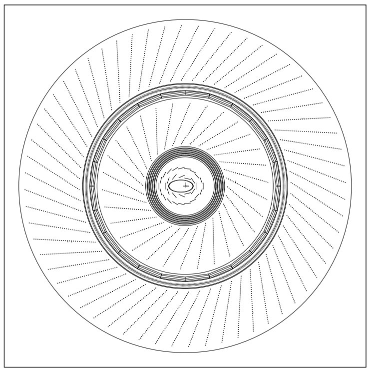
Central track finding and reconstruction

Track finding and reconstruction in H1 is traditionally done (but not planned!) in separate moduls for every detector: repeated vertex, primary and secondary track treatment.

The central track detectors are, with

- coordinates in $r\varphi$:
  - Central Jet Chamber CJC
  - Central Silicon Tracker CST
- and with coordinates in z:
  - Central Jet Chamber CJC
  - Central Inner Propchamber CIP
  - Central Outer z chamber COZ
  - Central Silicon Tracker CST



First development of CJC reconstruction (MC) around 1990, with reorganisation of code in 1992 - 1994.

Work on a common Central track **finding** and reconstruction started > 1 year ago. Aim: improved track reconstruction for new high luminosity runs at HERA II, not for reprocessing the 1995 data.

Problems with current scheme

- Only the CJC has *redundant* hit information to allow track finding; the CJC has, by construction, big holes (e.g. 20 cm to the first hit), but still good resolution in $r\varphi$ (calibration!); and less good resolution in z.
- CJC vertex fit of primary tracks is essential for a good track measurement; a huge improvement is obtained by the vertex fit in z (introduced in 1994).
- Addition of precise $r\varphi$ data from the CST (installation 1997) is difficult due to the high precision: adding a wrong hit may result in the loss of the primary vertex track. For V^0's the CST is still not used.
- In z all other chambers are more precise than the CJC: adding a wrong hit may cause loss of the primary vertex track. All other detectors produce a large number of noise hits and have problems with the efficiency (i.e. often one can choose among many hits, no one may be correct).
- The reconstruction of low momentum tracks and radiating electron tracks is difficult.

Conclusion: The ultimate and essential test for adding an accurate hit to the CJC track is the overall vertex fit (or the V^0 fit). New techniques are required to succeed without spending a long time in combinatorical attempts. These require simultaneous access to **all** hits from **all** detectors.

Content of the talk

1. New reconstruction module for central tracks
2. New track fit methods
3. Track finding in the $r\varphi$-plane of the CJC
4. The event T_0 problem
5. Track reconstruction in z,
   and primary vertex determination
6. Assignment of z-hits from other track detectors
7. Further remarks and summary

Track finding in OO with H1TrkFinder is something different.

1. New reconstruction module for central tracks

Basic principle: use of all data from the CJC, CST, CIP, COZ at the hit level during track reconstruction, vertex determination and vertex fitting.

- New code for the CJC reconstruction, based on CJCREC principles and experience from CJCREC,
- but completely rewritten …
- …with access to space points from the other detectors after the initial CJC track finding.

I need the code for the access to the space points, preferentially in so-called BOS banks, for the

- CIP $\surd$ (from K. Müller)
- COZ $\surd$ (from Zeuthen)
- CST $\surd$ (simplified steering routine which creates the space point bank only)

CJC run and event initialization

The event reconstruction requires

- the raw event data, and
- reconstruction parameters (geometry etc.) from the data base.

A single subroutine performs the check, wether the relelant information is present:

```
CALL JCHECK(IERR)
IF(IERR.NE.0) GOTO ...
```

For `IERR.EQ.0` all relevant information is present and reconstruction can start.

`JCHECK` can be also called, at any time, outside the reconstruction program for tests etc., multiple calls allowed.

Tasks of JCHECK

1. The presence of the bank `[HEAD,0]` is checked.
2. The presence of the bank `[XRJE,0]` is checked. If the bank is present, the subroutine returns immediately.
3. In case of a new run: all necessary data base banks are read from the data base, and the geometry subroutine `JCGEOM` is called, where all geometry parameters are calculated.
4. The presence of the raw data information in bank `[CRJE,0]` is checked. The information is copied in expanded form to the bank `[XRJE,0]` for direct use (wire-dependent T_0 is already subtracted) A pointer array `IPCTR` is prepared to allow a direct access to hits of a certain wire.
5. The bank `[CXYZ,0]`, is created (empty), parallel to `[XRJE,0]`. Later in the reconstruction it will contain all coordinates (x, y, z, z') and all other relevant information $(\cos\beta, \mathrm{d}E/\mathrm{d}x, \ldots)$.
6. An estimate of the event T_0 is determined from the distribution of the (corrected) drift times t_i.
7. The run xy-vertex information and the average drift velocity v_{drift} is fetched from the data base.

2. New track fit methods

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:

$$\eta(S) = \Delta d_{ca} \quad + \quad \Delta\varphi \cdot S + \frac{1}{2}\Delta\kappa \cdot S^2$$

$$\text{(+ after each scattering plane)} \quad + \quad \Delta\varphi_{1,pl} \cdot (S - S_{pl}) + \Delta\varphi_{2,pl} \cdot \frac{D_{pl}}{\sqrt{12}}$$

and additional terms in the least squares expression:

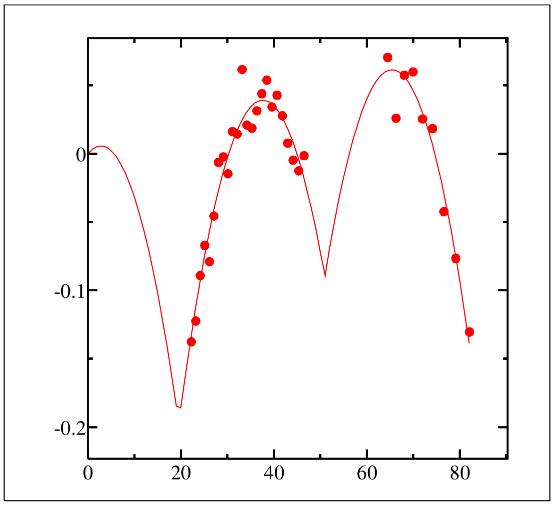
$$+\left(\frac{\Delta\varphi_{1,pl}}{\vartheta_0}\right)^2 + \left(\frac{\Delta\varphi_{2,pl}}{\vartheta_0}\right)^2 \qquad \text{with} \quad \vartheta_0 \approx \frac{0.0136 \text{ GeV/c}}{\beta\, p \text{ in GeV/c}} \sqrt{\frac{x}{X_0}} \times \ln \ldots$$

