Software Alignment for Tracking Detectors

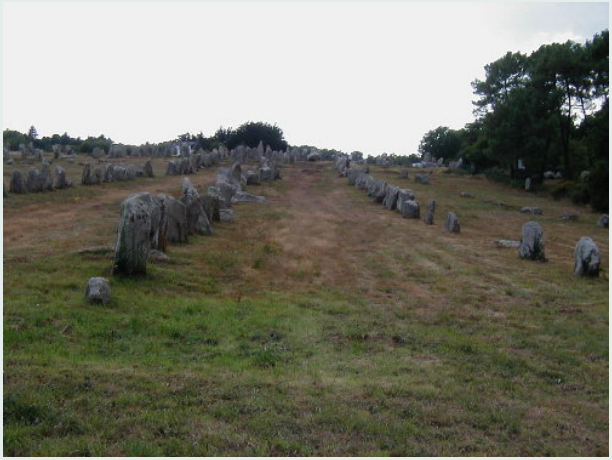
Abstract

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. Several different methods are in use, ranging from simple residual-based procedures to complex fitting systems with many thousands of parameters. The methods are reviewed with respect to their mathematical basis and accuracy, and to aspects of the practical realization.

1. Alignment and calibration

What is alignment and calibration? . . . a web search

Alignment:
1. an adjustment to a line; arrangement in a straight line.
2. the line or lines so formed.
3. the proper adjustment of the components of an electronic circuit, machine, etc., for coordinated functioning: The front wheels of the car are out of alignment.
4. a state of agreement or cooperation among persons, groups, nations, etc., with a common cause or viewpoint.
5. a ground plan of a railroad or highway.
6. (Archaeol.) a line or an arrangement of parallel or converging lines of upright stones or menhirs.



Alignement at Kermario (Bretagne)

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

Calibration seems to be the more general term, while *alignment* is related to *geometrical* calibration.

Methods in HEP: many papers and collaboration reports on the Web, with a lot of HEP folklore (residuals, pulls, fit errors, inversion of a rectangular matrix, ... with the usual gradient method, MINUIT, typically $\mathcal{O}(10\mu\text{m})$, chi-square minimization, constraints, convergence recognized by small change/iteration, < 20 iterations, ... our method is successful).

Sometimes it is difficult to understand what is really done: "The detector parameters are found via a χ^2 minimization of the residuals."

Aim of **alignment/calibration** with tracks, after an initial optical survey and corrections for electronics and mechanical effects:

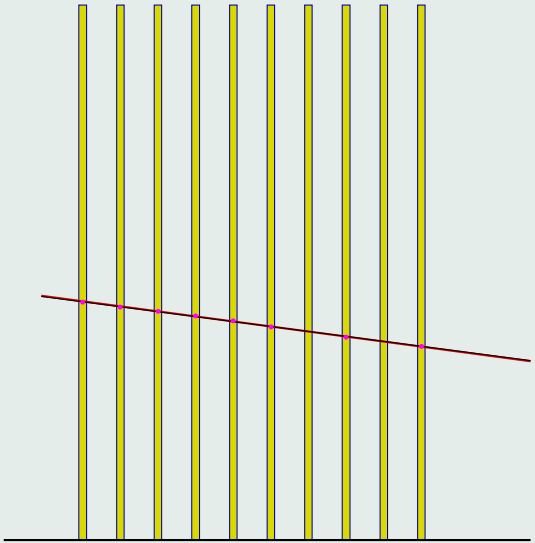
- eliminate or reduce bias in detector data;
- increase precision of reconstructed tracks and vertices;
- improve track and vertex recognition, reduce χ^2 of the fits;
- essential for accurate vertex detectors with potential precision of a few μm for e.g. heavy quark physics.

Alignment/calibration requires to *understand* the detector (functional relationship) and to optimize thousands of parameters – not a simple task. **Residuals** play a central role.

2. Alignment of a toy detector

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

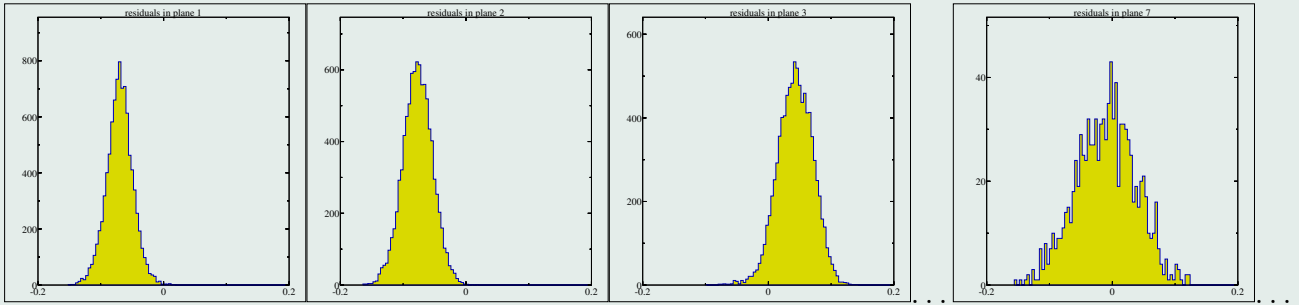
- 10 planes of tracking chambers, 1 m high, 10 cm distance, no magnetic field;
- accuracy $\sigma \approx 200\mu$m, with efficiency $\epsilon = 90\%$;
- plane 7 sick: accuracy $\sigma \approx 400\mu$m, with efficiency $\epsilon = 10\%$;
- 10 000 tracks with 82 000 hits available for alignment;
- Misalignment: 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:

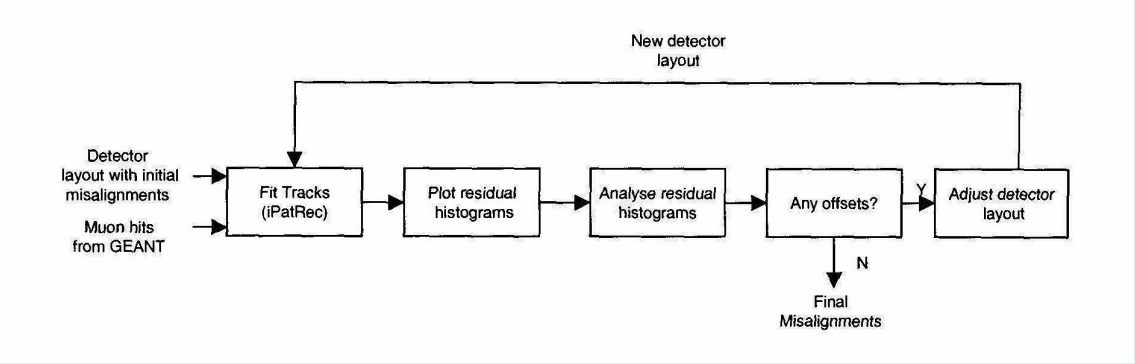
- A straight line is fitted to the track data.
- The residuals (= measured vertical coordinate minus fitted coordinate) are histogrammed, separately for each plane.



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

This is the standard method used in many experiments.

