2nd LHC Detector Alignment Workshop 25th and 26st June 2007

Millepede II

Volker Blobel – Universität Hamburg

Abstract

The large track detectors of the LHC experiments require an accurate alignment with the determination of several 10 k parameters in order to allow to make use of the potential high spatial resolution, necessary for the physics goals. The experiment-independent Millepede program performs a simultaneous fit of (global) alignment parameters and (local) track parameters, and allows to include e.g. laser and survey data and equality constraints in the fit. The Millepede II version, now on the web, uses fast methods in the non-iterative fit.

Design: experiment-independent program, not specific to alignment and tracks.

	Year	What happened?
Millepede I	1996	First studies at CERN (Opal)
	1997	First version used in H1 (with simultaneous fit)
	1998	Used in H1 for Vertex det. and Central Jet Chamber
	1999	Used with up to 4 800 parameters (HERAb)
	2000	Millepede I on the web, last program change
	2001	
	2002	
	2003	
	2004	
Millepede II	2005	Start of new development for large nr of parameters
	2006	Test with H1 and cms data, up to 50 k parameters
	2007	Millepede II on the web (25.th May)

```
tar -xzf Mptwo.tgz
make
./pede -t
```

MILLEPEDE I used by: H1, ZEUS, HERAb, CMS, LHCb, ALICE, PHENIX, STAR ...

⇒ Talk by M. Stoye: Track-based alignment of the CMS Tracker with Millepede II (includes studies on χ^2-invariant deformations)

Goal of Millepede II development

Version II should align a track detector, within hours, with:
100 000 alignment parameters, 100 constraints, Million tracks (+ Laser + survey data)

Construct and **minimize** "global" objective function $F(\boldsymbol{p}, \boldsymbol{q})$, which depends on the alignment corrections $\boldsymbol{p}$ and **all** track parameters $\boldsymbol{q}$ and ...

$$F(\boldsymbol{p}, \boldsymbol{q}) = \frac{1}{2} \sum_{\text{data sets}} \left[\sum_{\text{events}} \left(\sum_{\text{tracks}} \left(\sum_{\text{hits}} \Delta_i^2 / \sigma_i^2 \right) \right) \right] + \sum \left[\text{terms depending on Laser data and Survey data} \right]$$

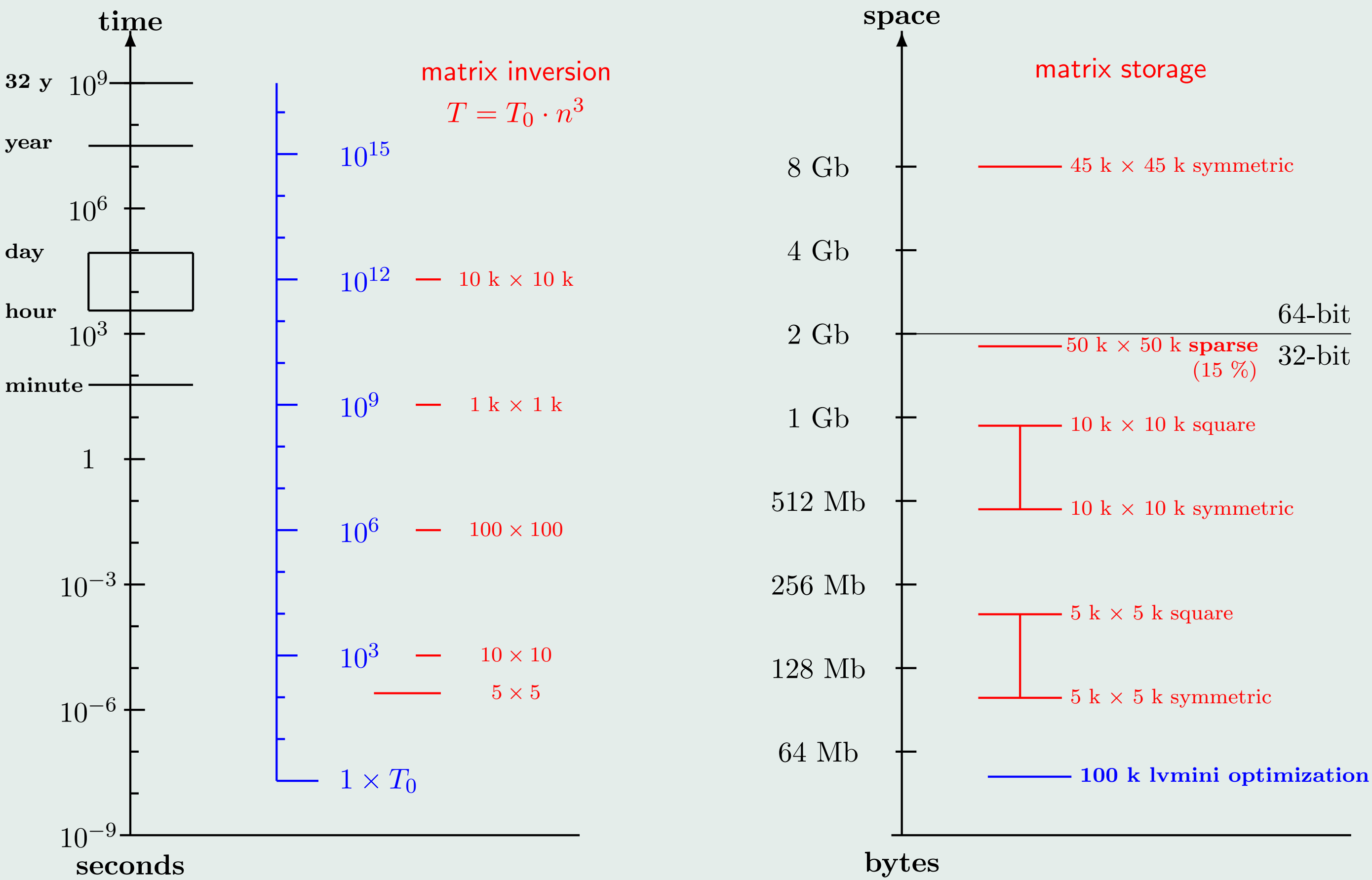
with fastest and most precise method [⇒] (References):

