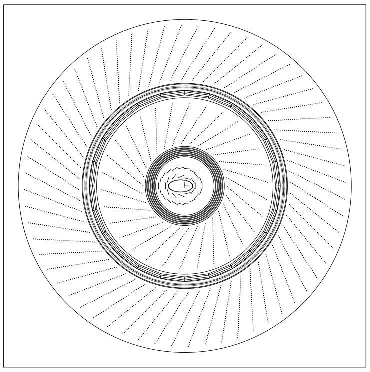
Tracking methods overview

V. Blobel, University Hamburg

HERMES meeting in Zeuthen 27. + 28. March 2003

My personal experience in track finding/fitting is from
- Hydrogen bubble chamber physics
- PLUTO detector at DORIS and PETRA
- Muon tracks in CHARM II
- Central track detector (central jet chamber CJC) of H1

My track reconstruction code development for H1
- started for the CJC at the end of the 80's (MC)
- first data taken in 1992
- reorganization of code 1992 - 1994
- new version for CJC + CIP + COP + COZ + CST started last year.



Content of the talk

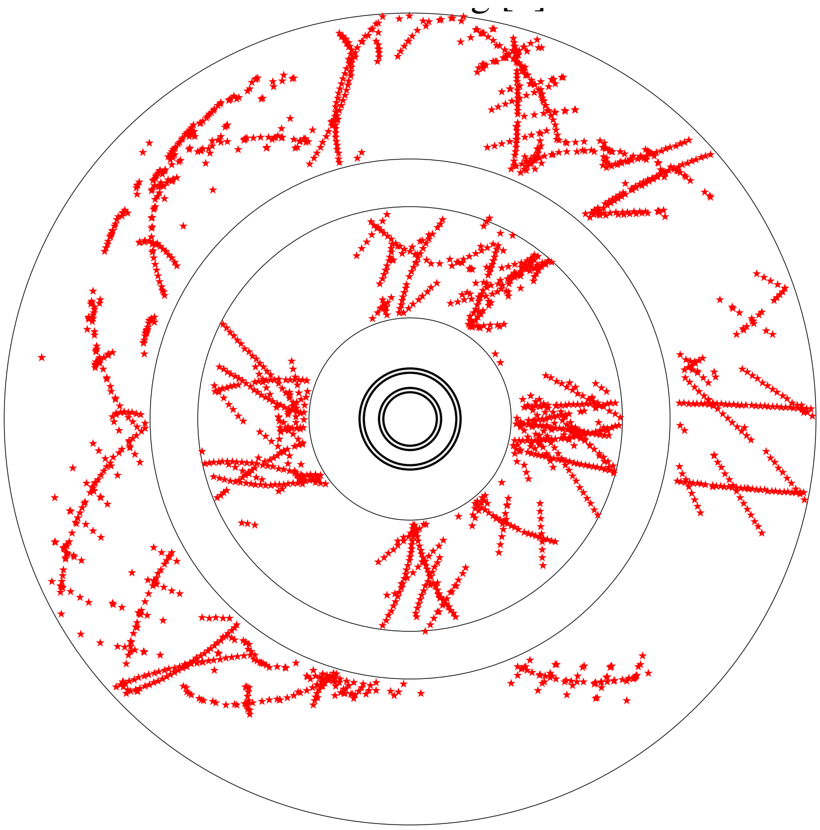
Track reconstruction is more than track fitting:
- Track finding and fitting
- Primary vertex finding and vertex/track fitting
- Secondary vertex finding and vertex/track fitting (V^0's)
- Alignment and calibration
- Monitoring

1. A few principles
2. Track finding methods
3. Mathematical methods and algorithms
4. Alignment and calibration

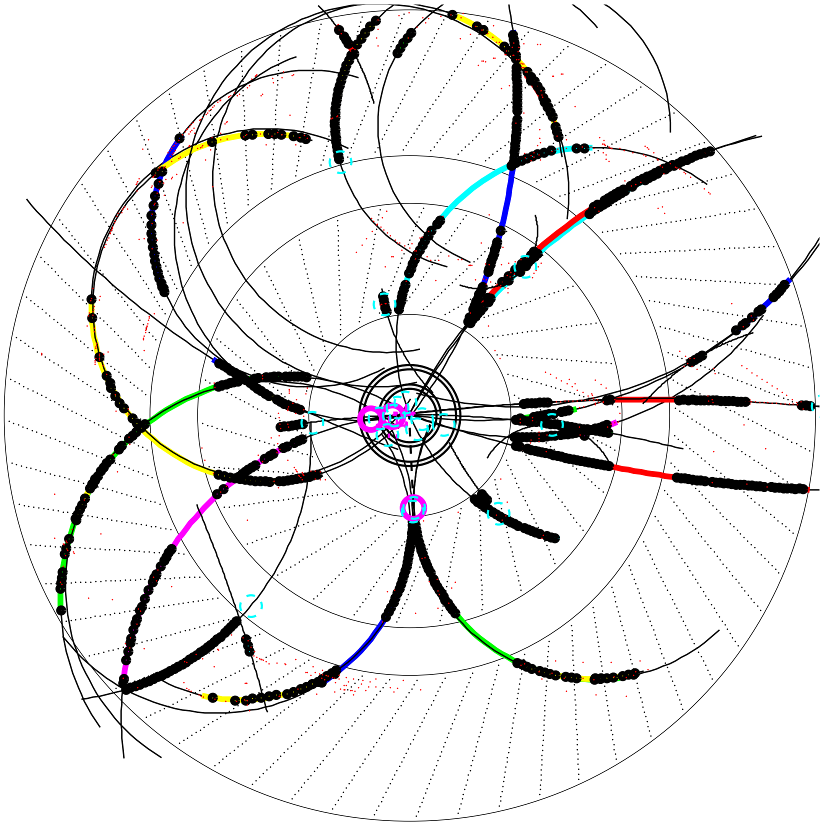
Principles of track reconstruction

- Long tracks with many hits are much more reliable and accurate ($\sigma \propto 1/(\text{length})^{5/2}$) than short tracks → start track finding with long tracks.
- Democratic approach: use **all** hits during initial track-seed finding (no *selected* planes), whenever possible, otherwise track finding effiency can become worse for detector inefficiencies. Do not rely on a high effiency of certain detector components.
- Avoid (outside) steering – the reconstruction should find its condition automatically.
- Avoid internal parameters as far as possible. Example: find cluster in 1D by sorting etc. instead of a histogram method, which requires limits, number of bins and bin size definition (and is often slower).
- Allow repeated call of track finding module – flag hits used already for a reliable track and remove those hits before (new) track finding.
- Track finding and fitting can not be separated – most sensitive test of the goodness of a hit is the “final” track fit
- Use physical constraint equations during fits – accuracy and reliablility is increased.
- Use hits from all track detectors simultaneously (synergy effects) – finally hits from all detectors ara to be used.
- Align and calibrate all detectors simultaneously.

