters are used to describe the alignment position and orientation of a measured point. The solution for the corrections Δp is simple and fast: instead of having to solve a large matrix equation one has to solve a large number of independent matrix equations of dimension six. The introduction of equality constraints between alignment parameters seems to be impossible in this method (but see section 3.3.4).

Because the track fit parameters are determined *before* the alignment step, the residuals are biased and the resulting alignment parameter corrections, determined either directly from the means of residuals or by the least squares fit are biased. In order to reduce or remove the bias in the result, iterations of the procedure are necessary, where the track fits are repeated with the improved alignment hoping, that the procedure converges. It is difficult to relate the procedure to any iterative method found in mathematical textbooks. If the convergence is linear, then the convergence rate depends on the eigenvalue spectrum of the complete system, and one could expect a good progress for the initial iterations and a slower and slower progress with increasing iteration number. Reported are iteration numbers until convergence in the range from 20 to 100.

2.2 Unbiased algorithms

The alternative to the fits of the previous section is the attempt to perform a simultaneous least squares fit of all global *and* all local (track) parameters in a single step with an objective function $F(\boldsymbol{\Delta p}, \boldsymbol{q})$. The total number of unknowns is of course large; in addition to the perhaps ten thousand alignment parameters there a perhaps five track parameters for each of one million tracks. However the aim of the fit is to get the alignment parameters and there are mathematical methods to reduce the size the system of equations, without any approximation, to the alignment parameters. This method, as explained below, has been used in the general Millepede program [7]. An equivalent algorithm has been developed for the ATLAS experiment [8].

Now the linearization for a single point measured on a track with index k includes the derivatives of the track residual with respect to the (local) track parameters:

$$\text{a single track point:} \qquad y = f(x_i; \boldsymbol{p}^{\text{global}}, \boldsymbol{q}^{\text{local}}) + \left(\frac{\partial f(x)}{\partial \boldsymbol{p}}\right)^{\mathrm{T}} \Delta \boldsymbol{p}^{\text{global}} + \left(\frac{\partial f(x)}{\partial \boldsymbol{q}_k}\right)^{\mathrm{T}} \Delta \boldsymbol{q}_k^{\text{global}} + \epsilon$$

It is assumed that a track fit has been done before and only small track parameter corrections $\Delta \boldsymbol{q}_k$, caused by a change in the alignment parameters, have to be fitted. The complete matrix equation for global and local parameters includes sums over track index k and contains many single-track-related

sub-matrices: the n-by-n matrix $\boldsymbol{C}$ for n global parameters and m-by-m matrices $\boldsymbol{C}_k^{\text{local}}$ and n-by-m matrices $\boldsymbol{H}_k^{\text{global-local}}$, if m track parameters are assumed:

$$\left(\begin{array}{c|c|c|c} \sum_k \boldsymbol{C}_k^{\text{global}} & \cdots & \boldsymbol{H}_k^{\text{global-local}} & \cdots \\ \hline \vdots & \ddots & 0 & 0 \\ \hline \left(\boldsymbol{H}_k^{\text{global-local}}\right)^{\mathrm{T}} & 0 & \boldsymbol{C}_k^{\text{local}} & 0 \\ \hline \vdots & 0 & 0 & \ddots \end{array}\right) \times \left(\begin{array}{c} \Delta\boldsymbol{p}^{\text{global}} \\ \hline \vdots \\ \hline \Delta\boldsymbol{q}_k^{\text{local}} \\ \hline \vdots \end{array}\right) = \left(\begin{array}{c} \sum_k \boldsymbol{b}_k^{\text{global}} \\ \hline \vdots \\ \hline \boldsymbol{b}_k^{\text{local}} \\ \hline \vdots \end{array}\right) .$$

