25th January 2007

Draft

Broken-line fits
for charged-particle trajectory reconstruction

The Manual

by

Volker Blobel

Abstract

The reconstruction of trajectories of charged particles measured in a tracking chamber is complicated by multiple scattering, especially at low momenta and for dense detectors. A parametrization of the trajectory with a small number of parameters is impossible because of many small-angle scatters along the trajectory, caused by multiple scattering. The expected angular deflection in layers of material depends on the momentum and on the mass of the particle and can be calculated. Fast algorithms and subprograms based on a novel approach to treat the multiple-scattering effects with broken-line fits are described. The algorithm requires to solve matrix equations with a band matrix. Fast routines for the solution of the matrix equations and for the determination of elements of the inverse matrix are described; they have an execution time proportional to the number of measured points. Included are fast circle-fit routines and utilities for circle parameters. These routines can be used for a first fit through the data and can give an estimate of the particle momentum.

During the track recognition from raw hits many fits have to be performed, often to data with a certain fraction of outliers, hits from other tracks and noise hits. Subroutines are included for robust fits, which allow to recognize the outliers.

Contents

1 Introduction to broken-line fits

1.1 Track parameters

In this paper the track measurement in a typical storage ring detector is considered with a z-axis in the beam axis. In a constant magnetic field $\boldsymbol{B}$ in z-direction the trajectory of a particle is a helix with axis in z-direction and a radius R in the bending plane, the $r\phi$-plane, if multiple scattering and field inhomogeneities are absent. The radius of curvature R (in cm) and the momentum (in GeV/c) is related to the momentum component perpendicular to $\boldsymbol{B}$ (in the xy-plane) by the equation

$$p\perp \equiv p\cos\lambda \equiv \frac{p}{\sqrt{1+\tan^2\lambda}} = 0.00299792458\ B\,R$$

for singly charged particles, with $\boldsymbol{B}$ in Tesla; here λ is the pitch angle. Instead of the radius of curvature R the inverse value, the signed curvature $\kappa(=\pm 1/R)$ is used as a parameter, with the sign given by the sign of the charge ($\pm e$) of the particle. Instead of pitch angle λ sometimes the polar angle ϑ is used: the relation between the two angles is

$$\lambda + \vartheta = \pi/2 \qquad \tan\lambda = \cot\vartheta = \frac{1}{\tan\vartheta}$$

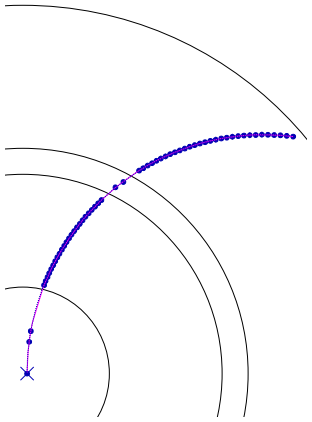

Figure 1: A 200 MeV/c track in a storage ring detector, shown in the $r\phi$-plane from a Monte Carlo simulation. The curve shown is a circle fitted to the data points. The data points are two accurate hits measured in a silicon vertex detector, and hits measured in two drift chambers. Two hits with a reduced accuracy are measured between the two drift chambres in a special z-chamber. The χ^2 of the fit is, due to multiple scattering, rather large. The deviations of the hits from the circle are however not visible in this plot, but see Figure 2.

The Figure 1 shows a low-momentum track in the $r\phi$-plane, with a circle fitted to the data. Here a usual xyz-coordinate system is assumed. The equation for the residual ε_i of a point x_i, y_i (in a xyz-coordinate system) in the $r\phi$- or xy-plane from a circle is given by

$$\varepsilon_i = \frac{1}{2}\kappa\left(x_i^2 + y_i^2 + d_{\rm ca}^2\right) - (1+\kappa d_{\rm ca})\left(x_i\sin\phi - y_i\cos\phi\right) + d_{\rm ca}\ . \tag{1}$$

The parametrization has three variables: the curvature $\kappa = \pm 1/R$ (R = radius of curvature), the direction ϕ (direction of propagation at the point of closest approach to the reference point), and the distance of closest approach $d_{\rm ca}$ (impact parameter). This equation is valid over a full trajectory loop, and also in the straight-line limit $\kappa \to 0$. The parametrization of the trajectory provides a continuous transition between negative and positive curvature as well as between negative and positive distance of closest approach.
The track length s_i is usually defined in the $r\phi$-plane. Using this track length the residuals of a point s_i, z_i from a straight line is given by

$$\varepsilon_i = z_i - (z_0 + (\tan\lambda)\cdot s_i)\ . \tag{2}$$

without any deflection by the magnetic field.

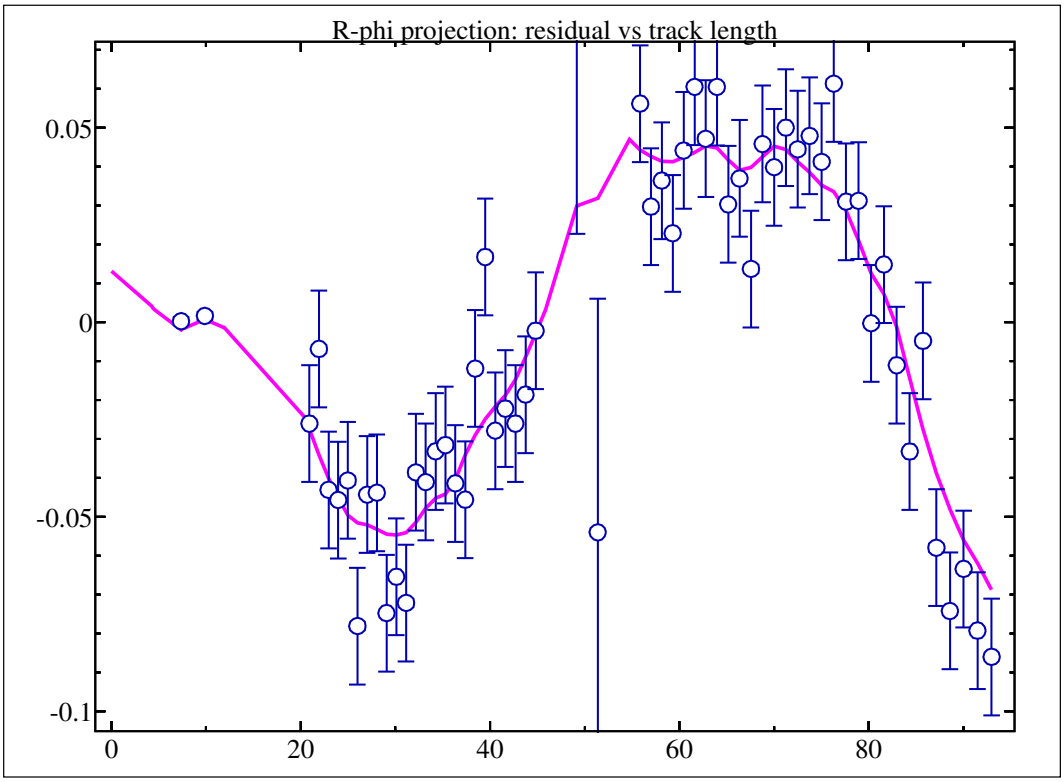

Figure 2: The track residuals of the track of Figure 1 are shown as a function of the track length. The curve shows the true trajectory in this Monte Carlo simulation. The effect of multiple scattering is clearly visible, especially in regions with dense material like beam pipe and detector walls.

1.2 Multiple scattering and broken-line algorithm

If a charged particle traverses material, there will be, in addition to the systematic deflection due to a magnetic field, many small random deflections; this is called multiple scattering. Due to the random nature of the many small deflections no simple track parametrization is possible. The effect of multiple scattering is clearly visible in Figure 2, which shows the track residuals, the distances of the data points from the fitted circle for the case of Figure 1. It is clear that the χ^2 of the circle fit is large, due to the multiple scattering. The distance of a single hit from the fitted circle cannot be used to decide whether the hit is acceptable or an outlier. The fitted parameters, the curvature and the track direction will be biased. Taking into account multiple scattering requires special fit methods.

One method is to include all multiple scattering effects in the covariance matrix of the n measured data, which becomes non-diagonal. Such a method requires a large computing time proportional to almost the third power of n, and the parameters are determined only w.r.t. a certain reference point, e.g. the vertex. Information at the track end and for the single points is not directly available.

The introduction of the Kalman filter method was a great progress. The large non-diagonal covariance matrix is avoided and the computing time is essentially proportional to n. It became a standard method of track reconstruction because of this linear dependence and because of the fact that the fitted data can give a complete description of the track including each single point. This Kalman fit is a sequential fit, which start from one track end and includes then hit by hit, improving the track parameters with every hit. The sequential procedure has also to be performed in the opposite direction, to obtain good

parameters over the whole track length.
Another possibility is to make a two-level fit of the track. In the first fit a helix is approximately fitted in the $r\phi$-plane through all measured values, for example by the robust method of Karimäki; no starting values are necessary in this fit, which is fast and delivers a good average description of the trajectory, with a constant value for the curvature and in the absence of multiple scattering. In the sz-plane a straight-line (two parameters) is fitted to the data. The curvature allows to determine an approximate momentum value and the size of the multiple scattering variances can be calculated. Then residuals of the measured coordinates with respect to the fitted circle are calculated together with the track length, and these data can be used as input for a second fit, which takes multiple scattering into account.

The broken-line fit is a method for track fits including multiple scattering, with computing time proportional to n, and applicable for certain track detector environments. In this section the following coordinates are considered for the broken line fit:

$$\begin{aligned} y_i &= \text{residual}, \quad i = 1, 2, \ldots n \\ s_i &= \text{track length in the } r\phi\text{-plane} \\ w_i &= \text{weight}(\equiv 1/\sigma_i^2) \end{aligned}$$

with $s_i < s_{i+1}$. The values s_i are the track length in increasing order, usually with $s_1 = 0$. The measured values y_i may be the distances of the original measured value from a simple track parameterization with a minimum number of parameters. The accuracy of the measured value is given by the weight w_i, which is the inverse variance of the measurement.

1.2.1 Broken-line fit for tracks without curvature

The trajectory of a particle in a in a plane of a detector without deflection by a magnetic field is a straight line in the absence of multiple scattering effects. The parametrization

$$y \cong a_1 + a_2\, s$$

where s is the length along the track, could be used in a fit to data $y_i, i = 1, 2, \ldots n$ to determine the track parameters intercept a_1 and slope a_2. This parametrization can also be used if multiple scattering effects are small, but would deliver inaccurate parameter values in case of larger multiple scattering effects.

Figure 3 shows the trajectory of a charged particle with multiple scattering, and the intersection points of the trajectory with the detector planes. The coordinates y_i, with standard deviation σ_i, represent measurements of the intersection points at the detector planes with coordinates s_i, for $i = 1, 2, \ldots n$. The measured coordinates y_i are transverse to the average track direction, and are uncorrelated.

Instead of fitting a track parametrization with e.g. intercept and slopes as parameters, the track-fit method developed here uses two phases in the track reconstruction:

Reconstruction of the trajectory: The trajectory, represented by the intersection points of the trajectory with the detector planes, is determined in a least squares fit; the estimates of the intersection points are denoted by u_i. Although a matrix equation with a n-by-n matrix has to be solved, the computation time is linear in n because the matrix is a band matrix of small band width.

Track parameter determination: From the fitted u_i-values the two track parameters intercept and slope, required for the physics analysis, are determined including the covariance matrix; simple error propagation is sufficient. The track parameters can be determined at both sides of the track.

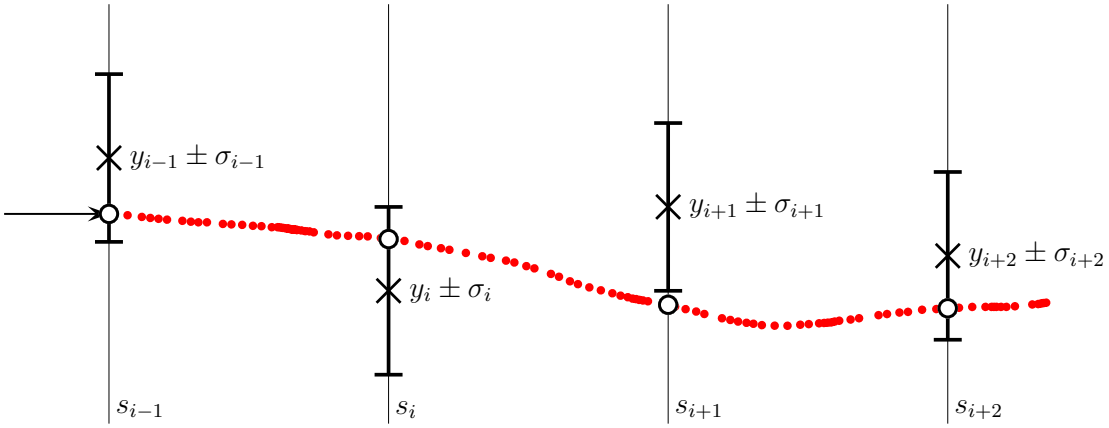

Figure 3: A charged particle incident from the left in the plane of the figure traverses a medium with several detector planes s. The particle track, given by a dotted curve, intersects the detector planes, the intersection points are drawn as circles. The result of the measurement are data points $y \pm \sigma$, given by crosses with error bar.

The least squares fit thus has n parameters u_i to be determined, one for each measured value y_i. The sum of squares to be minimized includes the sum of $(y_i - u_i)^2/\sigma_i^2$. The intersection points u_i to be fitted are shown in Figure 4. Each two adjacent points are connected by straight lines. The angles between the connecting straight line and the true particle direction to the left and to the right of a material layer are called ψ_L and ψ_R. The expectation values of these angles are zero, and the variance can be calculated from the multiple scattering. The result, discussed in detail in section 2, is a covariance matrix

$$V \begin{bmatrix} \psi_L \\ \psi_R \end{bmatrix} = \begin{pmatrix} V_L & V_{LR} \\ V_{LR,} & V_R \end{pmatrix} ,$$

determined by the material between s_i and s_{i+1}. There is a kink angle β_i,

$$\beta_i = \psi_{R,i-1} - \psi_{L,i}$$

due to multiple scattering, between each straight line segments. Expectation values $E[\beta_i]$ are zero and the variances $V[\beta_i]$ can be calculated from the variances of the angles ψ_L and ψ_R (section 2) of the layers between detector planes. As an example, for a homogenous material the variance is

$$V[\beta_i] = V[\psi_{\text{right},i}] + V[\psi_{\text{left},i+1}] = 1/3 \left(\theta_{0,i}^2 + \theta_{0,i+1}^2\right)$$

More general the covariance matrix of two adjacent β-values is given by (see equation (21))

$$V \begin{bmatrix} \beta_i \\ \beta_{i+1} \end{bmatrix} = \begin{pmatrix} V_{R,i} + V_{L,i+1} & V_{LR,i+1} \\ V_{LR,i+1} & V_{R,i+1} + V_{L,i+2} \end{pmatrix} .$$

The correlation between the angles ψ_{left} and ψ_{right} in one layer is according to equation (21) positive and usually small, it can be zero (thin material on one side of the layer) and it can be large (thin material in the center). This correlation propagates into a correlation between adjacent β_i values. The fact, that the correlation coefficient is non-zero and positive corresponds to the fact that a large scatter in one layer influence both adjacent β-values. As in other methods (where this correlation is usually never mentioned) the correlations between adjacent layers are neglected in the following.