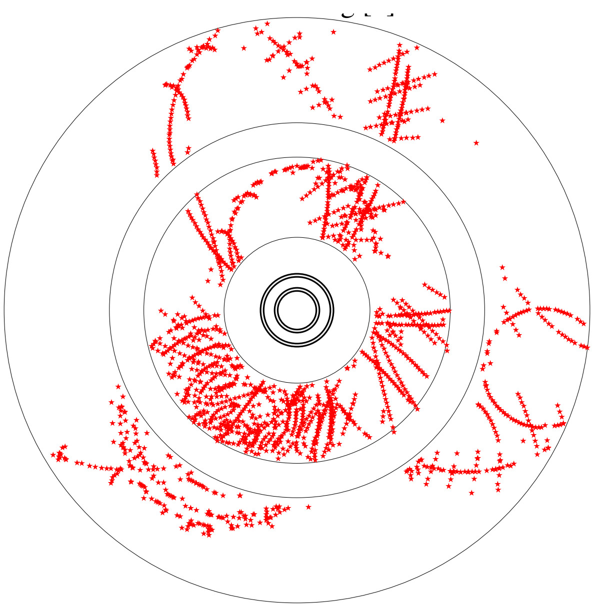
Initially no T_0, no vertex, no tracks – just hits and mirror hits

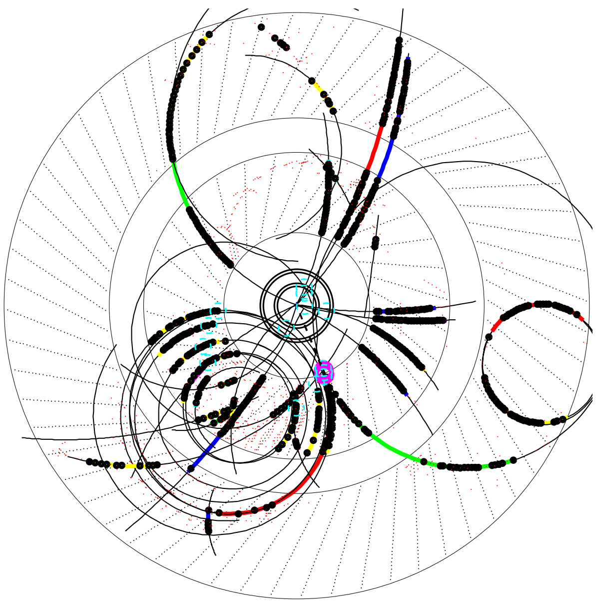


... after CJC reconstruction: tracks and vertices

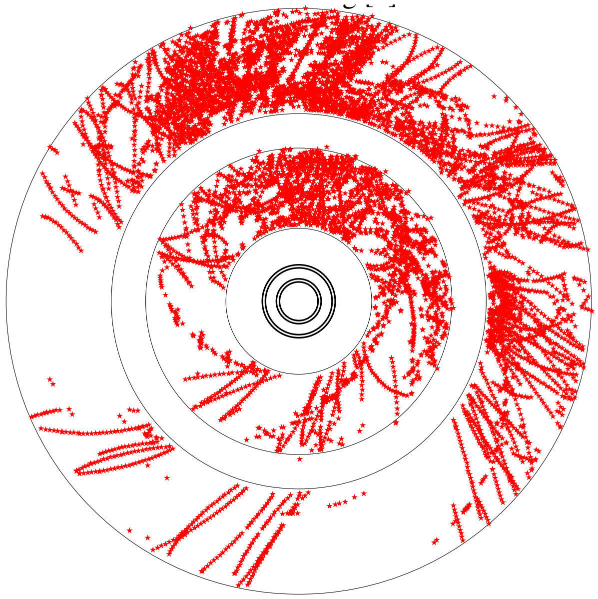


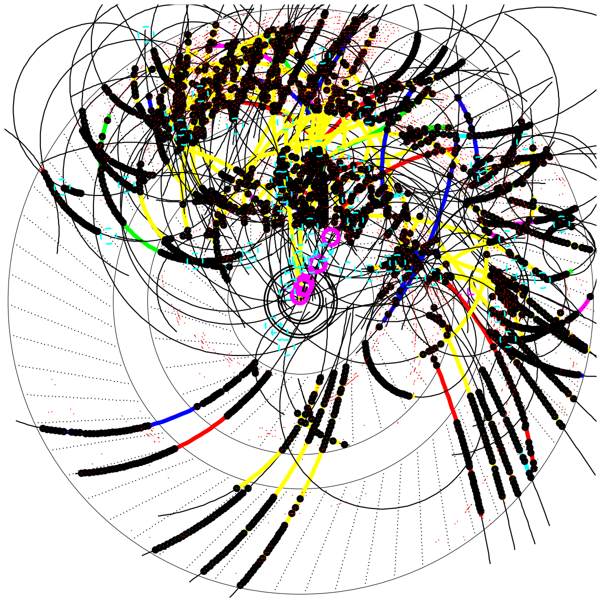
Another example ...





...and another more heavy one.





Principles of track finding

My basic strategy for track finding has two steps:

Initial track finding: Using different methods the parameters of potential tracks are *estimated* → track seed based on simple geometrical relations (the method has to be robust with respect to the basic parameters T_0, v_{drift} and α_L). Track seeds are expressed by the triple $(\kappa, d_{ca}, \varphi)$ (or $(\kappa, 0, \varphi)$) and the number of hits (track length).

Book keeeping only includes $(\kappa, d_{ca}, \varphi, n_{\text{hits}})$, no book keeping on hits. It allows sorting according to the number of hits.

Track improvement: Tracks are always treated as a whole considering **all** potential hits. Starting from the knowledge of the triple $(\kappa, d_{ca}, \varphi)$ from a track seed it is possible to calculate for all detector planes the expected values of the detector signals (like MC track following). Then **all** data hits close to the expected signal can be extracted, and this set of data (= coordinate values) has to be examined. The result is again a triple $(\kappa, d_{ca}, \varphi)$, and the track length.

Assumption: Hits of a track follow a smooth curve.

Principles of track finding, continued

The two steps of the basic strategy for track finding repeated:

Initial track finding: There is a dimensional gap between data and tracks:

- Data are *points* – with one coordinate value in a given plane.
- Tracks (in projection) are objects with 2 + 1 parameters: coordinate and slope, and curvature.

This gap has to be closed in initial track finding.

Track improvement: The selected hits are examined.

- Often hits from a single track are extracted – no problem.
- There may be outliers – removed by smoothing and fitting.
- There may be two crossing tracks, or a track with a kink – smooth, fit with few parameters, . . . fit by broken line, analyse and split eventually. The longest track candidate (if any) should remain.

In the second step the precision of the parameter values is improved (or the track is discarded). The track improvement is also used to improve the precision of basic parameters like T_0.

