LHC Detector Alignment Workshop

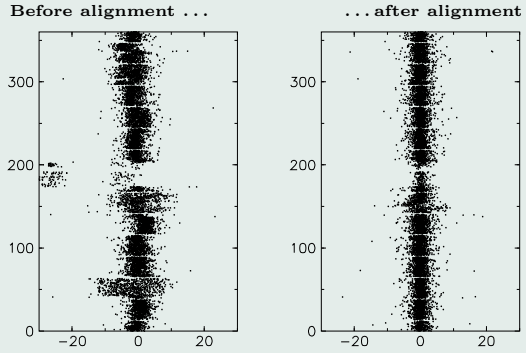
September 4 – 6 **2006, CERN**

Alignment Algorithms

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. Many different methods are used in HEP, which either require a large number of iterations, due to a linear convergence behaviour, or which require the solution of a large matrix equation. 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 and classification

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, inproved mass and vertex resolution, 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.

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 on survey data, perhaps on Laser alignment data, and **mainly** on track residual minimization (*incomplete* data, several degrees of freedom undefined);
- no "reference standards with known values" exist ($\approx$ exceptions are data from $e^+e^- \to \mu^+\mu^-$ and cosmics with $\boldsymbol{B} = 0$).

Alignment parameter corrections Δp

Geometrical corrections: six parameters are required for each individual detector element or larger detector component: translations: $(\Delta u,\ \Delta v,\ \Delta w)$ and angles: $(\alpha,\ \beta,\ \gamma)$.
Often three parameters (two translations, one rotation) are very sensitive.

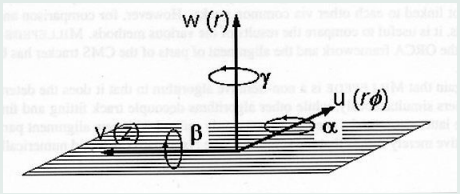
Planar sensors (silicon pixel or strip detectors): local (sensor) coordinates (u, v, w) and global detector coordinates $\boldsymbol{r} = (x, y, z)$ are transformed by

$$(u, v, w) = \boldsymbol{R}\,(\boldsymbol{r} - \boldsymbol{r}_0)$$
$$(u, v, w)_{\text{aligned}} = \Delta\boldsymbol{R}\,\boldsymbol{R}\,(\boldsymbol{r} - \boldsymbol{r}_0) + (\Delta u, \Delta v, \Delta w)$$

$\boldsymbol{R},\ \boldsymbol{r}_0 =$ nominal rotation, position

The correction matrix $\Delta\boldsymbol{R}$ is given by small rotations by α, β, 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.

Drift chamber: Lorentz-angle, T_0, drift velocity v_{drift}, ... global, and per plane, per layer ... and e.g. correction dependent on angle between track and drift direction.

Optimization

Alignment as an optimization problem:

- definition of an objective function $F(\Delta \boldsymbol{p})$, to be minimized;
- large number n of correlated parameters, up to $n \approx 6 \times 5\,000$ (Atlas) or $n \approx 6 \times 20\,000$ (CMS);
- non-linearities are expected to be not large.

The standard minimization method to obtain $\boldsymbol{\Delta p}$ = corrections to alignment parameters requires the solution of a system of linear equations:

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

($\boldsymbol{C}$ is a n-by-n matrix) or of a sequence of such systems of linear equations in case of non-linearities.

Most (not all) *physicists think:*

- *The solution of a matrix equation requires a matrix inversion!* **not true!**
- *Matrix inversion is impossible for a large matrix!* **not true!**

> *"The solution of 4200 equations in 4200 unknowns is computationally infeasible. Even worse, non-linear fit won't converge."*

Therefore most alignment methods work only with small matrices.

Classification

- **Minimization of sum of squares of residuals:** A "global" objective function $F(\ldots)$ or χ^2-function is constructed

$$F(\ldots) = \sum_{\text{data sets}} \left(\sum_{\text{events}} \left(\sum_{\text{tracks}} \left(\sum_{\text{hits}} \Delta_i^2 / \sigma_i^2 \right) \right) \right)$$

  Biased algorithms: methods with solution of many small system of equations; requires iterations to reduce or remove bias.
  $F(\boldsymbol{\Delta p}) = \ldots$ depends on the alignment parameters corrections $\boldsymbol{\Delta p}$.
  Almost all alignment methods tried in HEP experiments are of this type.

  Unbiased algorithms: direct methods with solution of large system of equations, allowing to apply constraints.
  $F(\boldsymbol{\Delta p}, \boldsymbol{q}) = \ldots$ depends on the corrections to alignment parameters $\boldsymbol{\Delta p}$ and **all** track parameters $\boldsymbol{q}$
  Examples: Millepede (used by several experiments), Brueckman de Renstroem (Atlas)

- **Special methods** (discussed in other talks)