An approximation of the true trajectory is given by the polyline, connecting the points (s_i, u_i). The angles of the line segments to the s-axis are denoted by α. The $\tan\alpha_i$ and $\tan\alpha_{i+1}$ values of the line

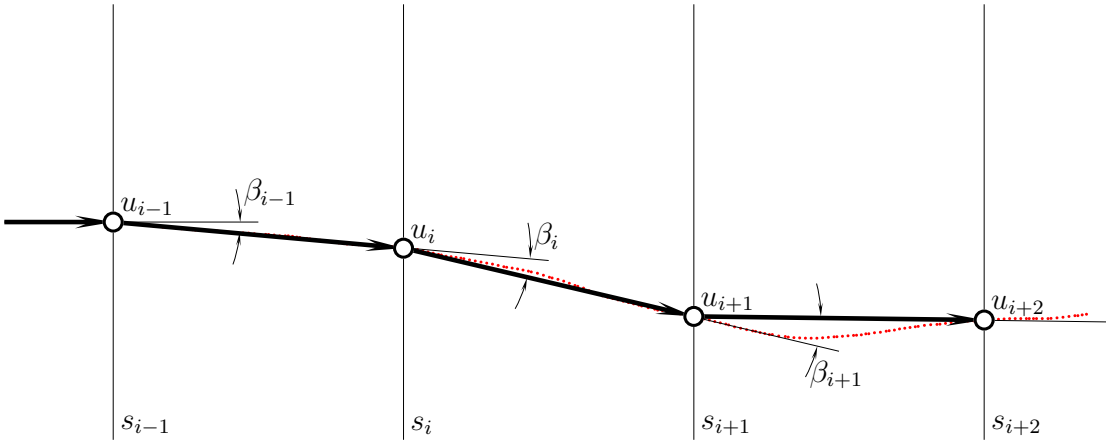

Figure 4: A charged particle incident from the left in the plane of the figure traverses a medium with several detector planes s. The intersection points u of the particle track with detector planes, drawn as circles, are connected by straight lines. The kink angles β are the angles between adjacent straight lines.

segments between s_{i-1} and s_i, and between s_i and s_{i+1} are given by

$$\tan\alpha_i = \frac{u_i - u_{i-1}}{s_i - s_{i-1}} \qquad \tan\alpha_{i+1} = \frac{u_{i+1} - u_i}{s_{i+1} - s_i} \tag{3}$$

and the angle β_i between these two line segments is given by

$$\tan\beta_i = \tan\left(\alpha_{i+1} - \alpha_i\right) = \frac{\tan\alpha_{i+1} - \tan\alpha_i}{1 + \tan\alpha_i \cdot \tan\alpha_{i+1}} \tag{4}$$

The difference between the $\tan\alpha$-values and hence the angle β_i will be small, thus for small angles $\tan\beta$ is well approximated by the angle β and

$$\beta_i \approx f_i \cdot (\tan\alpha_{i+1} - \tan\alpha_i) \qquad \text{with factor } f_i = \frac{1}{1 + \tan\alpha_i \cdot \tan\alpha_{i+1}} \tag{5}$$

is a good approximation. The angle β can be expressed by u-values and after some rearrangement the formula for the angles β_i can be written as

$$\beta_i = f_i \cdot \left[u_{i-1} \frac{1}{s_i - s_{i-1}} - u_i \frac{s_{i+1} - s_{i-1}}{(s_{i+1} - s_i)\,(s_i - s_{i-1})} + u_{i+1} \frac{1}{s_{i+1} - s_i} \right] , \tag{6}$$

or, with the definition $\delta_i = 1/(s_{i+1} - s_i)$, the angle is

$$\beta_i = f_i \cdot [u_{i-1}\,\delta_{i-1} - u_i\,(\delta_{i-1} + \delta_i) + u_{i+1}\,\delta_i] \tag{7}$$

The angles β_i are linear functions of the values u_i if the factors f_i are assumed to be constants, and thus the problem is a linear least squares problem, which is solved without iterations. Usually factors f_i are very close to 1 and the difference to 1 can be neglected[1].

The angles β_i, which all have a mean value of zero and a variance given by the multiple scattering theory, can be considered as $(n-2)$ measurements in addition to the n measurements y_i, and these

[1] If they have to be taken into account two iterations become necessary.

total $(2n-2)$ measurements allow to determine estimates of the n true points u_i in a least squares fit by minimizing the function S

$$\boxed{S(\boldsymbol{u}) = \sum_{i=1}^{n} \frac{(y_i - u_i)^2}{\sigma_i^2} + \sum_{i=2}^{n-1} \frac{\beta_i^2}{\sigma_{\beta,i}^2}} \tag{8}$$

with respect to the values u_i. No explicit parametrization of the trajectory is defined, but instead the improved measured values are assumed as the parameters and the sum depends on the *rms* multiple scattering angles.

The vector $\boldsymbol{u}$ that minimizes the sum-expression S is given by the solution of the standard normal equations of least squares

$$\boldsymbol{C}\boldsymbol{u} = \boldsymbol{r} \tag{9}$$

The matrix $\boldsymbol{C}$ is a symmetric band matrix, where all elements outside a band along the diagonal vanish. A band matrix is characterized by the *bandwidth* m, with $C_{ik} = 0$ for all i and k with $|i-k| \geq m-1$. In the case above the band width is $m = 3$ (five adjacent diagonals), and because of the symmetry an 3-by-n array is sufficient to store the band matrix. Operations with band matrices are discussed in detail in section 6.

The computing time to solve the system of normal equations is only proportional to n, because of the *band structure* of the matrix. The values u_1 and u_n are the intercepts at the two ends to the track. The values u_i obtained from the solution, can be used to calculate the initial and the final slope of the track by

$$\text{initial slope} \quad = \frac{u_2 - u_1}{s_2 - s_1} \qquad\qquad \text{final slope} \quad = \frac{u_n - u_{n-1}}{s_n - s_{n-1}} \; . \tag{10}$$

The complete inverse of the band matrix is a full matrix (all elements non-zero); it is the covariance matrix of the values u_i. This matrix is not calculated, because it would take a time proportional to n^3. However there is a fast method with a computing time proportional to n to calculated the band part of the inverse matrix, and this band contains enough information to calculated the individual variances of the u_i-values and the full covariance matrices of the initial and final intercept and slope values.

1.2.2 Tracks with curvature

The formalism developed for track without curvature has to be modified to describe the superposition of the deflection by the magnetic field and the random walk, caused by the multiple scattering. The segments between the points $(s_i,\, u_i)$ and $(s_{i+1},\, u_{i+1})$, which were straight lines in the case of field $\boldsymbol{B} = 0$, are now curved, with curvature κ. The chord a_i of the segment is given by

$$a_i = \sqrt{\Delta s_i^2 + \Delta y_i^2} \qquad \text{with } \Delta s_i = s_{i+1} - s_i \qquad \Delta y_i = y_{i+1} - y_i \; .$$

The angle γ_i between the chord and the circle at one end of the chord is given by[2]

$$\gamma_i \approx \sin\gamma_i = a_i \kappa / 2$$

and thus in a very good approximation proportional to the curvature κ.

The mean value of the angle β_i, as defined before in equation (5), is now different from zero. By the new definition

$$\beta_i \approx f_i \cdot (\tan\alpha_{i+1} - \tan\alpha_i) + (a_{i-1} + a_i)\,\kappa/2 \tag{11}$$

[2]The difference between $\arcsin\gamma$ and γ is about 1 % at a angle of $\gamma = 0.24 \equiv 14°$.

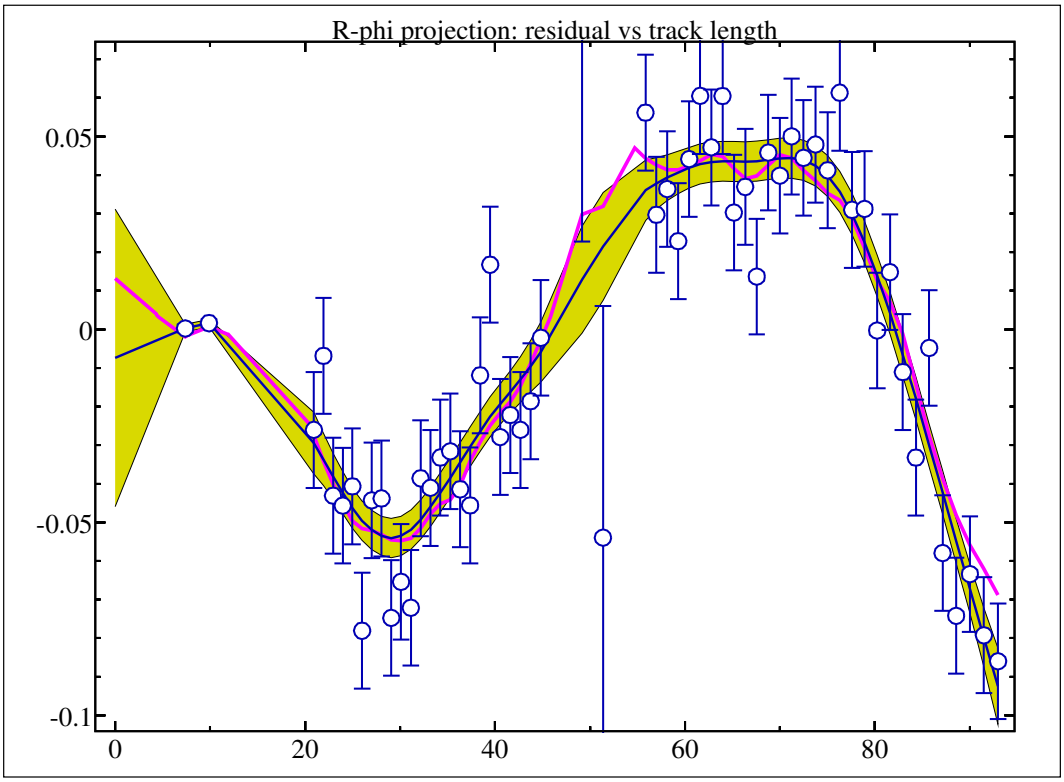

Figure 5: The track residuals of the track of Figure 1 are shown as a function of the track length. The center line within the band shows the result of the broken line fit, one standard deviation is given by the width of the band. The broken line fit is a reconstruction of the true trajectory in this Monte Carlo simulation, shown by the thin curve (compare Figures 1 and 2).

the magnetic deflection is taken into account and in this new definition mean value of the angle β_i is again zero. No further modifications are necessary in the definition of the function S (equation (8)) to be minimized. Expressed in the parameters u_i and κ the kink angle

$$\beta_i = f_i \cdot [u_{i-1}\,\delta_{i-1} - u_i\,(\delta_{i-1} + \delta_i) + u_{i+1}\,\delta_i] + (a_{i-1} + a_i)\,\kappa/2 \tag{12}$$

has to be used in the function S

$$\boxed{S\,(\boldsymbol{u}, \kappa) = \sum_{i=1}^{n} \frac{(y_i - u_i)^2}{\sigma_i^2} + \sum_{i=2}^{n-1} \frac{\beta_i^2}{\sigma_{\beta,i}^2}} \tag{13}$$

which has to be minimized with respect to the values u_i and κ.

The additional parameter curvature κ of course enlarges the matrix equation to be solved: a band matrix is bordered by one additional row and column, and the complete matrix is no longer a band matrix. However the special structure of this matrix still allows to take advantage of the band structure of the submatrix and the computing time is still short, and proportional to n. Also the band part of the inverse can be calculated and allows to compute the full covariance matrix of curvature, intercept and slope on both ends of the track.

1.3 Input data

The subroutines of this paper in the following sections all have identical input data, the coordinates of the measured data. These are

$$x_i,\ y_i,\ w_i\ (\equiv 1/\sigma_i^2) \qquad i = 1,\ 2, \ldots n \qquad \text{with } x_i < x_{i+1}$$

The coordinate x_i is the abscissa value, assumed to be known without error; the values x_i have to be in strict ascending order, i.e. $x_i = x_{i+1}$ is not allowed. The measured value at the abscissa x_i is y_i. The accuracy of the measured value is expressed by the weight w_i, defined by $w_i \equiv 1/\sigma_i^2$, where σ_i^2 is the variance. The weight may take on, for certain coordinates, the value 0 (i.e. *no* measurement). The resulting fit value is called u_i, and these values are returned by the broken-line fits. After the fit the variance of all values u_i is available.

Array(dimension)	description
`XM(N)`	abscissa values (track length)
`YM(N)`	measured function value
`WM(N)`	weight
`RM(N)`	radius
`TM(N)`	velocity (for T_0-fit, only subroutine `BRTFIT`)
`VT(7,N-1)`	multiple scattering matrices
`U(N)`	reconstructed function value

The data are usually track data: x_i is the track length up to the i-th measured point y_i, which usually is the track residual of the measured point with respect to a simple track fit with a small number of parameters, like a circle fit or a straight-line fit. The value of x_1 is usually zero. The important aspect of the broken-line fits is to take into account the multiple scattering effects in all detail. This requires the information on the multiple scattering in the array `VT(7,N-1)`. This information has to be calculated before the broken-line fit. It requires usually the knowledge of the radius r_i of the measured point with track length x_i, at least in cylindrical detectors. A special subroutine for the T_0 fit for drift chamber data requires the information on the velocity, in the array `TM(N)`.
The subroutines can of course also be used for general data which are not track data. The arguments are listed in the table above.

1.4 Robust fit methods

The fast broken-line algorithms described before take all data with their given weight. The results are optimal parameters if the weights were correctly assigned. In a track fit all hits have to belong to the track. During track finding many track fits are necessary which usally contain bad hits too: either hits from another track or random hits or hits from the correct track with a larger deviation due e.g. to δ-rays. In a standard least-squares fit the influence of outliers is large and can result in bad parameters. Especially in the track finding phase fast but robust methods, which are insensitive to outliers, but still take into account multiple scattering effects, are required.

The standard least squares fits of a straight line and a parabola to a set of data $x_i, y_i,\ i = 1,\ 2, \ldots n$ can take into account the known accuracy of individual data points. A weight $w_i = 1/\sigma_i^2$ is assigned to each data point, where σ_i is an estimate of the standard deviation of the y-measurement. In the least squares fit the squares of the scaled residuals

$$z_i^2 = w_i \left(f(x_i,\, \boldsymbol{a}) - y_i\right)^2$$

w.r.t. the parameters are considered. The parameters are determined by minimization of the sum of

the squares of the residuals

$$S(\boldsymbol{a}) = \sum_{i=1}^{n} w_i \left(f(x_i, \boldsymbol{a}) - y_i\right)^2 \ .$$

The minimum is reached if the derivatives are zero:

$$\text{solve} \ \sum_i w_i \left(f(x_i, \boldsymbol{a}) - y_i\right) \frac{\partial f}{\partial a_j} = 0 \qquad j = 1, 2 \ldots p$$

The determination of optimal coefficients for linear functions is a linear problem, which requires to solve the linear system of equations (normal equations); no iteration is necessary.

