ATLAS *Inner Detector Alignment and Commissioning Workshop*

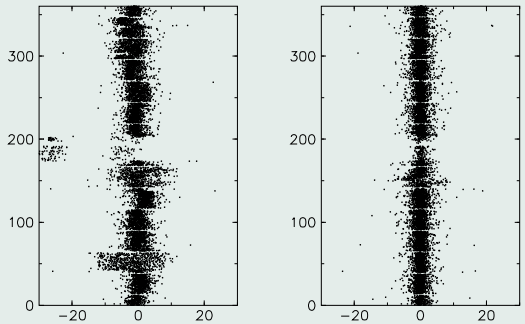
Ringberg 12 June – 16 June 2006

Detector Alignment by Tracks with Millepede II

Volker Blobel – Universität Hamburg

Abstract

Tracking detectors in high energy physics experiments require an accurate determination of a large number of alignment parameters in order to allow a precise reconstruction of tracks and vertices. In addition to the initial optical survey and corrections for electronics and mechanical effects the use of tracks in a special software alignment is essential. The general program MILLEPEDE performs a simultaneous least squares fit of a large number of tracks, with determination of (global) alignment parameters and (local) track parameters. The version II, which is under development, provides several options for the solution of large matrix equations. Aim is a program that is able to determine up to 100 000 alignment parameters in a reasonable time on a standard PC.



---

1. Introduction

Purpose of instrument calibration: Instrument calibration is intended to **eliminate or reduce bias** in an instrument's readings over a range for all continuous values. For this purpose, **reference standards** with known values for selected points covering the range of interest are measured with the instrument in question. Then a **functional relationship** is established between the values of the standards and the corresponding measurements [. . . from NIST].

Alignment/calibration of HEP track detectors:

- based **only** on track residual minimization (*incomplete* data, several degrees of freedom undefined) and on survey data;
- no "reference standards with known values" exist ($\approx$ exceptions are data from $e^+e^- \to \mu^+\mu^-$ and cosmics without field).

Alignment/calibration requires to *understand* the detector (functional relationship) and to optimize thousands or ten thousands of parameters. Aim is – after an initial optical survey and corrections for electronics and mechanical effects:

- reduce χ^2 of the track fits, in order to improve track and vertex recognition, and
- increase precision of reconstructed tracks and vertices, eliminating or reducing bias in detector data.

. . . essential for important aspects of physics analysis with large accurate vertex detectors with potential precision of a few μm.

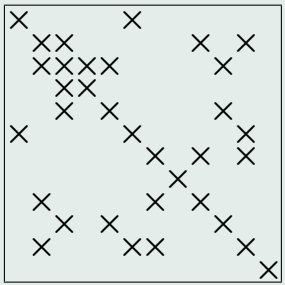
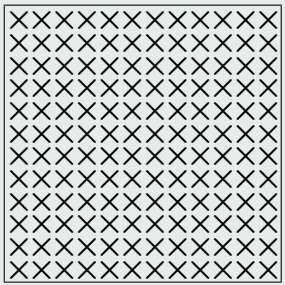
Optimization

Alignment as an optimization problem: The standard method for the determination of a large number of *correlated* parameters results is the definition of an objective function $F(\Delta \boldsymbol{p})$, and the minimization of the function by the solution of a large system of linear equations ($\boldsymbol{\Delta p}$ = corrections to alignment parameters)

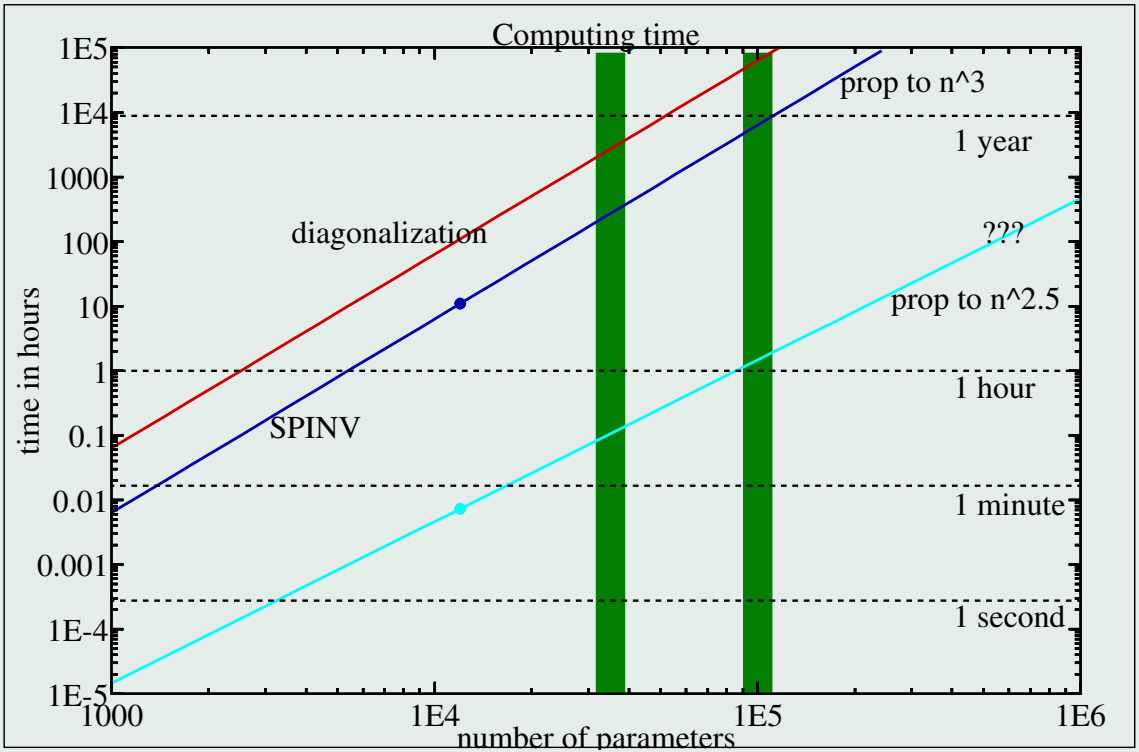
$$\boxed{\boldsymbol{C}\ \boldsymbol{\Delta p} = \boldsymbol{b}}$$

$\boldsymbol{C}$ is a sparse symmetric $n \times n$ matrix, with $n = \ldots 100\,000$

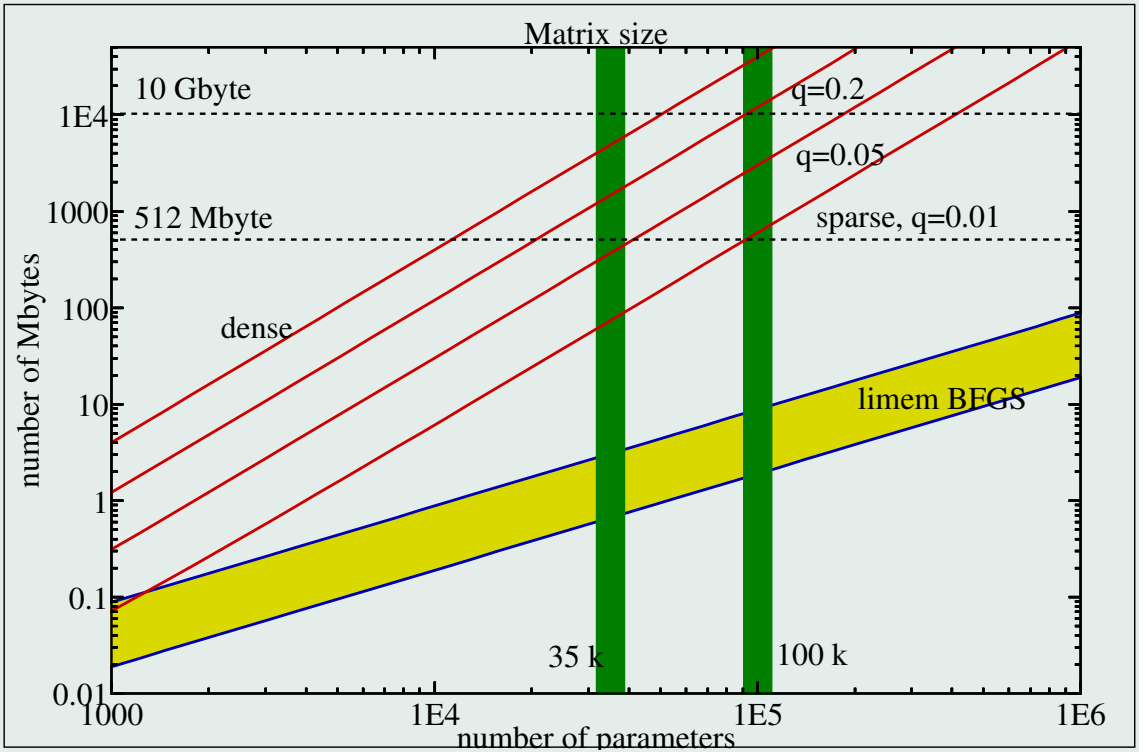
- Construction of correct matrix $\boldsymbol{C}$ and vector $\boldsymbol{b}$ from the data, or at least of a good approximation to the matrix $\boldsymbol{C}$ and vector $\boldsymbol{b}$, and ...
- correct solution of the large matrix equation, or at least a good approximation to the correct solution.