Principles of track finding, continued (2)

The two CJC's are separated by a gap. Initial track-seed finding and track improvement is separately done for the CJC 1 and 2.

Track-seeds are always sorted (decreasing hit number).

Improved tracks for separated CJC's are also sorted (decreasing hit number).

Then both CJC's are considered for all tracks (order of decreasing hit number).

There are two hits flags:

- A final flag marks a hit as used after examination of both CJC's. Those hits are never used again or even tried later for additional tracks.
- A preliminary flag marks hits used as a track candidate – these flags are cleared at the end of a loop.

Principles of track finding, continued (3)

Several completely different method or algorithms are used for the initial track-seed finding, which uses only hits not used by tracks found earlier:

- First a simple method for tracks with isolated track parts;
- Second (repeated) standard efficient method (avoiding dense regions). This is repeated until no further track is found
- Finally special short tracks are searched for with special method.

In the second step new tracks are finally found and their hits get the final “used” flag.

Track finding methods

- Initial track finding – closing the dimensional gap
  Example for hit triples from a single cell
  Example for seed finding in one cell
- Track improvement separately in CJC 1 and 2
- Track improvement in whole CJC
- Repeated seed finding and track reconstruction

- The track fit in detail . . .
- Track fit in $r\varphi$ in four steps
  Example for fits . . .

Initial track finding – closing the dimensional gap

Seed finding is done in a loop over all cells, both CJC's. For each cell the hits unused so far are copied, and seed finding is done in the copy.

Standard method: Hit triples (i, j, k) with wire distances of 1 and/or 2 layers are checked* ($d_i = v_{\text{drift}}(t_i - t_{i,0} - T_0)$ = drift distance):

$$d_j \approx (d_i + d_k)/2 \text{ ?} \qquad \text{all hits on the same side of the wire plane}$$
$$d_j \approx (|d_i - d_k|)/2 \text{ ?} \qquad \text{track crosses wire plane, requires good } T_0$$

(relations modified by staggering and unequal layer distances).

Eventually several triples are combined. A circle fit is done constrained to $(x, y) = (0, 0)$ for 4 or less hits, or unconstrained for more than 4 hits.

Eventually track seeds from different cells are combined.

---

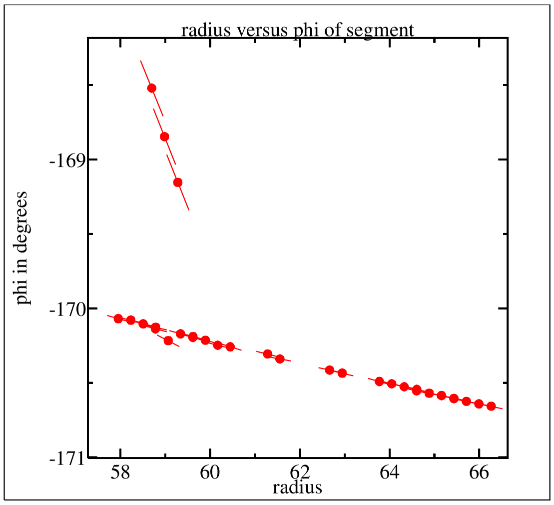
* Note: dense regions are excluded to avoid heavy combinatorical search.

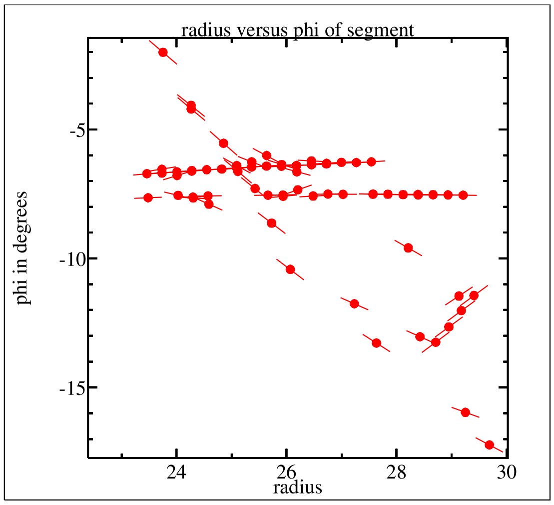
Example for hit triples from a single cell

A circle with through the origin $(\kappa, 0, \varphi)$ is fitted to hits triples.

The angle φ is calculated at the center of the triples, and plotted versus the radius (layer number); the fitted curvature κ is the slope of such a "point".

Low momentum tracks have a large slope.



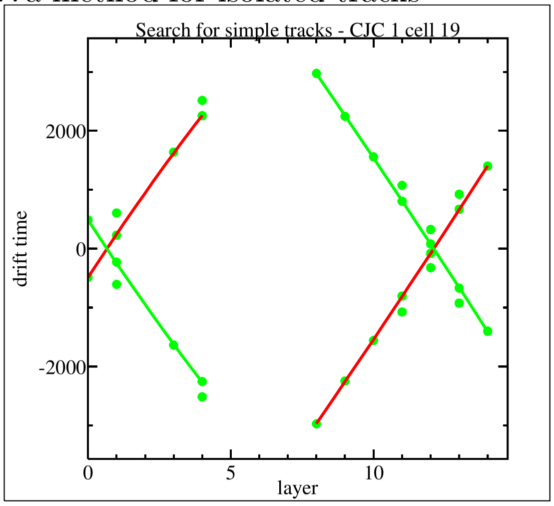


Hit triples may be combined (next slide), making use of the slope = curvature.

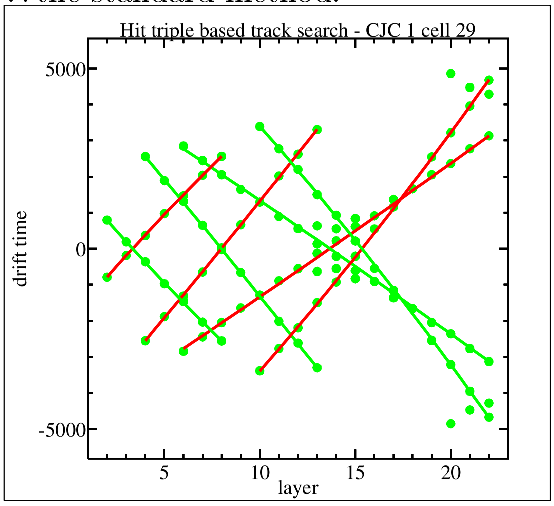
Example for seed finding in one cell

Shown are examples from two different methods for seed finding:

. . . a method for isolated tracks



. . . the standard method.



Hits and mirror hits are shown as green dots, together with the potential two solutions (green and red) for the seed.

Crossing the zero-line (wire plane) requires good knowledge of the event T_0.

Track improvement separately in CJC 1 and 2

Track seeds are sorted according to decreasing number of hits, i.e. track improvement will start from the seed with the largest number of hits, separately for CJC 1 and CJC 2.