In order to explain the mathematical method, the matrix equation $\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b}$ with a symmetric matrix is partitioned ($\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{22}$ are symmetric matrices):

$$\left(\begin{array}{c|c} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \hline \boldsymbol{C}_{12}^{\mathrm{T}} & \boldsymbol{C}_{22} \end{array}\right) \left(\begin{array}{c} \boldsymbol{a}_1 \\ \hline \boldsymbol{a}_2 \end{array}\right) = \left(\begin{array}{c} \boldsymbol{b}_1 \\ \hline \boldsymbol{b}_2 \end{array}\right) .$$

In the case of a zero rectangular matrix C_{12} the result for the vector $\boldsymbol{a}_2$ would be $\boldsymbol{C}_{22}\ \boldsymbol{a}_2^* = \boldsymbol{b}_2$, which is solved by $\boldsymbol{a}_2^* = \boldsymbol{C}_{22}^{-1}\ \boldsymbol{b}_2$; this is called a local solution and corresponds to an improved least squares track fit. In the non-zero case for C_{12} the complete solution for the two vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ can be written in the form

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

where the submatrix $\boldsymbol{B}$ is the inverse of the expression $\left(\boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^{T}\right)$, called Schurs complement. The solution for vector $\boldsymbol{a}_1$ can be obtained from the solution of the reduced matrix equation

$$\left(\boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^{T}\right)\boldsymbol{a}_1 = \left(\boldsymbol{b}_1 - \boldsymbol{C}_{12}\boldsymbol{a}_2^*\right) .$$

This method can be applied for each track. The local track fit is performed, and the related matrices $\boldsymbol{V}_k = \boldsymbol{C}_k^{-1}$ and $\boldsymbol{H}_k$ are calculated. Finally for each track the term $-\boldsymbol{H}_k\boldsymbol{V}_k\boldsymbol{H}_k^{T}$ is added to the global matrix $\boldsymbol{C}$ and the term $-\boldsymbol{H}_k\left(\boldsymbol{V}_k\boldsymbol{b}_k\right)$ is added to the vector $\boldsymbol{b}$. These additional terms transfer the *local* information to the matrix equation of the global parameters. All elements C_{ij}, where the parameters p_i and p_j appear in the fit of the track, get a contribution and (compare the case of the biased fit in section 2.1) the whole matrix will become a good approximation of the true second derivative matrix of function (1). After the loop on all tracks the complete information is collected and the matrix equation (2) for the global parameters has to be solved. Matrices $\boldsymbol{C}$ and vectors $\boldsymbol{b}$ from several data sets can be simply added to get the combined result. This method has been used in the general Millepede program. First development of the Millepede principle with the reduction of matrix and matrix inversion for the solution of the matrix equation was done in 1996, the code was used routinely for the vertex detector and the drift chamber in the H1 collaboration since 1997 [7]; with an increasing number of parameters the

drift chamber resolution has now been improved by a factor two [9]. The code and a manual was made available in 2000 on the web page `http://www.desy.de/~blobel`. Being an experiment-independent program it has been used since then by many experiments for alignment problems with up to 5 00 parameters. Several times it has been converted from Fortran to C++ (unpublished).

The development of a new version of Millepede, called Millepede II, started in May 2005 after discussions with the Hamburg cms group. The aim was to allow the alignment with up to 100 000 parameters in a reasonable time on a standard PC, using the same principle as before, i.e., unbiased and simultaneous fit of an arbitrary number of tracks and of alignment parameters. The plan was to use different, direct and iterative, methods either for the solution of the large matrix equation or for the direct minimization of function (1) using mathematical methods from the mathematical community e.g., [4], no home-made iterative methods. The design included a strong separation of experiment-dependent code and the experiment-independent Millepede alignment computation. A small C++ or Fortran routine is called within the experiments event-processing program to write the alignment information (derivatives, measured data, ...) to a special file. One or several of such files are then input to a general stand-alone Millepede program to produce the alignment parameters, with automatic recognition of existing alignment parameters, allowing to fix parameters with too few data. Equality constraints and in addition measured data from different sources (for example survey data) can be included. Preliminary versions of Millepede II have been used since summer 2005 by a PhD student for the cms experiment. At present almost all solution methods described in section 3 are included and working.