- Simultaneous fit of all alignment and local (track, Laser, ...) parameters (Millepede principle) in a single step, using large Hessian matrix in global fit,
- introduction of constraints; possible (only) with global fit, and
- include detailed outlier treatment: reject or down-weight bad data (method of M-estimates – no pure least squares fit).[⇒]
  Note: initial deviations may be large due to misalignment!

Note: standard methods require space $\propto n^2 \rightarrow$ 80 Gbyte and cpu-time $\propto n^3 \rightarrow$ 1 year

Translation table ...

HE Physics $\Rightarrow$	$\Leftarrow$ Mathematics, Statistics
χ^2-function, χ^2 formalism	objective function (log-likelihood function) ....
constraint	measurement-term in objective function ....
? (exact constraint)	constraint (equality, inequality constraints) ....
unconstrained parameters	undefined, ill-defined parameters ....
unbiased residual	? ....
pull = residual$/\sigma_m$	pull = residual$/\sqrt{\sigma_m^2 - \sigma_f^2}$ ....
"linear system of equations requires inversion"	"never solve a system of equations by inversion"
"solve 4200 equations in 4200 unknowns: computational infeasible; even worse, non-linear fit wont't converge"	"current algorithms ... for generally constrained optimization routinely solve systems in the tens and, perhaps even, hundreds of thousands of unknowns and constraints" ....


time
32 y
10^9
year
10^6
day
hour
10^3
minute
1
10^{-3}
10^{-6}
10^{-9}
seconds
matrix inversion
$T = T_0 \cdot n^3$
10^{15}
10^{12}
10 k × 10 k
10^9
1 k × 1 k
10^6
100 × 100
10^3
10 × 10
5 × 5
$1 \times T_0$
space
matrix storage
8 Gb
45 k × 45 k symmetric
4 Gb
64-bit
2 Gb
50 k × 50 k **sparse** (15 %)
32-bit
1 Gb
10 k × 10 k square
512 Mb
10 k × 10 k symmetric
256 Mb
5 k × 5 k square
128 Mb
5 k × 5 k symmetric
64 Mb
100 k lvmini optimization
bytes

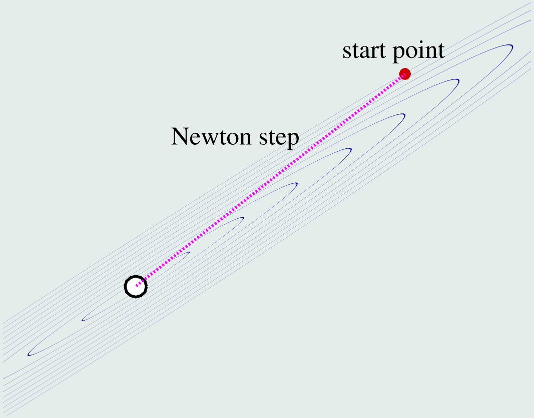
(0) Construction of objective function $F(\boldsymbol{x})$.
$\boldsymbol{x} \in \mathcal{R}^n$ with start value $\boldsymbol{x}_0$.