- Using $(\kappa, d_{ca}, \varphi)$ the track path in the CJC 1 or 2 is calculated (magnetic field table), and the expected drift times (dt) for all possible layers are calculated (also relevant geometrical quantities are calculated for corrections);
- for each wire the hit closest to the expected dt is selected, and the dt difference is calculated;
- hits far from other hits are excluded, and a parabola with T_0 optimization is fitted (done twice);
- a 6 σ cut w.r.t to the parabola is done for all originally selected hits, followed by a *broken line* fit (see next slide) and a 3 σ cut w.r.t to the broken line;
- a circle is fitted to the remaining hits.

This procedure is repeated up to three times. Finally a seed with ≥ 3 remaining hits is accepted as a candidate, the hits get a preliminary "used" flag and will not be selected with other seeds.

Track improvement in whole CJC

- Track candidates from CJC 1 and 2 are sorted according to decreasing number of hits. The track improvement procedure is repeated, this time for the complete CJC.
- Tracks are treated during improvement as a whole (i.e. as many hits as possible). Long tracks with a large number of hits allow safely to recognize outliers or hits from other tracks.
- Hits from accepted tracks are recorded, the hits are flagged as "used" (and will not be tried for seed finding later).
- The event T_0 is continuously updated during track improvement.

Repeated seed finding and track reconstruction

- Seed search and track improvement always work on hits *unused* so far, and can be repeated (with reduced combinatorics) again and again.
- Usually some tracks (or parts) are isolated and are easily found as seeds with a special algorithm [1], executed first. Those *early* tracks already improve the event T_0.
- The standard seed algorithm [2] is repeated until no further track is found.
- Finally a special seed algorithm [3] is used for very short (steep) tracks.

Examples:

```
JCHECK error code= 0, 295 hits, T0= 503
Track finding [1], + 7 tracks, T0= 465
Track finding [2], + 2 tracks, T0= 466
Track finding [2], + 0 tracks, T0= 466
Track finding [3], + 0 tracks, T0= 466
Total 9 tracks found
```

```
JCHECK error code= 0, 4429 hits, T0= 366
Track finding [1], + 42 tracks, T0= 435
Track finding [2], + 38 tracks, T0= 426
Track finding [2], + 37 tracks, T0= 422
...
Track finding [2], + 7 tracks, T0= 422
Track finding [2], + 9 tracks, T0= 423
Track finding [2], + 5 tracks, T0= 423
Track finding [2], + 0 tracks, T0= 423
Track finding [3], + 0 tracks, T0= 423
Total 185 tracks found
```

100 % of the tracks are found in the seed algorithms, but some tracks may be lost during the track improvement.

The track fit in detail

A particle track is a circle in the $r\varphi$-plane and a straight line in the zS-plane, modified

- by non-constant magnetic field $\boldsymbol{B}$,
- by multiple scattering in material (mainly change of direction), especially for low momentum tracks.
- by photon radiation (electrons),
- eventually by decays charged $\rightarrow$ charged + neutrals,
- by non-perfect calibration/alignment.

A track fit is required already during track finding and for final track determination. Required are

- optimal track parameters at the track-start,
- optionally with vertex constraint, and
- track parameters at the track-end, for extrapolation to other detectors.

Track fit in $r\varphi$ in four steps

Steps in the track fit:

- Circle fit, unweighted, using the method of KARIMÄKI, either with three parameters (κ, d_{ca} and φ), or in vertex fit with two parameters (κ and φ),
- track following using magnetic field table, using above parameters, in steps through the central detector, creating a list of xyz-values,
- calculation of track residuals from measured hits and list of xyz-values,
- weighted least squares fit of multiple parabolas to track residuals η.

Parametrization of multiple parabola function:

$$\begin{aligned} \eta(S) = \Delta d_{ca} \quad & + \quad \Delta\varphi \cdot S + \frac{1}{2}\Delta\kappa \cdot S^2 \\ (+\text{ after each scattering plane}) \quad & + \quad \Delta\varphi_{1,pl} \cdot (S - S_{pl}) + \Delta\varphi_{2,pl} \cdot \frac{D_{pl}}{\sqrt{12}} \end{aligned}$$

and additional terms in the least squares expression:

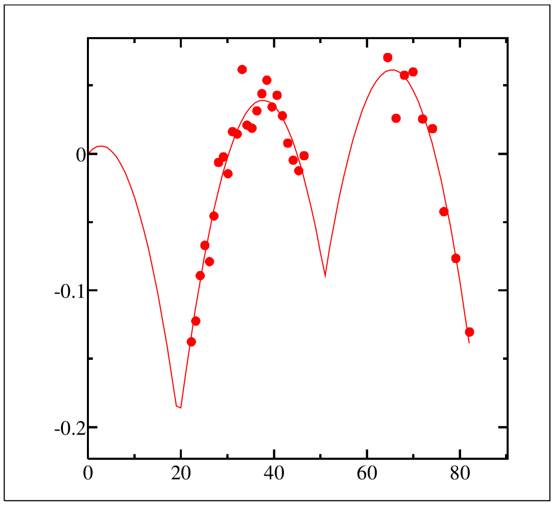
$$+\left(\frac{\Delta\varphi_{1,pl}}{\vartheta_0}\right)^2 + \left(\frac{\Delta\varphi_{2,pl}}{\vartheta_0}\right)^2 \qquad \text{with} \quad \vartheta_0 \approx \frac{0.0136 \text{ GeV/c}}{\beta\, p \text{ in GeV/c}} \sqrt{\frac{x}{X_0}} \times \ln \ldots$$

$\rightarrow 3 + 2 \cdot n_{pl}$ parameters + optionally ΔT_0.

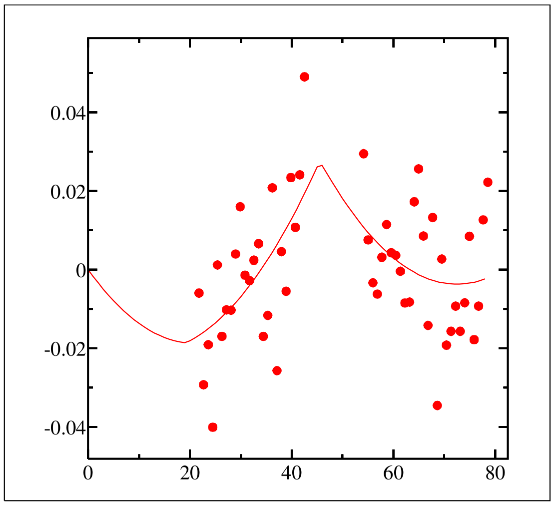
No parameter Δd_{ca} in vertex constrained fit. The primary vertex fit is different from the fit in the present CJCREC.

Examples for fits of multiple parabolas

Low momentum $p = 116$ MeV/c:



Momentum $p = 1.16$ GeV/c:



For high momentum (e.g. > 10 GeV/c) multiple scattering has no effect, but electrons may radiate photons.