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

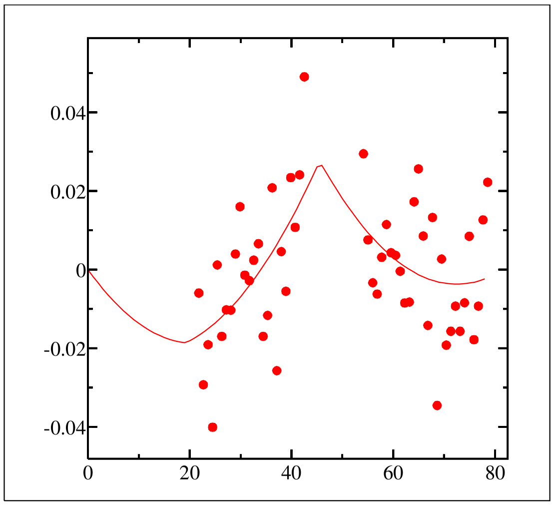
No parameter Δd_{ca} in vertex constrained fit. The primary vertex fit is different from the fit in the present CJCREC.

Examples for fits of multiple parabolas

Low momentum $p = 116$ MeV/c:



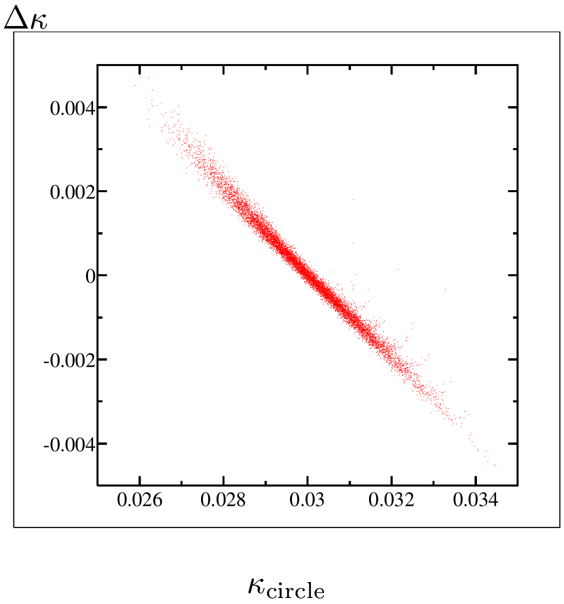
Momentum $p = 1.16$ GeV/c:

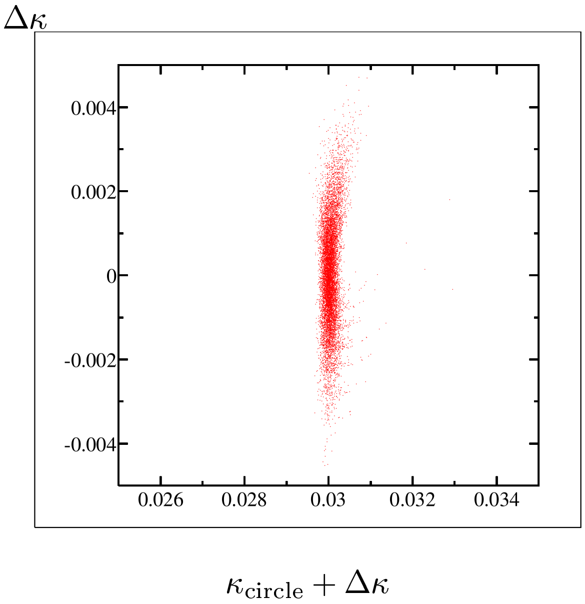


For high momentum (e.g. > 10 GeV/c) multiple scattering has no effect, but electrons may radiate photons.

MC study of momentum resolution

The sum κ from the circle fit $+\Delta\kappa$ from the fit of multiple parabolas has a narrower distribution $\rightarrow$ improvement of momentum resolution.





The curvature corresponds to a momentum of 116 MeV/c and a radius of 33.3 cm.

Straightline track fit in zS-plane

Hit value z from charge division:

$$z = \text{length} \cdot \frac{Q^+ - C_{1,k} \cdot Q^-}{Q^+ + C_{1,k} \cdot Q^-} + z_0$$

(correction of charge by a wire-dependent factor $C_{1,k}$).

The z-values of hits, accepted in the $r\varphi$-plane, as measured by the CJC have the properties

- standard deviations σ_z of several cm's,
- non-negligible fraction of outliers with deviations of > 100 cm.

The least squares fit of a straight line, minimizing the sum of squares of deviations, is computationally straightforward and simple, but the result is very often unsatisfactory.

Robust method (with well-defined minimum):
Minimizing, with respect to the parameters intercept and slope, the sum of the absolute values of deviations - which is unfortunately computationally not so simple (see next slide)

For simultaneous straight line fits (common z-vertex) normal least squares has to be used: a robust method (when a good approximations is already known) is given below.

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

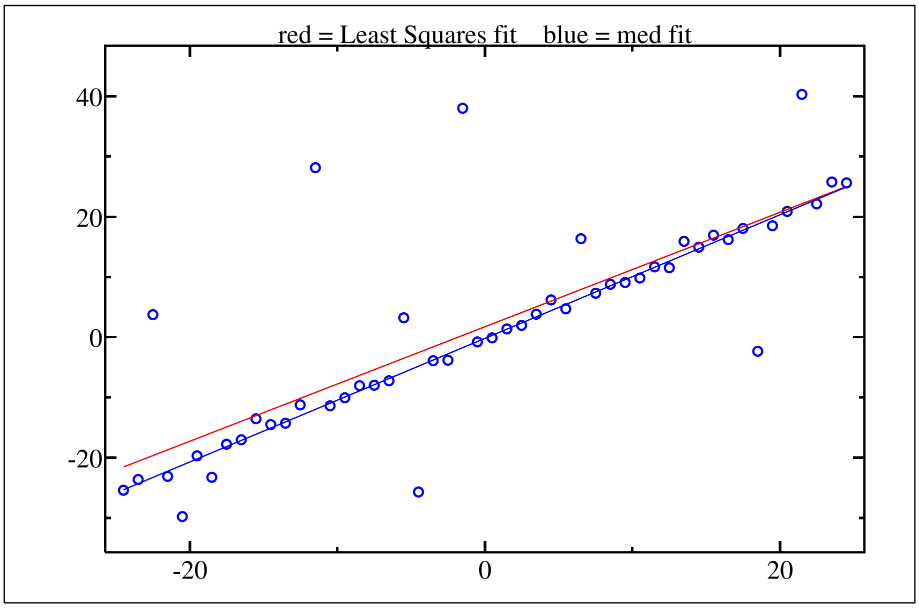
$$\frac{\partial F}{\partial b} = f(b) = \sum_{i=1}^{n} \text{sign}\, (y_i - a - bx_i) \cdot x_i \qquad \text{and} \quad f(\hat{b}) = 0$$