The inverse of a sparse matrix  is a dense matrix: 

Calculation of the – dense – matrix $\boldsymbol{C}^{-1}$ requires $(n^2 + n)/2$ words and a time $\propto n^3$ $\Rightarrow$ solution by inversion is (almost) impossible $\Rightarrow$ use other methods.



The two circles are points, measured with Millepede II for $n = 12\ 015$. The lower curve is for an iterative method, which requires only space for the sparse matrix $\boldsymbol{C}$.



q = fraction of non-zero off-diagonal elements

Note: printing the elements of a n-by-n matrix for 100 000 requires ≈ 10 m^3 paper (double-sided printing).

Direct solution: no iterations necessary (but inversion time $\propto n^3$)

Rate of convergence of iterative methods: $\{\boldsymbol{x}_k\}$ be sequence in $\mathbb{R}^n$ that converges to solution $\boldsymbol{x}^*$

linear convergence	$\dfrac{\|\|\boldsymbol{x}_{k+1}-\boldsymbol{x}^*\|\|}{\|\|\boldsymbol{x}_k-\boldsymbol{x}^*\|\|} \leq r$	$r \in (0,1)$
superlinear convergence	$\lim\limits_{k\to\infty} \dfrac{\|\|\boldsymbol{x}_{k+1}-\boldsymbol{x}^*\|\|}{\|\|\boldsymbol{x}_k-\boldsymbol{x}^*\|\|} = 0$	
quadratic convergence	$\dfrac{\|\|\boldsymbol{x}_{k+1}-\boldsymbol{x}^*\|\|}{\|\|\boldsymbol{x}_k-\boldsymbol{x}^*\|\|^2} \leq M$	M positive

for all k sufficiently large.

Speed of convergence depends on the eigenvalue spectrum – slow convergence for small eigenvalues.

Note: an iterative method can be faster than a direct method!

Rate of convergence

Linear convergence: Example is steepest-descent;

constant r may be close to 1; iteration numbers of 100 or 1000 not uncommon; 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 objective function:

$$\frac{||\boldsymbol{x}_{k+1} - \boldsymbol{x}^*||}{||\boldsymbol{x}_k - \boldsymbol{x}^*||} \leq r = \frac{\kappa - 1}{\kappa + 1} \qquad \kappa = \text{ Condition } = \frac{\lambda_{\max}}{\lambda_{\min}} \quad (\gg 1)$$

Often in applications a cut on small $||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k||$ is used, which means

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

Superlinear convergence: example is Quasi-NEWTON method;

Quadratic convergence: example is NEWTONs method;

roughly the number of exact digits double at each iteration; typically 3 to 5 iterations needed.

Least squares

$$\textbf{Minimize objective function} \qquad F(\boldsymbol{q}) = \tfrac{1}{2}\sum_{j=1}^{m} r_j^2(\boldsymbol{q})$$

The derivatives of $F(\boldsymbol{q})$ can be expressed in terms of the Jacobian of $\boldsymbol{r}$ (m-by-n matrix of first partial derivatives) defined by

$$\boldsymbol{J}(\boldsymbol{q}) = \left[\frac{\partial \boldsymbol{r}_j}{\partial q_i}\right]_{j=1,\ldots m;\, i=1,\ldots n}$$

Exploit the special structure of least squares:

$$\nabla F(\boldsymbol{q}) = \boldsymbol{J}(\boldsymbol{q})^{\mathrm{T}}\boldsymbol{r}(\boldsymbol{q}) \qquad \nabla^2 F(\boldsymbol{q}) = \boldsymbol{J}(\boldsymbol{q})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{q}) \left[+\sum_{j=1}^{m} r_j(\boldsymbol{q})\nabla^2 r_j(\boldsymbol{q})\right]$$

where the contribution in brackets [...] is usually neglected (Gauss-Newton method). Note that the full matrix $\boldsymbol{J}$ has never to be stored: one row at a time is sufficient.

For a linear problem (fixed first partial derivatives) only one step is necessary to determine $\boldsymbol{q}$:

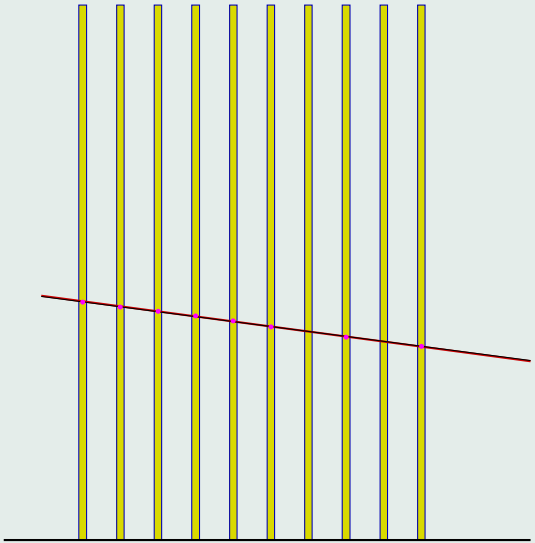
$$\underbrace{\left(\boldsymbol{J}(\boldsymbol{q})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{q})\right)}_{\text{symmetric matrix}}\boldsymbol{q} = \underbrace{\left(\boldsymbol{J}(\boldsymbol{q})^{\mathrm{T}}\boldsymbol{r}(\boldsymbol{q})\right)}_{\text{vector}}$$

Alignment problems are usually almost linear $\Rightarrow$ one least squares step may be sufficient ...
... but there may be other reasons for iterations.

2. Alignment of a toy detector

Test of alignment method with a MC toy track detector model:

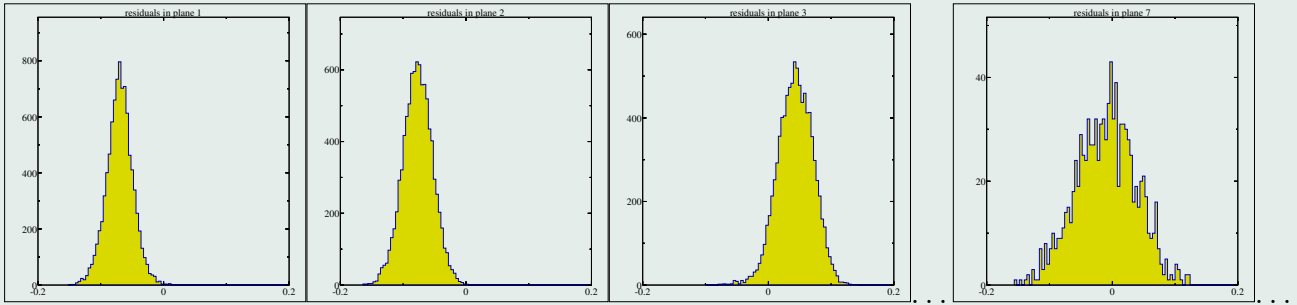
- 10 planes of tracking chambers, 1 m high, 10 cm distance, no magnetic field;
- accuracy $\sigma \approx 200\mu$m, with efficiency $\epsilon = 90\%$;
- plane 7 sick: accuracy $\sigma \approx 400\mu$m, with efficiency $\epsilon = 10\%$;
- 10 000 tracks with 82 000 hits available for alignment;
- Misalignment: the vertical position of the chambers are displaced by ≈ 0.1cm (normal distributed).



First attempt based on residuals

The first alignment attempt is based on the distribution of hit residuals:

- A straight line is fitted to the track data.
- The residuals (= measured vertical coordinate − fitted coordinate) are histogrammed, separately for each plane.



- The mean value of the residuals is taken as correction to the vertical plane position.