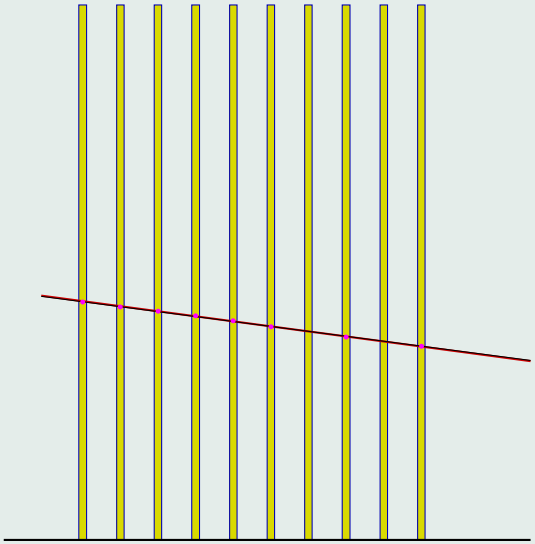
  Kalman filter: ⇒ Talk by Rudi Frühwirth: "Alignment using a Kalman Filter Technique"
  Update of alignment parameters (sequential parameter estimation) during track processing.

  SLD: ⇒ Talk by Fred Wickens: "Alignment experience from SLD"
  Fits of different functional forms to various residual types, and extraction of alignment parameters from fitted functional forms.

2. Alignment of a toy detector

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

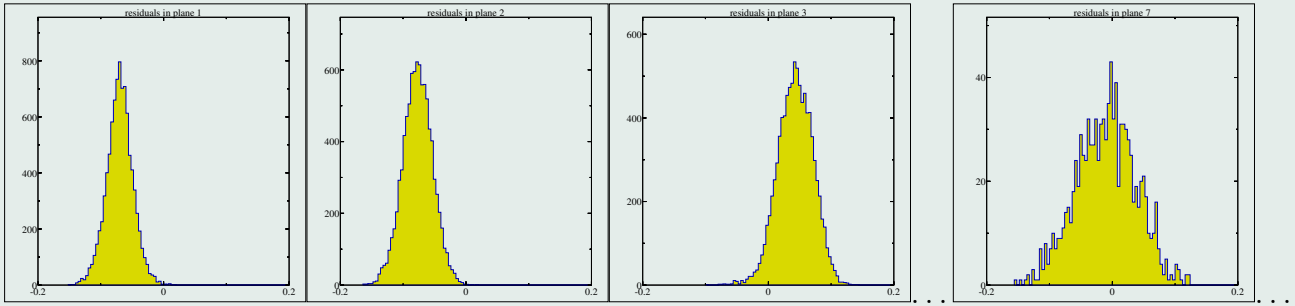
- 10 planes of tracking chambers, 1 m high, 10 cm distance, no magnetic field;
- accuracy $\sigma \approx 200\mu$m, with efficiency $\epsilon = 90\%$;
- plane 7 sick: accuracy $\sigma \approx 400\mu$m, with efficiency $\epsilon = 10\%$;
- 10 000 tracks with 82 000 hits available for alignment;
- Misalignment: 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, ignoring the misalignment.
- The residuals (fitted coordinate f_i − measured vertical) are histogrammed, separately for each plane.



- The mean value of the residuals is taken as correction to the vertical plane position.
- One iteration reduces the residuals significantly, . . . but fitted track parameters are essentially unchanged (biased).

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

Result from the first attempt

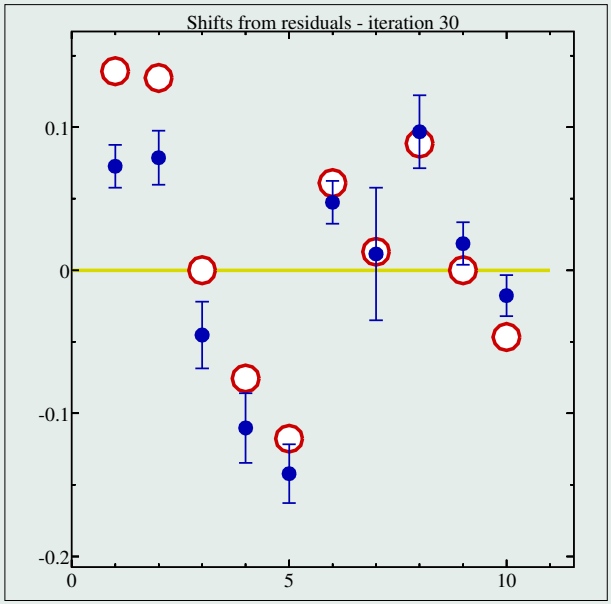
Large changes in first iteration, small changes in second iteration, almost no progress afterwards.

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

Units are cm.

red circle = true shift (displacement)
blue disc = displacement, determined from residuum



First attempt – Discussion

The result is not (yet) encouraging!

Method is equivalent to attempt, to solve a linear system of 10 unknowns, which is singular.

The reason for non-convergence is simple:

> Two degrees of freedom are undefined: a simultaneous shift and an 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.