(with $\text{sign}\,(0) = 0$) 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^*:

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

  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*).

3. Track finding in the $r\varphi$-plane of the CJC

The CJC is the only central track detector with sufficient redundant information to *find* and define tracks. Track finding in the CJC is done exclusively in the $r\varphi$- or xy-plane, reconstruction is later extended to z.

Initially

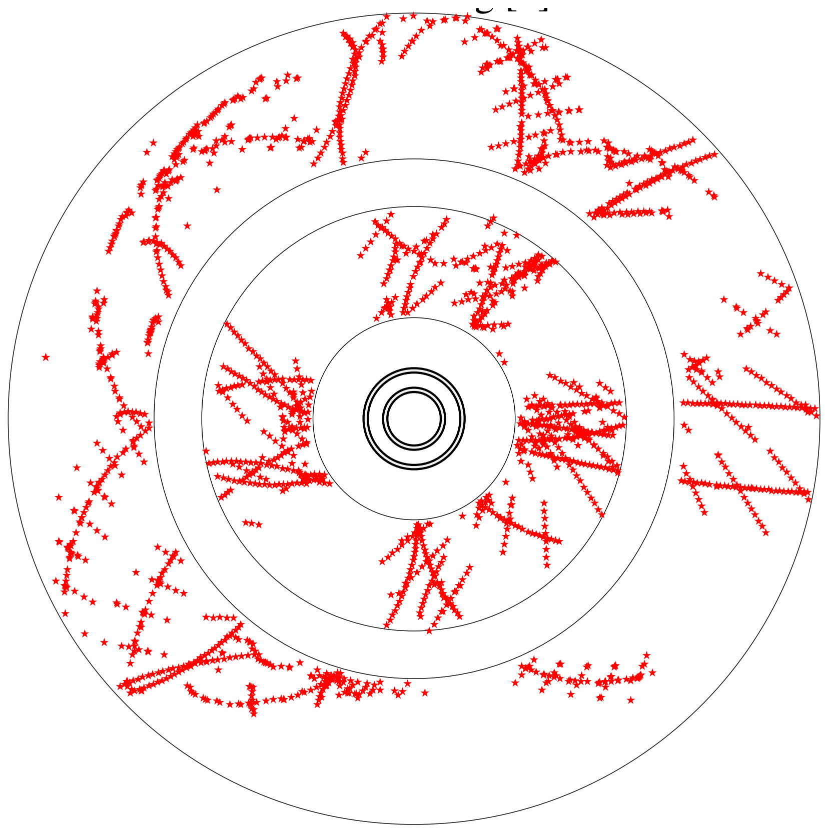
- event T_0 almost unknown,
- primary xy- and z-vertex unknown (z-vertex may be outside the CJC).

The basic strategy for track finding has two steps:

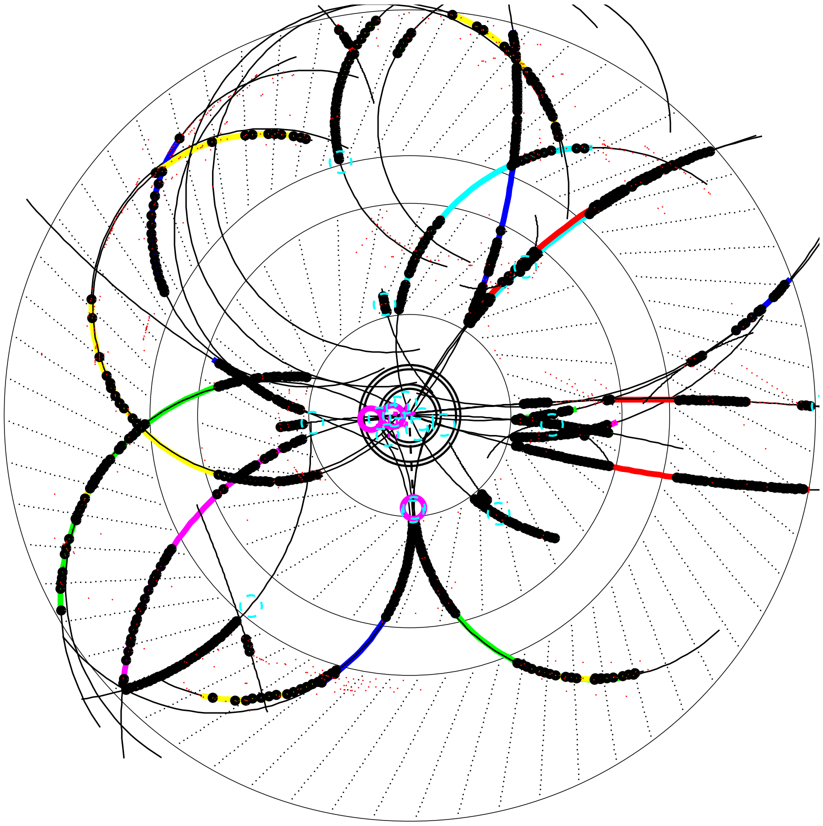
Initial track finding: Using different methods the parameters of potential tracks are estimated → track seed. The method has to be robust with respect to the basic parameters T_0, v_{drift} and α_L. Track seeds are expressed by the triple $(\kappa, d_{ca}, \varphi)$ (or $(\kappa, 0, \varphi)$) and the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

Track improvement: Starting from the knowledge of the triple $(\kappa, d_{ca}, \varphi)$ from a seed track 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.