(1) Quadratic model of $F(\boldsymbol{x})$:
$\rightarrow M_k(\boldsymbol{d}) = F_k + \boldsymbol{d}^{\mathrm{T}} \boldsymbol{\nabla} F_k + \frac{1}{2} \boldsymbol{d}^{\mathrm{T}} \boldsymbol{C}_k \boldsymbol{d}$

(2) Newton-step $\boldsymbol{d}_k$ from $\boldsymbol{C}_k \, \boldsymbol{d}_k = -\boldsymbol{\nabla} F_k$
expected decrease $\delta F = \frac{1}{2} \boldsymbol{d}^{\mathrm{T}} \boldsymbol{\nabla} F_k$
improve by line-search $\phi(\alpha) \equiv F(\boldsymbol{x}_k + \alpha \cdot \boldsymbol{d}_k)$
$\rightarrow$ never divergent!
$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \cdot \boldsymbol{d}_k$ and $k := k + 1$
repeat (1) and (2) until δF small

(3) Covariance matrix $= \boldsymbol{C}^{-1}$

$F(\boldsymbol{x})$ with large correlation



Matrix-based Newton method: making use of matrix $\boldsymbol{C}$, quadratic function is minimized by one step, non-quadratic function minimized with *quadratic convergence* rate.

Methods without matrix: e.g. parameter variation and steepest-descent (no matrix $\boldsymbol{C}$) with only linear convergence are slow; the convergence may never occur as the iteration stagnates (can be misinterpreted as indication for convergence).

Reduction of matrix size

Hessian matrix $\boldsymbol{C}$ for simultaneous fit of (global) alignment *and* track parameters, of order $(n_{\text{global}} + 5 \times n_{\text{tracks}}$, can be reduced to order n_{global} (MILLEPEDE principle, simple formalism from linear algebra, based on *Schur complement*). [⇒]

Element $(\boldsymbol{C})_{jk} \neq 0$, if parameters j and k appear together in a track:

$$\boldsymbol{C} = \begin{pmatrix} {\color{red}x} & & x & x & x & x & & x & x \\ & {\color{red}x} & x & x & & x & x & x & \\ x & x & {\color{red}x} & & & x & & x & x \\ x & x & & {\color{red}x} & x & & & & x \\ x & & & x & {\color{red}x} & & x & x & x \\ x & x & x & & & {\color{red}x} & x & & \\ & x & & & x & x & {\color{red}x} & & x \\ x & x & x & & x & & & {\color{red}x} & x \\ x & & x & x & x & & x & x & {\color{red}x} \end{pmatrix}$$

Matrix is *sparse*: fraction q of non-diagonal elements $\neq 0$, with $q = 2\% \ldots 15\%$.

$$\boldsymbol{C}^{-1} = \begin{pmatrix} {\color{red}x} & x & x & x & x & x & x & x & x \\ x & {\color{red}x} & x & x & x & x & x & x & x \\ x & x & {\color{red}x} & x & x & x & x & x & x \\ x & x & x & {\color{red}x} & x & x & x & x & x \\ x & x & x & x & {\color{red}x} & x & x & x & x \\ x & x & x & x & x & {\color{red}x} & x & x & x \\ x & x & x & x & x & x & {\color{red}x} & x & x \\ x & x & x & x & x & x & x & {\color{red}x} & x \\ x & x & x & x & x & x & x & x & {\color{red}x} \end{pmatrix}$$

Inverse matrix = covar. matrix would be dense matrix: correlation between each index pair $\neq 0$.

If track parameters fixed:

Element $(\boldsymbol{C})_{jk} \neq 0$, if parameters j and k appear together at a measured point:

$$\boldsymbol{C} = \begin{pmatrix} {\color{red}x} & & & & & & & & \\ & {\color{red}x} & & & & & & & \\ & & {\color{red}x} & & & & & & \\ & & & {\color{red}x} & & & & & \\ & & & & {\color{red}x} & & & & \\ & & & & & {\color{red}x} & & & \\ & & & & & & {\color{red}x} & & \\ & & & & & & & {\color{red}x} & \\ & & & & & & & & {\color{red}x} \end{pmatrix}$$

Matrix is (block)-diagonal.

$$\boldsymbol{C}^{-1} = \begin{pmatrix} {\color{red}x} & & & & & & & & \\ & {\color{red}x} & & & & & & & \\ & & {\color{red}x} & & & & & & \\ & & & {\color{red}x} & & & & & \\ & & & & {\color{red}x} & & & & \\ & & & & & {\color{red}x} & & & \\ & & & & & & {\color{red}x} & & \\ & & & & & & & {\color{red}x} & \\ & & & & & & & & {\color{red}x} \end{pmatrix}$$

(Block)-diagonal: correlations ignored.