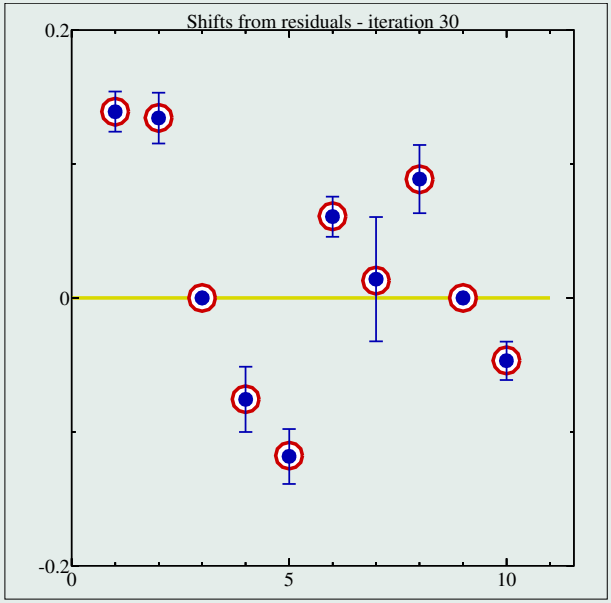
After 30 iterations with planes 3 and 9 fixed (displacement = 0) ...

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

Units are cm.

red circle = true shift (displacement)

blue disc = displacement, determined from residuum



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

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

for all k sufficiently large. Speed of convergence depends on the eigenvalue spectrum – slow convergence for small eigenvalues.

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:

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

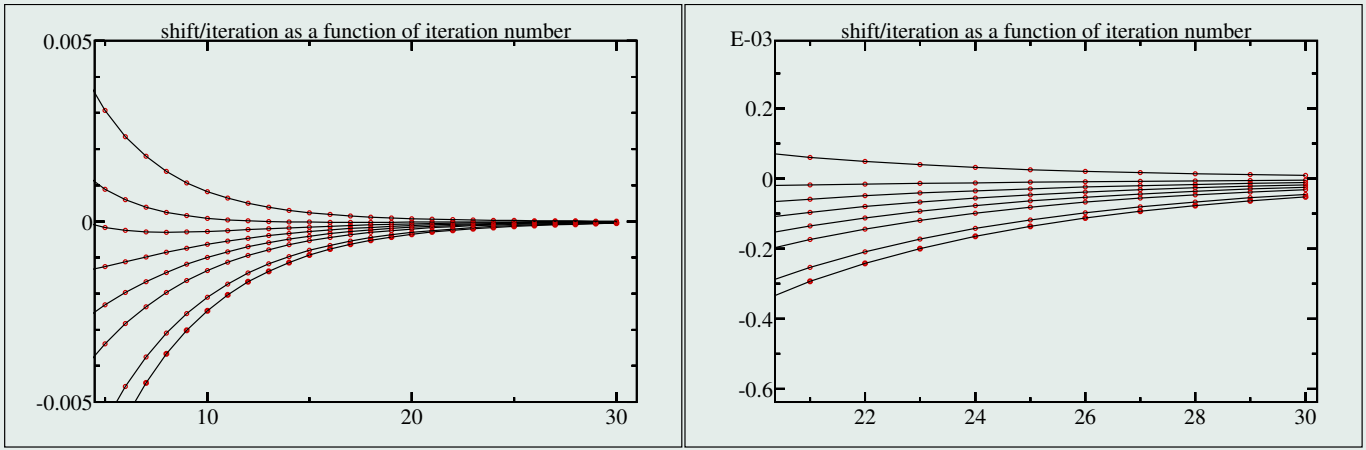
Often in applications small $||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k||$ is assumed to indicate convergence, but

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

i.e. a small $||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k||$ does not mean small distance to the solution!

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

Many small shifts $\boldsymbol{x}_k - \boldsymbol{x}^*$ add up during the iterations.



... linear convergence.

Units are cm.

Alternative: unbiased algorithm

Include the alignment parameters in the parameters fitted in track fits – requires a simultaneous fits of many tracks, with simultaneous 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 a 20010-by-20010 matrix.

The MILLEPEDE principle allows to reduce the least squares problem to $n = 10$ parameters, which has a direct unbiased solution (no iterations)!

Results from a simultaneous fit

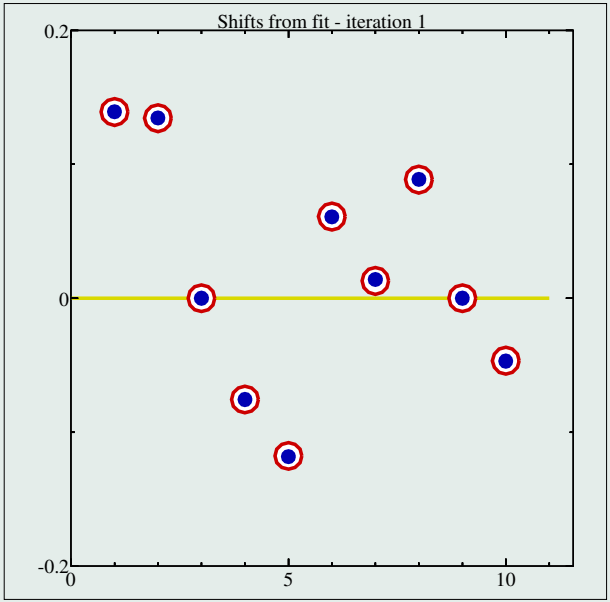
After one step (with planes 3 and 9 fixed at displacement = 0) ...

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, units are cm.)

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.