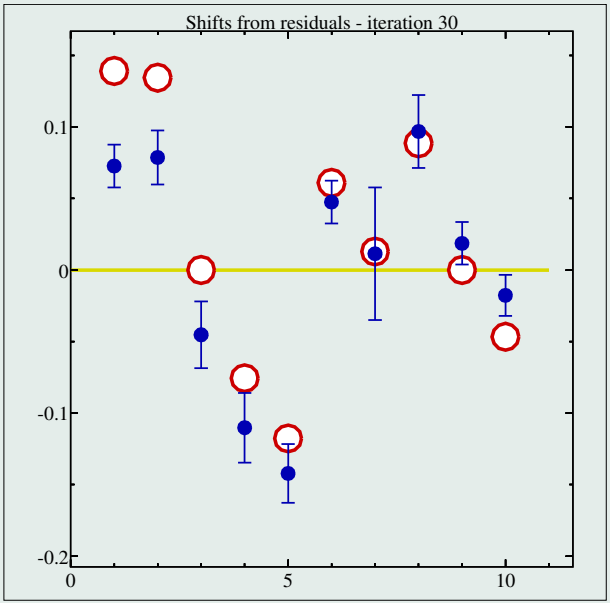
This is the standard method used in many experiments. What is the convergence behaviour?

Result from the first attempt

Large changes in first iteration, small changes in second iteration, almost no progress afterwards.

After 30 iterations ...

ID	true shift	determined	mean residual
1	0.1391	0.0727	0 ± 150
2	0.1345	0.0786	0 ± 189
3	0.0000	−0.0453	0 ± 234
4	−0.0756	−0.1102	0 ± 244
5	−0.1177	−0.1422	0 ± 205
6	0.0610	0.0475	0 ± 150
7	0.0130	0.0114	0 ± 464
8	0.0886	0.0968	0 ± 255
9	0.0000	0.0186	0 ± 149
10	−0.0467	−0.0176	0 ± 143



red circle = true shift (displacement)

blue disc = displacement, determined from residuum

First attempt – Discussion

The result is not (yet) encouraging!

The reason for non-convergence is simple:

> Two degrees of freedom are undefined: a simultaneous shift and a overall shearing of the planes!

(. . . this simple fact is not always mentioned in reports on the method.)

Improvement for second residual attempt:

> Fix the displacement (i.e. displacement = 0) of two planes, which are assumed to be carefully aligned externally (e.g. planes 3 and 9).

Other possibilities are:

- Use only fixed planes (planes 3 and 9) in the fit, and determine the residuals of other planes;
- for the determination of the displacement of a certain plane use all other planes in the fit.

These possibilities are in fact used by several collaborations!

Results from the second attempt

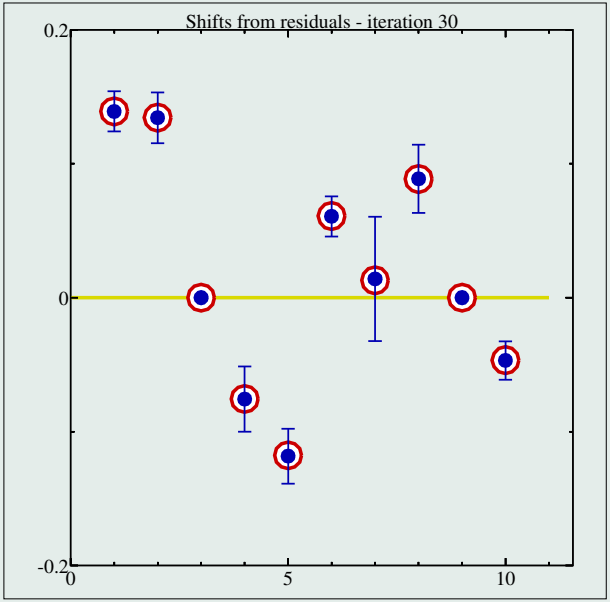
Large changes in first iteration, then many smaller and smaller changes: convergence is **linear** and slow, because the determination of displacements is based on biased fits.

After 30 iterations with planes 3 and 9 fixed (displacement = 0) ...

ID	true shift	determined	mean residual
1	0.1391	0.1391	-1 ± 150
2	0.1345	0.1344	0 ± 189
3	0.0000	0	2 ± 234
4	−0.0756	−0.0758	0 ± 244
5	−0.1177	−0.1183	0 ± 205
6	0.0610	0.0607	0 ± 150
7	0.0130	0.0140	0 ± 464
8	0.0886	0.0888	0 ± 255
9	0.0000	0	0 ± 149
10	−0.0467	−0.0469	0 ± 143

red circle = true shift (displacement)

blue disc = displacement, determined from residuum



Use of higher mathematics?

Residual-based methods work with biased results. Can the bias be avoided by an improved fit?

Yes: include the alignment parameters in the parameters fitted in track fits – requires a simultaneous fits of many tracks, with determination of (global) alignment parameters and (local) track parameters.

model: $y_i \cong a_1^{\text{local}} + a_2^{\text{local}} \cdot x_i \quad + a_j^{\text{global}}$ $\qquad a_j^{\text{global}}$ = shift for plane j, where y_i is measured

1 tracks	$2 + 10 = 12$ parameters	9 equations
2 tracks	$4 + 10 = 14$ parameters	18 equations
...	...	...
10 000 tracks	20 010 parameters	82 000 equations

... a linear least squares problem of $m = 82\,000$ equations (measurements) and $n = 20\,010$ parameters with $n \ll m$, which requires the solution of a matrix equation with 20010-by-20010 matrix.

The MILLEPEDE principle allows to reduce the least squares problem to $n = 10$ parameters!

Results from a simultaneous fit

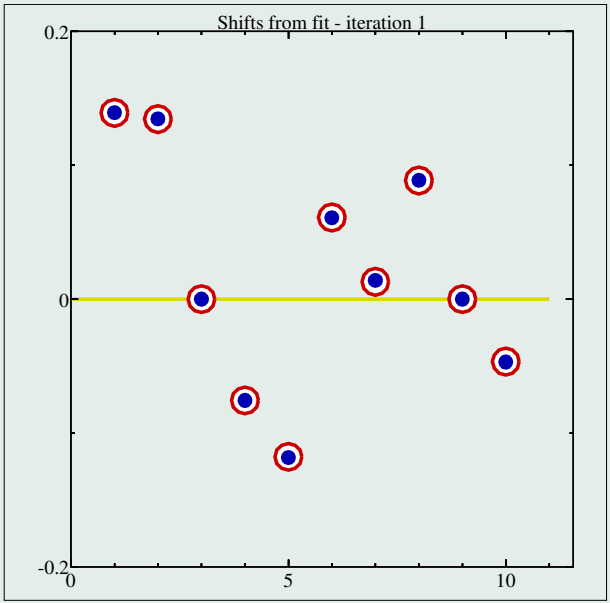
After one step (with planes 3 and 9 fixed at displacement = 0) ...

ID	true shift	determined	ρ	mean residual
1	0.1391	0.1393 ± 0.0004	0.68	0 ± 150
2	0.1345	0.1346 ± 0.0003	0.66	0 ± 189
3	0.0000			0 ± 234
4	-0.0756	-0.0756 ± 0.0003	0.58	0 ± 244
5	-0.1177	-0.1182 ± 0.0003	0.53	0 ± 205
6	0.0610	0.0608 ± 0.0003	0.50	0 ± 150
7	0.0130	0.0141 ± 0.0007	0.20	0 ± 464
8	0.0886	0.0888 ± 0.0003	0.53	0 ± 255
9	0.0000			0 ± 149
10	-0.0467	-0.0469 ± 0.0003	0.57	0 ± 143

(ρ = global correlation coefficient)

red circle = true shift (displacement)

blue disc = displacement, determined in fit



One step is sufficient: 1. step $\Delta\chi^2 = 1.277 \times 10^6$ 2. step $\Delta\chi^2 = 1.159 \times 10^{-5}$

Determination of drift velocities ... 10 additional parameters

Improvement: include, in addition, corrections to the drift velocities for each plane: $\Delta v_{\text{drift}}/v_{\text{drift}}$

$$y_i \cong a_1^{\text{local}} + a_2^{\text{local}} \cdot x_i \quad + a_j^{\text{global}} + \ell_{\text{drift},i} \cdot \left(\frac{\Delta v_{\text{drift}}}{v_{\text{drift}}}\right)_j$$

a_j^{global} = shift for plane j

$\left(\frac{\Delta v_{\text{drift}}}{v_{\text{drift}}}\right)_j$ = relative v_{drift} difference

reduction of residual σ by 30 - 40 %

ID	true shift	determined	ρ	$\Delta v_{\text{drift}}/v_{\text{drift}}$	determined	ρ	mean residual
1	0.1391	0.1393 ± 0.0004	0.68	0.0020	0.0019 ± 0.0002	0.016	0 ± 119
2	0.1345	0.1346 ± 0.0003	0.66	-0.0153	-0.0150 ± 0.0002	0.020	0 ± 128
3	0.0000			0.0193	0.0194 ± 0.0002	0.017	0 ± 137
4	-0.0756	-0.0756 ± 0.0003	0.58	0.0200	0.0197 ± 0.0002	0.013	0 ± 139
5	-0.1177	-0.1182 ± 0.0003	0.53	-0.0138	-0.0136 ± 0.0002	0.013	0 ± 141
6	0.0610	0.0608 ± 0.0003	0.50	0.0003	0.0004 ± 0.0002	0.019	0 ± 139
7	0.0130	0.0141 ± 0.0007	0.20	-0.0306	-0.0303 ± 0.0006	0.038	0 ± 348
8	0.0886	0.0888 ± 0.0003	0.53	0.0237	0.0238 ± 0.0002	0.018	0 ± 134
9	0.0000			-0.0044	-0.0044 ± 0.0002	0.008	0 ± 127
10	-0.0467	-0.0469 ± 0.0003	0.57	0.0021	0.0019 ± 0.0002	0.013	0 ± 117

... this would be rather difficult with a pure residual-based method.