3 Mathematical methods

The introduction of overall equality constraints seems to be necessary for track-based alignment and requires the solution of large systems of equations. Several different methods with different memory-consumption and cpu-time requirements exist and are discussed in this section.

3.1 Equality constraints

Equality constraints in the solution of the matrix equation are essential. Undefined degrees of freedom can be fixed by adding *linear equality constraint* equations of the type

$$\boldsymbol{a}^{\mathrm{T}}\Delta\boldsymbol{p} = c \qquad \text{e.g. overall translation} \quad d_x = \sum_i \Delta x_i = 0 \ ,$$

"zero average displacement", or "zero rotation of the whole detector". The standard method for equality constraints is the method of Lagrange multipliers, where an additional parameter, the Lagrange multiplier, is introduced for each constraint. A single equality constraint is introduced by appending a term $\lambda\left(\boldsymbol{a}^{\mathrm{T}}\Delta\boldsymbol{p} - c\right)$ to the function (1). Many equality constraints are combined to the matrix equation $\boldsymbol{A}\boldsymbol{\Delta p} = \boldsymbol{c}$, and the objective function is extended to the Lagrange function, leading to the matrix equation

$$\left(\begin{array}{c|c} \boldsymbol{C}^{\text{global}} & \boldsymbol{A}^{\mathrm{T}} \\ \hline \boldsymbol{A} & \boldsymbol{0} \end{array}\right) \left(\begin{array}{c} \Delta\boldsymbol{p}^{\text{global}} \\ \hline \lambda \end{array}\right) = \left(\begin{array}{c} \boldsymbol{b}^{\text{global}} \\ \hline \boldsymbol{c} \end{array}\right) \ .$$

The task is to find, with a solution of the equation, a stationary point of the Lagrange function. The matrix is no longer positive definite, but will have positive and negative eigenvalues. Certain iterative methods for the solution of matrix equations require a positive definite matrix and can thus not be used.

An alternative method is the elimination method (for linear constraints), where, according to the constraint equations, certain parameters are expressed by linear combinations of other parameters and thus eliminated. The problem is reduced to an unconstrained problem with a reduced number of parameters; the matrix is positive definite.

3.2 Outlier rejection

Outliers in the data have a large influence on the result and can deteriorate the alignment result. The difficulty is the fact that wrong initial alignment parameters can fake outliers. One method is an iterative solution with a large initial cut (for example 10 standard deviations), which is reduced to a final cut for example of three standard deviations. This method has been used in Millepede I. The same procedure is available in Millepede II, but in addition the local track fits are made robust against outliers by the technique of M-estimates in iterations of the local fits after the first one. Technically the influence of outliers on local fits is reduced by down-weighting. In least squares the objective function is the sum $\sum z_i^2/2$ of squares of scaled residuals $z_i = \Delta_i/\sigma_i$. In M-estimates the square $\rho(z) = z^2/2$ is replaced by a dependence with reduced influence for larger residuals, for example the Huber function $\rho(z)$, which results in an additional weight factor ω_i for the i-th measurement:

$$\rho(z) = \begin{cases} z^2/2 & \text{if } |z| \le c = 1.345 \\ c\,(|z| - c/2) & \text{if } |z| > c = 1.345 \end{cases} \qquad \text{weight } \omega = \begin{cases} 1 \\ c/|z| \end{cases} .$$

The influence of outliers on local fits is thus reduced by down-weighting. The asymptotic efficiency of the fits for pure Gaussian data is still 95 % for the value of the constant $c = 1.345$.