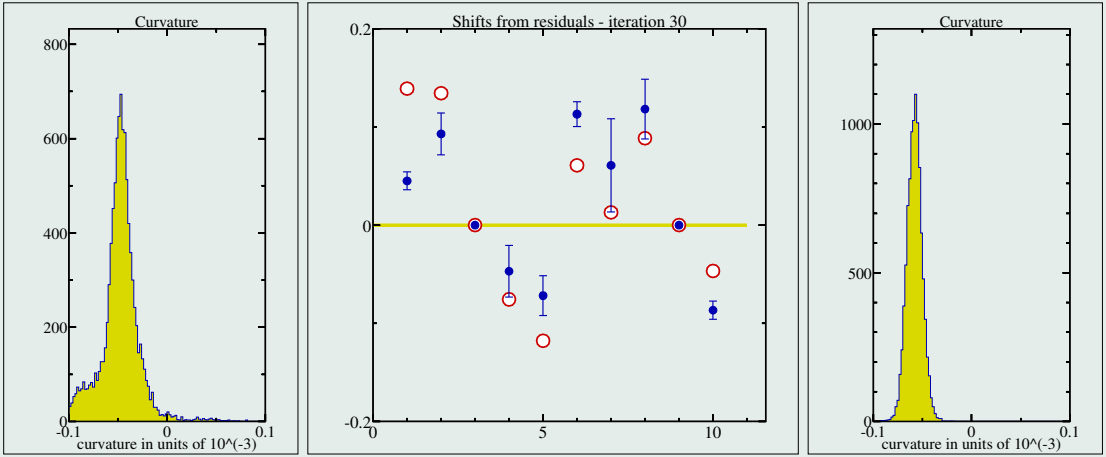
A more realistic scenario ("incomplete data")

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 alignment

3. Millepede

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.

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

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

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 $\boldsymbol{H}_k^{\text{global-local}}$

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

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: $\Rightarrow$

Solution by partitioning

Partitioning the matrix equation $\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b}$ ($\boldsymbol{C}$ symmetric, $\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{22}$ are square matrices):

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

If $\boldsymbol{C}_{12} \equiv 0$: $\quad \boldsymbol{C}_{22}\, \boldsymbol{a}_2^* = \boldsymbol{b}_2 \quad \boldsymbol{a}_2^* = \boldsymbol{C}_{22}^{-1}\, \boldsymbol{b}_2$ (called *local* solution)

Submatrix $\boldsymbol{B}$ of the complete inverse matrix correponding to the upper left part $\boldsymbol{C}_{11}$ is easy to calculate afterwards: $\boldsymbol{B} = \left(\boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^{T}\right)^{-1}$ and complete inverse matrix equation in terms of $\boldsymbol{B}$:

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

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

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 covariance matrix $\boldsymbol{V}_k = \left(\boldsymbol{C}_k^{\text{local}}\right)^{-1}$ 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 two '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.

Solution by matrix inversion in Millepede I ... 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

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

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

Subroutine SPMINV (in Millepede); in-space inversion of symmetric matrix in $(n^2 + n)/2$ words: choose always 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.

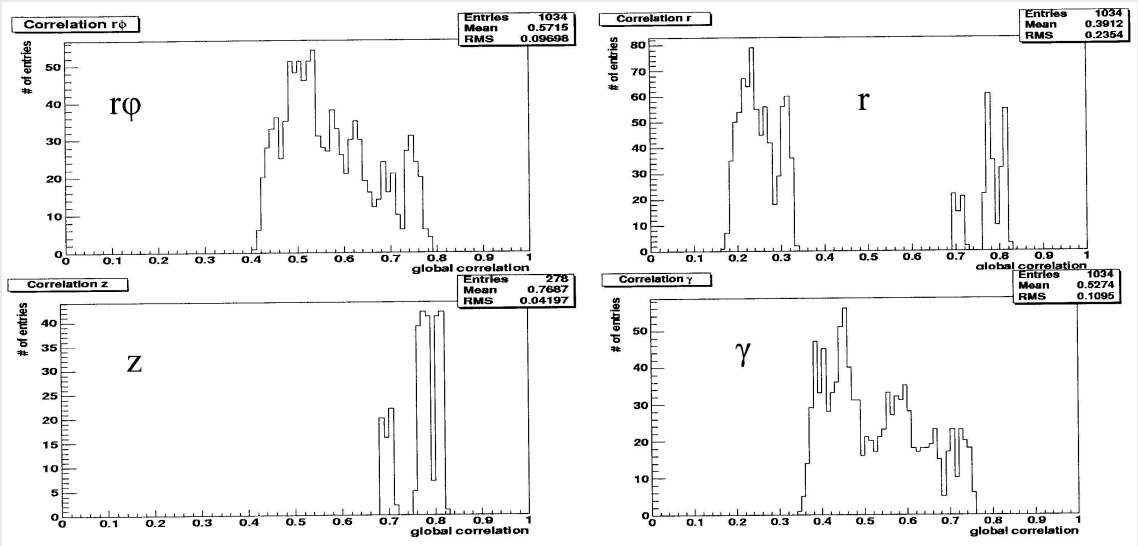
The **global correlation coefficient**, ρ_j is a measure of the total amount of correlation between the j-th parameter and *all* the other variables. It is the largest correlation between the j-th parameter and every possible linear combination of all the other variables.

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

Matrix is ill-conditioned (almost singular), if any ρ close to 1, with large condition number κ.