Alignment using iterative track fitting



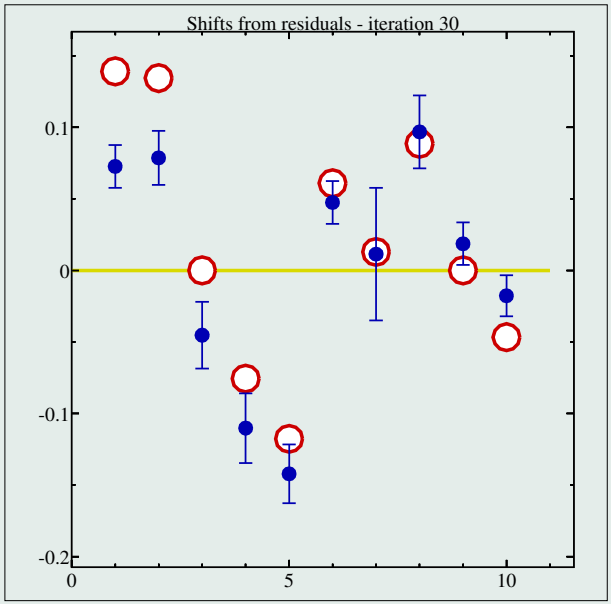
Properties of the method:

- simple, only basic operations $+ - \times /$ used, no matrices, n-tuples can be used;
- "simple, intuitive, quick", "method very robust" (i.e. small changes expected from residuals);
- but based on biased straight line fit!
- Is the method convergent, if applied iteratively, and if convergent, what is the order (linear of quadratic) of convergence?

Result from the first attempt

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

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

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 rotation of all 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:

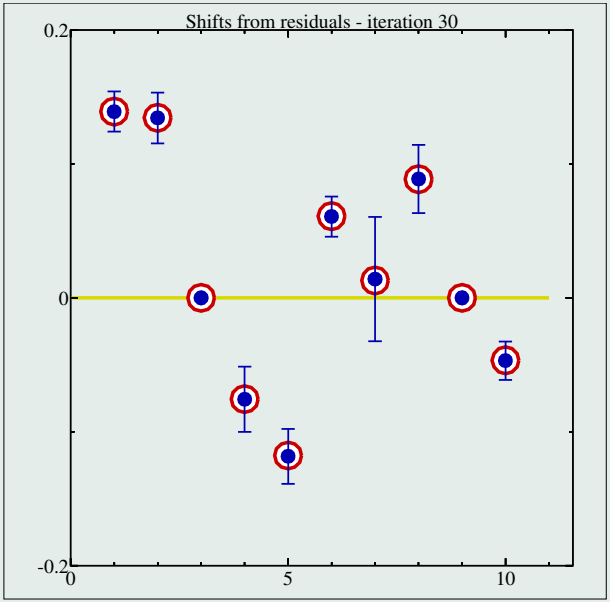
- Use only fixed planes (planes 3 and 9) in the fit, and determine the residuals of other planes;
- for the determination of the displacement of a certain plane use all other planes in the fit.

These possibilities are in fact used by several collaborations!

Results from the second attempt

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

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.

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, 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.

... a nice problem!

Results from a simultaneous fit

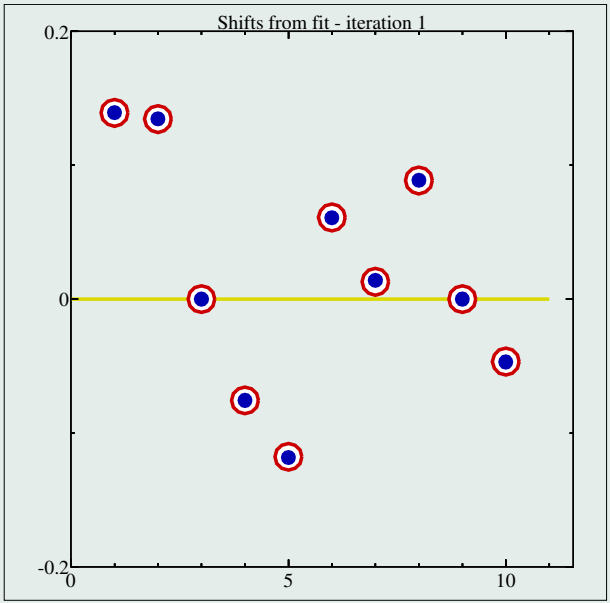
After one step (with planes 3 and 9 fixed at displacement = 0) ...

ID	true shift	determined	ρ	mean residual
1	0.1391	0.1393 ± 0.004	0.68	0 ± 150
2	0.1345	0.1346 ± 0.003	0.66	0 ± 189
3	0.0000			0 ± 234
4	-0.0756	-0.0756 ± 0.003	0.58	0 ± 244
5	-0.1177	-0.1182 ± 0.003	0.53	0 ± 205
6	0.0610	0.0608 ± 0.003	0.50	0 ± 150
7	0.0130	0.0141 ± 0.007	0.20	0 ± 464
8	0.0886	0.0888 ± 0.003	0.53	0 ± 255
9	0.0000			0 ± 149
10	-0.0467	-0.0469 ± 0.003	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 $\Delta\chi^2 = 1.159 \times 10^{-5}$

But how can this problem be solved in less than a second?

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

$\left(\frac{\Delta v_{\text{drift}}}{v_{\text{drift}}}\right)_j$ = 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 ± 0.004	0.68	0.0020	0.0019 ± 0.0002	0.016	0 ± 119
2	0.1345	0.1346 ± 0.003	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 ± 0.003	0.58	0.0200	0.0197 ± 0.0002	0.013	0 ± 139
5	−0.1177	-0.1182 ± 0.003	0.53	-0.0138	-0.0136 ± 0.0002	0.013	0 ± 141
6	0.0610	0.0608 ± 0.003	0.50	0.0003	0.0004 ± 0.0002	0.019	0 ± 139
7	0.0130	0.0141 ± 0.007	0.20	-0.0306	-0.0303 ± 0.0006	0.038	0 ± 348
8	0.0886	0.0888 ± 0.003	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 ± 0.003	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.

3. Linear least squares and matrix equations

Gauss, Laplace and Lagrange

Johann Carl Friedrich Gauss
(1777 – 1855)

Pierre-Simon Laplace
(1749 – 1827)

Joseph-Louis Lagrange
(1736 – 1813)

The standard linear least squares method I

Measurement: data $y_i,\ i = 1, 2, \ldots m$, measured independently and *without bias* at coordinate x_i, with standard deviation σ_i.

Mathematical model: data y are described by the mathematical model

$$\boxed{y = f(x; \boldsymbol{a})}$$

with a **function** $f(x; \boldsymbol{a})$, depending on the coordinate x and **parameters** $a_j,\ j = 1, 2, \ldots n$.

The function $f(x; \boldsymbol{a})$ is linear or can be linearized. The method of linear least squares can be used to perform one step in improving the parameter estimate $\boldsymbol{a}_\ell$ by a correction vector $\Delta\boldsymbol{a}$:

$$y_i \cong f(x_i; \boldsymbol{a}_\ell) + \sum_{j=1}^{n} d_j \, \Delta a_j \qquad \text{with} \quad d_j = \left.\frac{\partial f(x)}{\partial a_j}\right|_{x=x_i}$$

The principle of least squares requires to minimize the residuals

$$r_i = y_i - f(x_i; \boldsymbol{a}_\ell) \cong \boldsymbol{d}_i^{\mathrm{T}} \Delta\boldsymbol{a}$$

taking into account the measurement accuracy σ_i.