3.3 Solution of large matrix equations

3.3.1 Solution by matrix inversion

The standard method for the solution of the matrix equation $\boldsymbol{C}\Delta\boldsymbol{p} = \boldsymbol{b}$ with a symmetric matrix $\boldsymbol{C}$ is the stable Gauss algorithm with pivot selection on the diagonal. The computing time is $T =$ constant $\times\, n^3$. The constant for the subroutine used in Millepede is about 20×10^{-9} sec on a standard PC, corresponding to a solution time below one hour for $n =$ several thousands.

A standard inversion routine will usually fail – at least a few parameters out of many thousands will be badly defined; the matrix is almost singular and is destroyed during computation without result for the correction vector $\Delta\boldsymbol{p}$. Inversion will of course also fail, if undefined degrees of freedom remain undefined without equality constraints. In the Millepede inversion the largest pivot is selected in each step, but the inversion stops if no acceptable pivot is found, i.e., the largest possible submatrix is inverted and zero corrections are returned for the remaining parameters.

All variances and covariances are available in the inverse matrix. The *global correlation coefficient*, ρ_j is a measure of the total amount of correlation between the j-th parameter and *all* the other variables. It is the largest correlation between the j-th parameter and every possible linear combination of all the other variables, defined by

$$\rho_j = \sqrt{1 - \frac{1}{(\boldsymbol{V})_{jj} \cdot (\boldsymbol{C})_{jj}}} \qquad \text{or} \qquad (\boldsymbol{V})_{jj} \cdot (\boldsymbol{C})_{jj} = \frac{1}{1 - \rho_j^2} \qquad \text{with} \qquad \boldsymbol{V} = \boldsymbol{C}^{-1} .$$

The range of global correlation coefficients is $0 \ldots 1$. The matrix is ill-conditioned (almost singular), if any ρ_j is close to 1, with a large condition number κ of the matrix (iterative methods would be slow). The values of the global correlation coefficients depend on the geometry and the type of data; additional data (cosmics, vertex and mass-constrained tracks) can reduce the global correlations significantly and improve the alignment.

3.3.2 Solution by diagonalization

The diagonalization of the symmetric matrix $\boldsymbol{C}$ allows to recognize singularity of the matrix, or near singularity, by the determination of eigenvalues and eigenvectors:

$$\boldsymbol{C} = \boldsymbol{U}\ \boldsymbol{D}\ \boldsymbol{U}^{\mathrm{T}}$$

with the diagonal matrix $\boldsymbol{D}$ of eigenvalues, and the square matrix $\boldsymbol{U}$ of eigenvectors (with $\boldsymbol{U}\,\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{U}^{\mathrm{T}}\,\boldsymbol{U} = \boldsymbol{1}$. Note: $\boldsymbol{C}^{-1} = \boldsymbol{U}\;\boldsymbol{D}^{-1}\;\boldsymbol{U}^{\mathrm{T}}$). The eigenvalues λ_i in $\boldsymbol{D}$ are ordered with

$$\boldsymbol{D} = [\mathrm{diag}\,(\lambda_i)]: \quad \lambda_1 \;\geq\; \ldots \geq \lambda_k \geq \lambda_{k+1} = \ldots \lambda_n\;.$$

Algorithms for diagonalization are iterative, with a computing time ≈ 10 times larger than inversion. Zero and small positive eigenvalue correspond to undefined or weakly defined degrees of freedom amd the corresponding linear combinations can be suppressed. The eigenvalues are used to write the solution of $\boldsymbol{C}\,\boldsymbol{\Delta p} = \boldsymbol{b}$ in the form

$$\boldsymbol{\Delta p} = \boldsymbol{U}\;\left[\mathrm{diag}\,\left(\frac{1}{\sqrt{\lambda_i}}\right)\right]\underbrace{\left[\mathrm{diag}\,\left(\frac{1}{\sqrt{\lambda_i}}\right)\right]\;(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{b})}_{=\,\boldsymbol{q}\quad\text{with}\quad \boldsymbol{V}[\boldsymbol{q}]\,=\,\boldsymbol{1}}$$

with the replacement $1/\lambda_i = 0$ for $\lambda_i = 0$ or small. The solution can also be written in the form

$$\boldsymbol{\Delta p} = \boldsymbol{U}\;\left[\mathrm{diag}\,\left(\frac{1}{\sqrt{\lambda_i}}\right)\right]\boldsymbol{q} \qquad \boldsymbol{q} = \left[\mathrm{diag}\,\left(\frac{1}{\sqrt{\lambda_i}}\right)\right]\;(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{b})\;.$$