Global correlations

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



Values ≈ 1 means strong correlation and almost singular matrix – inversion may be impossible (and biased iterative methods would be extremely slow).

Values depend on geometry and type of data – additional data (cosmics, vertex and mass-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).

Degrees of freedom can be fixed, if necessary,

- 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 e.g. by fixed planes;
- by equality constraints (e.g. $d_x = 0$) 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 $\boldsymbol{g} = \partial g(\boldsymbol{p})/\partial \boldsymbol{p}$
  (determination of stationary point of Lagrange function, matrix not positive definite.):

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

- alternative is penalty function ($\ldots + |g(\boldsymbol{p})|^2$) or combination of both (augmented Lagrangian);
- elimination method, with reduction of the total number of parameters.

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 I, code 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`

MILLEPEDE design: experiment-independent program, with well-defined interface to experiment-dependent data.
Used (or under test) by H1(1997), CDF(2001), HERA-b, ZEUS, CMS, ATLAS, LHC-b, Compass, Phenix . . . ??? and rewritten in C++ several times (unpublished).

⇒ Talk by Claus Kleinwort: combined alignment/calibration of H1 vertex detector and drift chamber (resolution improved by factor 2).

Formalism equiv. to Millepede principle derived for ATLAS:
P.Bruckman, S.Haywood, Least Squares Approach to the Alignment of the Generic High Precision Tracking System, PHYSTAT05, Oxford 12-15 Sept. 2005

Principle of reducing matrix size (perhaps) used already in 19.th century in surveying.

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 (unbiased, simultaneous fit of arbitrary number of tracks and of alignment parameters);
- allow different (direct and iterative) methods for the solution of large matrix equation, using mathematical methods from the (mathematical) community (literature) (no home-made iterative methods);
- design with even stronger separation of experiment-dependent program and the Millepede alignment computation;
- 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

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

Pede Normal equation loop: read special file and form matrix and vector

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 MILLEPEDE 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 ($n < 5000$), or iterative solution (MINRES, or L-BFGS)

The introduction of overall equality constraints requires the solution of large systems of equations!

How to solve very large systems of equations?

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

Matrix inversion: • e.g. routine 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); possible for large n on special hardware;

Sparse matrix storage: • allows to store big sparse matrices

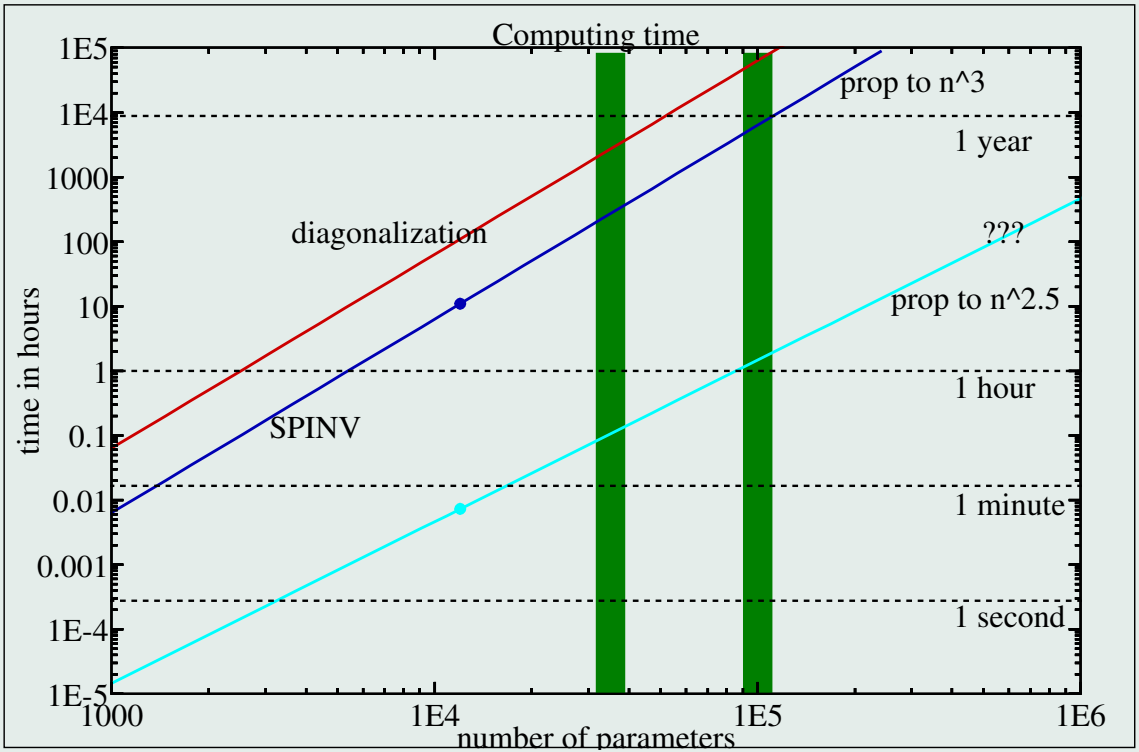
Generalized minimal residual method: • fast method for large sparse matrices, factor $> 1\,000$ faster than inversion for n =12 000. Routines MINRES • (and SYMMLQ ⊖);