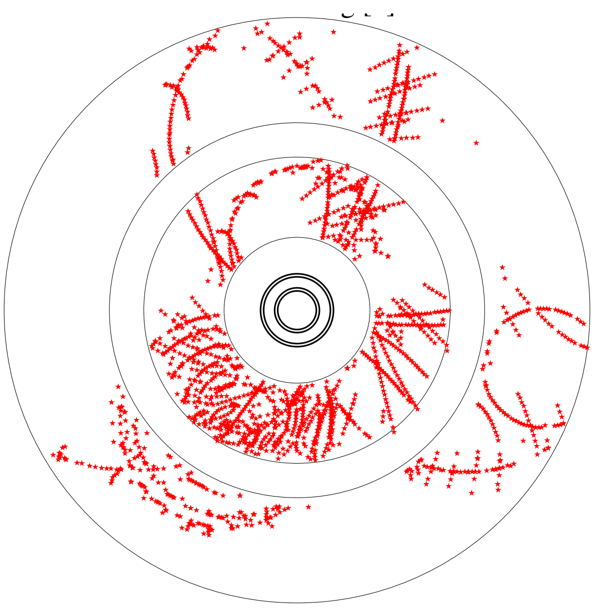
Initially no T_0, no vertex, no tracks – just hits and mirror hits

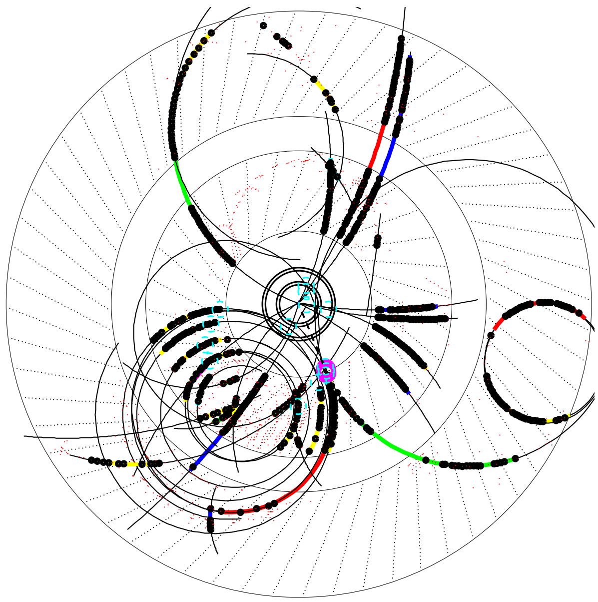


. . . after CJC reconstruction: tracks and vertices

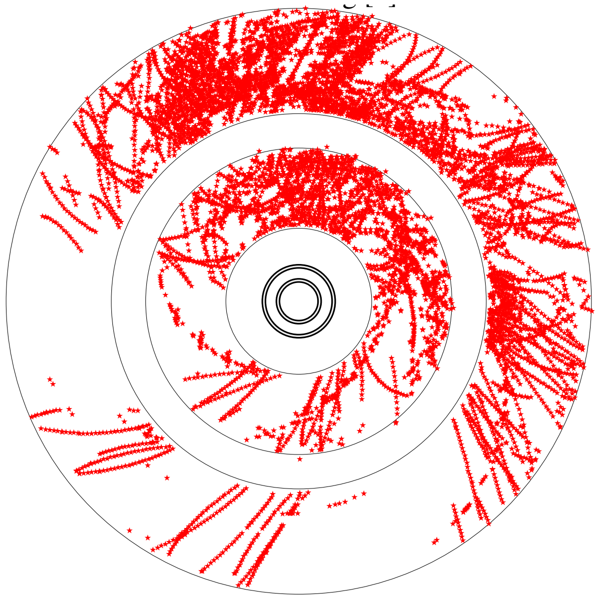


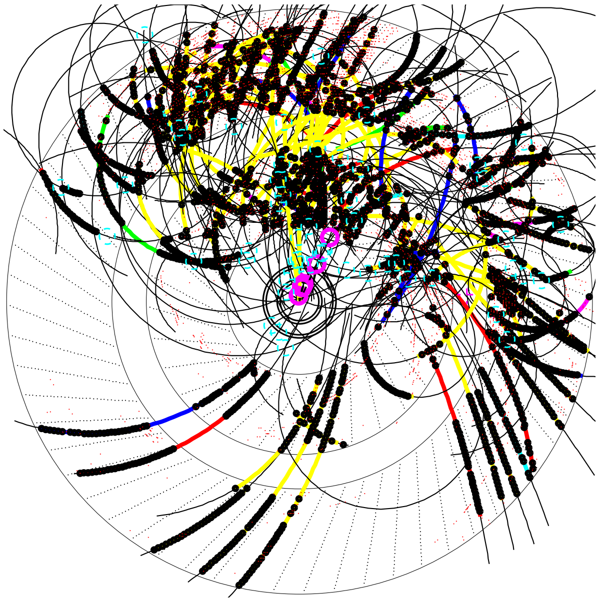
Another example . . .





. . . 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.

Initial track finding

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):

$d_j \approx (d_i + d_k)/2$?	all hits on the same side of the wire plane
$d_j \approx (\lvert d_i - d_k \rvert)/2$?	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.

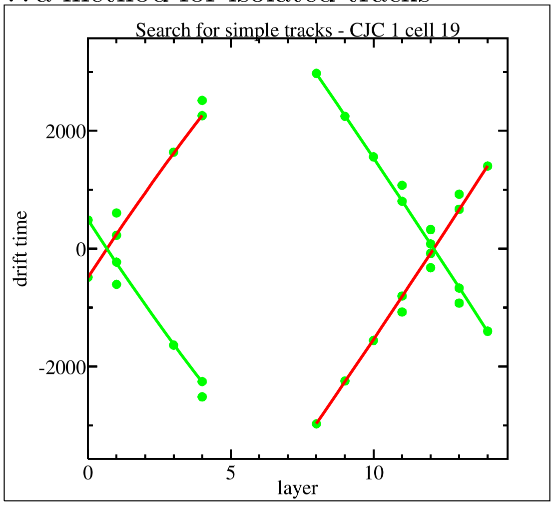
Eventually track seeds from different cells are combined.