The standard linear least squares method II

The least squares requirement $\qquad \text{minimize } S = \sum_{i=1}^{m} w_i r_i^2 \qquad$ with weight $w_i = \dfrac{1}{\sigma_i^2}$

requires to solve the system of m linear equations (normal equations) or matrix equation

$$\boxed{\boldsymbol{C}\,\Delta\boldsymbol{a} = \boldsymbol{b}}$$

with the symmetric n-by-n matrix $\boldsymbol{C}$ and the n-vector $\boldsymbol{b}$:

$$\boldsymbol{C} = \sum_{i=1}^{m} w_i\,\boldsymbol{d}_i\,\boldsymbol{d}_i^{\mathrm{T}} \qquad\qquad \boldsymbol{b} = \sum_{i=1}^{m} w_i\,r_i\,\boldsymbol{d}_i$$

This completes one step of the iteration

$$\boldsymbol{a}_{\ell+1} := \boldsymbol{a}_\ell + \Delta\boldsymbol{a} = \boldsymbol{a}_\ell + \boldsymbol{C}^{-1}\boldsymbol{b} \qquad \text{and} \qquad \ell := \ell + 1$$

No iteration is necessary for a function, which is linear in the parameters. After convergence $\boldsymbol{a} \equiv \boldsymbol{a}_\ell$ is an *unbiased estimate* of the parameter vector, with covariance matrix given by the inverse matrix $\boldsymbol{C}^{-1}$ (error propagation).

$$\text{residual} \quad r_i = y_i - f(x_i;\boldsymbol{a}) \qquad\qquad \text{pull} \quad p_i = \frac{y_i - f(x_i;\boldsymbol{a})}{\sqrt{\sigma_i^2 - \sigma_{i,\mathrm{fit}}^2}} \quad \text{with} \quad \sigma_{i,\mathrm{fit}}^2 = \boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{C}^{-1}\boldsymbol{d}_i$$

Qualitative investigations using histograms of the residuals and quantitative tests by checking pull distributions: $p_i \sim N(0,1)$ expected.

4. The MILLEPEDE algorithm

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

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

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

$$\begin{pmatrix} \sum_k \boldsymbol{C}_k^{\text{global}} & \cdots & \boldsymbol{H}_k & \cdots \\ \hline \vdots & \ddots & 0 & 0 \\ \hline \boldsymbol{H}_k^{\text{T}} & 0 & \boldsymbol{C}_k^{\text{local}} & 0 \\ \hline \vdots & 0 & 0 & \ddots \end{pmatrix} \times \begin{pmatrix} \Delta\boldsymbol{a}^{\text{global}} \\ \hline \vdots \\ \hline \Delta\boldsymbol{a}_k^{\text{local}} \\ \hline \vdots \end{pmatrix} = \begin{pmatrix} \sum_k \boldsymbol{b}_k^{\text{global}} \\ \hline \vdots \\ \hline \boldsymbol{b}_k^{\text{local}} \\ \hline \vdots \end{pmatrix}$$

If the $\boldsymbol{H}_k^{\text{global-local}}$ are neglected, the complete equation decays into $1+K$ independent matrix equations.
But the solution $\Delta\boldsymbol{a}^{\text{global}}$ can be calculated without approximation with a great simplification: $\Rightarrow$

Inversion by Partitioning

A square n-by-n matrix $\boldsymbol{C}$ and its inverse, $\boldsymbol{C}^{-1} \equiv \boldsymbol{B}$ can be partitioned into submatrices

$$\boldsymbol{C} = \left(\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right) \qquad \boldsymbol{B} = \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right) ,$$

Submatrices $\boldsymbol{C}_{11}$ and $\boldsymbol{B}_{11}$ are p-by-p square matrices and submatrices $\boldsymbol{C}_{22}$ and $\boldsymbol{B}_{22}$ are q-by-q square matrices, with $p+q=n$. A complete set of equations for the submatrices of the inverse matrix $\boldsymbol{B}$ can be derived from the matrix product $\boldsymbol{C}\,\boldsymbol{B} = \mathbf{1}_n$:

$$\begin{aligned} \boldsymbol{C}_{11}\,\boldsymbol{B}_{11} + \boldsymbol{C}_{12}\,\boldsymbol{B}_{21} &= \mathbf{1}_p & \boldsymbol{B}_{11} &= (\boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{21})^{-1} \\ \boldsymbol{C}_{11}\,\boldsymbol{B}_{12} + \boldsymbol{C}_{12}\,\boldsymbol{B}_{22} &= \mathbf{0} & \boldsymbol{B}_{12} &= -\boldsymbol{B}_{11}\boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1} \\ \boldsymbol{C}_{21}\,\boldsymbol{B}_{11} + \boldsymbol{C}_{22}\,\boldsymbol{B}_{21} &= \mathbf{0} & \boldsymbol{B}_{21} &= -\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{21}\boldsymbol{B}_{11} \\ \boldsymbol{C}_{21}\,\boldsymbol{B}_{12} + \boldsymbol{C}_{22}\,\boldsymbol{B}_{22} &= \mathbf{1}_q & \boldsymbol{B}_{22} &= \boldsymbol{C}_{22}^{-1} + \boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{21}\boldsymbol{B}_{11}\boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1} \end{aligned}$$

Only the inversion of a p-by-p and of a q-by-q matrix ($\boldsymbol{C}_{22}$) is required.
The special structure of a matrix to be inverted may allow a significant reduction of the computational effort of the inversion by the method of partitioning (e.g. $\boldsymbol{C}_{22}$ diagonal). Further simplification for symmetric matrices.

Partitioning the solution vector — matrix $\boldsymbol{C}$ symmetric: $\boldsymbol{C}_{21} = \boldsymbol{C}_{12}^T$

Partitioning of the whole matrix equation including the vectors $\Delta\boldsymbol{a}$ and $\boldsymbol{b}$:

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

Special case $\boldsymbol{C}_{12} = \boldsymbol{0}$:

$$\text{independent solutions} \qquad \Delta\widetilde{\boldsymbol{a}}_1 = \left(\boldsymbol{C}_{11}^{-1}\right)\boldsymbol{b}_1 \qquad \Delta\widetilde{\boldsymbol{a}}_2 = \left(\boldsymbol{C}_{22}^{-1}\right)\boldsymbol{b}_2\ .$$

General case $\boldsymbol{C}_{12} \neq \boldsymbol{0}$: solution $\Delta\boldsymbol{a}_1$ can be expressed by $\Delta\widetilde{\boldsymbol{a}}_2$

$$\begin{aligned}\Delta\boldsymbol{a}_1 &= \boldsymbol{B}_{11}\boldsymbol{b}_1 + \boldsymbol{B}_{12}\boldsymbol{b}_2 = \boldsymbol{B}_{11}\boldsymbol{b}_1 - \boldsymbol{B}_{11}\boldsymbol{C}_{12}\left(\boldsymbol{C}_{22}^{-1}\boldsymbol{b}_2\right) = \boldsymbol{B}_{11}\left(\boldsymbol{b}_1 - \boldsymbol{C}_{12}\Delta\widetilde{\boldsymbol{a}}_2\right) \\ &= \left(\boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^T\right)^{-1}\left(\boldsymbol{b}_1 - \boldsymbol{C}_{12}\Delta\widetilde{\boldsymbol{a}}_2\right)\end{aligned}$$

i.e. the full solution $\Delta\boldsymbol{a}_1$ can be obtained by "corrections" to $\boldsymbol{C}_{11}$ and $\boldsymbol{b}_1$ (without calculating $\Delta\boldsymbol{a}_2$). This equation can be applied repeatedly with a problem – opens the possibility to solve problems with a huge number of parameters, if the interest is in a sub-vector of the parameters

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 $\boldsymbol{V}$ of the local parameters

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

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

and finally for the track $\boldsymbol{C} := \boldsymbol{C} - \boldsymbol{H}\boldsymbol{V}\boldsymbol{H}^T$

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:

$$\Delta \boldsymbol{a}^{\text{global}} = \boldsymbol{C}^{-1} \boldsymbol{b}$$

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

Constraints in MILLEPEDE

Undefined degrees of freedom can be avoided by adding parameter constraint equations of type

$$g(\boldsymbol{a}) = 0$$

e.g. "zero average displacement", or "zero rotation of the whole detector".
There are two possibilities:

- fix certain parameters
  simply set rows/columns of the fixed parameter to zero – the matrix inversion program will take care of that.