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

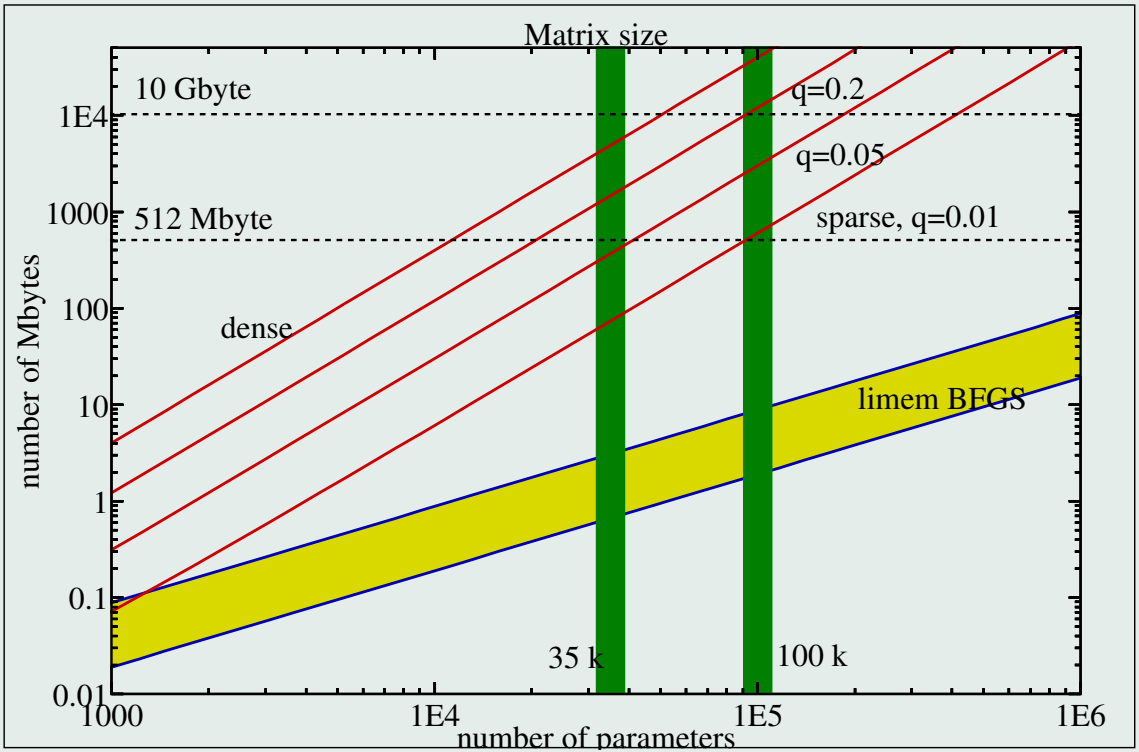
Limited memory BFGS: • uses only *virtual* matrix, low space requirement, but many iterations(?);

Millepede II Code: • =included, ⊖ = not yet tried.

Method of M-estimates instead of cuts against outliers; square (of least squares) replaced by density with larger tails for outliers.



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



q = fraction of non-zero off-diagonal elements

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 replacement $1/\lambda_i = 0$ 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.

Reduction of the number of parameters

(**1**) Consider the six parameters $\Delta\boldsymbol{p}$ of a single sensor (translation + rotation) and the corresponding 6×6 matrix $\boldsymbol{C}_{6\times6}$ and right-hand side 6-vector $\boldsymbol{b}$:

$$\boldsymbol{C}_{6\times6}\,\Delta\boldsymbol{p} = \boldsymbol{b} \qquad \text{with diagonalization} \quad \boldsymbol{C}_{6\times6} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\mathrm{T}}$$

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.

(**2**) One (small) set of alignment parameters for large detector section, added to single-sensor alignment parameters which are either *active* (with $\sum = 0$ constraint), or *non-active* (fixed parameters).

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 $\sigma_i^2 = \lambda_i$.

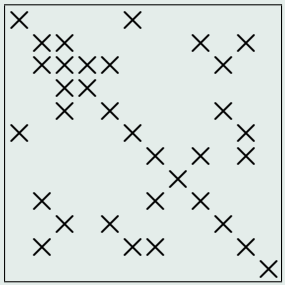
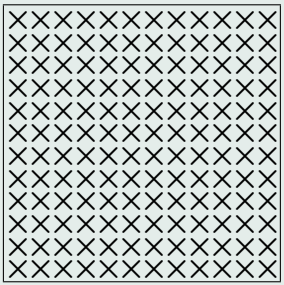
$$\text{Solution of } \boxed{\min \|\boldsymbol{J}\,\boldsymbol{\Delta p} - \boldsymbol{r}\|_2} \text{ by } \qquad \boldsymbol{\Delta p} = \boldsymbol{U}\ \left[\mathrm{diag}\left(\frac{1}{\sigma_i}\right)\right]\ (\boldsymbol{V}^{\mathrm{T}}\boldsymbol{r})$$

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 q of non-zero off-diagonal elements. The automatic generation of parameter-index relations requires a large number of comparisons. A fast method using a combination of hashing, sorting and binary search is used in MP II.

Note: the inverse of a sparse matrix  is a dense matrix: 

Mathematical methods for the solution exist, which only require the product of the matrix with vectors: $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$.