Minimization with constraints

Constraint equations for m linear (**equality**) constraints described by

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c} \qquad (\boldsymbol{A} \text{ has } m \text{ rows})$$

Task: minimize $F(\boldsymbol{x})$ subject to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}$

Joseph-Louis Lagrange
(1736 – 1813)

Step $\boldsymbol{d}$ calculation with Lagrange method: introduce m multipliers $\boldsymbol{\lambda}$

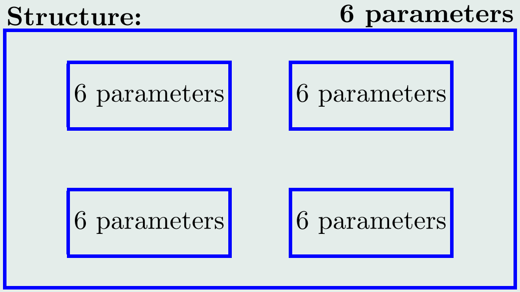
$$\mathcal{L}(\boldsymbol{x}) = F(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{T}} (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{c}) \qquad \left(\begin{array}{c|c} \boldsymbol{C} & \boldsymbol{A}^{\mathrm{T}} \\ \hline \boldsymbol{A} & 0 \end{array} \right) \left(\begin{array}{c} \boldsymbol{d} \\ \hline \boldsymbol{\lambda} \end{array} \right) = \left(\begin{array}{c} -\boldsymbol{\nabla} F \\ \hline \boldsymbol{c} \end{array} \right)$$

Matrix equation has unique solution (for sufficient constraints) even for singular matrix $\boldsymbol{C}$.[⇒]

Why constraints?

- Remove singularity of matrix – **essential**
  Constrain overall translation + rotation to zero.

- Introduction of structural constraints – **optional**
  Parameters for larger unit + for all individual sensors
  with overall zero-effect of individual sensors (constraint)
  → individual sensors can be fixed for quick check, using reduced
  nr of parameters and tracks.



$(6 + 4 \times 6)$ parameters $-\,6$ constraints

Solution of matrix equation in Millepede II

Sparse matrix structure is constructed dynamically from the data, with solution by ...

GMRES= generalized minimal residuals*)

solve $\boldsymbol{C}\,\boldsymbol{x} = \boldsymbol{y}$ or minimize $\|\boldsymbol{C}\,\boldsymbol{x} - \boldsymbol{y}\|_2$ (needs only product $\boldsymbol{C} \times$ vector)

Fast for sparse (and dense) matrix $\boldsymbol{C}$. Iterative method is related to *conjugate gradients* and to Lanczos tridiagonalization; convergence speed depends on eigenvalue spectrum.

Convergence is accelerated by *preconditioning*. In Millepede II the variable-band matrix Cholesky decomposition [⇒] is recommended for preconditioning.

Example:

solution takes **10 minutes** (factor 5000 faster than inversion) for 50 000 parameters plus 130 constraints.

... allows to calculate, for selected parameters, the standard deviation and the global correlation.[⇒]

Also direct methods for sparse exist: MA27 (1983) and MA57 (2004) (variant of Gaussian elimination, Schur complement)

Other methods in MP II: inversion, diagonalization [⇒], variable-band matrix Cholesky decomposition [⇒];
methods may include large number of constraints (Lagrange).

*) 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` **Software MINRES from July 2003**

Cpu-times for alignment

Approximate formula for cpu-time:

$$\textbf{cpu-time} = T(N_{\text{tracks}}, N_{\text{parameters}}) = N_{\text{iterations}} \times \left(\alpha \cdot N_{\text{tracks}} + \beta \cdot N_{\text{parameters}}^{\gamma} \right) \qquad \gamma \geq 1$$

Values of α, β, γ, $N_{\text{iterations}}$ depend on algorithm.

Experiment	lang	$N_{\text{parameters}}$	N_{tracks}	$N_{\text{iterations}}$	$N_{\text{constraints}}$	**cpu-time** (+ remarks)
	Global non-iterative method: (simultaneous alignment and track parameter fit)					
Millepede (cms) (M. Stoye study)	F77	50 000	> 3 Mio		130	1:40 hour + file times (below 2 Gbyte)
	Local methods: (track parameters fixed, bias removed by large number of iterations)					
D0	C^{++}	6 000	0.7 Mio	70 – 100	–	1 – 3 days
BaBar Si tracker	C^{++}	1 440	?	$\sim$ 100	–	< 24 hours