- add equality constraint equation
  append linearized Lagrange multiplier equation $\lambda \left(g(\boldsymbol{a}) + \boldsymbol{g}^{\mathrm{T}} \cdot \Delta\boldsymbol{a} = 0 \right)$ with $\boldsymbol{g} = \partial g(\boldsymbol{a}) / \partial \boldsymbol{a}$:

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

V. Blobel: Experience with Online Calibration Methods, Contribution to CHEP'97, Berlin 1997 (including the Millepede method), not accepted.
Used/tested in H1(1997), CDF(2001), Hera-B(?), ZEUS(?), Atlas(?) ...
O. Behnke: Directions in Tracking (talk), H1 Collaboration meeting at Zürich, 1999
V. Blobel, Linear Least Squares Fits with a Large Number of Parameters, (2000), and 1. Update (2000) `http://www.desy.de/~blobel` including Fortran code.
C++ interface for Millepede by Jesko Merkel: `http://www.desy.de/~jmerkel/cmillepede.html`
V. Blobel and C. Kleinwort: A New Method for the High-Precision Alignment of Track Detectors, Proc. of the Conference on: ADVANCED STATISTICAL TECHNIQUES IN PARTICLE PHYSICS, Durham, March 18th - 22nd, 2002. `arXiv-hep-ex/0208021`

The algorithm is general and can be applied to other problems with large number of (global) parameters and a huge number of measurements with local parameters.

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.

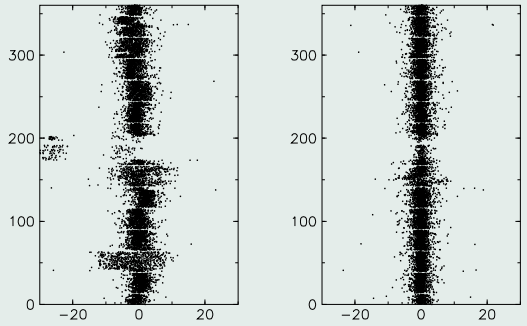
The MILLEPEDE program

The matrix-equations require about 100 lines of code (including optimization for sparse matrices); in addition the routine for the solution of the matrix equation has also about 100 statements [subroutine `MMPEDE` – Mini MILLEPEDE].

The more complete program MILLEPEDE on the Web has ≈ 1700 lines of code and includes iterations.
Outliers are present and have to be removed by cuts – but initial cuts have to be wide in order to accept badly misaligned hits.

Unfortunately this requires iterations e.g. in MILLEPEDE with

- 10σ cut in the first iteration,
- decreasing cut value in further iterations, and
- 3σ cut in the last iteration.



Z-residuals (units are 10 μm) versus ϕ in degrees for a LEP vertex detector – before and after MILLEPEDE alignment.

The region around $\phi = 180°$ with initially very large deviations (deviation $\approx 10\sigma$) is moved to small values.

5. Drift chamber calibration

M. J. Fero et al., Performance of the SLD central drift chamber, Nucl. Instr. methods **A 367** *(1995) 111-114*

Drift velocity calibration v_{drift} : first estimate of relation between v_{drift} and $\boldsymbol{E}$-field based on electrostatic model; relation iteratively corrected by minimizing the fit residuals as a function of drift distance.

Note: $v_{\text{drift}} \propto 1/\text{pressure}$. Tracks are used to measure changes in v_{drift}, which is allowed to be one of the variables in the track fit, and averaging of values from individual tracks.

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

Examples:

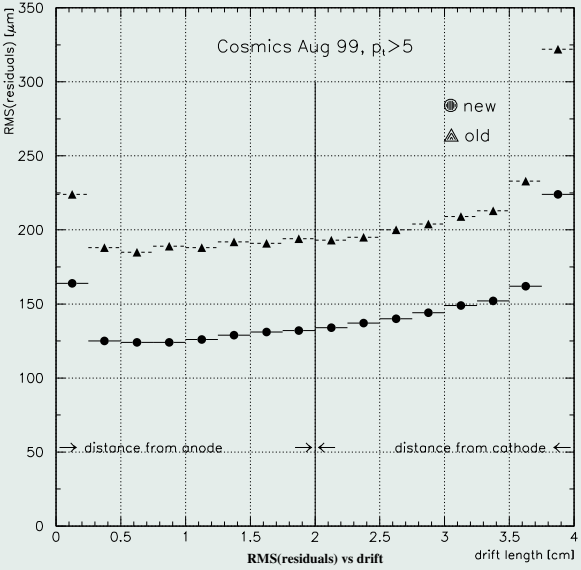
- *common* alignment of the drift chamber and the silicon detector;
- for both CJC1 and CJC2 14 global parameters representing an overall shift or tilt are introduced;
- local variations of the drift velocity v_{drift} for cells halfs and layers halfs are observed, which are parametrized by 180 + 112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

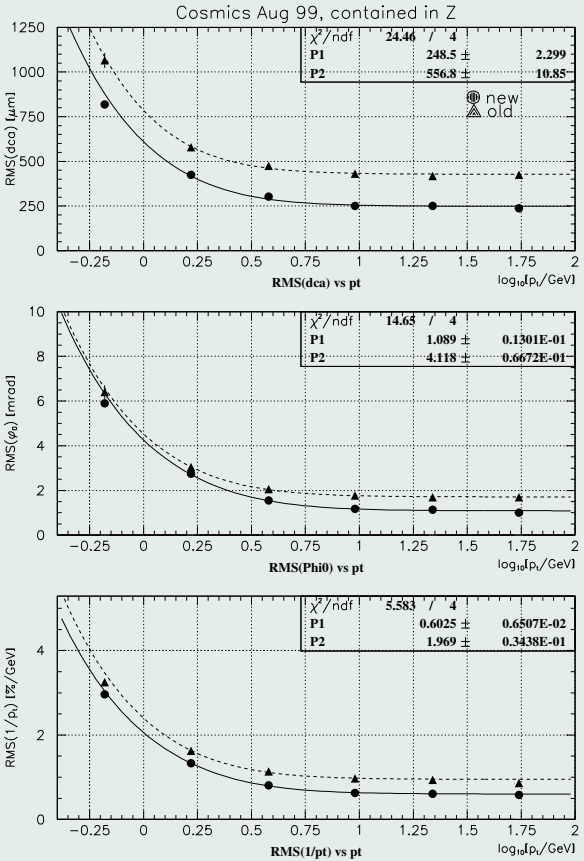
H1 Determination of 1400 parameters . . . from 50 000 tracks

row.	number	parameter	σ	unit
1	2	Δx	1	μm
2	2	$\Delta x/\Delta z_r$	2	μm
3	2	Δy	1	μm
4	2	$\Delta y/\Delta z_r$	2	μm
5	2	$\Delta\varphi$	10	μrad
6	2	$\Delta\varphi/\Delta z_r$	10	μrad
7	2	$\Delta\alpha_{\mathrm{Lor}}$	100	μrad
8	2	$\Delta v_{\mathrm{drift}-}/v_{\mathrm{drift}}$	10^{-5}	
9	2	$\Delta v_{\mathrm{drift}+}/v_{\mathrm{drift}}$	10^{-5}	
10	2	$\Delta T_0 \times v_{\mathrm{drift}}$	< 1	μm
11	2	wire staggering in wire plane	few	μm
12	2	wire staggering perp wire plane	few	μm
13	2	sagging in wire plane	few	μm
14	2	sagging perp. wire plane	few	μm
15	180	$\Delta v_{\mathrm{drift}}/v_{\mathrm{drift}}$ per cell half	few	10^{-4}
16	112	$\Delta v_{\mathrm{drift}}/v_{\mathrm{drift}}$ per layer half	few	10^{-4}
17	330	$\Delta T_0 \times v_{\mathrm{drift}}$ per group	10	μm
18	56	wire position in driftdir. per layer	10	μm
19	56	$\Delta T_0 \times v_{\mathrm{drift}}$ per layer	10	μm
20	56	wire pos. perp. driftdir. per layer	few 10	μm
21	112	$\Delta v_{\mathrm{drift}}/v_{\mathrm{drift}}$ for $I_e/50$ mA	few 10^{-4}	
22	90	$\Delta v_{\mathrm{drift}}/v_{\mathrm{drift}}$ per layer	few 10^{-4}	
23	90	Δy_W per layer	few 10	μm
24	90	Δy_W per layer^2	few 10	μm
25	64	Δ in ladder	few	μm
26	64	Δ perp. ladder	few	μm
27	64	rel. Δ in ladder (z_r)	few	μm
28	64	rel. Δ perp. ladder (z_r)	10	μm
29	64	rel. Δ perp. ladder (φ)	few	μm

Alignment parameters determined in the **Millepede** fit. The **Millepede** accuray is given by a standard deviation σ.