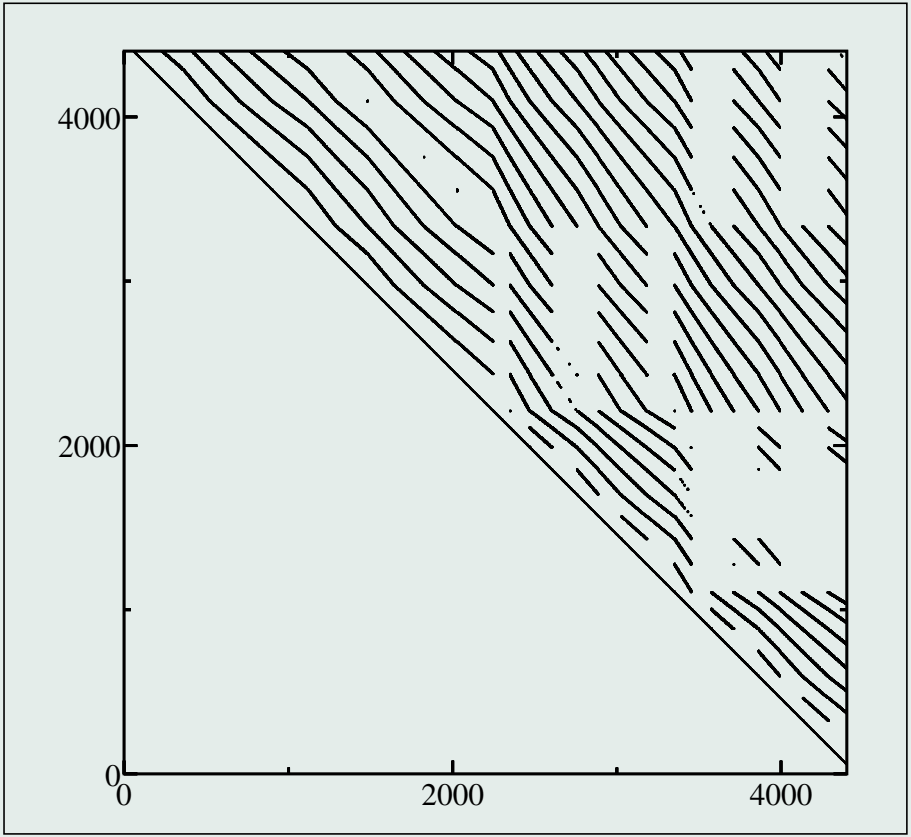
The indexed storage scheme of `PCGPACK`, modified for a symm. matrix, is used: it requires arrays with

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

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)

A sparse matrix example

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).



Generalized minimal residual method (GMRES)

Solution of a very large system of linear equations with sparse matrix, by an optimized 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, 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 $\boldsymbol{x}$.

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 $\boldsymbol{x}$. The matrix $\boldsymbol{M}^{-1}$ should be an approximation to $\boldsymbol{C}^{-1}$, such that $\boldsymbol{M}^{-1}\boldsymbol{C} \approx \boldsymbol{1}$. (not yet implemented).

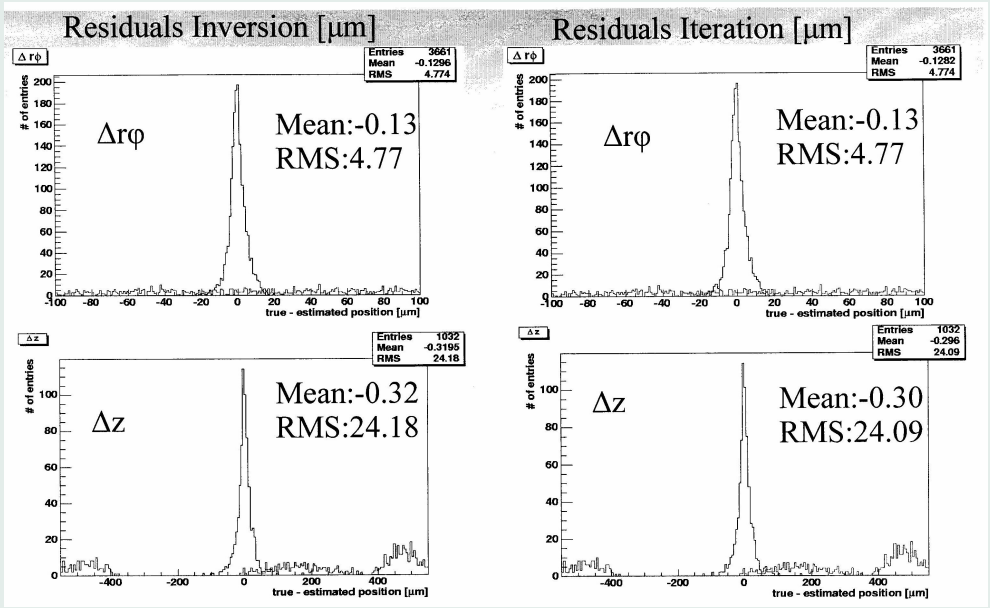
C. C. Paige and M. A. Saunders (1975), Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12(4), pp. 617-629.