The covariance matrix of the transformed vector is the unit matrix, ($\boldsymbol{V}[\boldsymbol{q}] = \boldsymbol{1}$) and this allows a simple significance test for the (independent) components of the vector $\boldsymbol{q}$. Singular value decomposition (SVD), if applied to the square matrix $\boldsymbol{C}$, is equivalent to diagonalization.

3.3.3 Generalized minimal residual method (GMRES) and sparse matrix storage

Large matrices are usually *sparse*, with a small fraction, often a few percent only, of non-zero off-diagonal elements. The inverse of a sparse matrix is a dense matrix, and the inversion method can not make use of the sparse structure. There are iterative methods for the solution of large matrix equations, which can work on a sparse matrix. Only products of the form $\boldsymbol{Cx}$ are required, and the matrix $\boldsymbol{C}$ is never modified. The method of conjugate gradients (Hestenes, Stiefel 1952) has been developed for the solution with a positive definite matrix. The generalized minimal residual method (GMRES) has been developed for the solution of a very large system of linear equations, with a symmetric matrix of logical size $n \times n$, which may be indefinite, very large and sparse, by an optimized solution of a quadratic minimization problem, in analogy to the method of conjugate gradients.

One example is the subroutine MINRES [10]. The matrix is accessed *only* by means of a subroutine call

`call Aprod ( n, x, y )` to return $\boldsymbol{y} = \boldsymbol{Cx}$

for any given vector $\boldsymbol{x}$. In a test with 12 000 parameters the solution with MINRES took a cpu time of 32 sec, compared to a cpu-time 12 h, 46 min, 5 s for matrix inversion, with essentially the same result. The computing time is sensitive to the eigenvalue spectrum of the matrix, which can be improved by preconditioning. Preconditioning is an option in MINRES by means of a subroutine call

`call Msolve( n, x, y )` to solve $\boldsymbol{My} = \boldsymbol{x}$ for $\boldsymbol{y}$

without altering vector $\boldsymbol{x}$. The matrix $\boldsymbol{M}^{-1}$ should be an approximation to $\boldsymbol{C}^{-1}$, such that $\boldsymbol{M}^{-1}\boldsymbol{C} \approx \boldsymbol{1}$ (see section 3.3.4).

Parameter errors or more general, the elements of the covariance matrix $\boldsymbol{V}$, depend on the hits statistics and the geometry. The inverse of matrix $\boldsymbol{C}$ is the covariance matrix $\boldsymbol{V}$ of the alignment parameters (with $\boldsymbol{CV} = \boldsymbol{1}$). This is available with matrix inversion and diagonalization, but not with MINRES. However some elements of the covariance matrix $\boldsymbol{V}$ can be calculated with MINRES. The column vector of matrix $\boldsymbol{V}$ with index j is determined by the solution of the matrix equation $\boldsymbol{Cv}_j = \boldsymbol{e}_j$, where $\boldsymbol{e}_j$ is the j-th column vector of the unit matrix.

The MINRES method can be used for a full or for a sparse matrix C. For a sparse matrix with a fraction q of non-zero off-diagonal elements a sparse index storage scheme can be used, which requires

$$n + q \cdot n(n-1)/2 \quad \text{double precision (data) and integer (indices) words}$$

and is optimized for the product (9 lines of code). The automatic generation of parameter-index relations and the definition of the sparse storage as well as the numerical matrix generation by sums requires a large number of comparisons. A fast method using a combination of hashing, sorting and binary search is used in Millepede II.