The next improvement would be the introduction of wire T_0's – additional $10 \times 25 = 250$ parameters.

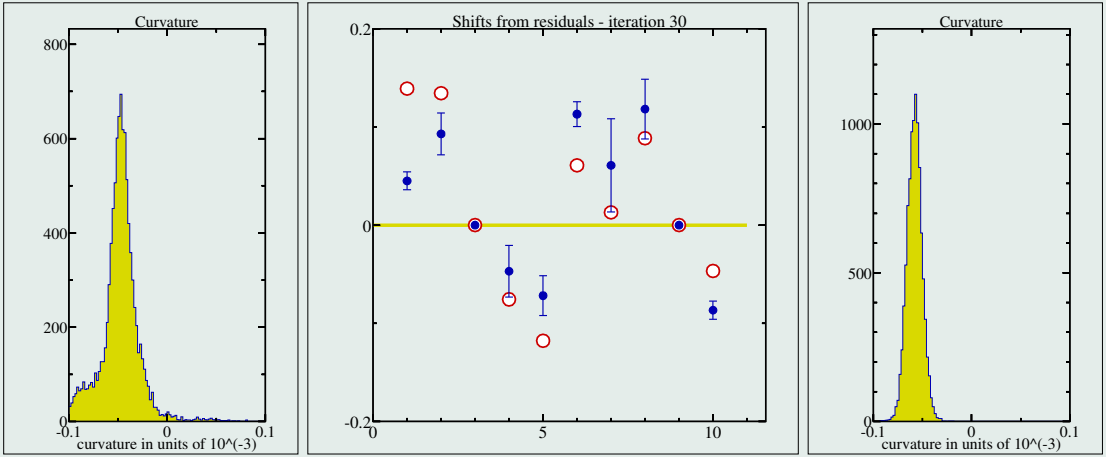
A more realistic scenario

So far the tracks are fitted with a straight line. Now a third parameter is added to the parameterization and a parabola is fitted (i.e. case of unknown momentum):

$$\text{model:} \qquad y_i \cong a_1^{\text{local}} + a_2^{\text{local}} \cdot x_i + a_3^{\text{local}} \cdot x_i^2$$

Th initial misalignment will create a curvature $\neq 0$ – the data are now insufficient to determine the true shifts.

Alignment by residual method



fitted curvature before ... ... after aligment

3. Millepede I

Determination of corrections $\boldsymbol{\Delta p}$ for alignment parameters $\boldsymbol{p}$ is based on minimization of residuals – the difference between fitted and measured track position:

$$\Delta_i = \text{ fitted value } - \text{ measured value}$$

A "global" objective function $F(\boldsymbol{\Delta p}, \boldsymbol{q})$ or χ^2-function is constructed, which depends on the corrections $\boldsymbol{\Delta p}$ and **all** track parameters $\boldsymbol{q}$

$$F(\boldsymbol{\Delta p}, \boldsymbol{q}) \equiv \chi^2(\boldsymbol{\Delta p}, \boldsymbol{q}_j) = \sum_{\text{data sets}} \left(\sum_{\text{events}} \left(\sum_{\text{tracks}} \left(\sum_{\text{hits}} \Delta_i^2/\sigma_i^2 \right) \right) \right)$$

Data sets are

- Physics data: track data from e^+e^--, $e^-\,p$-, $p\,p$-reactions,
- Cosmics with magnetic field (large distance to IP) and without magnetic field (straight tracks, curvature zero),
- vertex- or mass-constrained track data,
- external alignment data.

A mixture of different data is recommended, in order to introduce different correlations between the alignment parameters and to increase the precision.

Minimization requires first derivatives of hit residuals w.r.t. the parameters – *linearization*. Residuals and derivatives are used to determine matrix and vector of the so-called normal equations of least squares:

$$\left(\begin{array}{c|ccc} \boldsymbol{C} & \dots & \boldsymbol{H}_k & \dots \\ \hline \dots & \dots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{H}_k^{\mathrm{T}} & \mathbf{0} & \boldsymbol{C}_k^{\mathrm{track}} & \mathbf{0} \\ \dots & \mathbf{0} & \mathbf{0} & \dots \end{array}\right) \times \left(\begin{array}{c} \boldsymbol{\Delta p} \\ \hline \dots \\ \Delta \boldsymbol{q}_k^{\mathrm{track}} \\ \dots \end{array}\right) = \left(\begin{array}{c} \boldsymbol{b} \\ \hline \dots \\ \boldsymbol{b}_k^{\mathrm{track}} \\ \dots \end{array}\right) \qquad k = \text{ track index}$$

which defines the alignment corrections $\boldsymbol{\Delta p}$ and all track parameters $\boldsymbol{q}_k$ from the data used in the fit

The size of the huge matrix depends on the number of alignment parameters **and** on number of tracks.

Note: ignoring the alignment parameters, the (biased) track parameters are determined from the matrix equation

$$\boldsymbol{C}_k^{\mathrm{track}}\, \Delta \boldsymbol{q}_k^{\mathrm{track}} = \boldsymbol{b}_k^{\mathrm{track}} \qquad \text{with cov. matrix} \quad \boldsymbol{V}_k^{\mathrm{track}} = \left(\boldsymbol{C}_k^{\mathrm{track}}\right)^{-1}$$

with a small matrix and vector – or ignore track fits and calculate

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

The Millepede principle allows to find the solution of the huge equation, because the sparse structure of the matrix allows a reduction to the matrix size for the alignment parameters, using the results from the track fits and the rectangular *mixed* matrices $\boldsymbol{H}_k$.

Normal equations

Simultaneous least squares fit of all global and all local parameters (i.e. all tracks).

$$k'\text{th track:} \qquad y_i \cong f(x_i; \boldsymbol{p}^{\text{global}}, \boldsymbol{q}^{\text{local}}) + \left(\boldsymbol{d}_i^{\text{global}}\right)^{\text{T}} \Delta\boldsymbol{p}^{\text{global}} + \left(\boldsymbol{\delta}_i^{\text{local}}\right)^{\text{T}} \Delta\boldsymbol{q}_k^{\text{local}}$$

The complete matrix equation for global and local parameters includes sums over track index k and contains many matrices: n-by-n matrices $\boldsymbol{C}$ for n global parameters and m-by-m matrices $\boldsymbol{C}_k^{\text{local}}$ and n-by-m matrices $\boldsymbol{H}_k^{\text{global-local}}$

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

If the $\boldsymbol{H}_k^{\text{global-local}}$ are neglected, the complete equation decays into $1 + K$ independent matrix equations.

$\Delta\boldsymbol{p}^{\text{global}}$ can be calculated without approximation with a great simplification: $\Rightarrow$

For each track in a loop, on all tracks:

1. **Track- or other fit:** perform fit by finding the best local parameter values for the actual track until convergence with determination of the covariance matrix $\boldsymbol{V}_k$ of the local parameters

2. **Derivatives:** calculate for all hits (index i) the vectors of derivatives $\boldsymbol{\delta}_i^{\text{local}}$ and $\boldsymbol{d}_i^{\text{global}}$ for all local and the relevant global parameters, and update matrices:

$$\boldsymbol{C} := \boldsymbol{C} + \sum_i w_i \boldsymbol{d}_i^{\text{global}} \left(\boldsymbol{d}_i^{\text{global}}\right)^{\text{T}} \qquad \boldsymbol{b} := \boldsymbol{b} + \sum_i w_i r_i \boldsymbol{d}_i^{\text{global}} \qquad \boldsymbol{H}_k = \sum_i w_i \boldsymbol{d}_i^{\text{global}} \left(\boldsymbol{\delta}_i^{\text{local}}\right)^{\text{T}}$$

and finally for the track $\boldsymbol{C} := \boldsymbol{C} - \boldsymbol{H}_k \boldsymbol{V}_k \boldsymbol{H}_k^T \qquad \boldsymbol{b} := \boldsymbol{b} - \boldsymbol{H}_k \left(\boldsymbol{V}_k \boldsymbol{b}_k\right)$

The 'blue' equations transfer the 'local' information to the global parameters.