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

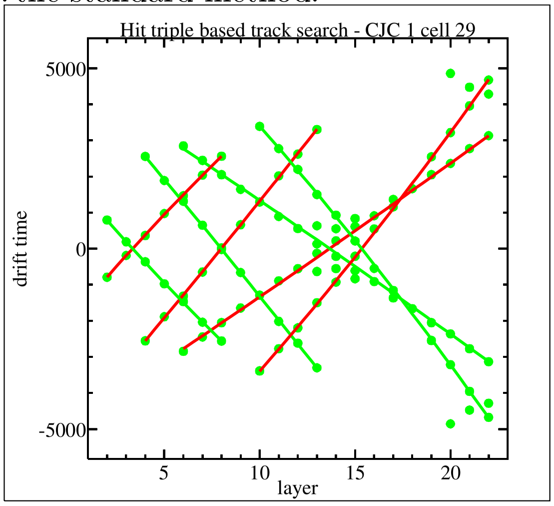
Example for seed finding in one cell

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.

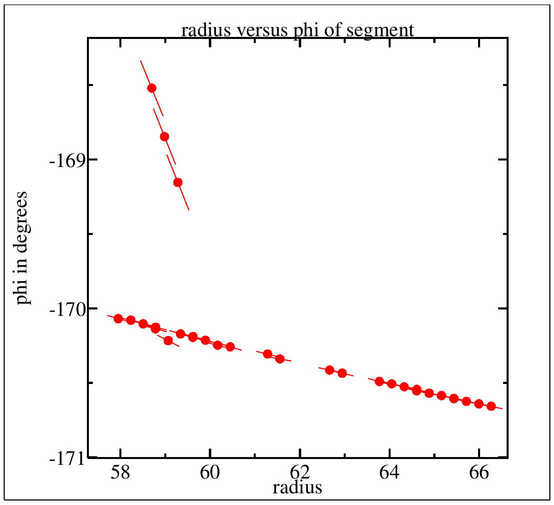
Crossing the zero-line (wire plane) requires good knowledge of the event T_0.

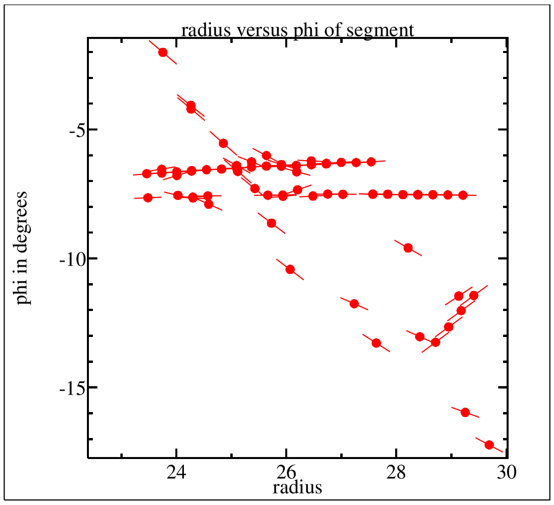
Example for hit triples from a single cell

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.

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.

Why and what is a 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 would 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

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

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.

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.

4. The event T_0 problem

The accurate determination of the event T_0 (from the tracks) is essential for an accurate track fit.

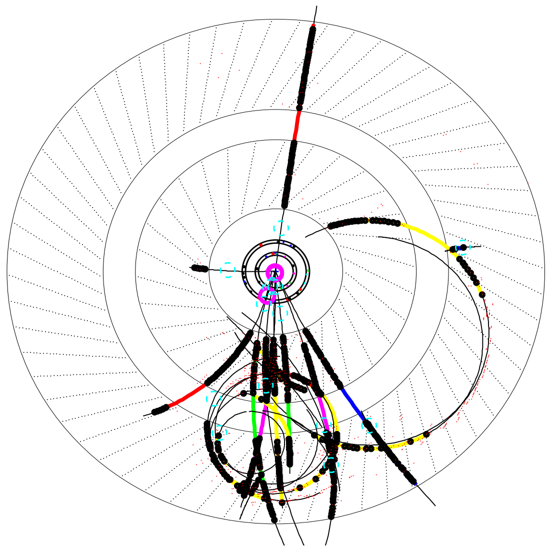
- $\rightarrow$ A wrong T_0 causes opposite shifts of hits left and right to the wire plane.
- $\leftarrow$ The T_0 is fitted for each track (and stored in the track bank since 1994); the accurate determination requires many crossings of the wire plane.

The present CJCREC combines all T_0-values to one event T_0 (by a median-like method), which is then used for all tracks. Curling tracks have a significantly different T_0. The presence of many non-vertex tracks may cause a shift in T_0, and such a shift can move e.g. a high-momentum electron track away from the primary vertex; this effect may be the most prominant reason for track-losses in the present CJCREC!

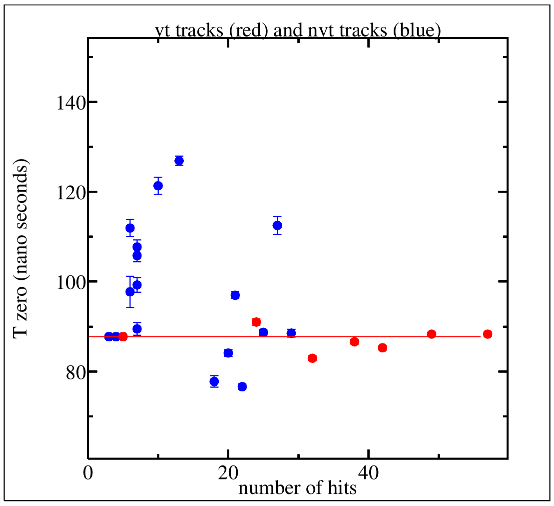
- Track fitting with T_0 determination and vertex constraint plus CST hits will increase the T_0 precision!

Example: a DIS event

...with 7 primary-vertex track and 17 non-vertex tracks:



$r\varphi$-view

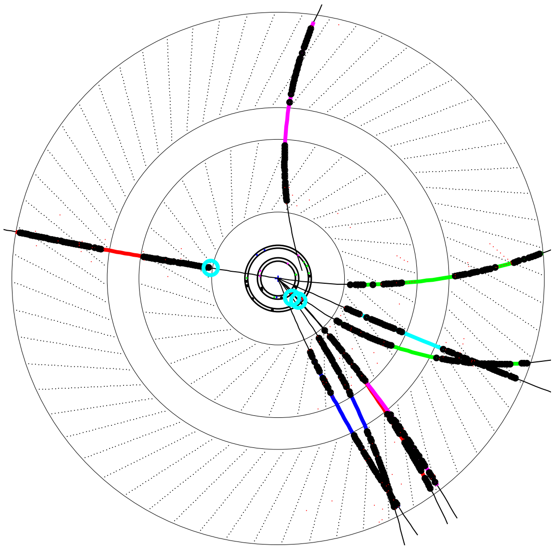


The T_0 values of the primary tracks (red) are close together. Many non-vertex tracks (blue) are "later" in time.