H1 Drift chamber alignment and calibration





Reduction of residual RMS by MILLEPEDE (left) and improvement of track data d_{ca}, ϕ and $1/p_t$ (versus $\log(1/p_t)$ right).

6. Vertex detector alignment

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 $\Delta\boldsymbol{q} = (\Delta u, \Delta v, \Delta w)$)

$$\boldsymbol{q}^c = \boldsymbol{R}_\alpha \boldsymbol{R}_\beta \boldsymbol{R}_\gamma \; \boldsymbol{R}\,(\boldsymbol{r} - \boldsymbol{r}_0) - \Delta\boldsymbol{q}$$

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

Detector	nr of elements	nr of parameters
SLD	96	$96 \times 6 = 576$
Aleph	144	$144 \times 6 = 864$
Delphi	24	72
CDF	352	$352 \times 3 = 1056$
Atlas		
CMS	≈ 20000, ≥ 13000	≈ 100000

Atlas plans/ideas: system of linear equations with 30000×30000 matrix; with hardware: 16 processors (64 bits achitecture) with 900 Mb memory for each processor, ScaLAPACK.

... from *O. Behnke, Directions in Tracking (Talk), Collaboration Meeting in Zürich 1999*

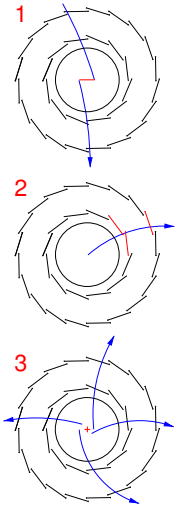
Alignment Method

192·6 = 1152 free parameters
microscope survey of half-ladders
(new 1998: 3d, „dromedar")
⇒ 384 free parameters

Three data samples:

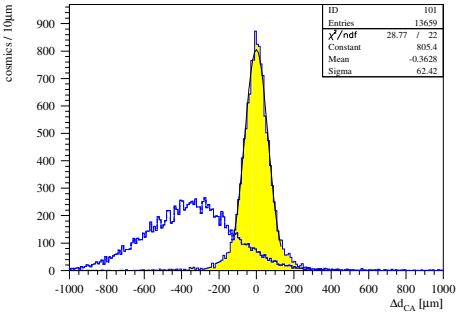
1. cosmics:
   Track $\vec{T}^t = (\kappa_{CJC}, \Phi_0, \theta, d_{ca}, z_0)$
   Covariance Matrix V includes:
   - ▶ CST intrinsic resolution
   - ▶ multiple scattering
   - ▶ CJC momentum resolution

   $\chi^2_{cosmics} = \sum_{events} \Delta\vec{T}^t \cdot V^{-1} \cdot \Delta\vec{T}$
2. ep tracks in overlap regions:
   residuals $\Delta\vec{u}^t = (\Delta x, \Delta z)$
   $\chi^2_{overlaps} = \sum_{events} \Delta\vec{u}^t \cdot V^{-1} \cdot \Delta\vec{u}$
3. vertex fit for ep data:
   distance $\vec{D}$ of track to 3d vertex
   $\chi^2_{vertex} = \sum_{events} \sum_{tracks} \vec{D}^t \cdot V^{-1} \cdot \vec{D}$



Minimize total χ^2 with respect to 384 alignment parameters

Improvement after Alignment



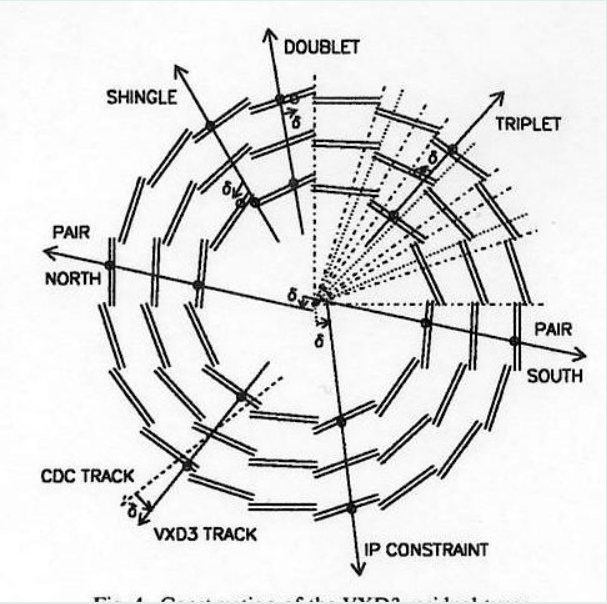
impact parameter resolution for cosmics
before and after alignment

Internal alignment of the SLD vertex detector

D. J. Jackson, D. Su and F. J. Wickens, Internal alignment of the SLD vertex detector using a matrix singular value decomposition technique. Nuclear Instr. Methods **A 491** *(2002) 351-365*

Classification of types of tracking constraints →

Good quality tracks were constrained to pass through two of the CCD hits with the corresponding residual measued to the third, reference, CCD.



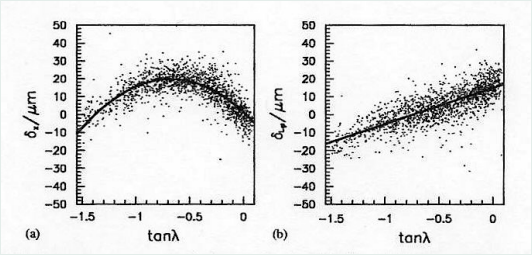
Residual types of the SLD pixel vertex detector

For each type of residuals, n-tuples were accumulated containing the deviation. Data in the n-tuples were fitted to the functional form given below to determine the coefficients and their covariance matrix (using MINUIT).
CCD shape corrections were taken into account from the optical survey data of the CCD surfaces (2108 → 5026 coefficients).

Table 1
Functional forms for fitting the various residual types. The superscripts $\parallel$ and $\perp$ indicate coefficients of fits to residuals measured parallel and perpendicular to the z-axis, respectively. N_I lists the total number of independent residual fits involved while N_C is the number of coefficients determined.