3. Using Millepede II for alignment

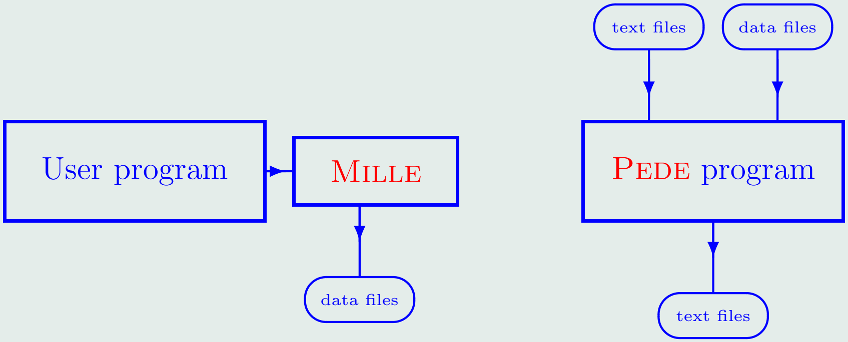
Input = sets of single measured data points from local fits (e.g. Kalman fit [ZEUS] with track hits):

$$y_i = \underbrace{f(x_i, \boldsymbol{q}, \boldsymbol{p})}_{\text{fit function}} + \underbrace{\sum_{j=1}^{\nu} \left(\frac{\partial f}{\partial q_j} \right) \Delta q_j}_{\text{local derivatives}} + \underbrace{\sum_{\ell \in \Omega} \left(\frac{\partial f}{\partial p_\ell} \right) \Delta p_\ell}_{\text{global derivatives}} + \epsilon \qquad \epsilon \in N(0, \sigma_i^2); \ \ell = \underbrace{\text{parameter label}}_{\text{positive integer}}$$

Derivatives express the change of residual $z_i = y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p})$, if q_j or p_ℓ is changed by Δq_j or Δp_ℓ.

(1) File with $z_i = y_i - f(x_i, \boldsymbol{q}, \boldsymbol{p})$, σ_i and all derivatives written within user program by Mille (by parallel processing).

Allows to repeat local fits (only last iteration) in Pede.



(2) Data files are processed in stand-alone program Pede, steered by text files: [⇒]

- select files and solution method (inversion, diagonalization, fast sparse method . . .)
- information on measurement of linear combinations of global parameters (e.g. survey data)
- status of global parameters (e.g initial values, fixed/variable, presigma)

Note: **all parameters are correlated**, and isolated optimization of a subset may distort results.

Alignment and calibration parameters:

- simultaneous fit of **all** parameters, no separate calibration or alignment of detector parts;
- include calibration of e.g. Lorentz angle, local values of drift velocity, T_0-values, coefficients for correction functions;
- include beam parameters: vertex position, beam direction;
- but: do not include too many (and ill-defined) parameters.

Use realistic data model $f(.,.)$ for the detector, i.e. understand the detector properties in detail, and adjust assumed accuracy of the detector parts.

A wrong "component" in the track data model may introduce alignment distortions!

- Simultaneous use of **all** available data types:
  - ⋆ (normal) **tracks**
  - ⋆ 2-track particles (given mass)
  - ⋆ tracks with common vertex
  - ⋆ cosmics (incl. horizontal)
  - ⋆ $\boldsymbol{B} = 0$ cosmics
  - ⋆ halo muons
  - ⋆ Laser data
  - ⋆ survey data (temperatur effect?)

  to reduce potential distortions; matrix becomes denser!
  (check accuracy of track reconstruction of "unusual" tracks (off-vertex tracks, halo muons)).
- linear equality constraints to fix undefined degrees of freedom (translation, rotation)
- (optionally) define detector parameter structure by constraints;
- adjust measurement accuracy (for aligned detector);
- outlier rejection and down-weighting of bad single hits (has to be adjusted).

Analyse alignment result and look for potential distortions or deformations (⇒ Talk by M.Stoye), and

- eventually introduce further constraints to fix weakly defined linear parameter combinations.

Summary

New MILLEPEDE II on the web

Download from: `www.desy.de/~blobel/` into fresh directory:

```
tar -xzf Mptwo.tgz
make
./pede -t
```

Use of $>$ 400 Mbyte memory requires to change 1 statement in code + makefile for 64-bit system.

MILLEPEDE II can be used:

- Feedback welcome and necessary!
- Perhaps several (small) changes during the coming weeks (if feedback $\neq 0$).
- Addition of
  - L-BFGS method, for even larger number of parameters(?)[⇒] ,
  - another solution method(?),
  - histogram viewer for histogram file?
- Do not forget to understand your detector.

References

[1] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer Series in Operations Research, Springer (1999)

[2] W.C. Davidon, *Variable metric method for minimization*, manuscript (1958), finally published SIAM J. Optimization **1** (1991) pp. 1-17.