The resulting parameters of the least squares fit are quite sensitive to outliers, i.e. to data with a deviation from the fit which is much larger than expected from the assigned standard deviation or weight. In fact in least squares the influence of a data point *increases* with increasing deviation of the data point from the fit. For data points where the assigned standard deviation is realistic this property causes no problem, but outliers can introduce a systematic bias in the coefficients. This bias can be rather large in certain situations.

In the program package the following two methods are used for robust estimates, where the resulting coefficients are insensitive to some fraction of outliers:

- Least median of squares: The median of the squares of scaled residuals is approximately minimized. Two (straight line) or three data points (parabola) are randomly selected and define the straight line or parabola, and the median of the squares of scaled residuals is determined. This is done repeatedly until a value of the expected magnitude is found, or until a certain number of trials has been tested. The method can be successful even for a fraction of 50 % outliers.

- M-estimation: Data points with a large value of the square of the scaled residual are down-weighted The method is robust against a certain fraction of outliers, but not as robust as the least-median-of-squares method. Therefore down-weighting requires reasonable starting values of the coefficients and is done after a least-median-of-squares determination of coefficients. The method works in iterations and requires more computing time then a standard least squares fit.

M-estimation. The method of down-weighting is called M-estimation and is closely related to the maximum-likelihood method. For data with a probability density pdf(z) the method of maximum-likelihood requires to minimize

$$S(\boldsymbol{a}) = -\sum_{i=1}^{n} \ln \mathrm{pdf}(z_i) = \sum_{i=1}^{n} \rho(z_i)$$

with $\rho(z) = \ln \mathrm{pdf}(z)$, the derivative $\psi(z) = \mathrm{d}\rho(z)/\,\mathrm{d}z$ is called the influence function. For a Gaussian distribution the expression to be minimized is

$$S(\boldsymbol{a}) = \sum_{i=1}^{n} \frac{1}{2} z_i^2$$

and the influence function is $\psi(z) = z$. Other density functions are proposed, which close to the center are similar to the Gaussian, but have large tails. If they are used in the maximum-likelihood expression, standard least-squares normal equation can be used with an extra weight-factor $\omega(z) = \psi(z)/z$. The table below gives some examples:

	$\rho(z) = \ln \mathrm{pdf}(z)$	influence function $\psi(z) = \mathrm{d}\rho(z)/\mathrm{d}z$	weight $\omega(z) = \psi(z)/z$
Least squares	$= \frac{1}{2} z^2$	$= z$	$= 1$
Cauchy	$= \frac{c^2}{2} \ln\left(1 + (z/c)^2\right)$	$= \frac{z}{1 + (z/c)^2}$	$= \frac{1}{1 + (z/c)^2}$
Tukey $\begin{cases} \text{if } \lvert z\rvert \le c \\ \text{if } \lvert z\rvert > c \end{cases}$	$= \begin{cases} c^2/6 \left(1 - \left[1 - (z/c)^2\right]^3\right) \\ c^2/6 \end{cases}$	$= \begin{cases} z\left[1 - (z/c)^2\right]^2 \\ 0 \end{cases}$	$= \begin{cases} \left[1 - (z/c)^2\right]^2 \\ 0 \end{cases}$
Huber $\begin{cases} \text{if } \lvert z\rvert \le c \\ \text{if } \lvert z\rvert > c \end{cases}$	$= \begin{cases} z^2/2 \\ c\left(\lvert z\rvert - c/2\right) \end{cases}$	$= \begin{cases} z \\ c \cdot \mathrm{sign}\,(z) \end{cases}$	$= \begin{cases} 1 \\ c/\lvert z\rvert \end{cases}$

Asymptotic efficiency of 95 % on the normal distribution obtained with $c = 2.3849$ (Cauchy), $c = 4.6851$ (Tukey) and $c = 1.345$ (Huber). In the subroutines described the Tukey function is used, which includes a cut-off at 4.6851 standard deviations.
M-estimation introduces a non-linearity into the solution and requires iterations e.g with weights $w(z)$, calculated from previous values.

Least-median-of-squares. In this method the median (50 % value) of the squares of the scaled residuals, z_i^2, is used to measure the goodness of a certain set of coefficients. It has been shown that the median is a value which is rather robust even in case of a large fraction of outliers. Two data points in the case of a straight line and three data points in the case of a parabola are selected randomly from the data points, and are used to calculated an interpolating straight line or parabola. This is repeated several times, until the probability to have found a set of "good data points" (no outliers) is high. The resulting parameters have to be improved by e.g M-estimation.

1.5 Overview over routines and computing times

The table below gives an overview over the various fit routines. The first four routines, described in detail in section 3, are for broken-line fits, i.e. in all routines a large number of parameters is determined, one fitted value for each measured data point. All four routines require information on the multiple scattering. The remaining three routines, described in detail in section 4, fit either a straight-line or a parabola, i.e. two or three parameters, with emphasis on *robustness*.

The first two routines, `BRLFIT` and `BRCFIT`, should be used for accurate fits, and all data points are used in these fits with the given weight. The data should contain no outliers. Result is a broken-line, expressed by an array of fitted function values for each value of x_i. Result is also a parameter vector with the full covariance matrix, for the initial *track* direction and the curvature (for `BRCFIT`), and the same for the final *track* direction. Both routines do not perform iterations and are very fast.

The next routine, `BRRFIT`, performs a *robust* fit to the data. In order to be robust even in case of a large fraction of outliers, the robust routines from the second group are called to obtain reasonable starting values for the broken-line fit. This method requires iterations and the fits takes therefore more time (often by a factor around 5) than the fast broken-line fits above. No explicit fit of the curvature is done in this routine, but an estimate of the curvature is done, if required. This estimation is done from the angular deviations during the iterations and the result is a good fit also for curved tracks; however no variance is returned. The main application of this routine is to make, within the

track finding stage, a robust fit, which allows, after the fit, to distinguish between *good* data points and *outliers*, which should be removed.

A special problem for drift chamber data is the determination of the reference time, the T_0, of the given track. A wrong value causes deviations with positive and negative value, depending on the side w.r.t. the wire plane. The routine `BRTFIT` is similar to `BRRFIT`, but includes a fit of the T_0-value with determination of the variance. A different technique has to be used for the determination of the starting values of the broken-line fit.

fit	page	name	explanation
broken-line, no curvature	20	`BRLFIT`	least squares fit of broken line, assuming no curvature, with calculation of covariance matrix
		`BRLCVP`	get χ^2, variances and pulls
broken-line, curvature	22	`BRCFIT`	least squares fit of broken line, including the determination of the curvature, with calculation of covariance matrix
		`BRCCVP`	get χ^2, variances and pulls
robust broken-line	24	`BRRFIT`	robust least squares broken-line fit, optionally with estimation of curvature; initial estimate by the median-of-least-squares method
		`BRRCVP`	get variances and pulls
T_0 determination	25	`BRTFIT`	robust fit of T_0 (for drift chamber data), with estimation of curvature
		`BRTCVP`	get variances and pulls
straight line	30	`MSQLIN`	robust least squares fit of straight line, with down-weighting of outliers; initial coefficients from `ROBLMS`; includes calculation of covariance matrix
parabola	30	`MSQPAR`	robust least squares fit of parabola, with down-weighting of outliers; initial coefficients from `ROBLMS`; includes calculation of covariance matrix
line, parabola	28	`ROBLMS`	robust estimate of straight-line or parabola coefficients, based on the median-of-least-squares method; allows a large fraction of outliers

The last three routines perform a robust fit either of a straight-line and a parabola, i.e. two or three parameters are determined. The method of M-estimates is used iteratively in the two fit routines `MSQLIN` and `MSQPAR`. These routines give accurate results with covariance matrix for the parameters. They reduce the effect of outliers, but require reasonable starting values. These starting values are determined using the subtroutine `ROBLMS`, which uses the median-of least-squares method, robust against up to 50 % outliers.

The broken-line fits require a computing time proportional to the number n of data points. The full reconstruction of a curved track with subroutine `BRCFIT` for $n = 100$ data points requires 17 μsec; this is about a factor of 10 faster than the Kalman filter fit. This comparison demonstrates the advantage in using a full fit with a compact band matrix.

During track finding many track fits, often using a mixture of good data points and outliers, are

necessary and the time for those fits will dominate the total track finding time, while the final fit to good points requires only a rather small time. Thus the behaviour of track fits in the mixture case is essential. The fits using subroutine `BRRFIT` are very robust, and are of course slower than `BRCFIT`. Due to the use of fast algorithms at all stages the computing time is still only half of the Kalman filter fit time in a fit to clean data. In practice a combinatorial Kalman method is used in the mixture case, which takes a much larger time than a fit to clean data.

2 Multiple scattering

The multiple-scattering matrix array `VT(7,N-1)` contains, for each layer from the measured point `XM(I)` to the measured point `XM(I+1)`, the information on the multiple scattering. For the fit the first three elements `VT(1,I)` to `VT(3,I)` have to contain the covariance matrix elements of the left and right deflection angle in the layer: three elements V_1, V_2 and V_3 of the covariance matrix of the deflection angle:

$$\boldsymbol{V}(\theta) = \begin{pmatrix} V_L & V_{LR} \\ V_{LR} & V_R \end{pmatrix}$$

with `VT(1,I)`$\equiv V_L$, `VT(2,I)`$\equiv V_{LR}$ and `VT(3,I)`$\equiv V_R$.

2.1 Multiple scattering angles

A charged particle will undergo a Coulomb scattering collision while traversing material. The cross section for Rutherford scattering grows rapidly for decreasing scattering angle and a particle will make a large number of small angle collisions, called multiple scattering. Multiple scattering is dominated by Coulomb scattering off the nuclei, except for the lightest elements, where the scattering off electrons contributes. The mean scattering angle with respect to the incident direction is zero.
Multiple scattering is parametrized by two mutually orthogonal, uncorrelated angles, assumed to be small. The Gaussian representation with a determination of the *rms* width θ_0 of the projected angle is an approximation of the real distribution, which has larger tails. The Review of Particle Properties of the PDG quotes the formula

$$V\,[\theta] = \theta_0^2 = \left(\frac{13.6\ \text{MeV}}{\beta pc}\right)^2\ t\ \left[1 + 0.038 \ln t\right]^2 \tag{14}$$

for the variance of the deflection angle of a singly charged particle and gives the dependence on momentum p (in GeV/c) and velocity β. The quantity t is the thickness of the material in units of the radiation length X_0, thus $t = \Delta s/X_0$. The logarithmic correction term due to Highland improved the length dependence.

In applications the particle traverses different materials with a mixture of elements. As pointed out by Lynch and Dahl it is wrong to calculate θ_0 for each material and to add the separate values in quadrature[3]; the result is too small. It is better to sum over the materials before the θ_0 calculation. Assuming that βp is constant one thus has to sum the t contributions.

The momentum vector of a charged particle traversing a detector is determined from measurements of the trajectory in several detector planes. The trajectory of a charged particle traversing a *homogeneous* medium of thickness Δs between two detector planes is shown in Figure 6, which also shows the quantities used to describe the effect of multiple scattering on the particle trajectory. The effect of multiple scattering after traversal of a homogeneous medium of thickness Δs can be described by two parameters, e.g. the *deflection angle* θ_{plane}, or short θ, and the *displacement* Δy, which are statistically correlated. The displacement Δy can be expressed[4] by the angle ψ with $\Delta y \approx \psi\,\Delta s$. The covariance matrix of the two angles θ and ψ is given by

$$\boldsymbol{V}\begin{bmatrix}\theta \\ \psi\end{bmatrix} = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{pmatrix} \theta_0^2\ . \tag{15}$$

The correlation coefficient between θ and ψ is $\rho_{\theta,\psi} = \sqrt{3}/2 = 0.87$.

[3] Note that the logarithmic correction term violates the rule of additivity of the variance
[4] All angles are here assumed to be small enough to allow approximation of e.g. $\tan\psi$ by ψ.

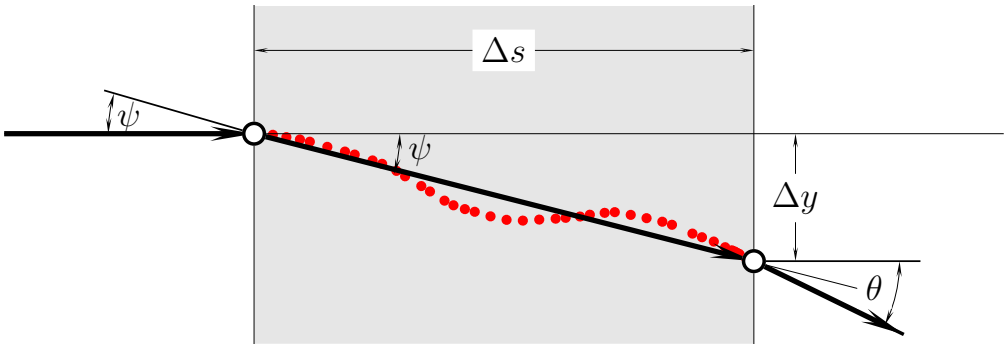

Figure 6: A charged particle incident from the left in the plane of the figure traverses a medium between two detector planes. The angle θ is the deflection angle, and the angle ψ is defined by the straight line between the two intersection points (circles).

The intersections of the detector planes by the particle trajectory are represented by the two circles in Figure 6. In an ideal detector these intersections could be measured with high precision and the straight line between the two intersections gives the complete information on the particle trajectory, which is available from the measurement. The angle between the direction of this line and the true particle direction is $\psi_{\text{left}} \equiv \psi$ on the left and $\psi_{\text{right}} \equiv \theta - \psi$ on the right of the medium. Expection and variance of these two angles are identical

$$E\begin{bmatrix}\psi_{\text{left}}\\ \psi_{\text{right}}\end{bmatrix} = \begin{pmatrix}0\\0\end{pmatrix} \qquad V\begin{bmatrix}\psi_{\text{left}}\\ \psi_{\text{right}}\end{bmatrix} = \begin{pmatrix}1/3 & 1/6\\ 1/6 & 1/3\end{pmatrix}\theta_0^2 \tag{16}$$

as is easily calculated from the covariance matrix in equation (15). This result, with a factor of $1/3$ for θ_0^2 on both sides, is valid for a homogeneous medium between the two detector planes. Multiple scattering introduces a kink on both sides even if there would be no material on the left and on the right.