Mathematical methods and algorithms

- Selecting the m-th largest element
- Cluster search in 1-D
- Cluster search in $>$ 1-D
- Smoothing of data $y = y(x)$ (SM353Htwice)
- Fitting a straight line by minimizing absolute deviations
- Robust fitting
- Simultaneous straight line fits with common intercept
- Fast circle fit
- Broken line fit
- Range search in 1-D and in many-D

Selecting the *m-th largest element*

Fast general method for selection (allowing rearrangement) is partitioning, exactly as done in Quicksort (see Press et al.):

- select a "random" partition element;
- go through array, forcing smaller elements to the left, larger elements to the right;
- ignore one subset and repeat with the subset which contains the desired m-th largest element, until element found.

No stack or extra space required (only one subset is used), but rearrangement is done.

Median determination is a special case: **the median is a robust estimate for the location of a distribution.**

Number of operations is $O(\log n)$.

Cluster search in 1-*D*

Often a cluster (signal) has to be searched for in a set $\{x\}$ of one-dimensional values x_i. The cluster is equivalent to a peak of a one-dimensional probability density.

One possible method: put values x_i into a histogram and search for maximum bin(s) content, which requires limits and bin size $\rightarrow$ loss of resolution.

Method in subroutine `CLUST1`, avoiding parameters like limits and bin size:

- Sort the values x_i into ascending order;
- find for each index i the *largest* index j with $x_j - x_i < \texttt{WID}$;
- calculate for each pair (i, j) with $n_{ij} = j - i + 1$ inner values

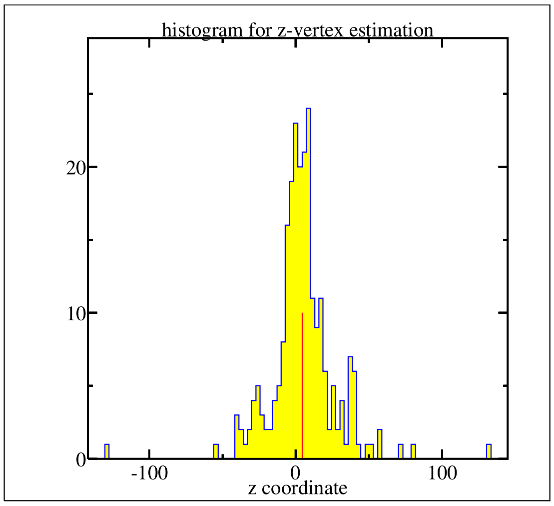
$$\text{density} = \frac{n_{ij}}{\max(x_j - x_i, \texttt{WID/2})}$$

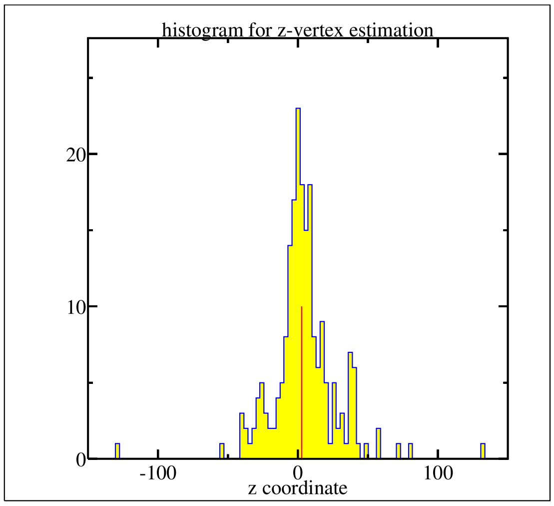
  and select pair (i, j) with maximum density;
- estimate standard deviation from $\{x_i, \ldots x_j\}$ and determine average x-value within two standard deviations around the peak.

Examples for cluster search

Cluster search used for κ- and φ-cluster in track seed search, and for initial primary z-vertex estimate:

z vertex distribution





…from vertex-track candidates

…and after removal of non-vertex track:

Cluster search in > 1-D

Sometimes a simple generalization of the 1-D cluster search method, based on sorting, may be useful.

Sorting is done without reordering data but using an index array (pointer).

Algorithm for 2-D, 3-D cluster search:

- Sort (by rearrangement of index array) the data in one dimension (x)
- Find largest peak in x by search for maximum density (defines a subset of data with a certain range in index array)
- Sort (by rearrangement of index array) the subset data in next dimension (y)
- Find largest peak in y of subset by search for maximum density
- …(continue)

❖ *Smoothing of data $y = y(x)$ (SM353Htwice)*

Algorithm by TUKEY for data smoothing and outlier rejection:

(3 - G - 5 - 3C - Q - H)twice

with several components:

3	replace each value by median of 3 values, with extrapolation at end
G	conditional hanning
5	replace each value by median of 5 values
3C	replace each value by median of 3 values, copy end points
Q	correct flat areas of three identical values by quadratic interpolation
H	replace each value by the average over 3 values
twice	perform all operations twice

Replacement of value by the median of 3 or 5 values removes efficiently and traceless isolated (far) outliers.

Also some components of the algorithm may be useful.

❖ *Fitting a line by minimizing absolute deviation*

Fit of the line $y(x;\, a,\, b) = a + b\, x$ by minimizing

$$F(a,b) = \sum_{i=1}^{n} |y_i - a - bx_i| \ . \qquad (*)$$

For fixed slope b the sum $(*)$ is minimized by the intercept a, given as a function of the fixed b

$$a(b) = \text{median}\,\{y_i - bx_i\} \ .$$

Fitted value of b from derivative of the sum $(*)$ w.r.t. slope b

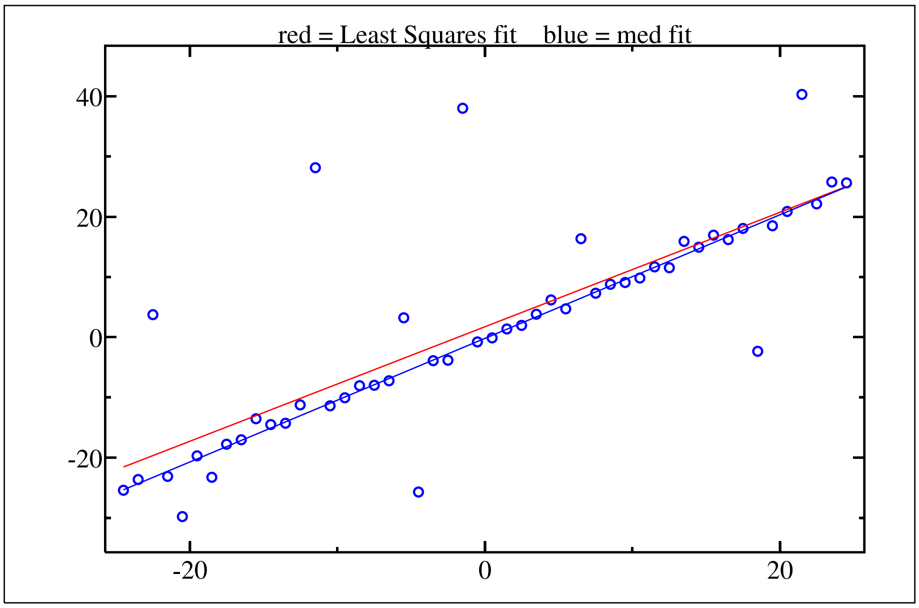
$$\frac{\partial F}{\partial b} = f(b) = \sum_{i=1}^{n} \text{sign}\,(y_i - a - bx_i) \cdot x_i \qquad \text{and} \quad f(\hat{b}) = 0$$

(with sign$\,(0) = 0$) with a replaced by $a(b)$. Function $f(b)$ has discontinuities and has to be solved for $\hat{b}$ by e.g. bracketing and bisection.