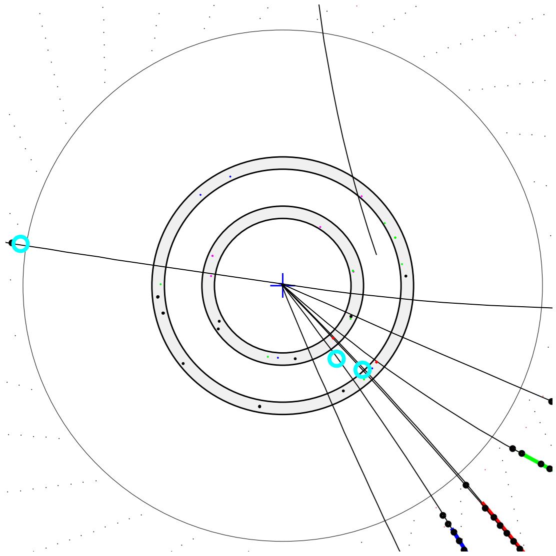
The event T_0 as calculated from the track T_0's is given by the red line.

Example: another DIS event

. . . with an e^-, 7 vertex tracks and 1 non-vertex track:

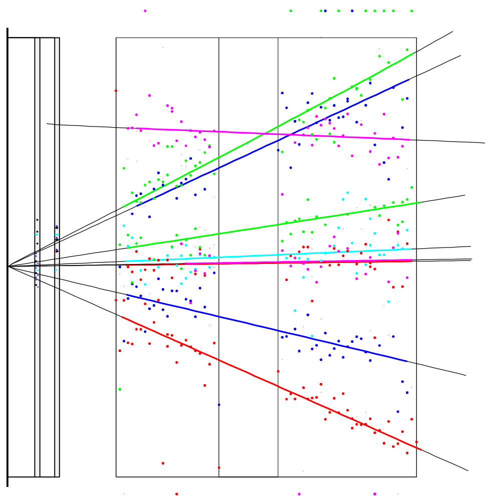


$r\varphi$-view

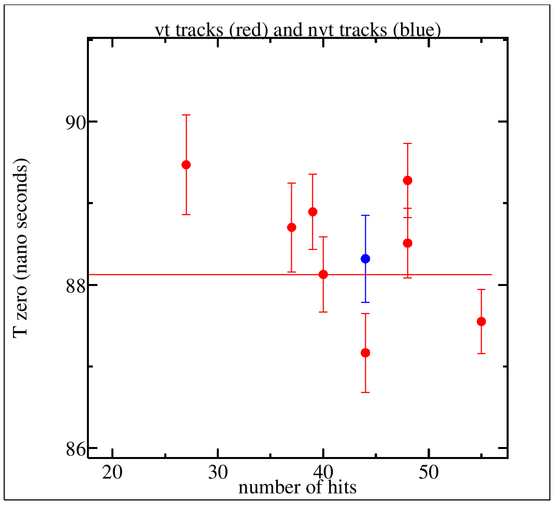


Region around the vertex, with CST hits.

. . . the same DIS event

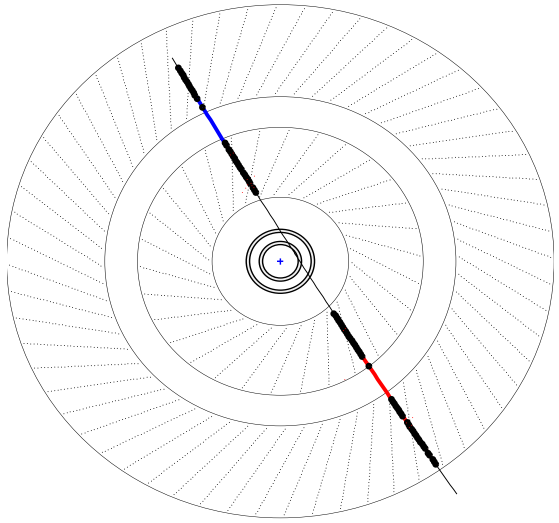


$r\,z$-view

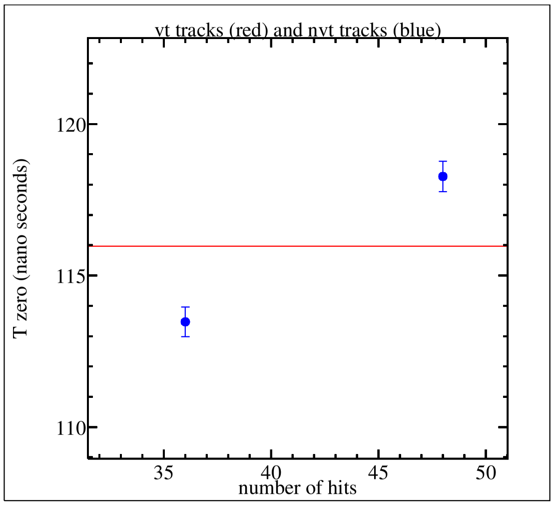


All T_0 values of the primary tracks (red) are close together.

Example: a cosmic



$r\varphi$-view



The two T_0 values of the cosmic tracks

The T_0 difference is 4.8 nano seconds equivalent to a distance of 1.50 m.

5. Track reconstruction in z and primary-vertex determination

Steps in the z and primary track reconstruction:

1. Single track fits in zS-plane (minimizing the sum absolute deviations):

$$z_i = z_0 + S_i \cdot \tan\lambda$$

2. Initial estimate of primary z-vertex z_0 from candidates for primary tracks. By a simple cut tracks with small d_{ca} are selected as candidates for primary tracks. All hits from these tracks are extrapolated to the beam line by

$$z_{\text{ext}} = z_i - S_i \cdot \tan\lambda$$

   A cluster search (*to* next slide) is performed in the set of all z_{ext} to obtain an initial estimate for z_0 (which may be far outside the detector).