After the loop on all tracks the complete information is collected; now the matrix equation for the global parameters has to be solved:

solve $\boldsymbol{C}\Delta\boldsymbol{p}^{\text{global}} = \boldsymbol{b}$ for $\Delta\boldsymbol{p}^{\text{global}}$ e.g. by $\Delta\boldsymbol{p}^{\text{global}} = \boldsymbol{C}^{-1}\,\boldsymbol{b}$

Note: matrices $\boldsymbol{C}$ and vectors $\boldsymbol{b}$ from several data sets can be simply added to get combined result.

Numerical linear algebra

Determination of *good* solution for $\boldsymbol{\Delta p}$ in

$$\boxed{\boldsymbol{C}\,\boldsymbol{\Delta p} = \boldsymbol{b}} \qquad \boldsymbol{C}^{-1} = \boldsymbol{V} = \text{ covariance matrix for parameters}$$

with large or huge $n \times n$ matrix depends on **algorithm** and **data**, and requires

0. **Scaling:** Use consistent units in data and variables – matrix and vector should be well scaled, i.e. elements should have similar precision.

1. **Algorithm – Stability:** With a stable algorithm the computed solution is the exact solution of a nearby problem. Gaussian Elimination with diagonal pivoting (restricted scan for largest next element) is considered to be a stable algorithm for positive definite matrices.

2. **Data – Conditioning:** Data are called ill-conditioned, if small changes in the data can cause large changes in the solution ($\rightarrow$ small eigenvalue(s)). Matrix is ill-conditioned if variables are undefined or poorly defined or strongly correlated.

Solution by matrix inversion — for n up to 5000

Standard method for solution of $\boldsymbol{C}\Delta\boldsymbol{p} = \boldsymbol{b}$ with symmetric matrix is stable Gauss algorithm with pivot selection of diagonal, with

$$\text{Computing time} \;=\; \text{constant} \times n^3 \qquad\qquad < 1 \text{ hour for } n = \text{ several thousand}$$

(in reality "constant" increases for large n).

A standard matrix routine will fail – at least a few parameters out of many thousands will be badly defined. – Matrix is destroyed without result for $\Delta\boldsymbol{p}$.

Subroutine SPMINV (in Millepede) for symmetric matrices in $(n^2+n)/2$ words: choose largest pivot, but stop inversion if no acceptable pivot found, i.e. invert largest possible submatrix; return zero corrections for remaining parameters.

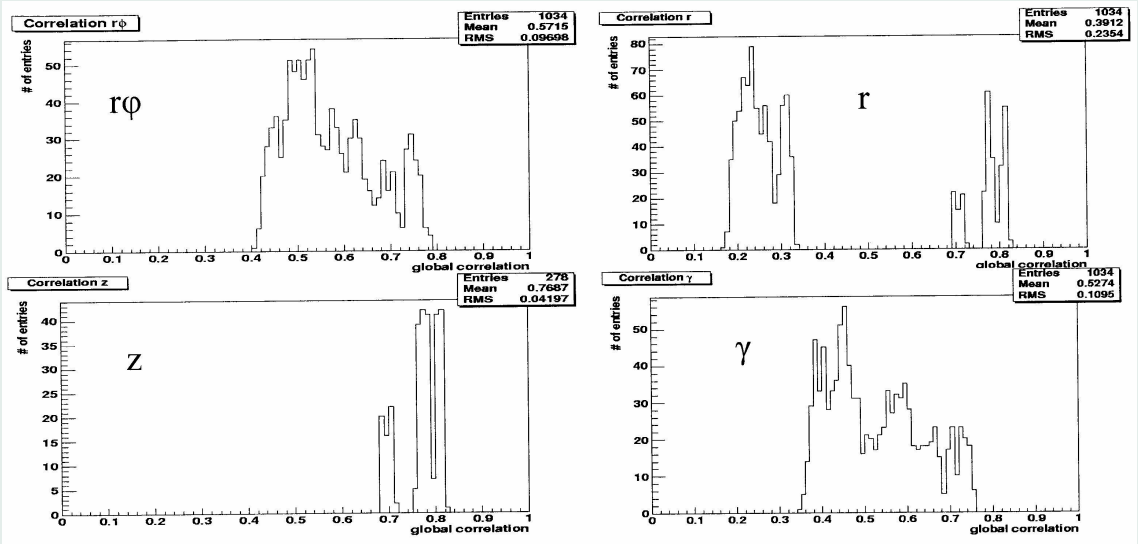
All variances and covariances available in inverse matrix.

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.

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

Global correlations

Range of global correlation coefficient is $0 \ldots 1$.



Values ≈ 1 means strong correlation and almost singular matrix – inversion may be impossible.

Values depend on geometry and type of data – additional data (cosmics, vertex constrained tracks) can reduce the global correlation and improve the alignment.

Undefined degrees of freedom ... or weakly defined degrees of freedom

Alignment of HEP track detectors ... if based **only** on track residual minimization: *incomplete* data, with several degrees of freedom undefined! Certain parameters are **undefined** or only weakly defined and could distort the detector.

General **linear** transformation of whole detector with translation and 3×3 matrix $\boldsymbol{R}$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} + \boldsymbol{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

defined by $3 + 9$ parameters, will not affect the χ^2 of the fits. The matrix can be decomposed into

- three rescaling factors of coordinate axes: f_x, f_y, f_y,
- three rotations: $\mathcal{D}_x$, $\mathcal{D}_y$, $\mathcal{D}_z$ and three shearings: $\mathcal{T}_{xz}$, $\mathcal{T}_{yz}$, $\mathcal{T}_{xy}$.

In addition there may be weakly defined **nonlinear** transformations (bend).

These degrees of freedom can be fixed

- by mixture of different data and by external measurements $\rightarrow$ hardware alignment devices, i.e. alignment by tracks has to be supplemented by external information, or fixed
- by equality constraints or by orthogonalization methods.

Equality constraints

Undefined degrees of freedom can be fixed by adding equality constraint equations of the type

$$g(\boldsymbol{p}) = 0 \qquad \text{e.g.} \quad d_x = \sum_i \Delta x_i = 0$$

e.g. "zero average displacement", or "zero rotation of the whole detector".

There are several possibilities:

- fix certain parameters (e.g. planes), or
- fix linear combinations of parameters (after diagonalization), or
- add equality constraint equation, i.e. append linearized Lagrange multiplier equation $\lambda\left(g(\boldsymbol{p}) + \boldsymbol{g}^{\mathrm{T}} \cdot \Delta\boldsymbol{p} = 0\right)$ with $\boldsymbol{g} = \partial g(\boldsymbol{p})/\partial \boldsymbol{p}$:

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

Notes: problem with Lagrange function can be solved by matrix inversion, but matrix is no longer positive definit. Alternative is penalty function ($\ldots + |g(\boldsymbol{p})|^2$) or combination of both (augmented Lagrangian).

Parameters of a Vertex detector

Planar sensors (silicon pixel or strip detectors): local (sensor) coordinates $\boldsymbol{q} = (u, v, w)$ and global detector coordinates $\boldsymbol{r} = (x, y, z)$ are transformed by

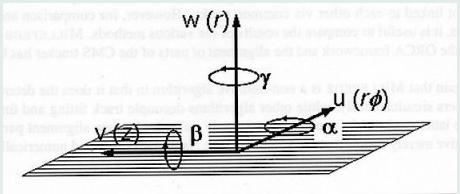
$$\boldsymbol{q} = \boldsymbol{R}\,(\boldsymbol{r} - \boldsymbol{r}_0) \qquad \boldsymbol{R} = \text{ nominal rotation}, \quad \boldsymbol{r}_0 = \text{ nominal position}$$

After alignment the transformation becomes (with $\Delta\boldsymbol{q} = (\Delta u, \Delta v, \Delta w)$)

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

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

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



where the approximation has sufficient accuracy for small angles α, β and γ.

Six parameters are required for each individual detector element, out of which **three** parameters (two translations, one rotation) are very sensitive.