Usually the material distribution between two detector planes is inhomogeneous. A thin layer of material is a special case; covariance matrices for this case are given below, for material concentrated on the left side (a), in the center (b) and on the right side (c).

$$\boldsymbol{V}\begin{bmatrix}\psi_{\text{left}}\\ \psi_{\text{right}}\end{bmatrix}_a = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\theta_0^2 \qquad \boldsymbol{V}\begin{bmatrix}\psi_{\text{left}}\\ \psi_{\text{right}}\end{bmatrix}_b = \begin{pmatrix}1/4 & 1/4\\ 1/4 & 1/4\end{pmatrix}\theta_0^2 \qquad \boldsymbol{V}\begin{bmatrix}\psi_{\text{left}}\\ \psi_{\text{right}}\end{bmatrix}_c = \begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix}\theta_0^2 \tag{17}$$

Thus the effect of the multiple scattering on the angles depends not only on the thickness in units of radiation length but also on the distribution of the material with the layer.
For the general case of an inhomogneous material distribution a layer of thickness Δs is assumed with sublayers characterized by thickness in units of radiation length $t_1, t_2, \ldots t_k \ldots$. The total thickness in units of radiation length is

$$t = \sum_k t_k \tag{18}$$

and the sum t is used to calculate the value of θ_0 from formula (14). To get the numerical factors of the covariance matrix the two sums

$$C_1 = \frac{1}{2\,t\,\Delta s}\sum_k (r_k + r_{k-1})\,t_k \tag{19}$$

$$C_2 = \frac{1}{3\,t\,(\Delta s)^2}\sum_k \left(r_k^2 + r_k r_{k-1} + r_{k-1}^2\right)t_k \tag{20}$$

have to be formed; here r_{k-1} and r_k are the coordinates of the left and right side of the sublayer with t_k. The covariance matrix is

$$\boldsymbol{V}\begin{bmatrix}\psi_{\text{left}}\\ \psi_{\text{right}}\end{bmatrix}_i = \begin{pmatrix}V_{L,i} & V_{LR,i}\\ V_{LR,i} & V_{R,i}\end{pmatrix} = \begin{pmatrix}1-2C_1+C_2 & C_1-C_2\\ C_1-C_2 & C_2\end{pmatrix}\theta_0^2\,, \tag{21}$$

where θ_0^2 is calculated from t by the formula (14).

2.2 Calculation of the multiple scattering matrix

Input data to the fits include the information of the covariance matrix $\boldsymbol{V}$ of the two angles ψ_{left} and ψ_{right} for each layer between two measured points. Subroutines described here can be used to calculate the covariance matrix elements for a track detector at a storage ring, which has a cylindrical shape in an almost homogeneous magnetic field with direction parallel to the beams.
The material is assumed to be in cylindrical layers and described in arrays `RDET(N)` and `TDET(N)`, as follows.

Variable	description
`NRT`	number of radii
`RDET(1)`	first radius
`...`	...
`RDET(I)`	radius i
`RDET(I+1)`	radius $i+1$
`...`	...
`RDET(NRT)`	last radius
`TDET(1)`	material thickness from `RDET(1)` to `RDET(2)`
`...`	...
`TDET(I)`	material thickness from `RDET(I)` to `RDET(I+1)`
`TDET(I+1)`	material thickness from `RDET(I+1)` to `RDET(I+2)`
`...`	...
`TDET(NRT-1)`	material thickness from `RDET(NRT-1)` to `RDET(NRT)`

The material value is the thickness in units of the radiation length X_0. Example: a copper layer of radius difference 0.1 cm has a thickness of 0.1 cm/1.43 cm = 0.07 (the radiation length of copper is $X_0 = 1.43$ cm); Vacuum has a thickness of zero.

The multiple scattering information for a fit of a given track has to be stored in an array `VDET(7,N)`. The elements are as follows.

Variable	description	used in fit
`VDET(1,I)`	variance V_L	yes
`VDET(2,I)`	covariance V_{LR}	
`VDET(3,I)`	variance V_R	yes
`VDET(4,I)`	thickness T, including the the logarithmic correction	
`VDET(5,I)`	geometrical factor C_L	
`VDET(6,I)`	geometrical factor C_{LR}	
`VDET(7,I)`	geometrical factor C_R	

In the fits only the variances `VDET(1,I)` and `VDET(3,I)` are used (the covariance `VDET(2,I)` is neglected).

The calculation of the array `VDET` for a given track is done in two steps:

- in the first step the material description and the measured data are used to calculate elements `J = 4, 5, 6, 7` of `VDET(J,I)`,
- in the second step the momentum and mass of the particle is given to calculate elements `J = 1, 2, 3`. This step can be repeated several times e.g. for different masses.

The first step of the calculation is done in subroutine `CVDET1`.

```
CALL CVDET1(RDET,TDET,NRT,SFAC,XM,RM,N, VDET)
```

Purpose: calculation of material and geometrical factors for the layers between measured values
`RDET :` array `RDET(NRT)` of `NRT` radii
`TDET :` array `TDET(NRT)` of `NRT` material thicknesses (the last element `TDET(NRT)` is not used).
`NRT :` number of radius values
`SFAC :` length factor
`XM :` abscissa values (track length) of the track in array `XM(N)`
`RM :` corresponding radius values of the track in array `RM(N)`
`N :` number of abscissa values of the track
`VDET :` array VDET(7,N) of material description, returned
Explanation: The radius values (array `RM`) are the perpendicular distances of the measured track points from the axis (detector axis = beam line). For each layer i the material values (array `RDET` and `TDET`) are integrated. The calculated material thickness in units of the radiation length includes the ratio `(XM(I+1)-XM(I))/(RM(I+1)-RM(I))`. Here we assume that the track length in the array `XM` are the track length values in the $r\phi$-plane. For a high-momentum track this ratio may be very close to 1. If the track length values are given in the $r\phi$-plane, then there is another factor, called `SFAC` here, which is 1 for a track perpendicular to the axis (angle $\theta = 90°$ or dip angle $\lambda = 0$). In general this factor is `SFAC` $= 1/\cos\lambda = \sqrt{1+\tan^2\lambda} = 1/\sin\theta$. A different convention is the following: track length values in the array `XM` are the true values in space; then the factor `SFAC` is 1.
The calculated thickness T in the element `VDET(4,I)` includes the logarithmic correction. The formula used is

$$T = T_u \left[1 + 0.038 \ln\left(\max(T_u, 1.0 \times 10^{-4})\right)\right]^2 ,$$

where T_u is the (uncorrected) thickness in unit of radiation length. Very small values of T_u are replaced by 1.0×10^{-4} in the logarithmic correction term; thus the correction factor in [ ] is at least 0.65.

The second step of the calculation requires the knowledge of the particle momentum and mass and is done in the subroutine `CVDET2`. Several calls with different masses are possible for different mass hypotheses.

```
CALL CVDET2(PTOT,XMAS,N, VDET)
```

Purpose: calculation of covariance matrix elements for the layers between measured values
`PTOT :` momentum value in GeV/c
`XMAS :` mass value in GeV/c^2
`N :` number of abscissa values
`VDET :` array `VDET(7,N)` of material description, updated in the first three elements at return (only elements up to `N-1)` are used)
The squared *rms* width θ_0 is calculated for each layer by

$$\theta_0^2 = \left(\frac{0.0136}{\beta pc}\right)^2 \cdot T = 0.0136^2 \, \frac{p^2 + m^2}{p^4} \cdot T$$

and this used to calculate the matrix elements:

$$\boldsymbol{V}\begin{bmatrix}\psi_{\text{left}}\\ \psi_{\text{right}}\end{bmatrix}_i = \begin{pmatrix} C_L\cdot\theta_0^2 & C_{LR}\cdot\theta_0^2 \\ C_{LR}\cdot\theta_0^2 & C_R\cdot\theta_0^2 \end{pmatrix} = \begin{pmatrix} V_{L,i} & V_{LR,i} \\ V_{LR,i} & V_{R,i} \end{pmatrix} ,$$

stored in the first three elements `VDET(1,I) ...VDET(3,I)`; the other elements of the array `VDET` are unchanged. In order to avoid problems in the fit the *rms* width θ_0 is forced to be at least 0.0001 rad.

The two routines `CVDET1` and `CVDET2` are suitable for a certain detector configuration with cylindical detectors and material distribution around the beam (and magnetic-field) axis. Corresponding routines have to be written for other detector configurations

3 Basic broken-line fits

3.1 Fits without and with curvature

A track detector at a storage ring has in general a cylindrical shape around the beam axis, in an almost homogeneous magnetic field with direction parallel to the beams. The trajectories of charged particles are approximately circles (radius R) in the plane perpendicular to the beam axis, usually called $r\phi$-plane. The determination of the curvature κ, defined by $\kappa = 1/R$, from a set of measured data points allows the calculation of the momentum. The dependence of the z-coordinate (z-axis parallel to the beam axis) on the track length s in the $r\phi$-plane has no curvature; ideally the z-dependence of the z-coordinate on the track length s is a straight line, without influence of the magnetic field.
The ideal trajectory of a charged particle is disturbed by the effects of multiple scattering, which causes many small random deflections of the particle. Subroutines for two different broken-line fits are described, one without a curvature and one with a fit of the curvature. Both require multiple scattering data (see section, array `VT`), which can only be calculated with the knowledge of the momentum. The strategy for the reconstruction of the particle trajectory is thus:

- fits of a circle in the $r\phi$-plane and of a straight line in the sz-plane, which allow a first determination of the momentum of the particle; the fits can be simple, without determination of covariance matrices;
- calculation of the multiple scattering array `VT`, e.g. by the routines described in section 2.2, using the momentum value;
- calculation of the residuals of the measured data points with respect to the fitted (idealized) trajectories (i.e. circle and straight line);
- broken-line fit of the residuals as a function of the track length, taking into account the multiple scattering effects. In the $r\phi$-plane a curvature is fitted, which is a correction to the curvature from the simple circle fit. In the sz-plane a broken-line without curvature is fitted. Values to the initial (and also final) intercept and slope are fitted, and can be used to correct the corresponding coordinates from the simple fits.

The steps with calculation of the multiple scattering array etc. can be repeated in case of a large correction of the curvature.
In the scheme explained above the broken-line fit is done to the residuals. Coordinates are called x and y below and correspond to the track length s and the residual. Each data point has assigned a weight, which in general is defined by $w = 1/\sigma^2$ for standard deviation σ. Certain weights may be zero, for example the first and the last value; then the y-value is irrelevant, but the multiple scattering effects are correctly calculated and the fitted values at the point with zero weight are calculated.

Fit, assuming no curvature. Subroutine `BRLFIT` performs a broken-line fit without curvature.

```
CALL BRLFIT(MODE,XM,YM,WM,N,VT, U,PAR,VPAR)
```

Purpose: least squares fit of broken line, assuming no curvature, with calculation of covariance matrix

`MODE :` unused argument, use `MODE=1`
`XM :` array `XM(N)` of `N` abscissa values x_i
`YM :` array `YM(N)` of `N` ordinate values y_i
`WM :` array `WM(N)` of `N` weights w_i
`N :` number of data points n

VT : array VDET(7,N) of material description (see section 2.2)
U : array U(N) of N fitted values u_i with reconstructed trajectory, returned
PAR : array PAR(2,2) with fitted intercept and slope, returned
VPAR : array VPAR(3,2) of covariance matrix elements, returned
The reconstructed trajectory is represented by the values U(I), which are the fitted values for the measured YM(I). The arrays PAR(2,2) and VPAR(3,2) contain trajectory parameters at both sides of the track:

$$\begin{array}{lll} \text{at } x = x_1: & \texttt{PAR(1,1)} = \text{intercept} & \texttt{PAR(2,1)} = \text{slope} \\ \text{at } x = x_n: & \texttt{PAR(1,2)} = \text{intercept} & \texttt{PAR(2,2)} = \text{slope} \end{array}$$

Thus the fitted trajectory is, close the point x_1, given by the linear function

$$f(x) = \texttt{PAR(1,1)} + \texttt{PAR(2,1)} \cdot (x - x_1)$$

The covariance matrix elements in VPAR(.,1) for one track end and in VPAR(.,2) for the other track end are given in the following order:

$$\boldsymbol{V} = \begin{pmatrix} 1 & \\ 2 & 3 \end{pmatrix}$$

i.e. in symmetric storage mode.
Information on the χ^2 contributions and the accuracy of the reconstructed trajectory is obtained by a subsequent call of BRLCVP.

```
CALL BRLCVP(CHSQ1,CHSQ2,UVAR,PULL)
```

Purpose: obtain χ^2 information, variances of fitted values and pulls from previous call of BRLFIT
CHSQ1 : χ^2 contribution from position measurement
CHSQ2 : χ^2 contribution from multiple scattering angles
UVAR : array UVAR(N) of N fitted variances of corresponding array U(N)
PULL : array PULL(2,N) with pulls from position measurement in PULL(1,I) and from multiple scattering angles in PULL(2,I) for i-th data point. Note that element PULL(2,N) is undefined and is set to zero.
The χ^2-contributions from position measurement and from the multiple scattering angles are returned separately. The values have to be added to obtain the total χ^2,

$$\chi^2_{\text{total}} = \chi^2_{\text{position}} + \chi^2_{\text{angles}} = \texttt{CHSQ1 + CHSQ2}$$

The number of degrees of freedoms is N-2.
The relative size of the two χ^2-contributions depends on the accuracy of the position measurement relative to the multiple scattering angles, which are determined by the amount of material and the momentum and mass of the particle. In general at high momentum multiple scattering effects are negligible and the contribution χ^2_{angles} is small; the main contribution is χ^2_{position} from the position measurement. At low momenta multiple scattering effects are large and the main χ^2-contribution is from χ^2_{angles}.
The array UVAR(N) contains at return the variances of the fitted values u_i of the reconstructed trajectory. For example the fitted value and the standard deviation of the data point with index $i = 7$ is obtained from

$$u_7 = \texttt{U(7)} \qquad \sigma_7 = \texttt{SQRT(UVAR(7))}$$

The array PULL(2,N) contains at return the pulls from position measurement and from multiple scattering angles. For example the pull values of the data point with index $i = 7$ are obtained from

$$\text{position pull: } p_{7,\text{position}} = \texttt{PULL(1,7)} \qquad \text{angle pull: } p_{7,\text{angle}} = \texttt{PULL(2,7)}$$

Pulls should follow a Gaussian distribution $N(0,1)$ with mean value of zero and a variance and standard deviation of 1. This statement should be valid for all momenta, i.e. unlike in the case of the χ^2-contributions the relation between the accuracy of the position measuremt to the multiple scattering angles should not matter. This is due to the general definition of pulls:

$$\text{pull } p = \frac{\text{measured value} - \text{fitted value}}{\sqrt{\text{variance of measured value} - \text{variance of fitted value}}} .$$

Often the variance of the fitted value is erroneously ignored in a quantity called pull; a too narrow pull distribution is expected in those cases. It is of course allowed to ignore the variance of the fitted value, but this quantity should then be called a scaled residual.