3.3.4 Cholesky decomposition for band matrices

A symmetrically structured matrix $\boldsymbol{C}$ with zero elements C_{ij} for $|i-j| > m$ is called a band matrix with semibandwidth m and bandwidth $2m+1$. The Cholesky decomposition

$$\boldsymbol{C} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}} \,, \tag{3}$$

with a left unit triangular matrix $\boldsymbol{L}$ (values 1 on the diagonal) and a diagonal matrix $\boldsymbol{D}$, can be done *in-place* (preserving the band structure), with matrices $\boldsymbol{D}$ and $\boldsymbol{L}$ taking the space of the (symmetric) matrix $\boldsymbol{C}$. For a fixed bandwidth the computing time depends *linearly* on the dimension n of the matrix. The decomposition allows to rewrite the matrix equation (2) in the form $\boldsymbol{L}\left(\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{\Delta p}\right) = \boldsymbol{b}$ and now the matrix equation can be solved in two steps by a forward and a backward substitution. Substitutions are straightforward because $\boldsymbol{L}$ is triangular.

The matrix $\boldsymbol{C}$ of alignment problems is of course not a band matrix, but a band matrix with e.g., a semibandwidth $m = 6$ can be considered as an approximation of the parameter part of the matrix $\boldsymbol{C}$ and one could try an iterative solution. The constraint part of the matrix $\boldsymbol{C}$ will in general have significant elements in *all* positions, and a narrow band matrix would be an unacceptable approximation. But for matrices with a *variable band width* the decompostion with Gaussian elimination without interchanges preserves the band structure too; the decomposition of variable-band (also called skyline, and profile) matrices can be done *in-space*. This allows to use the full Lagrange formalism for the equality constraints, and a narrow-bandwidth approximation for the parameter part of the matrix $\boldsymbol{C}$ in iterative solutions (compare section 2.1). Overall memory space consumption is low, on the cost of a perhaps larger number of iterations. The method can also be used for the preconditioning of the generalized minimal residual method of section 3.3.3; tests show a significant increase of the speed of that algorithm by preconditioning.

3.4 Quasi-Newton methods and limited-memory BFGS

The iterative Quasi-Newton methods for function minimization require only the calculation of the gradient ∇F of the objective function $F(\boldsymbol{p})$ at each iteration. In each iteration k a new step $\Delta\boldsymbol{p}$ can be calculated by

$$\Delta\boldsymbol{p} = -\boldsymbol{B}_k \nabla F \,, \tag{4}$$

where $\boldsymbol{B}$ is an approximation of the inverse of the second derivative matrix. The step is used in a line search minimization of the function $\Phi(\alpha) = F(\boldsymbol{p} + \alpha \cdot \Delta\boldsymbol{p})$. Usually the value $\alpha = 1$ is acceptable; the standard line-search method giving a sufficient improvement is the Wolfe line search [4]. Starting from a simple assumption (e.g. $\boldsymbol{B}_0 =$ scaled unit matrix $\gamma \cdot \mathbf{1}$) the approximate inverse Hessian $\boldsymbol{B}$ is improved by updates, using the difference vectors

$$\boldsymbol{s}_k = \boldsymbol{p}_{k+1} - \boldsymbol{p}_k \qquad\qquad \boldsymbol{y}_k = \nabla F_{k+1} - \nabla F_k \,.$$

Requiring the secant equation $\boldsymbol{B}_{k+1}\boldsymbol{y}_k = \boldsymbol{s}_k$, the most-popular update formula is

$$\rho_k = 1/\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k \qquad \boldsymbol{B}_{k+1} = \left(\mathbf{1} - \rho_k \boldsymbol{s}_k \boldsymbol{y}_k^{\mathrm{T}}\right)\, \boldsymbol{B}_k \,\left(\mathbf{1} - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^{\mathrm{T}}\right) + \rho_k \boldsymbol{s}_k \boldsymbol{s}_k^{\mathrm{T}} \qquad \text{(BFGS)}$$

Table 1: Overview over mathematical methods (for an explanation see text).