3. Common least squares vertex track fit of primary tracks, iterated.

$$z_i = z_0 + S_i \cdot \tan\lambda_j \qquad j = 1, 2 \ldots n \quad \text{(primary tracks)}$$

   with $(n+1)$ parameters, using the robust technique of pseudo-observations. Tracks are eventually removed from list of primary tracks, with fit repeated.

4. Run xy-vertex determination from $z0$, followed by $r\varphi$-plane track fits to primary vertex.

Steps 2 to 4 are repeated, if a track turn out to be not a primary track.

Cluster search

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`:

- 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

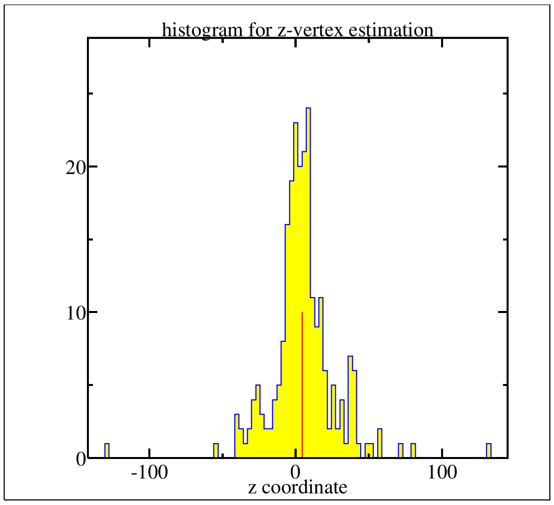
$$\text{density} = \frac{n_{ij}}{\max(x_j - x_i, \texttt{WID/2})}$$

  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.

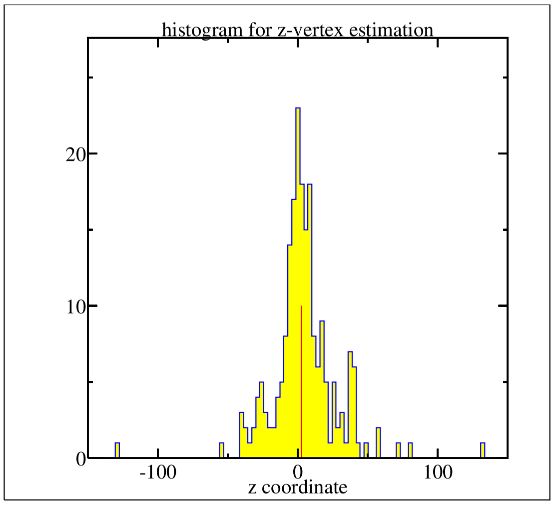
Examples for step 2: estimate of primary z-vertex z_0

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

z vertex distribution



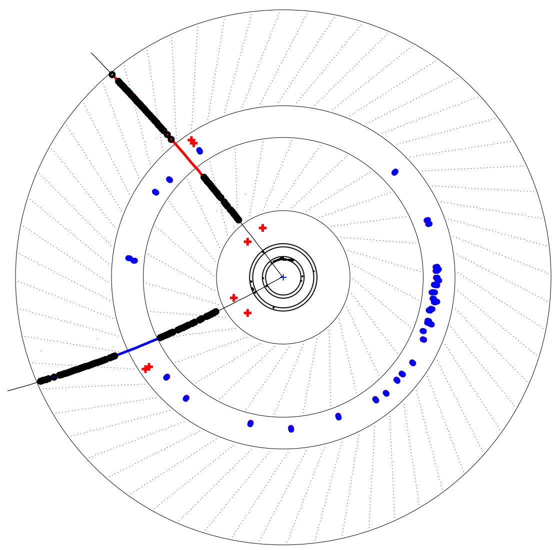
...from vertex-track candidates



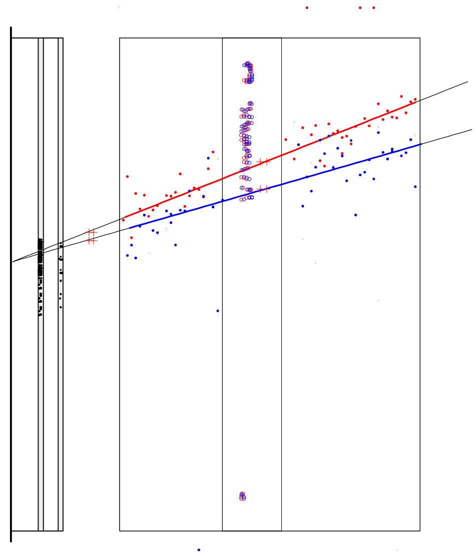
...and after removal of non-vertex track:

Hits from other central detectors

An elastic J/Ψ-event with CIP/COP/COZ/CST hits:



. . . $r\varphi$-view . . .

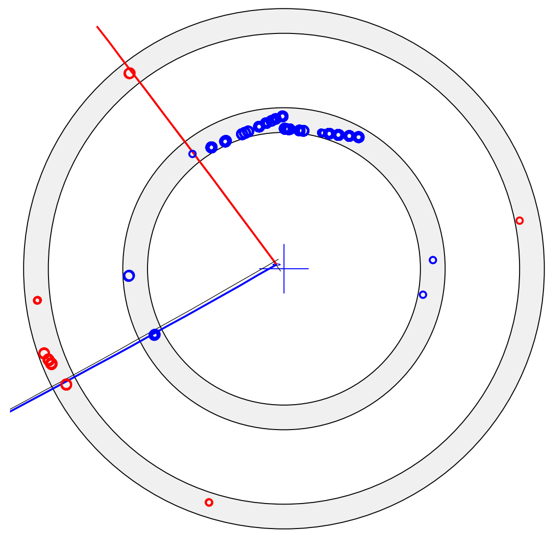


. . . and rz-view.

Hits from other central detectors – cleaning

Cleaning of set of z-hits from:

- **CIP:** accept only isolated hits; combine cluster of hits to one z-value or use single hits.
- **COZ:** remove all hits which are far from CJC tracks; combine cluster of hits or use single hits.
- **CST:** accept only hits which are close to CJC tracks; eventually selection of single hits if there are many close hits.



The CST region around the primary vertex.