The median c_{median} minimizes the sum $\sum |c_i - c_{\text{median}}|$ of absolute deviations (median based on sorting the c_i).

Example for straight line fit to noisy data

Blue line is fitted by minimizing the sum absolute deviation - not much disturbed by outliers.



Red line is result of least squares fit - because of square of deviations strongly influenced by outliers.

Method for robust least squares fit

Assume estimate for the standard error of y_i (or of residual r_i) to be s_i.

- Obtain e.g. by previous method an approximation of a fit to the observations y_i, yielding fitted values $\hat{y}_i$, and residuals $r_i = y_i - \hat{y}_i$.
- "Clean" the data by pulling outliers towards their fitted values: winsorize the observations y_i and replace them by pseudo-observations y_i^*:

$$\begin{aligned} y_i^* &= y_i\,, & \text{if} \quad & |r_i| \le c\,s_i\,, \\ &= \hat{y}_i - c\,s_i\,, & \text{if} \quad & r_i < -c\,s_i\,, \\ &= \hat{y}_i + c\,s_i\,, & \text{if} \quad & r_i > +c\,s_i\,. \end{aligned}$$

  The factor c regulates the amount of robustness, a goid choice is $c = 1.5$.
- Refit iteratively: the pseudo-observations y_i^* are used to calculate new parameters and new fitted values $\hat{y}_i$.

The method *works* in practice, but has no well-defined target (it has an *open end*).

❖ *Simultaneous straight line fits with common intercept*

With the line (track) index j the model is

$$y = a + b_j x \qquad j = 1, 2, \dots n$$

with $n+1$ parameters $\boldsymbol{p} = b_1, b_2 \dots b_n$ and a for n straight lines.

For the standard least squares normal equations $\boldsymbol{A}\boldsymbol{p} = \boldsymbol{r}$ with

$$\boldsymbol{A} = \left(\begin{array}{cccc|c} b_{11} & 0 & \dots & 0 & u_1 \\ 0 & 0 & \dots & 0 & u_2 \\ \dots & & & & \dots \\ 0 & 0 & \dots & b_{n,n} & u_n \\ \hline u_1 & u_2 & \dots & u_n & a_{nn} \end{array} \right) \qquad \boldsymbol{r} = \left(\begin{array}{c} r_1 \\ r_2 \\ \dots \\ r_n \\ \hline \rho \end{array} \right)$$

the computing time will be roughly $\propto (n+1)^3$.

A faster method based on the special structure of the normal equations has a computing time $\propto (n+1)$ ($\rightarrow$ next slide).

Special formulae for matrix inversion and solution

Elements of the inverse matrix with $\alpha = a_{nn} - \sum_{k=1}^{n} u_k^2 / b_{kk}$ are:

$$\left(\boldsymbol{A}^{-1}\right)_{ii} = \frac{1}{b_{ii}} + \frac{1}{\alpha}\left(\frac{u_i}{b_{ii}}\right)^2 \qquad \left(\boldsymbol{A}^{-1}\right)_{ij} = \frac{1}{\alpha}\frac{u_i u_j}{b_{ii} b_{jj}}$$

$$\left(\boldsymbol{A}^{-1}\right)_{i,n+1} = -\frac{1}{\alpha}\frac{u_i}{b_{ii}} \qquad \left(\boldsymbol{A}^{-1}\right)_{n+1,n+1} = \frac{1}{\alpha}$$

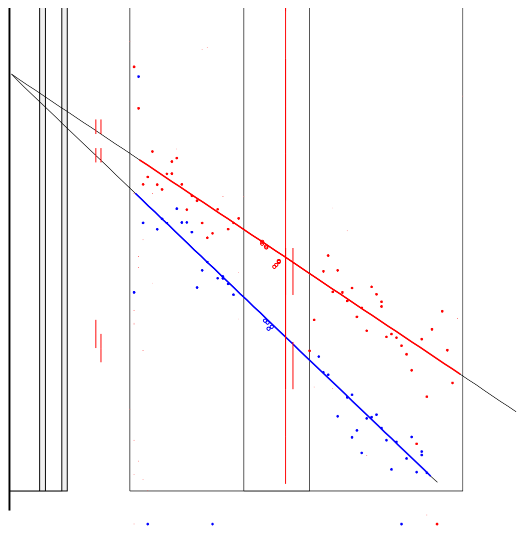
Results for common intercept a and slopes b_j and their variances are:

$$a = \frac{1}{\alpha}\left(\rho - \sum_{k=1}^{n} \frac{r_k\, u_k}{b_{kk}}\right) \qquad V(a) = \left(\boldsymbol{A}^{-1}\right)_{n+1,n+1} = \frac{1}{\alpha}$$

$$b_i = \frac{r_i - a\, u_i}{b_{ii}} \qquad V(b_i) = \left(\boldsymbol{A}^{-1}\right)_{ii}$$

The **addition of data** to a single track j modifies only the elements b_{jj}, u_j, r_j, ρ and a_{nn}; only few operations are required for a repeated fit, if relevant sums were saved before. Especially the modified intercept a and $\Delta\chi^2$ is easily calculated.

Example: an elastic J/Ψ event



$r\,z$-view

Shown are

- CIP and COP hits (red vertical lines), and
- COZ hits, if $\Delta z < 10$ cm, as colored circles.

Fast circle fit

A fast least squares *circle* fit method has been developed by Veikko Karimaeki:

- Parameters are $\kappa = 1/R$, d_{ca} and φ at origin,
- no starting values of parameters required,
- no iteration necessary,
- rather accurate result with full covariance matrix;
- special version allows to fix the origin (only 2 parameters).

Used in H1 central track reconstruction for initial track-seed determination (2 parameters) and approximation of final fit.

Veikko Karimaeki, Fast Code to fit circular Arcs, University of Helsinki Report HU-SEFT-1991-10

Broken line fit

The problem:

- Ideally only 0.27 % of the hits have a deviation from a parametrized track curve larger than 3 σ.
- In practice the hits do not follow the parametrized track curve so well, a larger fraction of hits is outside 3 σ and could be removed as outliers.
- The removed hits could form another (short) track (ghost tracks).