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

[4] J.J.Moré and D.J. Thuente, *Line search algorithms with guaranteed sufficient decrease*, ACM Transactions on Mathematical Software **20** (1994), pp. 286-307

[5] F. James and M. Roos, Minuit, Function Minimization and Error Analysis, Reference Manual, CERN Program Library Long Writeup D506 (1994)
F. James and M. Winkler, Minuit Users Guide (C^{++} Version), CERN (2004)

[6] Ph.E. Gill et al., *Practical Optimization*, Academic Press (1981)

[7] J.F. Bonnans et al., *Numerical Optimization – Theoretical and Practical Aspects*, Springer (2000)

[8] I.S. Duff et al., *Direct Methods for Sparse Matrices*, Oxford Science Publ. (1986)

[9] H.R. Schwarz, *Numerische Mathematik*, Teubner (1993)

[10] P.J. Rousseeuw and A.M. Leroy, *Robust Regression and Outlier Detection*, Wiley (2003)

[back]

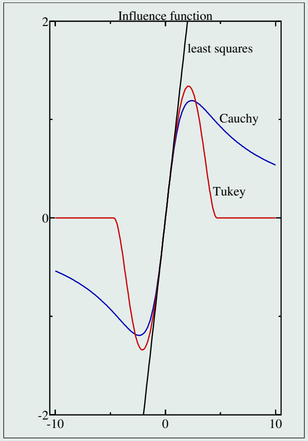
The presence of outliers in the data can deteriorate the alignment result.
Difficulty: wrong initial alignment parameters can fake outliers.

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

Millepede II: Same as Millepede I, in addition technique of M-estimates applied to local fits, after the first iteration.

M-estimates. 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 (used in local fits).

$z =$ residual/std. deviation	$\rho(z) = \ln \text{pdf}(z)$	influence function $\psi(z) = \text{d}\rho(z)/\text{d}z$	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 $\begin{cases} \text{if } \lvert z\rvert \le c = 1.345 \\ \text{if } \lvert z\rvert > c = 1.345 \end{cases}$	$= \begin{cases} z^2/2 \\ c\left(\lvert z\rvert - c/2\right) \end{cases}$	$= \begin{cases} z \\ c \cdot \text{sign}\,(z) \end{cases}$	$= \begin{cases} 1 \\ c/\lvert z\rvert \end{cases}$



[back]

The Hessian $\boldsymbol{C}$ of a simultaneous fit of 100 000 global parameters – a square $100\,000 \times 100\,000$ matrix – and of 1 Mio tracks with 5 parameters each – 1 Mio 5×5 matrices.

The Hessian $\boldsymbol{C}_{\text{total}}$ is in total a $5\,100\,000 \times 5\,100\,000$ matrix (100 Terabytes) …

$$
\boldsymbol{C}_{\text{total}} = \left(\begin{array}{ccccccccc|ccc|ccc|ccc|ccc|c}
\textcolor{red}{x} &&&&&&&&&&&&&&& x & x & x &&&& \cdots \\
& \textcolor{red}{x} &&&&&&&& x & x & x &&&&&&&&&& \cdots \\
&& \textcolor{red}{x} &&&&&&&&&&&&& x & x & x &&&& \cdots \\
&&& \textcolor{red}{x} &&&&&&&&& x & x & x &&&& x & x & x & \cdots \\
&&&& \textcolor{red}{x} &&&&&&&& x & x & x &&&&&&& \cdots \\
&&&&& \textcolor{red}{x} &&&& x & x & x &&&&&&&&&& \cdots \\
&&&&&& \textcolor{red}{x} &&&&&& x & x & x &&&&&&& \cdots \\
&&&&&&& \textcolor{red}{x} &&&&&&&&&&& x & x & x & \cdots \\
&&&&&&&& \textcolor{red}{x} &&&&&&&&&& x & x & x & \cdots \\
\hline
& x &&&& x &&&& x & x & x &&&&&&&&&& \\
& x &&&& x &&&& x & x & x &&&&&&&&&& \\
& x &&&& x &&&& x & x & x &&&&&&&&&& \\
\hline
&&& x & x && x &&&&&& x & x & x &&&&&&& \\
&&& x & x && x &&&&&& x & x & x &&&&&&& \\
&&& x & x && x &&&&&& x & x & x &&&&&&& \\
\hline
x && x &&&&&&&&&&&&& x & x & x &&&& \\
x && x &&&&&&&&&&&&& x & x & x &&&& \\
x && x &&&&&&&&&&&&& x & x & x &&&& \\
\hline
&&& x &&&& x & x &&&&&&&&&& x & x & x & \\
&&& x &&&& x & x &&&&&&&&&& x & x & x & \\
&&& x &&&& x & x &&&&&&&&&& x & x & x & \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots &&&&&&&&&&&&& \ddots
\end{array} \right)
\longrightarrow \quad
\boldsymbol{C}_{\text{global}} = \begin{pmatrix}
\textcolor{red}{x} && x &&&&&& \\
& \textcolor{red}{x} &&&& x &&& \\
x && \textcolor{red}{x} &&&&&& \\
&&& \textcolor{red}{x} & x && x & x & x \\
&&& x & \textcolor{red}{x} && x && \\
& x &&&& \textcolor{red}{x} &&& \\
&&& x & x && \textcolor{red}{x} && \\
&&& x &&&& \textcolor{red}{x} & x \\
&&& x &&&& x & \textcolor{red}{x}
\end{pmatrix}
$$