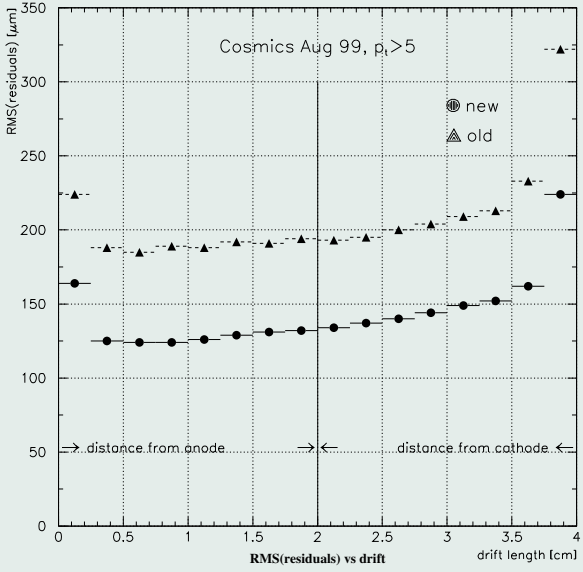
Note: Number of parameters per sensor can be reduced $\Rightarrow$ partially separable functions.

Driftchamber alignment

Claus Kleinwort et al., Detailed calibration of H1 drift chamber using MILLEPEDE with about 1400 parameters.

1400 parameters using 50 000 tracks:

- *common* alignment of the drift chamber and the silicon detector;
- for both CJC1 and CJC2 14 global parameters representing an overall shift or tilt are introduced;
- local variations of the drift velocity v_{drift} for cells halfs and layers halfs are observed, which are parametrized by 180 + 112 corrections, which change with the HV configuration;
- for each wire group (8 wires) corrections to T_0 are introduced (330 corrections).



Reduction of residual RMS by MILLEPEDE.

No problems with undefined degrees of freedom for parameters like T_0 or v_{drift} corrections.

First development of Millepede principle (reduction of matrix) 1996 . . . and used in H1 1997 . . .

V. Blobel: Experience with Online Calibration Methods, Contribution to CHEP'97, Berlin 1997 (including the **Millepede** principle), not accepted.

MILLEPEDE design: experiment-independent program, with well-defined interface to experiment-dependent data.

Available on web page `http://www.desy.de/~blobel`
V. Blobel, Linear Least Squares Fits with a Large Number of Parameters, (2000), 22 pages
and full Fortran code.

V. Blobel and C. Kleinwort: A New Method for the High-Precision Alignment of Track Detectors, PHYSTAT2002, Durham, `arXiv-hep-ex/0208021`

Used (or under test) by H1(1997), CDF(2001), HERA-b, ZEUS, CMS, ATLAS, LHC-b . . . ??? and rewritten in C++ several times (unpublished).

Principle of reducing matrix size (perhaps) used earlier:
Schreiber, O. (1877): Rechnungsvorschriften für die trigonometrische Abteilung der Landesaufnahme, Ausgleichung und Berechnung der Triangulation zweiter Ordnung. Handwritten notes. Mentioned in W. Jordan (1910): Handbuch der Vermessungskunde, Sechste erw. Auflage, Band I, Paragraph III: 429-433. J.B.Metzler, Stuttgart.

4. Millepede II

Start of development in May 2005 after discussions with Hamburg cms group, with aim:

- alignment with up to 100 000 parameters in a reasonable time on a standard PC;
- keep Millepede principle (simultaneous fit of arbitrary number of tracks and alignment parameters);
- allow different (direct and iterative) methods for the solution of large matrix equation, using mathematical methods from the (mathematical) community (literature) (not home-made iterative methods);
- design with even stronger separation of experiment-dependent program and the Millepede alignment;
- automatic recognition of existing alignment parameters, allowing suppression of parameters with too few data;
- constraints as equality constraints, or like measurements.

Test by PhD student (cms) since summer 2005. Not yet all options realized.

Millepede $\Rightarrow$ Mille + Pede

Mille: small C++ or Fortran routine, called within the experiments event-processing program

Pede: stand-alone experiment-independent alignment program, with many options and input files.

Mille Event loop: write alignment information (derivatives, hits, ...) to special file

Pede Normal equation loop: read data and text files and form $\boldsymbol{C}$ and $\boldsymbol{b}$

Solution of matrix equation: direct or iterative

solve $\boldsymbol{C}\boldsymbol{\Delta p} = \boldsymbol{b}$ for $\boldsymbol{\Delta p}$

End of solution

End of normal equation loop

End of event loop

12 h for event loop, $\frac{1}{2}$ h for ... PEDE II

Event loop: only once to extract the data (more than once, if large non-linearities)

Normal equation loop: once (or repeated, if outlier suppression necessary or for L-BFGS)

Solution of matrix equation: inversion direct ($n < 5000$), or iteratively

Overview over methods

Pede performs (like Millepede I) fits to single e.g. tracks, and generates the vector and matrix of the normal equations in different formats.

No single optimal method, different methods for different conditions (number of parameters, sparsity):

Matrix inversion: • same routine as in MP I, for up to 5 000 parameters, with time $\propto n^3$;

Diagonalization: • slower than inversion, allows to recognize insignificant linear combinations (no constraints necessary);

Sparse matrix storage: • allows to store big sparse matrices

Generalized minimal residual method: • fast method for large sparse matrices, factor 5 000 faster than inversion for n =12 000. Routines MINRES • (and SYMMLQ ⊖);

Preconditioning: ⊖ allows to reduce number of iterations, possible in MINRES (and SYMMLQ);

Band matrix decomposition: ⋆ fast (time $\propto m^2 \cdot n$), but only approximation (but useful for preconditioning?); bordered band matrix possible too;

Limited memory BFGS: ⋆ uses only *virtual* matrix, low space requirement, but many iterations(?);

Partially separable functions: ⋆ *trick*, to reduce number of parameters (but more iterations).

Code: • =included in MILLEPEDE II, ⋆ = prepared ⊖ = not yet tried.

Solution by matrix inversion — for n up to 5000

Standard method for symmetric matrices is stable Gauss algorithm with pivot selection of diagonal, with

$$\text{Computing time} \;=\; \text{constant} \times n^3 \qquad < 1 \text{ hour for } n = \text{ several thousand}$$

(in practice "constant" increases for large n).

A standard matrix routine will fail – at least a few parameters out of many thousands will be badly defined.

Subroutine SPMINV (in Millepede) for symmetric matrices in $(n^2+n)/2$ words: choose largest pivot, but stop inversion if no acceptable pivot found, i.e. invert largest possible submatrix; return zero corrections for remaining parameters.

All variances and covariances available in 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.

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

Solution by diagonalization

The diagonalization of the symmetric matrix $\boldsymbol{C} = \boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$ allows to recognize singularity or near singularity by the determination of eigenvalues, and to suppress corresponding linear combinations of parameters.
Algorithms are iterative, computing time ≈ 10 times larger compared to inversion, and solution less precise.

$$\boldsymbol{C} = \boldsymbol{U}\ \boldsymbol{D}\ \boldsymbol{U}^{\mathrm{T}} \qquad \text{Diagonalization of symmetric matrix}$$

with $\boldsymbol{D}$ diagonal, $\boldsymbol{U}$ square and orthogonal with $\boldsymbol{U}\,\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{U}^{\mathrm{T}}\,\boldsymbol{U} = \mathbf{1}$. Note: $\boldsymbol{C}^{-1} = \boldsymbol{U}\ \boldsymbol{D}^{-1}\ \boldsymbol{U}^{\mathrm{T}}$

eigenvalue ordering in $\boldsymbol{D} = [\mathrm{diag}\,(\lambda_i)]$: $\quad \lambda_1 \ \geq\ \ldots \geq \lambda_k \geq \lambda_{k+1} = \ldots \lambda_n = 0$ (or very small)

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

with $1/\lambda_i = 0$ for $\lambda_i = 0$ or small q_i with $|q_i| \lesssim 1$

$\Rightarrow$ Suppression of insignificant linear combinations, which could produce distortions of the detector.