In the following plots the *cleaning procedure* has always been applied.

The number of hits may be reduced by ≈ 90 %.

6. Assignment of z-hits (linking) from other central detectors

. . . other detectors are CST, CIP and COZ.

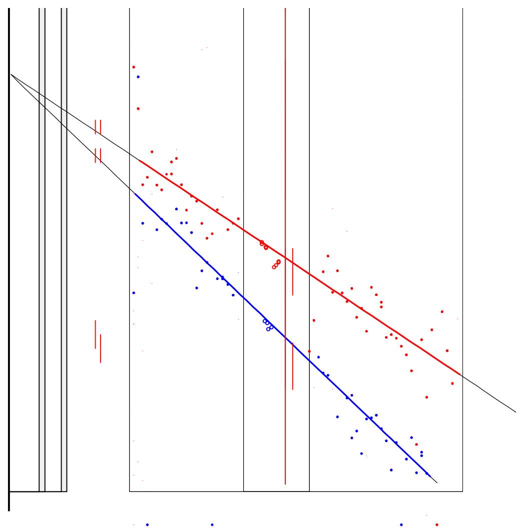
Assignment of z-hits to a track should never lead to the loss of a vertex track!

Method:

1. Make CJC-only vertex fit in z with $n+1$ parameters: n slopes and the value z_0 (note: z_0 may be wrong by cm's).
2. Select two z hits for one of the tracks from CST and CIP/COZ and repeat vertex fit with all CJC tracks plus the two additional z hits, store z_0-difference Δz_0, and repeat this for all possible z candidate hits.
3. Search for peak (cluster) in the set of Δz_0-values $\rightarrow \widehat{\Delta z_0}$-peak. No hits are accepted if no peak is visible.
4. If peak visible: select hits from CST and CIP/COZ, compatible with $\widehat{\Delta z_0}$-peak value, and make final vertex fit in z.

This may require ≈ 100 $(n+1)$-parameter vertex fits! CPU time problem?

Example: an elastic J/Ψ event



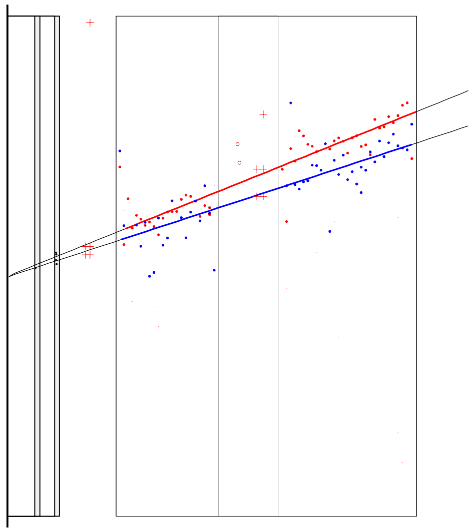
$r\,z$-view

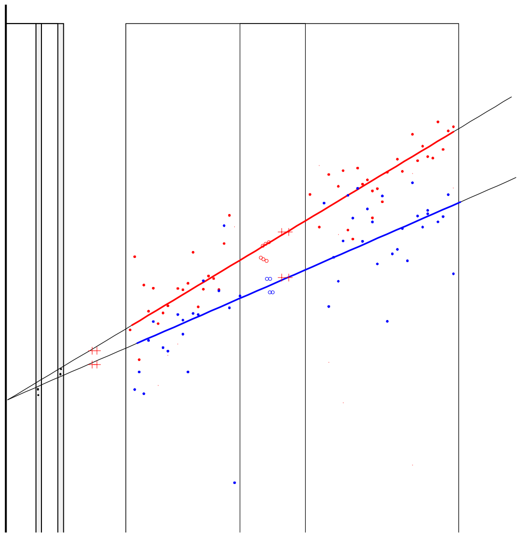
Shown are

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

rz-views after cleaning: Example 1 and 2

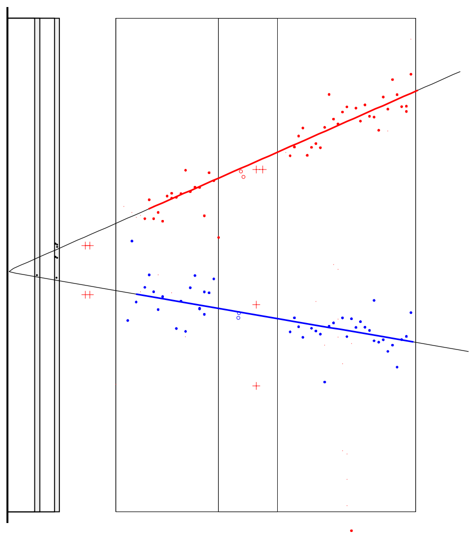
Two elastic J/Ψ-events with CIP/COP/COZ/CST hits:

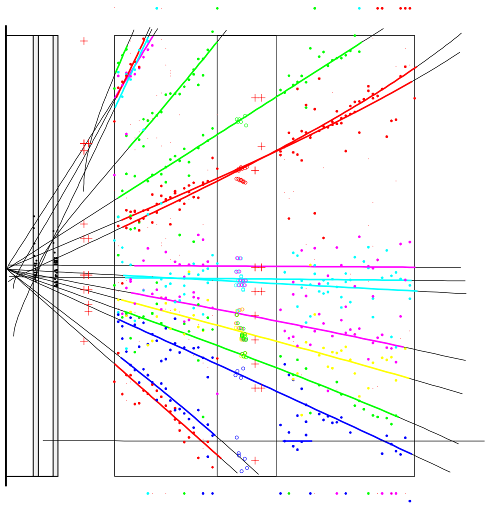




rz-views after cleaning: Example 3 and 4

An elastic J/Ψ-event and a multi-track event with CIP/COP/COZ/CST hits:





Simultaneous straight line fits with common intercept

With the line (track) index j the model is

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

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

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

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

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.

Further remarks and summary

Not mentioned:

- new output bank structure - eventually with decisions on track selection;
- new V^0 and $\gamma \to e^+e^-$ search and fit (different constraints with analytical derivatives);
- track extrapolation to calorimeter and clustering in the calorimeter;

…and missing:

- alignment code within the module.

Hopefully …

- higher efficiency in general by improved track finding and linking, and improved T_0 handling,
- improvement for low-momentum tracks (e.g. D^*) and V^0's.