Type	Functional form	N_I	N_C
Shingles	$\delta_z = s_1^{\parallel} + s_2^{\parallel} \tan\lambda + s_3^{\parallel} \tan^2\lambda$	96	288
	$\delta_{L\phi} = s_1^{\perp} + s_2^{\perp} \tan\lambda$	96	192
Doublets	$\delta_z = d_1^{\parallel} + d_2^{\parallel} L_\phi$	48	96
	$\delta_{L\phi} = d_1^{\perp} + d_2^{\perp} L_\phi + d_3^{\perp} L_\phi^2$	48	144
Triplets	$\delta_z = t_1^{\parallel} + t_2^{\parallel} \tan\lambda + t_3^{\parallel} \tan^2\lambda + t_4^{\parallel} L_\phi \tan\lambda + t_5^{\parallel} L_\phi$	80	400
	$\delta_{L\phi} = t_1^{\perp} + t_2^{\perp} L_\phi + t_3^{\perp} L_\phi^2 + t_4^{\perp} L_\phi \tan\lambda + t_5^{\perp} \tan\lambda$	80	400
Pairs	$\delta_{r\varepsilon} = p_1^{\parallel} + p_2^{\parallel} \tan\lambda + p_3^{\parallel} \tan^2\lambda$	28	84
	$\delta_{r\phi} = p_1^{\perp} + p_2^{\perp} \tan\lambda$	28	56
	$\delta_\phi = p_1^{\phi} + p_2^{\phi} \tan\lambda$	28	56
CDC angle	$\delta_\lambda = c_1^{\lambda} + c_2^{\lambda} \tan\lambda + c_3^{\lambda} \tan^2\lambda$	56	168
	$\delta_\phi = c_1^{\phi} + c_2^{\phi} \tan\lambda$	56	112
IP constraint	$\delta_{r\phi} = i_1^{\perp} + i_2^{\perp} \tan\lambda$	56	112
	Total	700	2108

Functional forms for fitting the various residual types.



Examples for residual fits, here as a function of $\tan\lambda$ in a layer before the alignment.

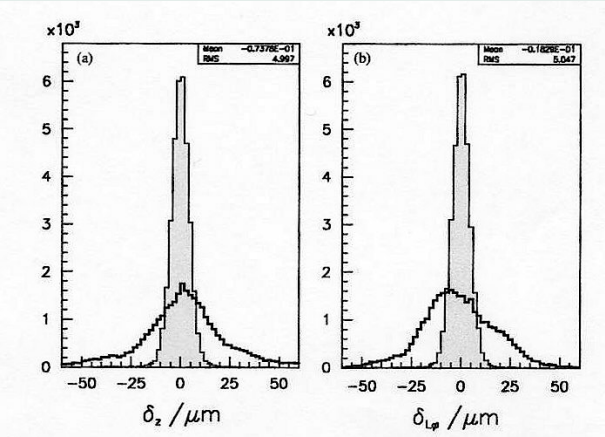
In total there are 2108 coefficients from 700 residual fits (exercise in book-keeping).

Taking into account the covariance matrices of the residual fits, 866 alignment corrections were determined from 5026 coefficients from the residual fits, using SVD techniques for the least-squares fit minimization.

Note: nr of coefficients determined by parametrization types, not by nr of events.

Only a single iteration is required.

With the aligned geometry the design performance is achieved →



Triplet residuals obtained with the original survey geometry and after alignmnent.

Mathematical aspects

From the paper published in 2002:

> "...for the VXD3 alignment (inversion of sparse matrices of order 5000×1000 elements) using double precision arithmetic in modest times on a standard workstation. Although the size of matrices involved appears daunting only $\approx 1\%$ or $\approx 35,000$ elements of the final 5026×866 design matrix $\boldsymbol{A}$ were given non-zero values ..."

Today the size of the matrices should be no problem on a standard PC.

On SVD:

> "...indeed the earliest text-book we found referring to the technique was published in 1983."

G. H. Golub and W. Kahan: Calculating the singular values and pseudo-inverse of a matrix. SIAM J. Numer. Anal Ser. B **2** *(1965) 205-224*
G.H. Golub and C. Reinsch, Singular value decomposition and least squares solutions, Numer. Math. **14** *(1970) 403-420*
J. H. Wilkinson and C. Reinsch, Handbook for Automatic Computation, Vol. II. Springer (1971) (with ALGOL *code)*
C. L. Lawson and R. J. Hanson: Solving Least Squares problems. Prentice-Hall (1974)

In the older psychometric literature, such decompositions were called Eckart-Young decompositions, after *C. Eckart and G. Young, The approximation of one matrix by another of lower rank. Psychometrika* **1** *(1936) 211-218.*

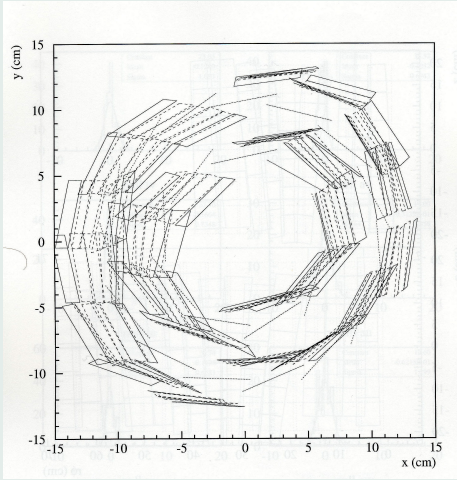
Alignment of the upgraded VDET at LEP2 (ALEPH)

A. Bonissent et al., Alignment of the upgraded VDET at LEP2, ALEPH 97-116

A global χ^2 involving all $864 = 144 \times 6$ degrees of freedom is built, using single tracks and vertex constraints. Selected information is used from the outer tracking.

Precise faces measurement are used to reduce the degrees of freedom, while allowing for parametrized distortions.

The method is shown to provide accurate results, even with a limited number of events.



Local alignment of the VDET in the $r\phi$ view. The dashed lines show the nominal position. The end wafers are drawn with solid lines, the other with dashed lines. Displacements are amplified by a factor of 100.

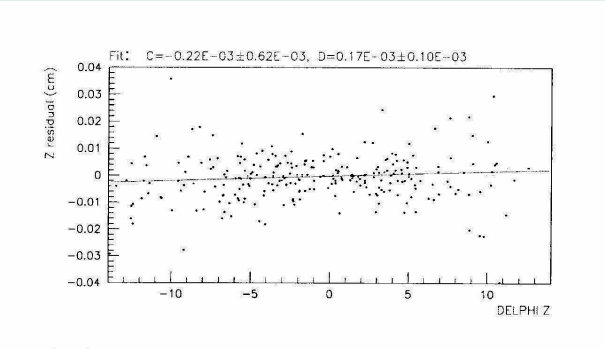
Delphi: final alignment of the Barrel Silicon Tracker

P. Brückmann de Renstrom: The Final Alignment of the Barrel Silicon Tracker at LEP2, DELPHI *2004-047 TRACK 098*

Fitting the alignment parameters to the track-hit residual distributions (with reference layers).

Individual parametrizations of hit residuals fitted using MINUIT, including flat background from miss-associated hits.

Iterative sequence of applying geometry corrections.



A fit to the di-muon residual distribution.

$$\delta_z = C + D \times \cot\theta + e \times r \times \cot\theta^2$$
$$\oplus \quad \text{flat background}$$

Final average single-layer resolution in $R\phi$ direction better than 8 μm.

Alignment using Kalman filter

CMS: About 20 000 silicon sensors, with resolution 10 μm to 40 μm. Expected precision from mechanical mounting and LASER beam alignment worse than intrincic resolution – alignment using tracks is necessary.

Extension of the Kalman filter method of track fitting, with updating the current alignment parameters after each track.

R. Frühwirth, T. Todorov and M. Winkler, Estimation of Alignment Parameters, using the Kalman Filter with annealing, CMS Note 2002-008

R. Frühwirth, T. Todorov and M. Winkler, Estimation of Alignment Parameters using the Kalman Filter with annealing, Journal of Physics G: Nuclear and Particle Physics **29** *(2003) 561–574*

Update formulae for global parameters $\boldsymbol{a}$ and their covariance matrix $\boldsymbol{E}$:

$$\boldsymbol{a}_1 := \boldsymbol{a}_0 + \boldsymbol{E}_0 \boldsymbol{D}^{\mathrm{T}} \boldsymbol{W} \left[\boldsymbol{m} - \boldsymbol{f}(\boldsymbol{p}_0, \boldsymbol{a}_0)\right] \qquad \text{with} \quad \boldsymbol{W} = \left[\boldsymbol{V} + \boldsymbol{H}\boldsymbol{C}_0\boldsymbol{H}^{\mathrm{T}} + \boldsymbol{D}\boldsymbol{E}_0\boldsymbol{D}^{\mathrm{T}}\right]^{-1}$$
$$\boldsymbol{E}_1 := \boldsymbol{E}_0 - \boldsymbol{E}_0 \boldsymbol{D}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{D} \boldsymbol{E}_0$$

. . . requires some matrix operations.