Solution by singular value decomposition

Singular value decomposition avoids the formation of normal equations ($\boldsymbol{C} = \boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$) and is numerically more accurate than normal-equation methods.

$$\boldsymbol{J} = \boldsymbol{V}\ \boldsymbol{D}\ \boldsymbol{U}^{\mathrm{T}} \qquad \text{Singular value decomposition (SVD) for } m \times n \text{ matrix } \boldsymbol{J}$$

with $\boldsymbol{D}$ diagonal, $\boldsymbol{U}$ square and orthogonal with $\boldsymbol{U}\,\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{U}^{\mathrm{T}}\,\boldsymbol{U} = \mathbf{1}$ and $m \times n$ matrix $\boldsymbol{V}$ column-orthogonal with $\boldsymbol{V}^{\mathrm{T}}\,\boldsymbol{V} = \mathbf{1}$. Diagonal elements σ_i of $\boldsymbol{D}$ are called singular values, with $\sigma_i^2 = \lambda_i$.

$$\text{Solution of } \boxed{\min \|\boldsymbol{J}\,\boldsymbol{\Delta p} - \boldsymbol{r}\|_2} \text{ by } \qquad \boldsymbol{\Delta p} = \boldsymbol{U}\ \left[\mathrm{diag}\left(\frac{1}{\sigma_i}\right)\right](\boldsymbol{V}^{\mathrm{T}}\boldsymbol{r})$$

with $1/\sigma_i = 0$ for $\sigma_i = 0$ or close to 0.

$\Rightarrow$ Suppression of insignificant linear combinations, which could produce distortions of the detector.

Advantage: numerically accurate due to small condition number $\kappa(\boldsymbol{D}) = \sqrt{\kappa(\boldsymbol{J}^{\mathbf{T}}\boldsymbol{J})}$
Disadvantage: requires to store huge $m \times n$ matrices $\boldsymbol{J}/\boldsymbol{V}$ and large $n \times n$ matrix $\boldsymbol{U}$, and can **not** be used (in this form) in large alignment problems. SVD can be applied to the matrix $\boldsymbol{C} = \boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$; then it is equivalent to diagonalization.

Sparse matrix storage

Large matrices are usually *sparse*, with small fraction of non-zero off-diagonal elements (fraction called q). Mathematical methods for the solution of matrix equations exist, which only require the product of the matrix with vectors.

The indexed storage scheme of `PCGPACK`, modified for a symmetric matrix, is used: it requires arrays with

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

and is optimized for the product (9 lines of code).
During matrix generation (sums): a (binary) search is necessary to find the location for an index pair (i, j)

The automatic generation of parameter-index relations requires a large number of comparisons.

Example: For 100 000 parameters and 1 Mio tracks, each 20 hits about $2q \times 10^{19}$ sequential comparisons would be necessary, for $q = 1\%$ this corresponds to 6 cpu-years (1 nanosec/comparison).

A faster method using a combination of hashing, sorting and binary search is used.

Generalized minimal residual method (GMRES)

Solution of a very large system of linear equations with sparse matrix, by an clever solution of a quadratic minimization problem, in analogy to the method of conjugate gradients (Hestenes, Stiefel 1952).

Example: MINRES (M. A. Saunders), designed to solve

system of linear equations $\boxed{\boldsymbol{C}\,\boldsymbol{\Delta p} = \boldsymbol{b} \qquad \text{or} \qquad \min ||\boldsymbol{C}\,\boldsymbol{\Delta p} - \boldsymbol{b}||_2}$

where $\boldsymbol{C}$ is a symmetric matrix of logical size $n \times n$, which may be indefinite and/or singular, very large and sparse. It is accessed *only* by means of a subroutine call

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

for any given vector $\boldsymbol{x}$.

Example of **compact storage**: row-index sparse storage *)

For $n =$ 100 000 and 512 Mbytes memory: $q = 1\%$
5 Gbytes memory: $q = 10\%$

C. C. Paige and M. A. Saunders (1975), Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12(4), pp. 617-629.

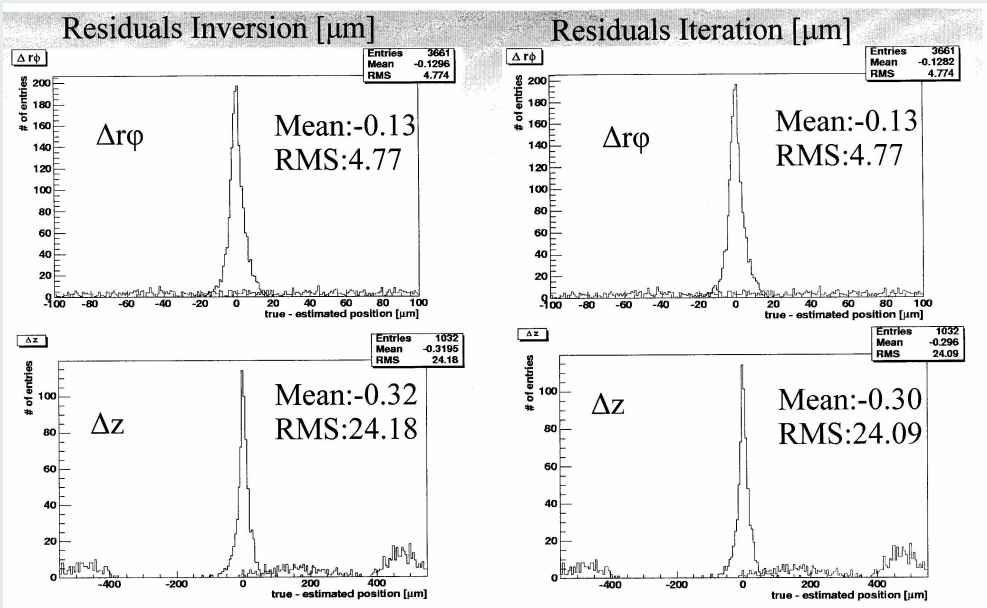
`www.stanford.edu/group/SOL/software/minres.html`

*) W. H. Press, S. A. Teukolsky, W. T. Vetterling, B.P. Flannery, Numerical Recipes – The Art of Scientific Computing, Cambridge Univ.

Comparison

For 12 000 parameters

- matrix inversion (cpu-time 12 h, 46 min, 5 s), and
- iterative solution with MINRES (cpu-time 32 s).



Example: 250 000 tracks, 500 Mbytes file, 4 400 variable parameters, 3.1 % non-zero off-diagonal elements (plot ⇒ ps-file), 100 sec for preparation, and 100 sec per iteration (250 000 track fits + solution).

Preconditioning

The convergence rate of iterative methods depends on the spectral properties of the matrix.

A transformation may improve the spectral properties:

$$\begin{aligned} &\text{instead of} && \boldsymbol{C}\,\Delta\boldsymbol{p} = \boldsymbol{b} \\ &\text{solve} && \boldsymbol{M}^{-1}\boldsymbol{C}\,\Delta\boldsymbol{p} = \boldsymbol{M}^{-1}\boldsymbol{b} \qquad \text{for } \Delta\boldsymbol{p}\,, \end{aligned}$$

which has the same solution as the original system, but the condition number of $\boldsymbol{M}^{-1}\boldsymbol{C}$ may be smaller.

The matrix $\boldsymbol{M}^{-1}$ should be an approximation to $\boldsymbol{C}^{-1}$, such that $\boldsymbol{M}^{-1}\boldsymbol{C} \approx \mathbf{1}$.

One possibility: band portion of the inverse of the band matrix approximation of matrix $\boldsymbol{C}$.

Preconditioning is an option in `MINRES` by means of a subroutine call

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

without altering vector $\boldsymbol{x}$.

Cholesky decomposition

Decomposition of the symmetric matrix $\boldsymbol{C}$

$$\boldsymbol{C} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}} \qquad \text{decomposition}$$

is "numerically extremely stable", and can be made *in-space*. Matrix $\boldsymbol{L}$ is a left unit triangular matrix (diagonal elements =1) and $\boldsymbol{D}$ is a diagonal matrix (less stable for semi-definite matrix).

Solution of $\boldsymbol{C}\boldsymbol{\Delta p} = \boldsymbol{L}\left(\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{\Delta p}\right) = \boldsymbol{b}$ by

$$\boldsymbol{L}\boldsymbol{v} = \boldsymbol{b} \quad \text{forward substitution} \qquad \boldsymbol{L}^{\mathrm{T}}\boldsymbol{\Delta p} = \boldsymbol{D}^{-1}\boldsymbol{v} \quad \text{forward substitution}$$

With clever ordering (?) of parameters the matrix C can be approximated by a band matrix.