Fit including the curvature. Subroutine `BRCFIT` performs a broken-line fit including the curvature.

```
CALL BRCFIT(MODE,XM,YM,WM,N,VT, U,PAR,VPAR)
```

Purpose: least squares fit of broken line, including the curvature parameter, with calculation of covariance matrix

`MODE :` unused argument, use `MODE=1`
`XM :` array `XM(N)` of `N` abscissa values x_i
`YM :` array `YM(N)` of `N` ordinate values y_i
`WM :` array `WM(N)` of `N` weights w_i
`N :` number of data points n
`VT :` array `VDET(7,N)` of material description (see section 2.2)
`U :` array `U(N)` of `N` fitted values u_i with reconstructed trajectory, returned
`PAR :` array `PAR(3,2)` with fitted curvature, intercept and slope, returned
`VPAR :` array `VPAR(6,2)` of covariance matrix elements, returned

The reconstructed trajectory is represented by the values `U(I)`, which are the fitted values for the measured `YM(I)`. The arrays `PAR(3,2)` and `VPAR(6,2)` contain trajectory parameters at both sides of the track:

at $x = x_1$:	`PAR(1,1)`= curvature	`PAR(2,1)` = intercept	`PAR(3,1)`= slope
at $x = x_n$:	`PAR(1,2)`= curvature	`PAR(2,2)` = intercept	`PAR(3,2)`= slope

(the two curvature values `PAR(1,1)` and `PAR(1,2)` are identical). Thus the fitted trajectory is, close the point x_1, given by the quadratic function

$$f(x) = 1/2\ \texttt{PAR(1,1)} \cdot (x - x_1)^2 + \texttt{PAR(2,1)} + \texttt{PAR(3,1)} \cdot (x - x_1)$$

The covariance matrix elements in `VPAR(.,1)` for one track end and in `VPAR(.,2)` for the other track end are given in the following order:

$$\boldsymbol{V} = \begin{pmatrix} 1 & & \\ 2 & 3 & \\ 4 & 5 & 6 \end{pmatrix}$$

i.e. in symmetric storage mode.
Information on the χ^2 contributions and the accuracy of the reconstructed trajectory is obtained by a subsequent call of `BRCCVP`.

```
CALL BRCCVP(CHSQ1,CHSQ2,UVAR,PULL)
```

Purpose: obtain χ^2 information, variances of fitted values and pulls from previous call of `BRCFIT`

`CHSQ1 :` χ^2 contribution from position measurement

`CHSQ2 :` χ^2 contribution from multiple scattering angles

`UVAR :` array `UVAR(N)` of `N` fitted variances of corresponding array `U(N)`

`PULL :` array `PULL(2,N)` with pulls from position measurement in `PULL(1,I)` and from multiple scattering angles in `PULL(2,I)` for i-th data point.

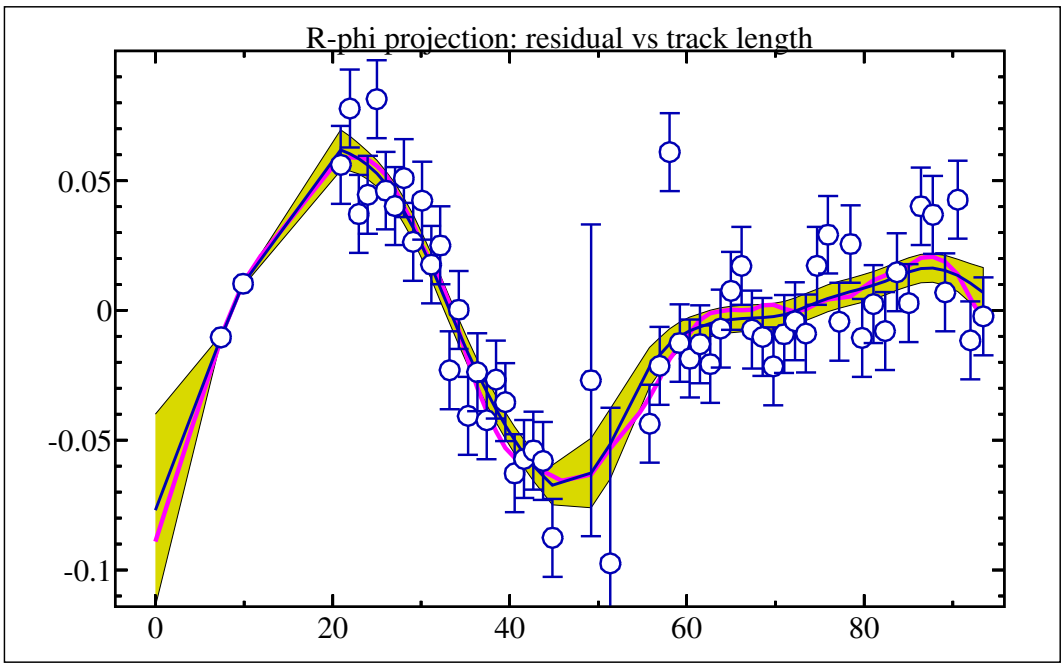

Figure 7: The track residuals of a 200 MeV/c track in a storage ring detector are shown as a function of the track length, together with a broken-line fit result. There is a $5\,\sigma$-outlier at a track length of 60 cm. The influence of this outlier on the track fit is only weak.

3.2 Robust broken-line fits

The influence of a single data point with a large deviation on the broken-line fit is only weak if the deviation is not too large. Figure 7 shows one example of data with one $5\,\sigma$-outlier. However the standard broken-line fit is sensitive to single outliers with a very large deviation and to data with a larger fraction of outliers. This is demonstrated in Figure 8 for data containing 30 % outliers, where the standard broken-line fit is compared with a robust version. A good reconstruction of the trajectory is obtained by the robust method, while the standard fit is unsatisfactory. The robust fit allows to recognize all bad data points. Those data points can be excluded from the track assignment within the track-finding phase of track reconstruction.

Colour code. Data points in the plot and also in further plots are coded by a colour code with the following meaning. *Blue* hits are within ± 1.645 standard deviations; for Gaussian data 90 % are expected in this class. *Cyan* hits have a deviation between ± 1.645 and ± 1.96 standard deviations (5 % expected for Gaussian data) and *magenta* hits have a deviation between ± 1.96 and ± 4.7 standard deviations (5 % expected for Gaussian data). All these hits still contribute to the fit, but hits with large deviations are down-weighted. *Red* hits have deviations beyond ± 4.7 standard deviations and are excluded from the fit.

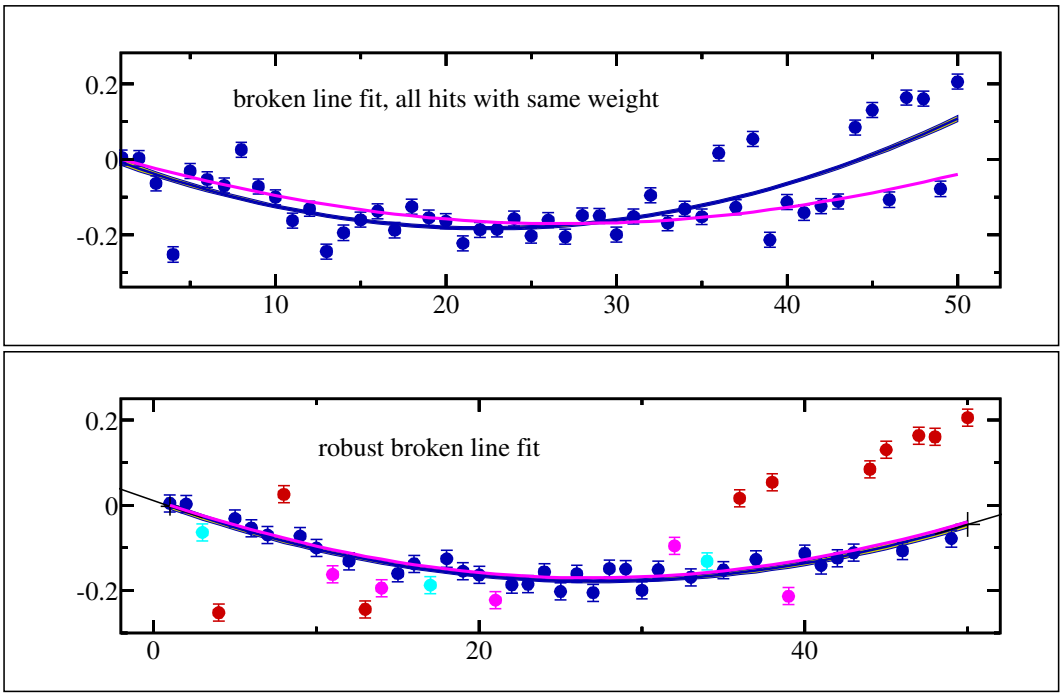

Figure 8: The upper plot shows the standard broken-line fit, where all hits have the same weight. The data points include in total 30 % outliers; 20 % are from a crossing track, 5 % are bad track hits and 5 % are random hits. The broken-line fit deviates from the true trajectory, shown by the magenta line. The outliers have a larger influence on the result. The lower plot shows the same data with a robust broken-line fit, where down-weighting using the Tukey function is applied in iterations. Starting parameters are determined by a robust parabola fit, using the method of least-median-of-squares. The fit follows closely the original trajectory, and allows to recognize outliers safely. The color code of the hits is explained on page 23.

Subroutine `BRRFIT` performs a broken-line fit with or without curvature.

```
CALL BRRFIT(LP,XM,YM,WM,N,VT, U,PAR)
```

Purpose: robust least squares fit of broken line, assuming either curvature or no curvature, with calculation of covariance matrix

`LP :` straight line (`LP=1`) or parabola (`LP=2`)
`XM :` array `XM(N)` of `N` abscissa values x_i
`YM :` array `YM(N)` of `N` ordinate values y_i
`WM :` array `WM(N)` of `N` weights w_i
`N :` number of data points n
`VT :` array `VDET(7,N)` of material description (see section 2.2)
`U :` array `U(N)` of `N` fitted values u_i with reconstructed trajectory, returned
`PAR :` array `PAR(2,3)` with fitted curvature, intercept and slope, returned returned
The reconstructed trajectory is represented by the values `U(I)`, which are the fitted values for the

measured YM(I). The array PAR(2,3) contains trajectory parameters at both sides of the track:

$$\begin{array}{lll} \text{at } x = x_1: & \texttt{PAR(1,1)} = \text{intercept} & \texttt{PAR(2,1)} = \text{slope} \\ \text{at } x = x_n: & \texttt{PAR(1,2)} = \text{intercept} & \texttt{PAR(2,2)} = \text{slope} \end{array}$$

In addition there is an estimate of the curvature:

$$\texttt{PAR(1,3)} = \text{curvature estimate} \qquad \texttt{PAR(2,3)} = \text{effective number of points}$$

Thus the fitted trajectory is, close the point x_1, given by the linear function

$$f(x) = 1/2\,\texttt{PAR(1,3)} \cdot (x - x_1)^2 + \texttt{PAR(1,1)} + \texttt{PAR(2,1)} \cdot (x - x_1)$$

Information on the covariance matrix elements, the accuracy of the reconstructed trajectory is obtained by a subsequent call of BRRCVP. Values of χ^2 are not returned.

```
CALL BRRCVP(VPAR,UVAR,PULL)
```

Purpose: obtain covariance matrix elements, variances of the fitted values and and pulls from previous call of BRRFIt

VPAR : array VPAR(3,2) of covariance matrix elements

UVAR : array UVAR(N) of N fitted variances of corresponding array U(N)

PULL : array PULL(2,N) with pulls from position measurement in PULL(1,I) and from multiple scattering angles in PULL(2,I) for i-th data point.

The covariance matrix elements in VPAR(.,1) for one track end and in VPAR(.,2) for the other track end are given in the following order:

$$\boldsymbol{V} = \begin{pmatrix} 1 & \\ 2 & 3 \end{pmatrix}$$

i.e. in symmetric storage mode.

3.3 Fit with T_0 determination

The subroutine BRTFIT is a special routine. Drift chamber data require an accurate determination of the time-zero value, T_0, for the transformation of measured time-values to coordinates. Subroutine BRTFIT performs a broken-line fit with or without curvature.

```
CALL BRTFIT(LPA,XM,YM,WM,TM,N,VT,DTZERO, U,PAR)
```

Purpose: robust least squares fit of broken line, assuming either curvature or no curvature, with calculation of covariance matrix

LP : straight line (LP=1) of parabola (LP=2)

XM : array XM(N) of N abscissa values x_i

YM : array YM(N) of N ordinate values y_i

WM : array WM(N) of N weights w_i

TM : array TM(N) of N time-zero values

N : number of data points n

VT : array VDET(7,N) of material description (see section 2.2)

DTZER : estimate of T_0 shift (input)

U : array U(N) of N fitted values u_i with reconstructed trajectory, returned

PAR : array PAR(2,4) with fitted curvature, T_0, intercept and slope, returned returned

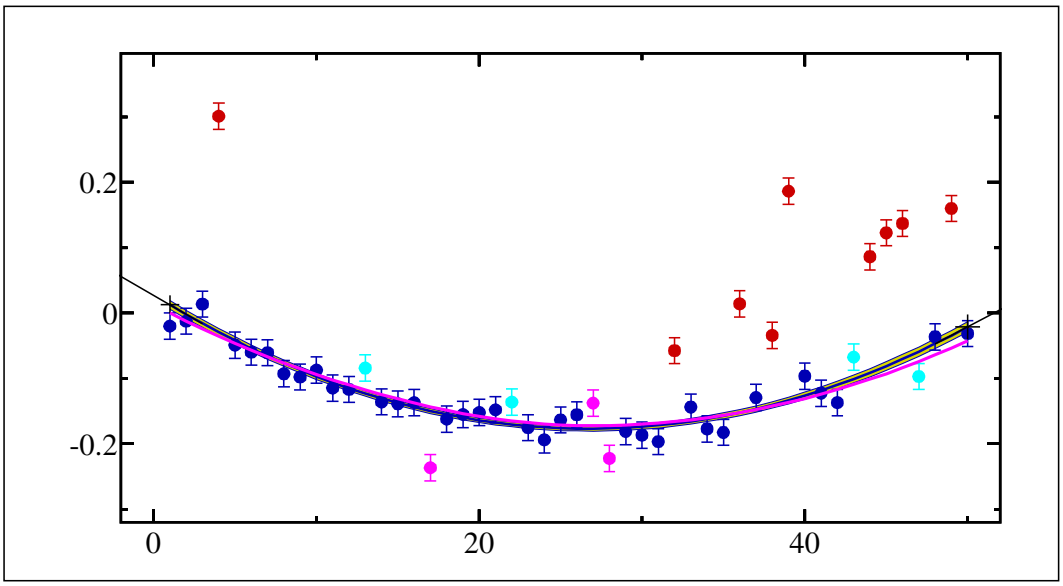

Figure 9: The data points in this plot contain 30 % outliers. The reconstruction follows the true trajectory rather close, not influenced by outliers. The color code of the hits is explained on page 23.