What is a *Broken line fit*?
It is a least squares fit of a set (x_i, y_i) *without* a fixed parametrization, but it *follows* the trend of the data, and allows to remove true outliers.

Fitted parameters are $\hat{y}_i$ with minimization* of

$$\sum \left(\frac{\hat{y}_i - y_i}{\sigma_i} \right)^2 + \tau \cdot \sum \left(\hat{y}_{i+1} - 2\hat{y}_i + \hat{y}_{i-1} \right)^2$$

with a certain parameter τ, which determines the degree of smoothness.

---

* The expression shown is somewhat simplified; in reality it takes into account the x-differences.

Range search in 1-D and in many-D

Unfortunately binary search can only be used in 1-D. But there is another more general algorithm:

Range searching =
finding all records in a data base of total n records that satisfy certain range restrictions

$$x_l \leq x < x_r$$

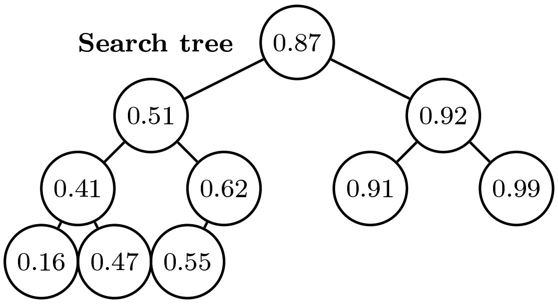
on a specified set of attributes

The algorithm:

- Preprocessing ('learning" phase): build the given points into a structure, based on links;
- range-searching: use the structure to return points falling into a given range.

Binary search trees

- all records with smaller keys are in the left subtree, and all records in the right subtree have larger (or equal) key values.



Properties:

- Build left/right link structure by simple operations, leaving keys untouched, in $O(n \log n)$ steps
- and $O(R + \log n)$ for one range search, where R is the number of points actually falling in the range.

Recursive code for search in 1-D tree

```
type interval = record xl,xr: integer end
procedure treerange (t : link; int: interval);
  var bxl, bxr: boolean;
  begin
  if t ≠ z then
    begin
    bxl := t↑.key ≥ int.xl;
    bxr := t↑.key ≤ int.xr;
    if bxl then treerange(t↑.l,int);
    if bxl and bxr
      then t↑.key is within the range
    if bxr then treerange(t↑.r,int);
    end
  end;
```

Implementation of code can be recursive or non-recursive, stack-based. In Fortran a completely general stack-based implementation for many-D requires 30 - 40 statements for preprocessing and for search, using an index array `INDLR(2,.)`.

Range search in many-D

Generalization from 1-D to 2-D to multi-D trees is straightforward: cycle through the dimensions while going down the tree. Branching at each node corresponds to cutting the k-dimensional region of interest with a $(k-1)$-dimensional hyperplane.

Property of 2-D tree:

- Construction of a 2-D tree from N random points requires $2n \ln n$ comparisons, on average;
- Range searching with a 2-D tree seems to use about $R + \log n$ steps to find R points in ranges in a region containing n points.

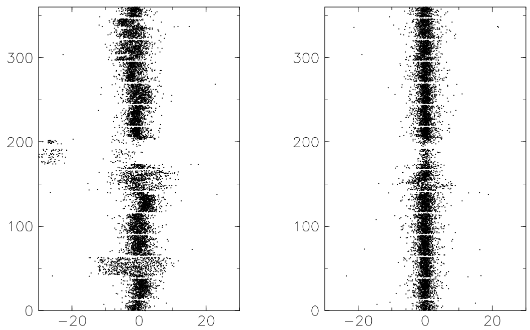
Alignment and calibration

Alignment and calibration can not be separated:

- Alignment: geometrical positions of detector components, valid for a run range.
- Calibration: Additional parameters needed for track reconstruction, which are valid for a run (range) or for single events.
  Examples: Beam parameters, detector parameters, Time (zero) of a drift chambers for single wires (run range) or for a single event; global and local drift velocities.
- Determination of alignment and calibration parameters should be integral part of the standard event reconstruction.

Motivation for alignment

Example: Residuals of track fit versus φ for a silicon tracker before . . . and after alignment:



Aim:

- smaller residuals!
- more accurate track parameters!!!

A simple method

An approximate alignment method:

- perform least squares fits on the e.g.track data *ignoring* initially the alignment parameters
- the *deviations* between the fitted and measured data, the *residuals*, are used to determine the alignment parameters afterwards.

The result of applying the alignment parameters on e.g. fits on the track data

- improved track residuals
- . . . but same track parameter values as before.

. . . because least squares alignment fits were based on a wrong track model.
The alignment problem is more general: in addition to pure alignment parameters there are drift velocities, variations of drift velocities, Lorentz angle . . . , which can not determined with residuals.

A better method?

The calculation of space coordinates from electronic signals requires alignment and many other parameters.

A better idea: perform a simultaneous least squares fit to the parameters of e.g. 20 000 tracks (each track has 50 hits and is described by three parameters) and all (e.g. 1000) alignment parameters i.e. solve normal equations $\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b}$
Add all necessary constraints e.g. zero average displacement and rotation of the detector.

Now the track model is correct, but ...

- $3\times$ 20 000 + 1000 parameters = 61 000 parameters
- 20 000 $\times$50 hits = 1 Mio hits

Can such a simultanenous fit be performed on a standard PC?

Constraints

Explicit relations between the parameters have to be taken into account e.g. zero average displacement and zero rotation of the whole detector.

Case of a single linear constraint between parameters:

$$\boldsymbol{a}^T \cdot \boldsymbol{f} = f_0$$

Modification of system of equations by Lagrange multiplier (λ) method:

$$\left(\begin{array}{c|c} \boldsymbol{C} & \boldsymbol{f} \\ \hline \boldsymbol{f}^T & \boldsymbol{0} \end{array}\right) \left(\begin{array}{c} \boldsymbol{a} \\ \hline \lambda \end{array}\right) = \left(\begin{array}{c} \boldsymbol{b} \\ \hline f_0 \end{array}\right) ,$$

Each constraint adds another parameter and equation to the whole system.

Solution by partitioning

Structure of the matrix equation: $\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b}$

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

where the submatrix $\boldsymbol{C}_{11}$ is a p-by-p square matrix and the submatrix $\boldsymbol{C}_{22}$ is a q-by-q square matrix, with $p + q = n$.

If the sub-vector $\boldsymbol{a}_1$ would not exist:

$$\boldsymbol{C}_{22}\, \boldsymbol{a}_2^* = \boldsymbol{b}_2 \qquad\qquad \boldsymbol{a}_2^* = \boldsymbol{C}_{22}^{-1}\, \boldsymbol{b}_2$$

The solution $\boldsymbol{a}_2^*$ requires only inversion of $\boldsymbol{C}_{22}$ and is called the *local* solution.