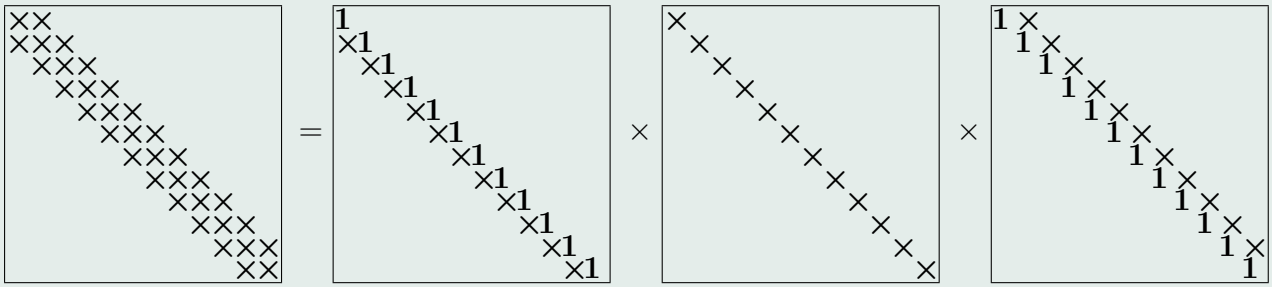
Band matrices with band-width m: the band structure is kept in this decomposition and computing time for solution is only $\propto n \times m^2$. Selected elements of the inverse matrix (corresponding to the band of the original matrix $\boldsymbol{C}$) can be calculated quickly.

Fast methods exist also for
Variable-bandwidth matrices ('sky-line" matrix), and for
Bordered band matrices ('arrow" matrix), where e.g. border (additional full rows and columns) is due to Lagrange multiplier constraints.

Decomposition is similar to the original Cholesky factorization; the introduction of the diagonal matrix $\boldsymbol{D}$ avoids the square-root operations.

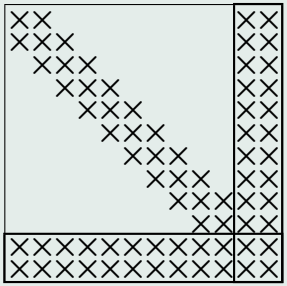
Band matrix decomposition

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



Note: the inverse of a bandmatrix is a full dense matrix, but
the bandpart of the inverse can be calculated fast.

In reality there will be *bordered* band matrices: $\Longrightarrow$ even for those cases there are fast *direct* methods for the solution – and for the determination of elements of the inverse matrix.



Quasi-Newton methods ...

... require only the gradient of the objective function $F(\boldsymbol{p})$ at each iteration. New step $\Delta\boldsymbol{p}$ for line search of $F(\boldsymbol{p} + \alpha \cdot \Delta\boldsymbol{p})$ is calculated from

$$\boldsymbol{H}\,\Delta\boldsymbol{p} = \nabla F \qquad \text{or} \qquad \boldsymbol{B}\,\nabla F = \Delta\boldsymbol{p} \qquad \text{with } \boldsymbol{B} = \boldsymbol{H}^{-1}$$

with approximate Hessian $\boldsymbol{H}$ or inverse Hessian $\boldsymbol{B}$.

Starting from a simple assumption (e.g. $\gamma\mathbf{1}$) the Hessian $\boldsymbol{H}$ or the 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 $\boldsymbol{H}_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k$ or $\boldsymbol{B}_{k+1}\boldsymbol{y}_k = \boldsymbol{s}_k$, the most-popular update formula is

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

To minimize $F(\boldsymbol{p} + \alpha \cdot \Delta\boldsymbol{p})$ the line-search has to satisfy the Wolfe (or strong Wolfe) conditions.

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 $\boldsymbol{B}$.

Quasi-Newton method, a revolutionary idea, invented by physicist W.C. Davidon (Argonne) in the mid 1950s; paper not accepted for publication. Different update formulas were developed during the following 20 years.

Limited memory BFGS

One step of the BFGS method has the form

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k - \alpha \cdot \boldsymbol{B}_k \nabla F_k \qquad \text{(line search, } \alpha \text{ usually close to 1)}$$

The limited memory BFGS (short: L-BFGS) method uses update information only from most recent iterations.
With $\boldsymbol{B}_0 = \gamma \mathbf{1}$ the product $\boldsymbol{B}_k \nabla F_k$ can be evaluated from the last m difference vectors $\boldsymbol{y}_k$, $\boldsymbol{s}_k$, a "memoryless" BFGS method.

- *Good* values for m are in the range $3 \ldots 20$;
- the storage requirement with $n\,(2m+4)$ is linear in n, and
- $\approx 3/2\, m^2 n$ operations are needed per iteration.
- The rate of convergence is *linear*.

. . . an optimization method for $n \gg 100\,000$ parameters ?

J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Mathematics of Computation **35** (1980), 773 - 782.
D.C. Liu and J. Nocedal, *On the limited-memory BFGS method for large scale optimization*, Mathematical Programming **45** (1989) 503 - 528

Partially separable functions

In *separable* optimization, the objective function $F(\boldsymbol{p})$ is decomposed into a sum of simpler functions that can be optimized separately, for example

$$F(\boldsymbol{p}) = F_1(\boldsymbol{p}_1) + F_2(\boldsymbol{p}_2) + F_3(\boldsymbol{p}_3)$$

. . . requires detailed information about the structure of the objective function.

Can the method be applied to alignment?

Consider the six parameters $\Delta\boldsymbol{p}$ of a single sensor (translation + rotation) and the corresponding 6×6 matrix $\boldsymbol{C}$ and right-hand side 6-vector $\boldsymbol{b}$:

$$\boldsymbol{C}\,\Delta\boldsymbol{p} = \boldsymbol{b} \qquad \text{with diagonalization} \quad \boldsymbol{C} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\mathrm{T}}$$

The transformed vector of linear combinations

$$\boldsymbol{q} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{b}$$

has a diagonal covariance matrix $\boldsymbol{V} = \boldsymbol{D}^{-1}$, and thus the linear combinations are uncorrelated.

Idea: consider only the three most-sensitive (i.e. with largest diagonal elements) or most-significant linear combinations, or try two pairs of each three linear combinations, which are *almost* independent. This reduces the storage space of the matrix to 1/4.

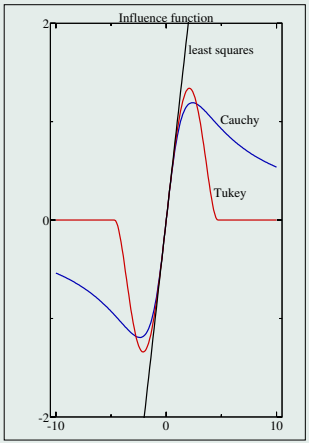
The presence of outliers in the data can deteriorate the alignment result. Difficulty: wrong initial alignment parameters can fake outliers.

Millepede I: Large initial cut at $\approx 10\sigma$ reduced to 3σ in ≈ 5 iterations.

Millepede II: No cut in first iteration, followed by technique of M-estimates in subsequent iterations.

The objective function in least squares is the sum of **squares** of scaled residuals z, with larger influence for larger residuals (outliers). The **square** is replaced in M-estimates by a dependence with reduced influence for larger residuals.

	$\rho(z) = \ln \mathrm{pdf}(z)$	influence function $\psi(z) = \mathrm{d}\rho(z)/\mathrm{d}z$	add. weight $\omega(z) = \psi(z)/z$
Least squares	$= \frac{1}{2} z^2$	$= z$	$= 1$
Cauchy($c = 2.3849$)	$= \frac{c^2}{2} \ln\left(1 + (z/c)^2\right)$	$= \frac{z}{1 + (z/c)^2}$	$= \frac{1}{1 + (z/c)^2}$
Huber $\begin{cases} \text{if } \lvert z\rvert \le c = 1.345 \\ \text{if } \lvert z\rvert > c = 1.345 \end{cases}$	$= \begin{cases} z^2/2 \\ c\,(\lvert z\rvert - c/2) \end{cases}$	$= \begin{cases} z \\ c \cdot \mathrm{sign}\,(z) \end{cases}$	$= \begin{cases} 1 \\ c/\lvert z\rvert \end{cases}$



Summary

A comparison of large scale optimization methods

Residuals	$F(\boldsymbol{p})$	∇F	$\boldsymbol{H} \equiv \nabla^2 F$	$\boldsymbol{H}^{-1}$	Method	MP
×	–	–	–	–	Residual	
–	×	×	–	–	Steepest descent	
–	×	×	–	×	Quasi-Newton (e.g. BFGS)	
–	×	×	×	×	Newton (standard optimization)	
–	–	×	×	×	Inversion	I,II
–	–	×	×	×	Diagonalization	II
–	–	×	(×)	–	Sparse $\boldsymbol{H}$ solution	II
–	–	×	(×)	(×)	Band or bordered band matrix	II
–	×	×	–	(×)	L-BFGS (virtual $\boldsymbol{H}^{-1}$)	II

- Several solution modes in Millepede II should allow alignment even with $n = 100\,000$ parameters.
- Options to add constraints and to reduce influence of outliers included.
- Simultaneous use of several (all) types of events and data – physics data: single tracks, vertices, invariant-mass constraints, and cosmics, overlaps . . . external measurements is essential for accurate alignment.

Contents