The array `PAR(2,4)` contains trajectory parameters at both sides of the track:

at $x = x_1$:	`PAR(1,1)` = intercept	`PAR(2,1)` = slope
at $x = x_n$:	`PAR(1,2)` = intercept	`PAR(2,2)` = slope

In addition there is an estimate of the curvature and a fitted T_0 value:

`PAR(1,3)` = curvature estimate	`PAR(2,3)` = effective number of points
`PAR(1,4)` = T_0	`PAR(2,4)` = variance of T_0

Information on the covariance matrix elements, the accuracy of the reconstructed trajectory is obtained by a subsequent call of `BRTCVP`. Values of χ^2 are not returned.

```
CALL BRTCVP(VPAR,UVAR,PULL)
```

Purpose: obtain covariance matrix elements, variances of the fitted values and and pulls from previous call of `BRTFIT`

`VPAR` : array `VPAR(3,2)` of covariance matrix elements

`UVAR` : array `UVAR(N)` of `N` fitted variances of corresponding array `U(N)`

`PULL` : array `PULL(2,N)` with pulls from position measurement in `PULL(1,I)` and from multiple scattering angles in `PULL(2,I)` for i-th data point.

The covariance matrix elements in `VPAR(.,1)` for one track end and in `VPAR(.,2)` for the other track end are given in the following order:

$$\boldsymbol{V} = \begin{pmatrix} 1 & \\ 2 & 3 \end{pmatrix}$$

i.e. in symmetric storage mode.

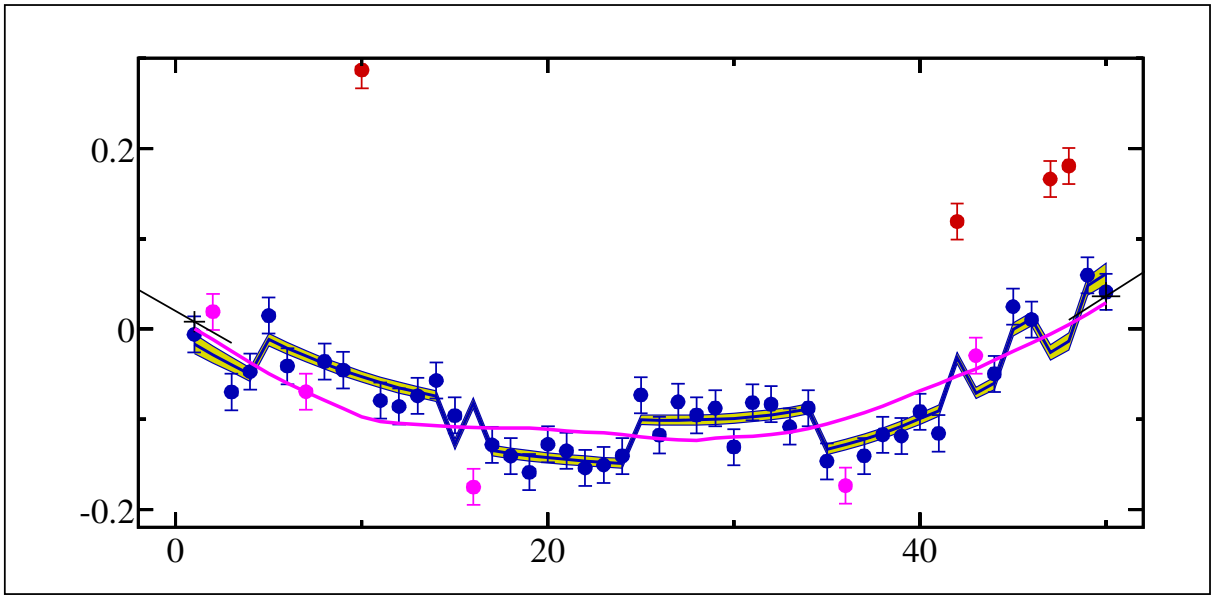

Figure 10: Data points with a fraction of 20 % outliers are shown, together with the true trajectory (magenta line). The shown residuals are calculated with an inaccurate value of T_0, which creates larger positive or negative deviations, depending on the position of the hits relative to the wire plane of a drift chamber. The robust broken-line fits includes a T_0-correction and is able to fit a precise value of T_0 and to reconstruct the true trajectory. The color code of the hits is explained on page 23.

4 Robust straight-line and parabola fit

4.1 Robust parameter estimation

The *least-median-of-squares* method is used to get robust estimates of the parameters of a straight line or parabola. The subroutine `ROBLMS` determines the parameters a_1 and a_2 of a straight line

$$f(x,\, a_1, a_2) = a_1 + a_2(x - x_1)$$

or the parameters a_1, a_2 and a_3 of a parabola

$$f(x,\, a_1, a_2, a_3) = a_1 + a_2(x - x_1) + a_3(x - x_1)^2$$

from 2 and 3 points, respectively, which give the least-median-of-squares of residuals. The residuals defined here include the accuracy of each data point, which is given by the standard deviation σ_i or the weight $w_i = 1/\sigma_i^2$. The square of the scaled residuals is defined by

$$z_i^2 = \left(\frac{y_i - f(x_i; a)}{\sigma_i}\right)^2 = w_i \left(y_i - f(x_i; a)\right)^2 \ . \tag{22}$$

For each parametrization the squared residuals are calculated and the median (the 50 % value) is determined. A fast median algorithm called *quick-select*, derived from the *quicksort* algorithm, is used in-line. The parametrization with the smallest median is kept. The parameter values returned by `ROBLMS` are, as can be seen from the above formulas, defined with reference to the first coordinate x_1. The coefficients are only *rough* estimates. This subroutine is called internally in other subroutines, for the determination of starting parameters, in a more precise parameter determination.

The results should be acceptable even in case of a large fraction of outliers, theoretically up to 50 %. In practice the algorithm should be rather safe (high probability to find *good* points) for outlier fractions of 30 % to 40 %. Figure 11 shows one example of robust fits, where the outlier fraction is 40 %, with a tendency to one side.

Up to 48 trials and at least 6 trials are made. The first 6 trials are systematic trials using the first and last three points. Usually a good result is obtained already within the first 6 trials, if the outlier fraction is only a few %. All following trials are done with randomly selected data points. If the total number n the data points is small, a combinatorial algorithm is used to select data points for the parametrization.

The programs returns with the best parameters as soon as a small value S of the median-of-squares is reached. Without outliers the squared residuals should follow χ^2-distribution with 1 degree of freedom (for gaussian data). If the data contain no outliers, a value of $S = 0.455$ is expected from the χ^2-distribution (50 % p-value) with 1 degree of freedom. Some expected values from the χ^2_1-distribution for different p-value are given below:

p-value	50 %	75 %	90 %	95 %	99 %
χ^2_1	0.455	1.32	2.71	3.84	6.63

If the outlier fraction is as high as 44.5 %, then the median S corresponds to about 90 % of the good data, and the expected value is 3.84. The general relations between the outlier fraction f and the p-value are $f = 1 - 0.5/p$ and $p = 0.5/(1-f)$.
An estimate of the probability to select in m trials 2 and 3 *good* points, respectively, is given by the formulas

$$\text{straight-line: } 1 - \left(1 - (1-f)^2\right)^m \qquad \text{parabola: } 1 - \left(1 - (1-f)^3\right)^m$$

The following tables gives the necessary number of trials in order to obtain a *good* sample with a probability of at least 95 %:

fraction $f =$	5 %	10 %	20 %	25 %	30 %	40 %	50 %
straight-line	2	2	3	4	5	7	11
parabola	2	3	5	6	8	13	23

If the fraction f of outliers is small, the probability to find *good* points is high and the expected value of S is small. For increasing fraction f of outliers the required number of trials increases fast and the expected value of S becomes larger. The cut-value for S to stop the search for *good* points increases with the number m of trials:

$$S < 0.5 \left[\frac{m+8}{4}\right] .$$

Thus larger values of S are accepted at return if the number m of trials is already larger.

```
CALL ROBLMS(LP,XM,YM,WM,N, PAR,SMEDIA)
```

Purpose: robust fit of straight line (`LP=1`) or of parabola (`LP=2`) using the *least-median-of-squares* method

`LP :` straight line is selected by `LP=1` and parabola is selected by `LP=2`

`XM :` array `XM(N)` of `N` abscissa values x_i

`YM :` array `YM(N)` of `N` ordinate values y_i

`WM :` array `WM(N)` of `N` weights w_i

`N :` number of data points n

`PAR :` array `PAR(3)` of polynomial coefficients a_1, a_2 and a_3, with $a_3 = 0$ in case of `LP=1`; returned. (see below)

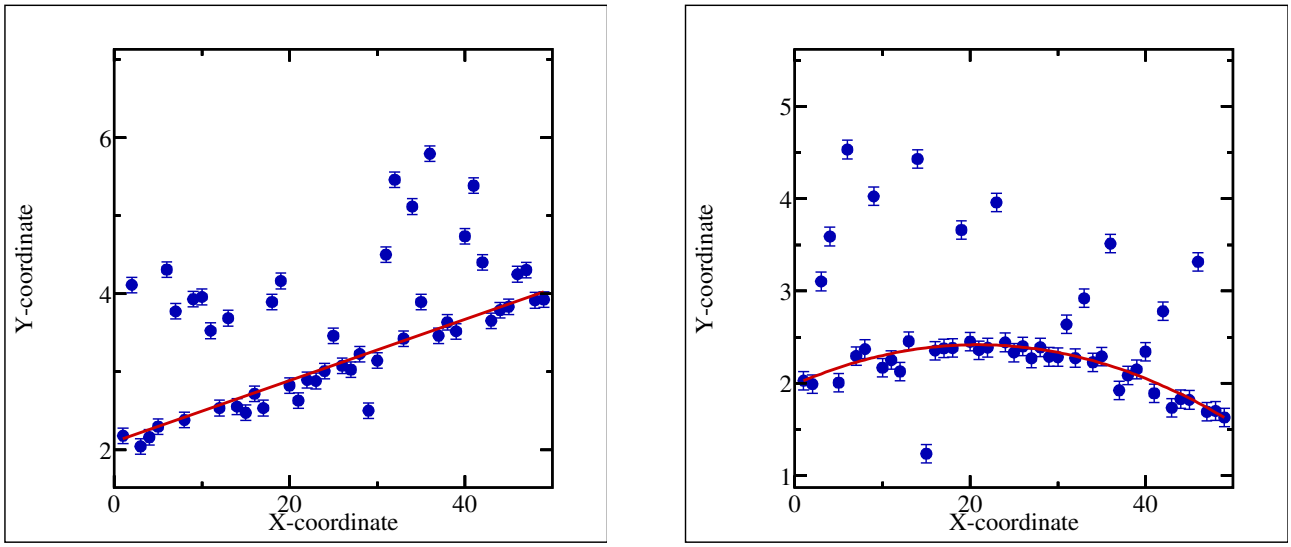

Figure 11: Examples for the determination of a straight line (left) and of a parabola (right) by the subroutine ROBLMS. The data contain 40 % outliers, most of them with y-value larger than the correct data.

SMEDIA: median of squares, returned.
The returned value of SMEDIA is the median of the squared scaled residuals, i.e. $\approx 50\%$ of the data points have a value of the squared residual smaller then SMEDIA.

4.2 Straight-line and parabola fit

The routines described in this section perform a robust fit of a straight line and a parabola, with iterative down-weighting of outliers using the weight function proposed by Tukey. The down-weighting follows the idea of M-estimates; the method of M-estimates is closely related to the maximum likelihood method. Instead of assuming the Gaussian distribution for the fluctuations about the mean, a different distribution with larger tails is assumed, to reduce or remove the influence of outliers. Down-weighting requires acceptable initial parameter values, and is, for each data point (x_i, y_i, w_i), based on the value of the scaled residual z_i defined in equation (22). A further weight ω_i (≤ 1) for the least-squares fit is calculated from the value of the scaled residual z_i. Thus the weight used in the least squares fit is $\omega_i \cdot w_i$; because of the down-weighting the linear least squares fit becomes non-linear and requires iterations.

Several down-weighting functions are proposed in the literature. Here the function proposed by Tukey is applied, which excludes data points with deviations larger than 4.681 standard deviations. The method requires reasonable starting values of the coefficients; in the subroutines described below the starting values are determined by the least-median-of-squares method.

The following steps are performed in both subroutines:

- estimation of parameter start values by calling ROBLMS, which also supplies the value SMEDIA, which is the median of the squared residuals;
- a standard least squares fit is performed, using only the data with a squared normalised residual value $z_i^2 \leq$ SMEDIA; thus about 50 % of the data points are used in the fit;

- up to 9 further iterations are performed with a least squares fit, where data points are down-weighted using the Tukey-formula

$$\omega(z_i) = \begin{cases} \left[1 - (z_i/c)^2\right]^2 & \text{if } |z_i| \le c \\ 0 & \text{if } |z_i| > c \end{cases} .$$

  The constant is $c = 4.681$. With this value the asymptotic efficiency of the fit is at least 95 % for Gaussian-distributed data. Data points outside 4.681 standard deviations are excluded.

- The iteration ends after a total 10 iterations or if the estimated change $\delta\chi^2$ is small compared to 1.