Convergence to local minima not excluded: introduction of "annealing" = gradually stepping up the weights of the observations in the course of the estimation process (not "simulated annealing").

Simulated annealing is a method, where, depending on a temperature T and some BOLTZMANN factor, a step can result in an *increase* of an energy function.

Further references

Danny Hindson, A Robust Procedure for Alignment of the ATLAS Inner Detector Using Tracks, Thesis, Oxford (2004)

R. Harr et al., Tracking Detector Alignment Using Constrained Vertex Fits, IEEE Transactions on Nuclear Science, Vol 41 (1994) 796

K. Abe et al., Design and performance of the SLD detector, a 307 Mpixel tracking system, Nucl. Instr. and Methods **A 400** *(1997) 287–343*

R. McNulty, T. Shears and A. Skiba, A Procedure for the Software Alignment of the CDF Silicon System, `CDF/DOC/TRACKING/GROUP/5700`

B. Mours et al., The design, construction and performance of the ALEPH silicon vertex detector. Nucl. Instr. and Methods **A 379** *(1996) 101–115*

V. Karimäki, A. Heikkinen, T. Lampén and T. Lindén, Sensor alignment by tracks, CMS Conference Report, CHEP03, La Jolla, California, March 24-28, 2003

7. Matrix inversion

Matrix inversion is an essential part of minimisation. Inversion is a n^3 process and can be time consuming for large matrix dimension. Complete matrix inversion fails for *singular* matrices and is *inaccurate* for almost singular matrices.

From HEP publications:

> "For experiments with many data points, the inversion of such large matrices may lead to numerical instabilities, in addition to being time-consuming."

> "Minimizing χ^2 ...is impractical because it involves the inversion of the measurement covariance matrix which, in global fits, tends to be very large."

Matrices to be inverted in statistical computation are *symmetric* and represent covariance/Hessian matrices:

$$\Delta \boldsymbol{a} = \boldsymbol{C}^{-1}\boldsymbol{b} \qquad \boldsymbol{C}^{-1} = \text{ covariance matrix of the parameters}$$

The result will reflect the statistical properties of data and model!

- The storage and the computation can make use of the symmetry ($\rightarrow$ factor 1/2).
- Matrices with highly correlated parameters are almost singular.
- Highly correlated parameters should be avoided (and are not meaningful). A strategy to get relevant results in difficult cases can be used.

Matrix inversion – timing

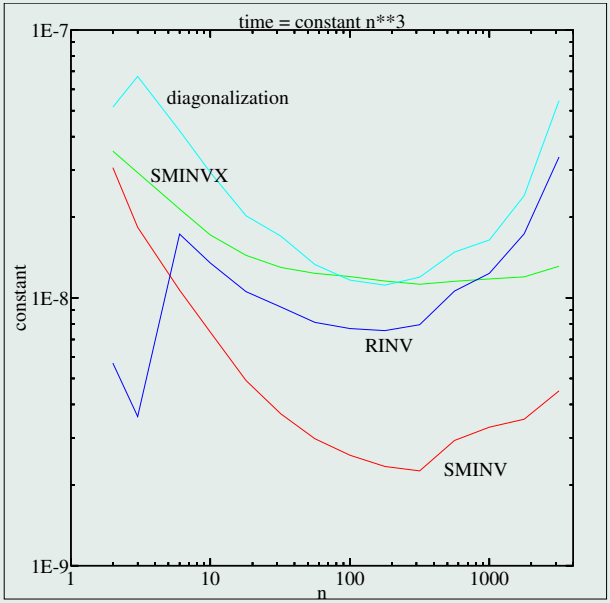
Time for inversion of a n-by-n matrix is expected to be $\approx \mathcal{O}(n^3)$.

$n =$	**RINV**	**SMINV**	**HHLROT**	unit
10	13.5	7.4	29.4	μsec
100	7.7	2.6	11.6	msec
1000	12.4	3.3	16.4	sec
3162	17.9	2.4	28.7	min
words	n^2	$1/2\ n^2$	$3/2\ n^2$	

Inversion with $n = 25000$ will take $\approx$ one day (SMINV) and requires 1.25 GB.

Atlas benchmark on 16-proc cluster: 95 hours for $n = 8000$ in double precision.

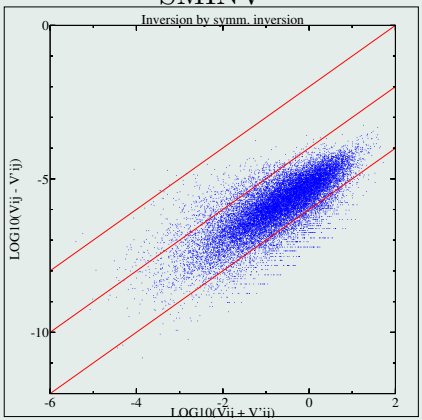
time $=$ constant $\times\ n^3$



2.6 MHz Pentium with 512 MB; single precison computation.

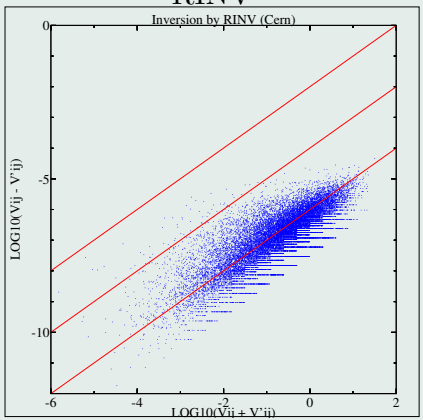
Check of accuracy based on $\boldsymbol{V}' = \left(\boldsymbol{V}^{-1}\right)^{-1}$. Plots show $\log_{10}$ of *difference* versus *sum* of elements; lines correspond to 10^{-2} 10^{-4} 10^{-6}.

SMINV



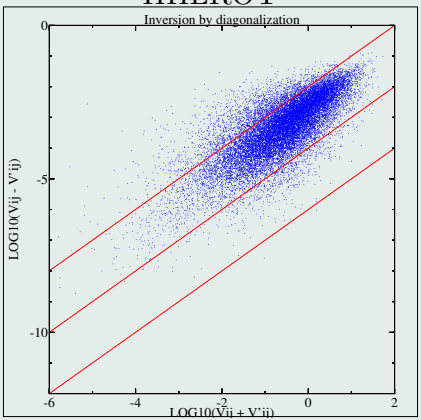
$\varepsilon \approx 10^{-5}$

RINV



$\varepsilon \approx 10^{-6}$

HHLROT



$\varepsilon \approx 10^{-3}$

Highest precision by RINV (Cern); lowest precision for inversion by diagonalisation.

Matrix programs

RINV Cern-Library program for full matrices, using triangular factorization with row interchange (special code for $n \leq 3$). Returns flag for singularity, but singularity will often go undetected.

SMINVX Special Gauss-Jordan algorithm for **symmetric matrices** with pivot selection on diagonal, and singularity detection.

SMINV Same as **SMINVX**, but with index calculation avoiding integer multiply and up to a factor of 3 faster.

Algorithm: use always largest pivot element (on diagonal), but avoid elements with very large $(\boldsymbol{V})_{jj} \cdot \left(\boldsymbol{V}^{-1}\right)_{jj}$. Stop inversion if no acceptable pivot can be found and clear corresponding matrix elements, i.e. invert the largest possible submatrix, and return zero correction for remaining parameters.