Solution by partitioning contnd.

Submatrix $\boldsymbol{B}$ of the complete inverse matrix correponding to the upper left part $\boldsymbol{C}_{11}$ is easy to calculate afterwards:

$$\boldsymbol{B} = \left(\boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^{T}\right)^{-1}$$

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: $\boldsymbol{a}_1 = \boldsymbol{B}\boldsymbol{b}_1 - \boldsymbol{B}\boldsymbol{C}_{12}\underbrace{\boldsymbol{C}_{22}^{-1}\boldsymbol{b}_2}_{\boldsymbol{a}_2^*} = \boldsymbol{B}\left(\boldsymbol{b}_1 - \boldsymbol{C}_{12}\boldsymbol{a}_2^*\right)$

Simultaneous fit of global and local parameters

Local parameters α_j are e.g. direct track parameters (belonging to a single track). For measurement z_k with (known) constant factors δ_j:

$$z_k = \alpha_1 \cdot \delta_{1k} + \alpha_2 \cdot \delta_{2k} + \ldots \alpha_\nu \cdot \delta_{\nu k} = \sum_{j=1}^{\nu} \alpha_j \cdot \delta_{jk}$$

Normal equation of linear least squares:

$$\boldsymbol{\Gamma}\,\boldsymbol{\alpha} = \boldsymbol{\beta}\,, \qquad \text{solved by} \quad \boldsymbol{\alpha} = \boldsymbol{\Gamma}^{-1}\boldsymbol{\beta}$$

Global parameters are e.g. alignment parameters and contribute to all the measurements. They represent corrections to ideal (design) values.

$$z = \underbrace{a_1 \cdot d_1 + a_2 \cdot d_2 + \ldots a_n \cdot d_n}_{\text{global parameters}} + \underbrace{\alpha_1 \cdot \delta_1 + \alpha_2 \cdot \delta_2 + \ldots \alpha_\nu \cdot \delta_\nu}_{\text{local parameters}}$$

Complete matrix equation for global and local parameters ...

...has huge matrices: n-by-n matrices $\boldsymbol{C}$ for n global parameters + N of small ν-by-ν matrices $\boldsymbol{\Gamma}$.

$$\left(\begin{array}{c|c|c|c} \sum \boldsymbol{C}_i & \cdots & \boldsymbol{G}_i & \cdots \\ \hline \vdots & \ddots & 0 & 0 \\ \hline \boldsymbol{G}_i^T & 0 & \boldsymbol{\Gamma}_i & 0 \\ \hline \vdots & 0 & 0 & \ddots \end{array}\right) \cdot \left(\begin{array}{c} \boldsymbol{a} \\ \hline \vdots \\ \hline \boldsymbol{\alpha}_i \\ \hline \vdots \end{array}\right) = \cdot \left(\begin{array}{c} \sum \boldsymbol{b}_i \\ \hline \vdots \\ \hline \boldsymbol{\beta}_i \\ \hline \vdots \end{array}\right)$$

Example: 3.7×10^9 matrix elements for $n = 1000$, $\nu = 3$ and $N = 20\,000$.

How to solve this linear system of equations?

The matrix of normal equations has a special structure with many vanishing sub-matrices. Only the global parameters $\boldsymbol{a}$ have to be calculated.

Use N times the partitioning method for the N sets of local parameters:

$$\boldsymbol{\alpha}_i^* = \boldsymbol{\Gamma}_i^{-1}\boldsymbol{\beta}_i$$

Finally the n normal equations

$$\begin{pmatrix} \quad \boldsymbol{C}' \quad \end{pmatrix} \begin{pmatrix} \boldsymbol{a} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}' \end{pmatrix}$$

are obtained with modified matrix and vector:

$$\boldsymbol{C}' = \sum_i \boldsymbol{C}_i - \sum_i \boldsymbol{G}_i \boldsymbol{\Gamma}_i^{-1} \boldsymbol{G}_i^T \qquad \boldsymbol{b}' = \sum_i \boldsymbol{b}_i - \sum_i \boldsymbol{G}_i \left(\boldsymbol{\Gamma}_i^{-1}\boldsymbol{\beta}_i\right)$$

The solution $\boldsymbol{a}$ is the complete solution.

Solution of final system

The solution requires no iterations. Even the case $n = 5000$ will not take much time (but: use special solution method, where undetermined parameters are set to zero).

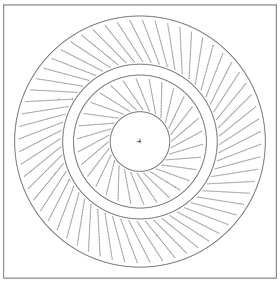
Iterations may be necessary for other reasons, namely

- the equations depend *non-linearly* on the global parameters; the equations have to be linearized;
- the data contain outliers, which have to be removed in a sequence of cuts, becoming narrower during the iteration;
- the accuracy of the data is not known before, and has to be determined from the data (after the alignment).

Method realized in the program **Millepede**. It provides a set of subroutines for the mathematical methods and allows to adapt the method to a particular problem with introduction of linear constraints.

Example: alignment of central track detectors (H1)

Alignment in the $r\varphi$-plane (perpendicular to the beam) of a 56-plane drift chamber and a 2-plane silicon vertex detector in the H1 detector



Components:

- drift chamber has a length (z) of about 2 m, and extending from 20.3 cm to 84.4 cm in radius r – resolution $\sigma = 150 \mu$m
- silicon vertex detector with two planes around the beampipe – resolution $\sigma = 15 \mu$m

Alignment fit . . .

. . . using ten thousands of tracks from *ep*-data, from cosmics with and without B-field.

Large number of correction parameters with small values:

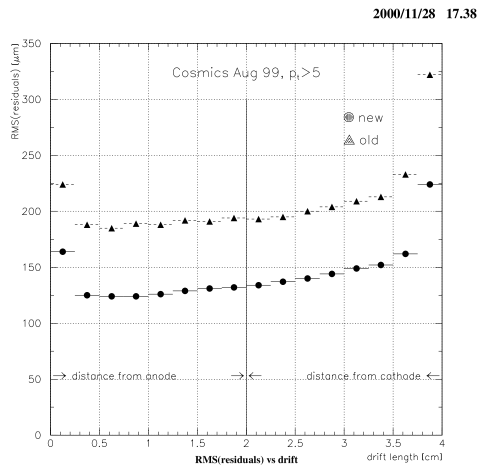
- geometrical shifts
- drift velocity and (relative) velocity corrections
- Lorentz angle corrections
- time corrections
- special correction for distortions of E field due to bad HV
- drift velocity change by electron beam current
- . . .

Table of parameters and precision

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

Improvement of drift chamber hit resolution

Mean track residuals from the pure drift chamber fit as function of the drift length – before and after improved alignment



→ improvement from $\sigma = 180\ \mu$m to $\sigma = 125\ \mu$m