The parameter values returned are more precise than the rough estimates provided by `ROBLMS`, in addition a covariance matrix of the parameters in symmetric storage mode is returned. Examples for the robust least squares fits are shown in Figure 12.

```
CALL MSQLIN(XM,YM,WM,N, PAR)
```

Purpose: robust least squares straight line fit with down-weighting of outliers
`XM :` array `XM(N)` of `N` abscissa values x_i
`YM :` array `YM(N)` of `N` ordinate values y_i
`WM :` array `WM(N)` of `N` weights w_i
`N :` number of data points n
`PAR :` resulting parameter array `PAR(5)`, with coefficients and matrix elements a_j, see below.
The parametrization of the straight line and the covariance matrix elements are:

$$f(x) = a_1 + a_2\,(x - x_1) \qquad \boldsymbol{V}\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} a_3 & \\ a_4 & a_5 \end{pmatrix}$$

```
CALL MSQPAR(XM,YM,WM,N, PAR)
```

Purpose: robust least squares parabola fit with down-weighting of outliers
`XM :` array `XM(N)` of `N` abscissa values x_i
`YM :` array `YM(N)` of `N` ordinate values y_i
`WM :` array `WM(N)` of `N` weights w_i
`N :` number of data points n
`PAR :` resulting parameter array `PAR(9)` with coefficients and matrix elements a_j, see below.
The parametrization of the parabola and the covariance matrix elements are:

$$f(x) = a_1 + a_2\,(x - x_1) + a_3\,(x - x_1)^2 \qquad \boldsymbol{V}\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} a_4 & & \\ a_5 & a_6 & \\ a_7 & a_8 & a_9 \end{pmatrix} \tag{23}$$

Note on the approximation of a small circle section by a parabola: Under a certain condition the fit of a parabola can be used instead of a circle fit for a section of a circle, although the curvature of a parabolic curve is not a constant along the curve. The general formula for the curvature of a curve described by a parametrization $f(x)$ is given by

$$\kappa(x) = \frac{\mathrm{d}^2 f(x)/\,\mathrm{d}x^2}{\left[1 + (\,\mathrm{d}f(x)/\,\mathrm{d}x)^2\right]^{3/2}} .$$

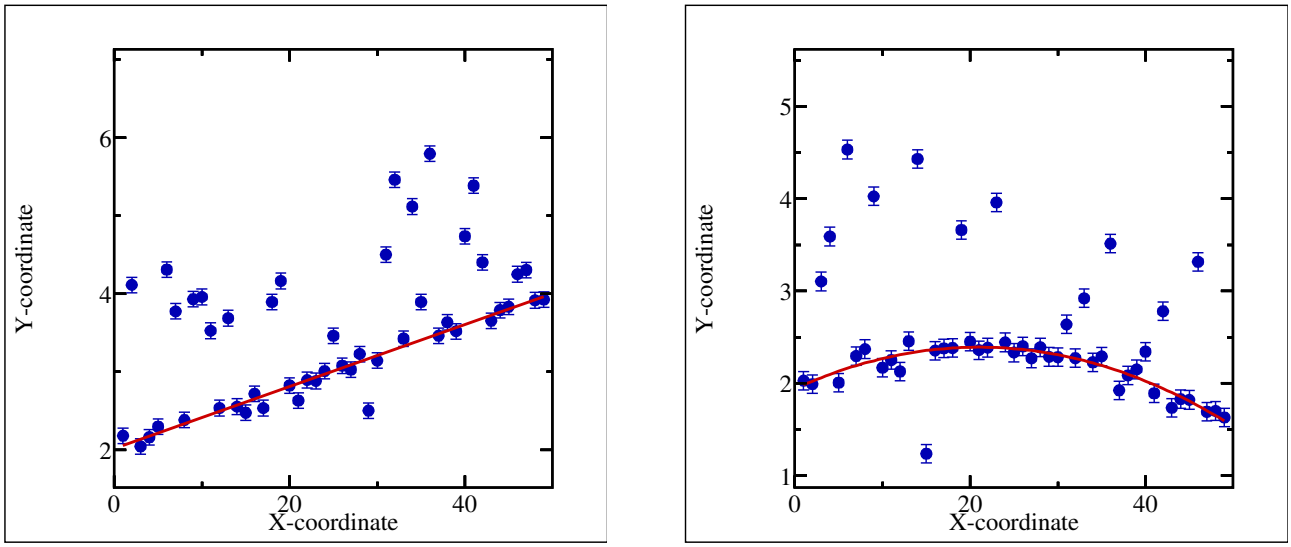

Figure 12: Examples for the determination of a straight line (left) by the subroutine `MSQLIN` and and of a parabola (right) by the subroutine `MSQPAR`. The data are identical to the data of figures 11; they contain 40 % outliers, most of them with y-value larger than the correct data. The difference of the fits to the result by `ROBLMS` is rather small in this example.

For the parabola of equation (23) the curvature, as a function of x, is given by

$$\kappa(x) = \frac{2a_3}{\left[1 + (a_2 + 2a_3 x)^2\right]^{3/2}}$$

The curvature is almost constant (as it is for a circle) under the condition $(a_2 + 2a_3 x)^2 \ll 1$. The robust broken-line fit in subroutine `BRRFIT` is done by first fitting a parabola with subroutine `ROBLMS`, even if the condition is not accurately valid. The small number of parameters of a parabola however allows a robust parameter estimation and subsequent fits with improved parametrization can initially use the outlier recognition by `ROBLMS`.

5 Circle fits and utilities

The full track fit requires an approximate circle fit. The parameters of the circle fit allow a first determination of the particle momentum and thus a calculation of the multiple scattering matrix (section 2.2).
The circle parametrization in the $r\phi$-plane was already given in the introduction. The parametrization has three variables: the curvature $\kappa = \pm 1/R$ (R = radius of curvature), the direction ϕ (direction of propagation at the point of closest approach to the reference point), and the distance of closest approach d_{ca} (impact parameter). The circle equation (24) is repeated here. The equation for the residual ε_i of a point x_i, y_i (in a xyz-coordinate system) in the $r\phi$- or xy-plane from a circle is given by

$$\varepsilon_i = \frac{1}{2}\kappa \left(x_i^2 + y_i^2 + d_{\mathrm{ca}}^2\right) - (1 + \kappa d_{\mathrm{ca}}) \left(x_i \sin\phi - y_i \cos\phi\right) + \frac{1}{2}\kappa + d_{\mathrm{ca}} \; . \tag{24}$$

The equation is valid over a full trajectory loop, and also in the straight-line limit $\kappa \to 0$. The parametrization of the trajectory provides a continuous transition between negative and positive curvature as well as between negative and positive distance of closest approach. A point with coordinates x_i and y_i can also be described by radius r_i and angle φ_i with the relation

$$x_i = r_i \sin\varphi_i \qquad\qquad y_i = r_i \cos\varphi_i \; .$$

The range of possible radii r_i is given by

$$|d_{\mathrm{ca}}| < r_i < \left|\frac{2}{\kappa} - d_{\mathrm{ca}}\right| \tag{25}$$

The smallest radius corresponds to the point of closest approach

$$x_{d_{\mathrm{ca}}} = +d_{\mathrm{ca}} \sin(\phi) \qquad\qquad y_{d_{\mathrm{ca}}} = -d_{\mathrm{ca}} \cos(\phi) \; .$$

The three circle parameters are available in single and in double precision. All other arguments are in single precision, but internal calculations are usually done in double precision.

5.1 Circle fits

Two routines for weighted circle fits are described. Input are single-precision coordinates, but all internal computation is in double precision. The parameters can be obtained in single precision (array `TR(3)`) and in double precision (array `TRD(3)`).

Array(dimension)	description
`X(N)`	x-coordinates
`Y(N)`	y-coordinates
`W(N)`	weights = inverse variance
`TR(3)`	circle parameters in single precision
	`TR(1)` = κ, curvature
	`TR(2)` = d_{ca}, distance of closest approach
	`TR(3)` = ϕ, angle of direction at d_{ca}
`TRD(3)`	circle parameters in double precision

The elements of the double-precision array `TRD(3)` have identical meaning. No covariance matrix and no χ^2 is returned, because it is assumed that a broken line fit with complete error treatment follows.

The circle fit uses the method of Veikko Karimäki[5]. However small parameter corrections given by Karimäki, which require certain additional computation (and cpu time), are neglected in this version. The fit method is non-iterative and independent of the orientation of the circle, and allows circle fits over the full circle, independent of the order of the coordinates.
Internally the circle fit is done in a shifted coordinate system in order to reduce certain round-off errors, which otherwise could reduce the precision for short tracks, which are far from the origin of the coordinate system.
In the second subroutine the circle is fitted exactly through a fixed reference point x_0, y_0 (for example primary vertex), specified without error. In a shifted xy coordinate system centered at the reference point the distance $d_{\rm ca}$ of closest approach is zero. However the returned circle parameters for both circle fit routines refer to the detector system, the system of the x, y-coordinates. In certain cases (e.g. within a track finding procedure) short track segments far from the origin of the coordinate system have to be fitted. A normal 3-parameter fit may give a rather inaccurate extrapolation to the origin, even if the segment is a part of a track from the origin. In those cases a fit with reference point $0, 0$ can improve e.g. the determination of the curvature of the track.

5.1.1 General circle fit

```
CALL SCIRFT(X,Y,W,N, TR)
```

Purpose: circle fit throught data points
`X :` array `X(N)` of `N` abscissa values x_i
`Y :` array `Y(N)` of `N` ordinate values y_i
`W :` array `W(N)` of `N` weights w_i
`N :` number of data points n
`TR :` resulting parameter array `TR(3)` in single precision

```
CALL DCIRFT(TRD)
```

Purpose: get circle parameters in double precision; to be called after the `SCIRFT(X,Y,W, TR)` call.
`TRD :` resulting parameter array `TRD(3)` in double precision

5.1.2 Circle fit exactly through given point

```
CALL SCIRXY(X,Y,W,N,XREF,YREF, TR)
```

Purpose: circle fit throught data points, exactly through the given reference point `(XREF,YREF)`.
`X :` array `X(N)` of `N` abscissa values x_i
`Y :` array `Y(N)` of `N` ordinate values y_i
`W :` array `W(N)` of `N` weights w_i
`N :` number of data points n
`XREF :` x coordinate of reference point
`YREF :` y coordinate of reference point
`TR :` resulting parameter array `TR(3)` in single precision

[5]Veikko Karimäki, Effective circle fitting for particle trajectories, NIM **A 205**, 187 - 191, 1991. Veikko Karimäki, Fast code to fit circular arcs, Computer Physics Communications **69**, 133 - 141, 1992.

```
CALL DCIRXY(TRD)
```

Purpose: get circle parameters in double precision; to be called after the `SCIRXY(X,Y,W, TR)` call.
`TRD :` resulting parameter array `TRD(3)` in double precision

5.1.3 Move circle parametrization

The current circle parametrization is transformed to a new reference point. These subroutines are called internally in the fit routines.

```
CALL SCIRMV(XNEW,YNEW,TR)
```

Purpose: move the parametrization to a new reference point
`XNEW :` x-coordinate of new reference point
`YNEW :` y-coordinate of new reference point
`TR :` circle parameter array `TR(3)` in single precision, returned with modified second and third element for new reference

```
CALL DCIRMV(XNEW,YNEW,TRD)
```

Purpose: move the parametrization to a new reference point
`XNEW :` x-coordinate of new reference point
`YNEW :` y-coordinate of new reference point
`TRD :` circle parameter array `TRD(3)` in double precision, returned with modified second and third element for new reference.

5.2 Function calls for circle quantities

A single value of a quantity derived from the circle parameters is returned by the following function calls. The first call for a given circle is `FUNPAR`.

```
PHI=FUNPAR(TR,RD)
```

Purpose: set the circle parameters in single precision for this and the following function calls. For radius `RD` $\neq 0$ the φ-angle of the point at the given radius on the circle is returned, otherwise 0 is returned.
`TR :` array `TR(3)` in single precision for this call and the other function calls.
`RD :` radius value for φ calculation. For a value of `RD=0` the value `PHI=0` is returned
In the following calls there is no argument `TR`, and the circle parameters given previously in `FUNPAR` are used.

```
PHI=FUNDPA(TRD,RD)
```

Purpose: set the circle parameters in double precision for this and the following function calls. For radius `RD` $\neq 0$ the φ-angle of the point at the given radius on the circle is returned, otherwise 0 is returned.
`TR :` array `TR(3)` in single precision for this call and the other function calls.
`RD :` radius value for φ calculation. For a value of `RD=0` the value `PHI=0` is returned
In the following calls there is no argument `TR`, and the circle parameters given previously in `FUNPAR`

are used

```
PHI=FUNPHI(RD)
```

Purpose: The φ-angle of the point at the given radius on the circle is returned; a value of 0 is returned, if the circle is not reached for the given radius.

`RD :` radius value for φ calculation.

```
S=FUNLEN(RD)
```

Purpose: The curve length from the d_{ca}-point is returned. The value 0 is returned if the circle is not reached for the given radius. Not that there is an ambiguity for the curvelength, which in principle could be negative, for a point *before* the d_{ca}-point. This function always returns a non-negative value (use the subroutine `CIRLRS` for the general case).

`RD :` radius value for the curve length calculation.

```
FAC=FUNRXY(X,Y,UX,UY)
```

Purpose: The factor `FAC` for the given point `(X,Y)` and the given vector `(UX,UY)` is returned. Eventually a value `FAC = 0` is returned. Internally a circle point `(XC,YC)` is calculated, which is the point on the circle closest to the given point `(X,Y)`, and the factor `FAC` is defined by the equations:

```
XC = X + FAC * UX          YC = Y + FAC * UY
```

`X :` x-coordinate of a data point
`Y :` y-coordinate of a data point
`UX :` x-component of a vector
`UY :` y-component of a vector

5.3 Circle utilities

5.3.1 Set circle parameters

The circle parameters have to be set before any of the following entries can be used. Either single or double precision circle parameters can be set. Internally double precision is used.

```
CALL SCIRTR(TR)
```

Purpose: set the circle parameters in single precision.

`TR :` circle parameter array `TR(3)` in single precision

```
CALL DCIRTR(TRD)
```

Purpose: set the circle parameters in double precision.

`TRD :` circle parameter array `TRD(3)` in double precision

5.3.2 Transformation to curve length and residuals

Entries described below allow to calculate the curve length s for a given xy point. The default reference point, which corresponds to curve length $s = 0$, after the calls of `RCIRCL` or `RCIRCL` for the curve length s is the point of closest approach, with coordinates

$$x_{d_{\rm ca}} = +d_{\rm ca}\,\sin(\phi) \qquad y_{d_{\rm ca}} = -d_{\rm ca}\,\cos(\phi)\ .$$

A different reference point is set by the following call of `CIRREF`.

```
CALL CIRREF(XREF,YREF)
```

Purpose: set the reference point for the following calculations of the curve length s; if the reference point is not on the circle, the nearest circle point is calculated. The call is ignored, if the radius of the reference point is too large.
`XREF :` x-coordinate of reference point
`YREF :` y-coordinate of reference point

The x_i- and y_i-coordinates of n data points are transformed to the curve length s_i and the orthogonal distance r_i (vertical distance of point (x_i, y_i) from the track). This calculation includes an internal calculation of a point on the circle, which has the smallest distance to the measured point (x_i, y_i).

```
CALL CIRLRS(N,X,Y, S,R)
```

Purpose: transform array of data points to curve length and orthogonal distance
`X :` array `X(N)` of `N` abscissa values x_i
`Y :` array `Y(N)` of `N` ordinate values y_i
`N :` number of data points n
`S :` array `S(N)` of `N` curve length values s_i; returned.
`R :` array `R(N)` of `N` orthogonal distance values r_i; returned.
The elements `S(I)` and `R(I)` are set to zero, if the point `(X(I),Y(I))` is at a too small or too large radius (the limits are given by equation (25)).

5.3.3 Calculation of the circle point and the tangent vector

```
CALL CIRTAN(XS,YS, XC,YC,TX,TY)
```

Purpose: calculated for the given data point `(XS,YS)` the circle point `(XC,YC)` and the unit vector in tangential direction
`XS :` x coordinate of a data point
`YS :` y coordinate of a data point
`XC :` x coordinate of the nearest circle point with the smallest distance to `(XS,YS)`; returned.
`YC :` y coordinate of the nearest circle point with the smallest distance to `(XS,YS)`; returned.
`TX :` x component of the unit vector in tangential direction at the circle point `(XC,YC)`; returned.
`TY :` y component of the unit vector in tangential direction at the circle point `(XC,YC)`; returned.
The arguments `XC, ... TY` are set to zero for a data point `(XS,YS)` outside the radius limits of equation (25).

6 Band matrices

In this section the fast solution of matrix equations with a symmetric band matrix using the Cholesky method is described. The Cholesky method allows to make efficient use of the band structure to reduce the number of operations. The method can also be used for general symmetric matrices and and those matrices are included in this section.