Element $(\boldsymbol{C}_{\text{global}})_{jk} \neq 0$, if parameters j and k in same local fit.

Note: the inverse of a sparse matrix $\boldsymbol{C}_{\text{global}}$ (= covariance matrix) is dense; **all parameters are correlated!**

… is reduced to a (sparse) $100\,000 \times 100\,000$ matrix $\boldsymbol{C}_{\text{global}}$ for the global parameters.

Millepede simultaneous fit

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

The Millepede principle: transfer of the local information to the global Hessian $\boldsymbol{C}$

$$\boldsymbol{C}_{\text{global}} = \sum_k \boldsymbol{C}_k - \sum_k \boldsymbol{H}_k \boldsymbol{C}_k^{-1} \boldsymbol{H}_k^T \qquad \text{(“Schur complement”)}$$

(transfer of the local information to the global Hessian $\boldsymbol{C}$.

$$\left(\quad \boldsymbol{C}_{\text{global}} \quad\right) \times \left(\Delta\boldsymbol{p}^{\text{global}}\right) = \left(\sum_k \boldsymbol{b}_k^{\text{global}}\right)$$

[back]

Cholesky decomposition $\quad \boldsymbol{C} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}} \quad$ cpu-time $\propto n \times m^2$

$\boldsymbol{L} =$ left unit triangular (band) matrix $\quad \boldsymbol{D} =$ diagonal matrix

$$
\boldsymbol{C} = \left(\begin{array}{cccccccccccc|ccc}
\color{red}{x} & x & x & & & & & & & & & & & x & \\
x & \color{red}{x} & x & x & & & & & & & & & x & x & \\
x & x & \color{red}{x} & x & x & & & & & & & & x & x & x \\
 & x & x & \color{red}{x} & x & x & & & & & & & & x & x \\
 & & x & x & \color{red}{x} & x & x & & & & & & x & x & x \\
 & & & x & x & \color{red}{x} & x & x & & & & & x & & x \\
 & & & & x & x & \color{red}{x} & x & x & & & & x & & \\
 & & & & & x & x & \color{red}{x} & x & x & & & x & x & x \\
 & & & & & & x & x & \color{red}{x} & x & x & & x & x & \\
 & & & & & & & x & x & \color{red}{x} & x & x & x & & x \\
 & & & & & & & & x & x & \color{red}{x} & x & x & x & x \\
 & & & & & & & & & x & x & \color{red}{x} & x & x & x \\
\hline
 & x & x & & x & x & x & x & x & x & x & x & o & o & o \\
x & x & x & x & x & & & x & x & & x & x & o & o & o \\
 & & x & x & x & x & & x & & x & x & x & o & o & o
\end{array} \right)
\quad \Longrightarrow \quad
\boldsymbol{L} = \left(\begin{array}{cccccccccccc|ccc}
\color{red}{x} & & & & & & & & & & & & & & \\
x & \color{red}{x} & & & & & & & & & & & & & \\
x & x & \color{red}{x} & & & & & & & & & & & & \\
 & x & x & \color{red}{x} & & & & & & & & & & & \\
 & & x & x & \color{red}{x} & & & & & & & & & & \\
 & & & x & x & \color{red}{x} & & & & & & & & & \\
 & & & & x & x & \color{red}{x} & & & & & & & & \\
 & & & & & x & x & \color{red}{x} & & & & & & & \\
 & & & & & & x & x & \color{red}{x} & & & & & & \\
 & & & & & & & x & x & \color{red}{x} & & & & & \\
 & & & & & & & & x & x & \color{red}{x} & & & & \\
 & & & & & & & & & x & x & \color{red}{x} & & & \\
\hline
 & x & x & x & x & x & x & x & x & x & x & x & \color{red}{x} & & \\
x & x & x & x & x & x & x & x & x & x & x & x & x & \color{red}{x} & \\
 & & x & x & x & x & x & x & x & x & x & x & x & x & \color{red}{x}
\end{array} \right)
$$

The matrix equation $\boldsymbol{C}\boldsymbol{x} = \boldsymbol{L}\left(\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}\right) = \boldsymbol{y}$ can be solved in two steps:

solve $\quad \boldsymbol{L}\boldsymbol{z} = \boldsymbol{y} \quad$ for $\boldsymbol{z}$ by forward substitution, and

solve $\quad \boldsymbol{L}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{D}^{-1}\boldsymbol{z} \quad$ for $\boldsymbol{x}$ by backward substitution.

