A kinematical constrained fit using the Lagrange multipliers method.

Juan J. Saborido
(CERN, EP)

Abstract

The bundling of tracks taking into account the kinematical constraint given by the invariant mass of the mother particle and the geometrical ones of having a common point in space, is a problem which involves finding the extrema of a multivariate function subjected to a certain set of analytical constraints. In the present note I will describe in detail an algorithm, based on the Lagrange multipliers method, to perform this kind of fit. This algorithm was implemented in a set of Fortran routines to be used in the context of the LHCb analysis library. This implementation is also briefly discussed. Finally, a brief note on the $K^0_s$ reconstruction is given.
The bundling of tracks taking into account kinematical constraints is of course an old problem which has received proper attention and treatment in many places. See for example [1], or the book by Siegmund Brandt [2], which contains also a clear exposition of the treatment of constrained measurements. In the LHCb analysis library

[1] [3] there is an implementation of a constrained fit procedure to bundle several (up to 6) tracks with any mass constraint. However, the output covariance matrix given by this method is not reliable and causes problems when trying to bundle again the new formed track with other ones. For the special cases of bundling 2 or 3 tracks, if the best performance is to be achieved, it should be better to use an specifically dedicated algorithm rather than an general one to treat N-tracks introducing extra approximations.

1 Introduction.

Let \( y = (y_1, \ldots, y_n) \) be a set of measurements\(^3\) which may be regarded as the sum of the true (unknown) quantities \( \eta \) and measurement errors \( \epsilon = (\epsilon_1, \ldots, \epsilon_n) \):

\[
y_j = \eta_j + \epsilon_j \quad j = 1, \ldots, n
\]  

The covariance matrix of measurements \( y \) will be written as \( C_y \), and we shall assume each \( \epsilon_j \) to be a random variable normally distributed around 0 with variance \( (C_y)_{jj} = \sigma_j^2 \). The joint probability density function of variables \( y \) may then be written in the following form:

\[
\phi(y) = \frac{1}{(2\pi)^{n/2} \sqrt{\det(G_y)}} \exp \left[ -\frac{1}{2}(y - \eta)^\top G_y (y - \eta) \right] \sim \exp \left( -\frac{1}{2} \epsilon^\top G_y \epsilon \right)
\]  

Where \( G_y = C_y^{-1} \) is the inverse of the covariance matrix, and \( \epsilon^\top \) is the transposed of \( \epsilon \). The corresponding logarithmic likelihood function is, apart from constant factors, \( l = -\epsilon^\top G_y \epsilon \), and has a maximum for \( \epsilon = 0 \). In other words, if no more information is provided to the system, the best estimators of the quantities \( \eta \) are obviously the very measurements \( y \).

Now suppose the system of true quantities \( \eta \) is known to verify certain conditions, and we want to use this information to further constrain the measurements \( y \). A typical example could be the measurement of the three angles of a triangle, where we know the sum is constrained to be 180\(^\circ\). Or, in our particular case, the track decay products of a certain particle, where we know all the tracks should have a common point in space and give the invariant mass of that particle. Let’s assume the constraint equations can be written in the form\(^4\):

\[
f_k(\eta) = 0 \quad j = k, \ldots, q
\]  

In general, the \( f_k \) will not be linear functions, but we shall assume they can be well described by a first order Taylor expansion in a neighbourhood of \( \eta_0 \), which denotes a

---

\(^1\)The so-called Axlib, which was used in 1997/98 for the preparation of the Technical Proposal.

\(^2\)Some of the diagonal elements are overestimated, some are underestimated, and the off-diagonal ones give most of the times correlation coefficients greater than 1. The origin of these failures is probably some bug in the Fortran implementation of the method rather than in the method itself, but I cannot assure it.

\(^3\)For example, the parameters of a track or several tracks, as given by the track fitting algorithm.