6.1 Band matrices and band matrix equations

Problems treated by the least squares method require the solution of a system of linear equations, the so-called normal equations of least squares, which are of the form

$$\boldsymbol{W}\boldsymbol{x} = \boldsymbol{y} \tag{26}$$

with a symmetric n-by-n matrix $\boldsymbol{W}$ and a known right-hand-side n-vector $\boldsymbol{y}$. Formally the solution for $\boldsymbol{x}$ can be expressed with the inverse matrix:

$$\boldsymbol{x} = \boldsymbol{W}^{-1}\boldsymbol{y}\ .$$

The inverse matrix $\boldsymbol{W}^{-1}$ is the covariance matrix of the parameter vector of the least squares problem; the diagonal elements are the variances of the elements of the solution vector. Matrices $\boldsymbol{W}$ of least squares are positive definite (positive eigenvalues). All diagonal elements of $\boldsymbol{W}$ and of $\boldsymbol{W}^{-1}$ are positive.
Several different methods exist for the solution of the matrix equation. The standard Gauss algorithm, which can make use of the symmetry of the matrix, takes a number of operations $1/2n^2$, where n is the dimension parameter of matrix and vector, if the solution and the inverse are calculated. The number of operations can be reduced to $1/3n^2$, if the inverse matrix is not needed.
Another method, applicable for positive definite matrices, is the solution by Cholesky decomposition. In a Cholesky decomposition the matrix is decomposed according to $\boldsymbol{C} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ where the matrix $\boldsymbol{L}$ is a left triangular matrix; all elements above the diagonal vanish. The Cholesky decomposition is possible for a positive-definite matrix and sometimes called the *square root* of a matrix. It requires the n (real) square roots. After decomposition the solution is the matrix equation is done in two steps, with an auxiliary vector $\boldsymbol{z}$.
There is a variant of the Cholesky decomposition with introduction of a diagonal matrix $\boldsymbol{D}$, which avoids the square-root operations. The decomposition is

$$\boldsymbol{W} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}} \qquad \text{decomposition} \tag{27}$$

where the matrix $\boldsymbol{L}$ is a left unit triangular matrix, which has values 1 on the diagonal.
The decomposition allows to rewrite the matrix equation (26) in the form

$$\boldsymbol{L}\left(\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}\right) = \boldsymbol{y}$$

and now the matrix equation can be solved with $\boldsymbol{z} = \boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}$ in two steps:

$$\text{solve} \quad \boldsymbol{L}\boldsymbol{v} = \boldsymbol{r} \qquad \text{for } \boldsymbol{v} \text{ by forward substitution, and} \tag{28}$$

$$\text{solve} \quad \boldsymbol{L}^{\mathrm{T}}\boldsymbol{u} = \boldsymbol{D}^{-1}\boldsymbol{v} \qquad \text{for } \boldsymbol{u} \text{ by backward substitution.} \tag{29}$$

Substitutions are straightforward because $\boldsymbol{L}$ is triangular. The number of operation is similar to the Gauss algorithm.
The symmetric n-by-n matrix $\boldsymbol{W}$ in the fit algorithms has a band structure, where all elements outside a band along the diagonal vanish. A band matrix is characterized by the *bandwidth* m, with $W_{ik} = 0$ for all i and k with $|i-k| \geq m$. There is a band of $(2m+1)$ elements in the diagonal.

The band matrix which appears in the fit algorithms has a band width $M = 2$, with five adjacent diagonals. The matrix equation is thus of the form

$$\boldsymbol{W}\boldsymbol{x} = \boldsymbol{y} \qquad \text{or} \qquad \begin{pmatrix} W_{11} & W_{12} & W_{13} & & & & \\ W_{21} & W_{22} & W_{23} & W_{24} & & & \\ W_{31} & W_{32} & W_{33} & W_{34} & W_{35} & & \\ & W_{42} & W_{43} & W_{44} & W_{45} & W_{46} & \\ & & W_{53} & W_{54} & W_{55} & W_{57} & W_{68} \\ & & & & & \ddots & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ \vdots \end{pmatrix}$$

In the case above the band width is $m = 2$, and because of the symmetry an 3-by-n array is sufficient to store the band matrix. One possible assignment of band matrix elements to array elements, used in the subroutines described below, is

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{22} & W_{23} & W_{24} \\ W_{33} & W_{34} & W_{35} \\ \dots & \dots & \dots \\ W_{i-2,i-2} & W_{i-2,i-1} & W_{i-2,i} \\ \boldsymbol{W}_{i-1,i-1} & \boldsymbol{W}_{i-1,i} & \boldsymbol{W}_{i-1,i+1} \\ \boxed{\boldsymbol{W}_{i,i}} & \boldsymbol{W}_{i,i+1} & W_{i,i+2} \\ \boldsymbol{W}_{i+1,i+1} & W_{i+1,i+2} & W_{i+1,i+3} \\ W_{i+2,i+2} & W_{i+2,i+3} & W_{i+2,i+4} \\ \dots & \dots & \dots \\ W_{n-2,n-2} & W_{n-2,n-1} & W_{n-2,n} \\ W_{n-1,n-1} & W_{n-1,n} & - \\ W_{n,n} & - & - \end{bmatrix} \tag{30}$$

Three array elements at the end (indicated by $-$) are unused (and never touched in band matrix operations). In an inversion of a band matrix the band structure is lost, and the inverse of a band matrix is a full matrix. The time for a standard inversion is essentially not reduced for a band matrix. However the band structure is kept in both forms of the Cholesky decomposition. The triangular matrix $\boldsymbol{L}$ has the band structure with non-zero elements only in $(m + 1)$ diagonals. Much faster solution methods are possible for band matrix with decomposition techniques.
The advantage of the Cholesky decomposition for a band matrix, is that the band structure is kept. The number of operations contains, instead of n^3 for a full matrix, only $m^2\, n$, and thus the number of operations is, for fixed band width, proportional to n. This linear dependence gives a much faster execution.
The diagonal elements of matrix $\boldsymbol{L}$ are all 1 and these values need not be stored; this property allows to store the decomposition with matrices $\boldsymbol{L}$ and $\boldsymbol{D}$ in the same array as $\boldsymbol{C}$.
Making use of the band structure reduces the calculation of the solution to a work $\propto n$ instead of $\propto n^3$, without forming the full inverse matrix. An overview over the number of instructions in several band matrix operations is given in table 1.

Inverse matrix. The inverse matrix $\boldsymbol{C}^{-1}$ corresponds to the covariance matrix of the parameters u_i. For quick calculations e.g. within track recognition this matrix will not be needed. If the inverse matrix or selected elements of the inverse matrix are needed, they can be calculated by different methods. One method is to determine selected columns (rows) of the inverse matrix by the solution of matrix equation e.g.

$$\boldsymbol{C}\boldsymbol{v}_k = \boldsymbol{e}_k \tag{31}$$

matrix operation	number of dot-instructions
decomposition	$n \cdot m^2$
solution of matrix equation	$2n \cdot m$
band part of inverse	$n \cdot m^2$
inverse	$1/2\, n^2 \cdot m$

Table 1: Leading term of the number of dot-instructions (mainly products) for matrix operations for a n-by-n symmetric band matrix with band width m. All operations are linear in n except the calculation of the full inverse.

for the k-th column $\boldsymbol{v}_k$ here $\boldsymbol{e}_k$ is the k-th column of the unit matrix. Another method based on the decomposed sparse matrix (equation (27)) is explained in the next section.

6.2 Elements of the inverse of a band matrix

Matrix equations of the type $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with a band matrix $\boldsymbol{A}$ can be solved with reduced computational effort without computing the inverse matrix $\boldsymbol{A}^{-1}$, which is a full matrix (no zero elements). In many applications elements of the inverse are however useful, for example the diagonal elements in least squares applications have a significance as estimates of the variances of the fitted parameters. Thus it is useful to compute certain elements of $\boldsymbol{A}^{-1}$, not to use them as an operator, but for the elements themselves. A method which allows to compute a certain part of the inverse is given below.
The band matrix $\boldsymbol{A}$ is considered in the symmetric decomposition (27) where $\boldsymbol{L}$ is unit lower triangular, i.e. the diagonal elements are all 1, with the band structure preserved, and $\boldsymbol{D}$ is diagonal. Let $\boldsymbol{Z} = \boldsymbol{A}^{-1}$ and consider the identity

$$\boldsymbol{Z} = \left(\boldsymbol{L}^{\mathrm{T}}\right)^{-1} \boldsymbol{D}^{-1} + \boldsymbol{Z}\,(\boldsymbol{I} - \boldsymbol{L}) \ . \tag{32}$$

Because the difference $(\boldsymbol{I} - \boldsymbol{L})$ is strictly lower triangular (diagonal elements are zero), the following relations are true:

$$Z_{ij} = -\sum_{k=j+1}^{n} Z_{ik}\, L_{kj} \qquad\qquad i > j \tag{33}$$

$$Z_{ii} = d_{ii}^{-1} - \sum_{k=i+1}^{n} Z_{ik}\, L_{ki} \tag{34}$$

Elements of $\boldsymbol{Z}$ required to evaluate Z_{ij} are those Z_{ik} in row i for which $k \geq j+1$ and $L_{kj} \neq 0$. A sequence of computation can be performed by calculating elements of $\boldsymbol{Z}$ in reverse order, starting with Z_{nn}; when calculating Z_{ij} all required elements of $\boldsymbol{Z}$ are already calculated. The formulas (33) allow to calculate the whole symmetric matrix Z, but often not all elements are required. An important property of the method is the fact, that *all elements of $\boldsymbol{Z}$ in the sparsity pattern can be computed without calculating any elements outside this pattern.* This fact allows to calculate the elements of the inverse $\boldsymbol{Z}$ lying within the band of the original matrix $\boldsymbol{A}$ by a number of operations which is proportional to n, not n^3.

6.3 Symmetric matrix subroutine

The symmetric matrix is stored in symmetric storage mode, with `(N*N+N)/2` elements. After decomposition the inverse diagonal elements of matrix $\boldsymbol{D}$, $1/D_{ii}$, are stored at the diagonal position. The off-diagonal elements of the triangular matrix $\boldsymbol{L}$ are stored in the remaining elements of `W`. The diagonal elements of $\boldsymbol{L}$ are all ones and are not stored.

```
CALL DCHDEC(W,N, AUX)
```

Purpose: decomposition
`W :` symmetric positive definite matrix, stored in symmetric storage mode, with `(N*N+N)/2` elements; the content is replaced by the decomposition;
`N :` number of rows and columns of the matrix;
`AUX :` auxiliary array of length `N`.

```
CALL DCHSLV(W,N,B, X)
```

Purpose: solution of matrix equation
`W :` decomposed symmetric matrix `W` (after decomposition by `DCHDEC`);
`N :` number of rows and columns of the matrix;
`B :` right hand side of the matrix equation;
`X :` result vector of the matrix equation; arrays `B` and `X` may be identical.

```
CALL DCHINV(W,N, V)
```

Purpose: calculation of inverse matrix
`W :` symmetric matrix `W` (after decomposition by `DCHDEC`);
`N :` number of rows and columns of the matrix;
`V :` inverse matrix in symmetric storage mode, i.e. `(N*N+N)/2` elements.

6.4 Bandmatrix subroutines

6.4.1 General band width

The symmetric band matrix of band width `M` is stored in an array `W(MP1,N)`, with `MP1 = M +1`. The diagonal elements are stored in `W(1,.)`. After decomposition the inverse diagonal elements of matrix $\boldsymbol{D}$, $1/D_{ii}$, are stored in `W(1,.)`. The off-diagonal elements of the triangular matrix $\boldsymbol{L}$ are stored in the remaining elements of `W(MP1,N)`. The diagonal elements of $\boldsymbol{L}$ are all ones and are not stored.

```
CALL DBCDEC(W,MP1,N, AUX)
```

Purpose: decomposition
`W :` symmetric positive definite matrix of bandwidth m, with dimension `W(MP1,N)`; the content is replaced by the decomposition;
`MP1 :` dimension parameter `MP1 = M +1`;
`N :` number of rows of the matrix;
`AUX :` auxiliary array of length `N`.

```
CALL DBCSLV(W,MP1,N,B, X)
```

Purpose: solution of matrix equation
`W :` decomposed symmetric band matrix `W` (after decomposition by `DBCDEC`);
`MP1 :` dimension parameter `MP1 = M +1`;
`N :` number of rows of the matrix;
`B :` right hand side of the matrix equation;
`X :` result vector of the matrix equation; arrays `B` and `X` may be identical.

```
CALL DBCIEL(W,MP1,N, V)
```

Purpose: calculation of band part of inverse matrix
`W :` symmetric band matrix `W` (after decomposition by `DBCDEC`);
`M :` band width;
`N :` number of rows of the matrix;
`V :` band part of inverse matrix; same dimension as `W`.

```
CALL DBCINV(W,MP1,N, VS)
```

Purpose: calculation of inverse matrix
`W :` symmetric band matrix `W` (after decomposition by `DBCDEC`);
`MP1 :` band width;
`N :` number of rows of the matrix;
`VS :` complete inverse matrix in symmetric storage mode, i.e. `(N*N+N)/2` elements.

6.4.2 Bandwidth $m = 1$

Band matrices with bandwidth $m = 1$ are called triangular matrices; they have $2m + 1 = 3$ diagonals not equal to zero. The symmetric matrix is stored in an array `W(2,N)`.

```
CALL DB2DEC(W,N, AUX)
```

Purpose: decomposition
`W :` symmetric positive definite matrix of bandwidth $m = 1$, with dimension `W(2,N)`; the content is replaced by the decomposition;
`N :` number of rows of the matrix;
`AUX :` auxiliary array of length `N`.

```
CALL DB2SLV(W,N,B, X)
```

Purpose: solution of matrix equation
`W :` decomposed symmetric band matrix `W` (after decomposition by `DB2DEC`);
`N :` number of rows of the matrix;
`B :` right hand side of the matrix equation;
`X :` result vector of the matrix equation; arrays `B` and `X` may be identical.

```
CALL DB2IEL(W,N, V)
```

Purpose: calculation of band part of inverse matrix
`W :` symmetric band matrix `W` (after decomposition by `DB2DEC`);
`N :` number of rows of the matrix;
`V :` band part of inverse matrix; same dimension as `W`.

6.4.3 Bandwidth $m = 2$

The symmetric matrix of bandwidth $m = 2$ is stored in an array `W(3,N)`.

```
CALL DB3DEC(W,N, AUX)
```

Purpose: decomposition

`W :` symmetric positive definite matrix of bandwidth $m = 2$, with dimension `W(3,N)`; the content is replaced by the decomposition;

`N :` number of rows of the matrix;

`AUX :` auxiliary array of length `N`.

```
CALL DB3SLV(W,N,B, X)
```

Purpose: solution of matrix equation

`W :` decomposed symmetric band matrix `W` (after decomposition by `DB3DEC`);

`N :` number of rows of the matrix;

`B :` right hand side of the matrix equation;

`X :` result vector of the matrix equation; arrays `B` and `X` may be identical.

```
CALL DB3IEL(W,N, V)
```

Purpose: calculation of band part of inverse matrix

`W :` symmetric band matrix `W` (after decomposition by `DB3DEC`);

`N :` number of rows of the matrix;

`V :` band part of inverse matrix; same dimension as `W`.