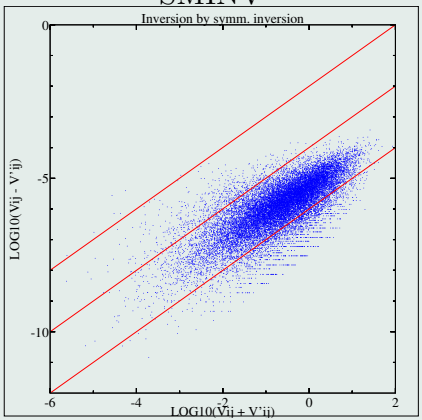
HHLROT Diagonalization (eigenvalues + eigenvectors) by Householder transformation followed by diagonalization of tridiagonal matrices. Allows to recognize insignificant components of the solution from eigenvalues.

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.

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

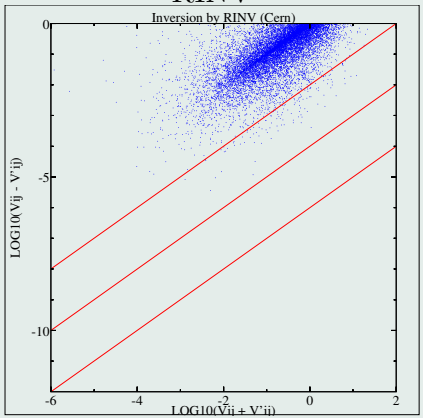
Precision of result for matrix, made singular with rank defect of 1.

SMINV



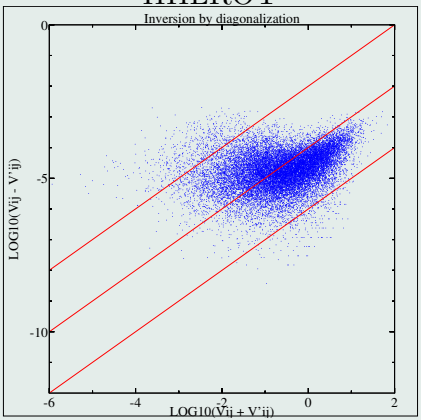
$\varepsilon \approx 10^{-5}$

RINV



IFAIL $= -1$

HHLROT



$\varepsilon \approx 10^{-3}$

RINV (Cern) fails without result. Other algorithms have still useful result for 999 by 999 submatrix, with unchanged precision.

Solution of very large least squares problems

What is the best method to solve a very large least squares problem?

Orthogonal decomposition by singular value decomposition (SVD)

Advantage: with single-precision arithmetic as accurate as other methods with double-precision.

Disadvantage: requires the full input matrix in memory.

Another method which allows "sequential" input of data:

Transformation by Householder triangularization to triangular form without requiring that the entire matrix be in computer storage.
C. L. Lawson and R. J. Hanson: Solving Least Squares problems. Prentice-Hall (1974)

8. Very large number of parameters

For 10 000 parameters a double-precision symmetric matrix $\boldsymbol{C}$ (symmetric storage mode) with 400 Mbytes of memory is required – this can still be handled w.r.t. to space and computing time (several hours).

Is there a strategy for the case of e.g. 100 000 parameters to solve a matrix equation

$$\boldsymbol{C}\Delta\boldsymbol{a} = \boldsymbol{b}$$

for $\Delta\boldsymbol{a}$ using a standard PC?

- All diagonal elements are non-zero, but off-diagonal element C_{ij} is non-zero only, if the parameter with index i and with index j appear both in a measurement.
- Usually the matrix $\boldsymbol{C}$ is a sparse matrix, where only a small fraction of elements is non-zero.
- Unfortunately the inverse matrix $\boldsymbol{C}^{-1}$ is a full matrix (all elements non-zero), even if the matrix $\boldsymbol{C}$ is a sparse matrix.
- But there are efficient iterative methods for the solution of the matrix equation (i.e. determination of $\Delta\boldsymbol{a}$) which only need the matrix product

$$\boldsymbol{C}\boldsymbol{v} \qquad\qquad \boldsymbol{v} = \text{ vector .}$$

CG methods for linear systems

The solution of the matrix equation (matrix symmetric, positive definite) and of the quadratic minimization problem are equivalent:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \qquad \text{minimize } f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}\ ,$$

because the minimum of $f(\boldsymbol{x})$ is determined by grad $f(\boldsymbol{x}^*) = \boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b} = 0$.

Vectors $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots \boldsymbol{d}_n$ are called conjugate w.r.t. matrix $\boldsymbol{A}$, if

$$\boldsymbol{d}_i^{\mathrm{T}} A \boldsymbol{d}_j = 0 \qquad \text{for } i, j = 1, 2, \ldots n \quad i \neq j$$

The minimum of $f(\boldsymbol{x})$ is reached after at most n one-dimensional minimizations along the directions $\boldsymbol{d}_i$, i.e. of $f(\boldsymbol{x_k} + t \cdot \boldsymbol{d}_k)$ with

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_k \cdot \boldsymbol{d}_k \qquad \text{with} \quad t_k = -\frac{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{d}}{\boldsymbol{d}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{d}_k} \qquad \boldsymbol{g} = \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b}$$

A good approximation is usually found already after $\ll n$ steps.

The convergence is improved by *preconditioning* (scaling of the elements of $\boldsymbol{A}$).

The method of generalized minimized residuals (GMRES) is even more efficient (H. R. Schwarz, Numerische Mathematik, Teubner 1993).

Indexed storage of sparse matrices

Many different schemes exist, which all allow the **product** of the sparse matrix $\boldsymbol{C}$ of logical size $N \times N$ with a vector.

Numerical Recipes Scheme: Use a real array `S(.)` and an integer array `IND(.)` of same length.

- The first N locations of `S(.)` store the diagonal elements.
- Each of the first N locations of `IND(.)` stores the index of the array `S(.)`, that contains the first off-diagonal element of the corresponding row of the matrix.
- Location 1 of `IND(.)` is always equal to $N + 2$.
- Location $N + 1$ of `IND(.)` is one greater than the index in `S(.)` of the last off-diagonal element of the last row. Location $N + 1$ of `S(.)` is not used.
- Entries in `S(.)` at locations $\geq N + 2$ contain the off-diagonal values, ordered by row and, within each row, by columns.
- Entries in `IND(.)` at locations $\geq N + 2$ contain the column numbers of the corresponding elements in `S(.)`.

This *row-indexed sparse storage mode* is optimized to multiply a vector to its right.

9. Conclusions and summary

Different types of alignment algorithms are in use in HEP experiments:

- Simple residual methods (which are simple, but may require many iterations).

  "The handle on alignment comes from the **residual distributions**." [Atlas report]

- Sequential and iterative ("classical") methods: hierarchy of parametrizations of residual distributions (residuals between the track extrapolation and the recorded cluster) and overall fit, iteratively.

- Simultaneous fits to (global) alignment parameters and (local) track parameters (MILLEPEDE).

Some recommendations:

- integration of alignment into reconstruction code;
- simultaneous use of all relevant detectors, but not too many parameters (orthogonality!);
- simultaneous use of several (all) types of events and data – physics data: single tracks, vertices, invariant-mass constraints, and cosmics, overlaps ...

Summary

Present and future detectors are very large and very accurate –
the requirements for good alignment and calibration are increasing.

Sometimes an observer has the impression that the alignment of certain objects can be improved with appropriate methods.

Data, computing power and also methods should be available to reach the goal.

Millepede: *any of numerous herbivorous nonpoisonous arthropods having a cylindrical body of 20 to 100 or more segments most with two pairs of legs.*

Software Alignment for Tracking Detectors