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

Comparison

For 12 000 parameters

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



Quasi-Newton methods ...

... require only the gradient ∇F 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

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

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

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

(used e.g. in MINUIT/MIGRAD).

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 $\boldsymbol{B}$ and thus cannot be used.

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 the m 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 $\boldsymbol{y}_k$, $\boldsymbol{s}_k$ only (no matrix storage).

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

. . . 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
Jorge Nocedal and Stephen J.Wright, Numerical Optimization, Springer

Summary

Two classes of alignment algorithms are developed in HEP experiments:

Biased algorithm: require a certain number of iterations and requires to fix certains planes (in 3D);

Unbiased algorithm: requires to solve a large system of equations, allows to **add equality constraints** in formalism.

Both methods reduce average residuals – what about behaviour for long-range correlations between alignment parameters ?

Solution methods, in order of increasing size parameter n:

$F(\boldsymbol{p})$	∇F	$\boldsymbol{H} \equiv \nabla^2 F$	$\boldsymbol{V} \equiv \boldsymbol{H}^{-1}$	Method
–	×	×	×	Diagonalization
–	×	×	×	Inversion
–	×	(×)	–	Sparse $\boldsymbol{H}$ solution
×	×	–	(×)	L-BFGS (virtual $\boldsymbol{H}^{-1}$)

All these solution methods are implemented in MILLEPEDE II, which should become able to perform alignment even with $n \approx 100\,000$ parameters.

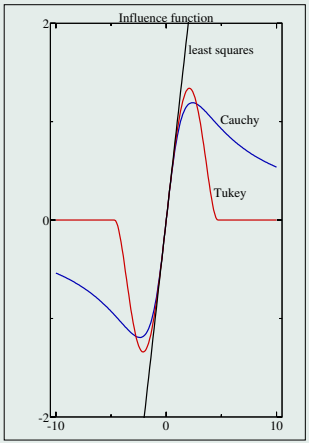
The presence of outliers in the data can deteriorate the alignment result. Difficulty: wrong initial alignment parameters can fake outliers.

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 $\begin{cases} \text{if } \lvert z\rvert \le c = 1.345 \\ \text{if } \lvert z\rvert > c = 1.345 \end{cases}$	$= \begin{cases} z^2/2 \\ c\,(\lvert z\rvert - c/2) \end{cases}$	$= \begin{cases} z \\ c \cdot \mathrm{sign}\,(z) \end{cases}$	$= \begin{cases} 1 \\ c/\lvert z\rvert \end{cases}$



Covariance matrix with MINRES

The inverse of matrix $\boldsymbol{C}$ is the covariance matrix $\boldsymbol{V}$ of the alignment parameters. This is available with matrix inversion and diagonalization, but not with MINRES.

Method to compute some elements of $\boldsymbol{V}$ with MINRES:

Solution of matrix equation $\boldsymbol{CV} = \boldsymbol{1}$ right hand-side is unit matrix

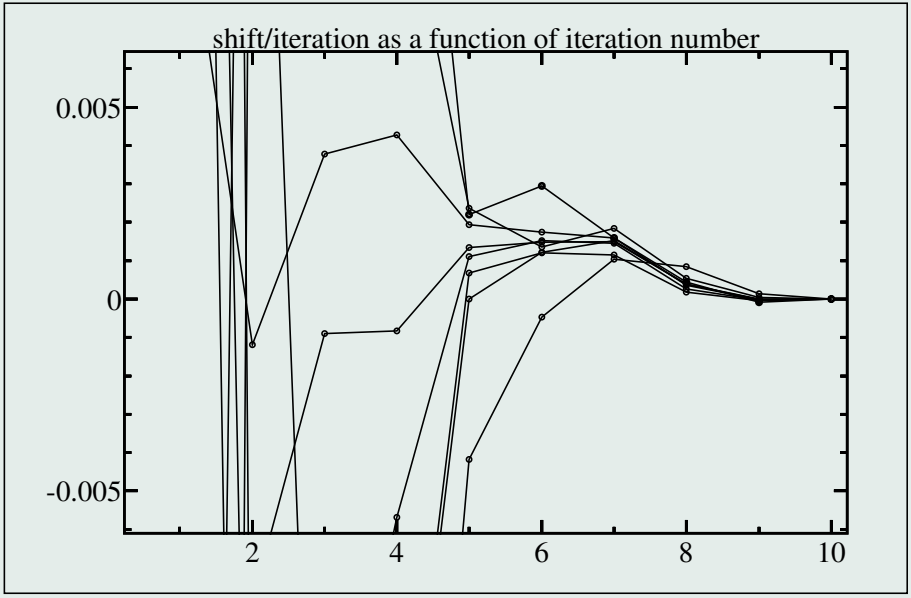
for $\boldsymbol{V}$ would give the complete covariance matrix $\boldsymbol{V}$.

Solution of matrix equation $\boldsymbol{Cv} = \boldsymbol{u}$ right hand-side is column-vector of unit matrix

for $\boldsymbol{V}$ will give on column of the covariance matrix $\boldsymbol{V}$.

Elements of covariance matrix are determined by hits statistics and by geometry.

Fast convergence for limited memory BFGS method after iteration 8.



. . . fast rate of linear convergence.

Units are cm.

Contents