[back]

Solution by diagonalization

The diagonalization of the symmetric matrix $\boldsymbol{C}$ allows to recognize singularity or near singularity by the determination of eigenvalues, and to suppress corresponding linear combinations of parameters.

Computing time and space requirement larger compared to inversion, and solution less precise (especially for small eigenvalues; mixing of eigenvectors).

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

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: $\boldsymbol{C}^{-1} = \boldsymbol{U}\,\boldsymbol{D}^{-1}\,\boldsymbol{U}^{\mathrm{T}}$

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

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

with replacement $1/\lambda_i = 0$ for $\lambda_i = 0$ or small q_i with $|q_i| \lesssim 1$; keep significant modes with small λ_i.

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

[back]

Elements of the 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 $\quad \boldsymbol{C}\,\boldsymbol{V} = \mathbf{1} \quad$ (right hand-side $\mathbf{1}$ is unit matrix)

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

…solution of matrix equation $\quad \boldsymbol{C}\boldsymbol{v}_j = \boldsymbol{e}_j \quad$ (right hand-side $\boldsymbol{e}_j$ is j-th column of unit matrix)

for $\boldsymbol{v}_j$ will give on j-th column of the covariance matrix $\boldsymbol{V}$.

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

[back]

Feasible parameters

A vector $\boldsymbol{x}$ compatible with constraint equations $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{c} = 0$ is called a *feasible* vector.

Round-off errors can introduce small deviations: $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{c} = \boldsymbol{\varepsilon}$.

In order to **force feasibility** a minimum-norm correction $\boldsymbol{\Delta x}$ with min $\|\boldsymbol{\Delta x}\|_2$ is calculated by

$$\boldsymbol{\Delta x} = -\boldsymbol{A}^{\mathrm{T}} \left(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}\right)^{-1} \boldsymbol{\varepsilon}$$

in each iteration.

The product $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ is a square m-by-m non-singular matrix for sufficient constraints.

[back]

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

Parameter                 ! set status for selected parameters
201 0.0      0.0          ! variable parameter (default), initial value = 0
202 1.732   -1.0          ! fixed parameter, initial value = 1.732
204 1.23     0.020        ! variable parameter with presigma
constraint 0.14           ! numerical value of constraint equation
713 1.0 720 0.5           ! pairs of parameter label and numerical factor
Measurement  10.3 0.1     ! survey distance [713]-[714] = 10.3 +- 0.1
713  1.0 714 -1.0

method sparseGMRES 5 0.1 ! Generalized residual minimization, sparse matrix
bandwidth 6               ! with variable-band matrix preconditioning
chisqcut  15  6           ! chisquare cut for first and second loop
outlierdownweighting 5    ! down-weighting in 5 local iterations
dwfractioncut  0.2        ! reject bad records
printrecord 13 -1         ! debug printout for record 13 and worst record
histprint                 ! print histograms
subito                    ! exit after first step
end
```

[back]

Limited memory BFGS (L-BFGS)

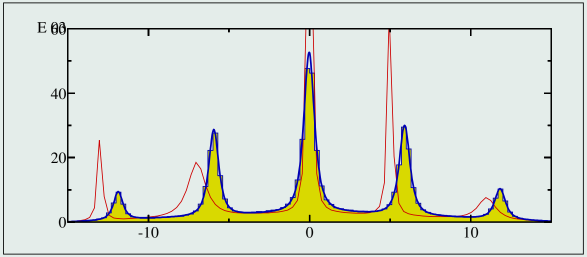
What to do, if the number of parameters is 200 000 or 500 000?

Large-scale limited memory BFGS algorithm has space requirement *proportional* to number of parameters, with e.g. only 60 Mbyte for 100 000 parameters. Information about the matrix $\boldsymbol{C}$ is stored in a limited number of vector pairs!

Minimization package `lvmini`, using L-BFGS, developed for $n = 2$ up to several 100 000 parameters, needs gradient $\boldsymbol{\nabla} F$. So far no constraints possible.

280-parameter Neural Net training and $> 100\,000$ parameter minimization under study.

Use of `lvmini` in Millepede II would require different method for constraints: elimination method under study.



`lvmini`-example of fit with 20 parameters

Initial parameter values correspond to red line.

Minimization requires ≈ 100 function evaluations.

Could also be used for large-scale optimization e.g in calorimeter calibration.

[back]

Contents