Method	$F(\boldsymbol{p})$	∇F	$\boldsymbol{C}$	$\boldsymbol{C}^{-1}$
Diagonalization	–	×	×	×
Inversion	–	×	×	×
Generalized residual minimization	–	×	(×)	(×)
Variable-band matrix	×	×	(×)	–
Limited memory BFGS	×	×	–	–

([4], used also e.g. in MINUIT/MIGRAD). The BFGS method has $\mathcal{O}(n^2)$ operations per iteration and has a superlinear rate of convergence but requires of course to store the full (dense) matrix $\boldsymbol{B}$ and thus cannot be used directly for alignment with a very large number of parameters.

However the matrix $\boldsymbol{B}_k$ is required only in the product $\boldsymbol{B}_k \nabla F$ of equation (4) and this product can be also calculated from a set of difference vectors $\boldsymbol{s}$ and $\boldsymbol{y}$, *without* forming the matrix $\boldsymbol{B}_k$ explicitly. The limited memory BFGS (short: L-BFGS) method [11, 12] uses update information only from recent iterations. With $\boldsymbol{B}_0 = \gamma \cdot \mathbf{1}$ the product $\boldsymbol{B}_k \nabla F_k$ is evaluated from the last m difference vectors $\boldsymbol{y}_k$, $\boldsymbol{s}_k$ only; good values for m are in the range $3 \ldots 20$. The storage requirement is, with $n\,(2m+4)$, *linear* in n, and $\approx 3/2\, m^2 n$ operations are needed per iteration; the iterative method has a fast rate of *linear* convergence, and is considered as the optimal method for optimization problems with $n \gg 100\,000$ parameters [5].

4 Summary

Track-based alignment with a large number of parameters requires advanced mathematical methods. The quantities required for different mathematical methods are given in Table 1. The matrix $\boldsymbol{C}$ is required for the method as a full matrix (symbol ×) or as a sparse matrix with reduced memory consumption (symbol (×)). In general the reduced memory consumption has to be compensated by a larger number of iterations. All methods allow to apply equality constraints in order to avoid distortions by weakly defined degrees of freedom and will reduce track residuals significantly. For the LHC track detectors a quality of alignment corresponding to the quality of the hardware can be expected.

Acknowledgements

I would like to thank the organizers of the workshop for arranging a superb program.

References

[1] V. Blobel, Software alignment for tracking detectors, NIM **A 566**, (2006) 5 - 13.

[2] R. Frühwirth, Track-based alignment using a Kalman Filter technique, these proceedings.

[3] F. Wickens, Internal alignment of the SLD vertex detectors, these proceedings.

[4] J. Nocedal and S.J. Wright, Numerical Optimization, Spinger Series in Operations Research, Springer, 1999.

[5] J.F. Bonnans, J.C. Gilbert, C. Lemarechal, and C.A. Sagastizabal, Numerical Optimization – Theoretical and Practical Aspects, (2003) Springer.

[6] M. Stoye, PhD thesis, in preparation, University of Hamburg

[7] V. Blobel and C. Kleinwort, A new method for the high-precision alignment of track detectors, PHYSTAT2002, Durham, and `arXiv-hep-ex/0208021`.

[8] P. Bruckman De Renstrom, Aligment strategy from ATLAS inner detector alignment, these proceedings.

[9] C. Kleinwort, Alignment Experience from ZEUS/H1, these proceedings.

[10] C.C. Paige and M.A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. **12(4)**, (1975) 617 - 629, and
`www.stanford.edu/group/SOL/software/minres.html`.

[11] J. Nocedal, Updating quasi-Newton matrices with limited storage, Mathematics of Computation **35** (1980), 773 - 782.

[12] D.C. Liu and J. Nocedal, On the limited-memory BFGS method for large scale optimization, Mathematical Programming **45** (1989) 503 - 528.