\(^4\)In the case of a particle decay with \( N \) tracks we would have \( (N - 1) \) such equations for the condition of all tracks having a common point in space, and another one for the condition of the invariant mass. Therefore, a total of \( q = N \) constraints.
first approximation to the true values $\eta$:

$$f_k(\eta) \approx f_k(\eta_o) + \left( \frac{\partial f_k}{\partial \eta} \right)_{\eta_o} (\eta - \eta_o) = 0 \quad (4)$$

We can write equations (4) in a compact matrix form using the following notation:

$$B_{kl} = \left( \frac{\partial f_k}{\partial \eta} \right)_{\eta_o} \quad c_k = f_k(\eta_o) \quad \delta = \eta - \eta_o \quad \delta^t \mathbf{B} \delta + \mathbf{c} = 0 \quad (6)$$

Now it is rather straightforward to extend the probability density function (2) to include the extra information given by equations (6). We do so by adding another Gaussian factor:

$$\phi(y) \sim \exp \left( -\frac{1}{2} \delta^t \mathbf{G}_y \delta \right) \exp \left[ -\mu^t (\mathbf{B} \delta + \mathbf{c}) \right] \quad (7)$$

where the vector $\mu = (\mu_1, \cdots, \mu_q)$ contains the so-called Lagrange multipliers, to be determined later. The logarithmic likelihood is now:

$$l \sim -\frac{1}{2} \delta^t \mathbf{G}_y \delta + \mu^t (\mathbf{B} \delta + \mathbf{c}) \quad (8)$$

By requiring its total differential with respect to $\delta$ to vanish we get $(\delta^t \mathbf{G}_y + \mu^t \mathbf{B} = 0)$, which, together with the constraint equations (6), forms a system of $n + q$ equations with $n + q$ unknowns ($\delta$ and $\mu$):

$$\delta^t \mathbf{G}_y + \mu^t \mathbf{B} = 0 \quad (9)$$

$$\mathbf{B} \delta + \mathbf{c} = 0 \quad (10)$$

After a little bit of matrix algebra, we can solve the previous system for $\delta$ and obtain:

$$\tilde{\delta} = -\mathbf{G}_y^{-1} \mathbf{B}^t (\mathbf{B} \mathbf{G}_y^{-1} \mathbf{B}^t)^{-1} \mathbf{c} = -\mathbf{C}_y \mathbf{B}^t (\mathbf{B} \mathbf{C}_y \mathbf{B}^t)^{-1} \mathbf{c} \quad (11)$$

As a first approximation to $\eta$ we will obviously take the measured values $\eta_o = y$, and use the solution for $\tilde{\delta}$ to get a set of improved measurements, $\tilde{\eta}$:

$$\tilde{\eta} = y - \mathbf{C}_y \mathbf{B}^t (\mathbf{B} \mathbf{C}_y \mathbf{B}^t)^{-1} \mathbf{c} \quad (12)$$

If the constraint equations (3) were linear, expression (12) would be an exact solution in the sense that the constraint equations would be exactly verified by the new values $\tilde{\eta}$. In the non linear case, however, one will have to follow an iterative procedure, using $\tilde{\eta}$ as the initial values for the next step. After this procedure has converged5, the covariance matrix of the final values $\tilde{\eta}$ is found by error propagation to be:

$$\mathbf{C}_{\tilde{\eta}} = \mathbf{C}_y - \mathbf{C}_y \mathbf{B}^t (\mathbf{B} \mathbf{C}_y \mathbf{B}^t)^{-1} \mathbf{B} \mathbf{C}_y \quad (13)$$

And the $\chi^2$ for $q$ degrees of freedom is obtained by doing:

$$\chi^2 = \tilde{\varepsilon}^t \mathbf{G}_y \tilde{\varepsilon} \quad \tilde{\varepsilon} = y - \tilde{\eta} \quad (14)$$

5For the problem discussed in this note, a vertex-mass constrained fit of 2 or 3 tracks, a very good convergence is reached most of the times after 2 iterations, and only very seldom 3 iterations or more are needed.
2 Vertex and Mass constrained fit of $N = 2, 3$ tracks.

As it was already mentioned, for $N$ tracks we’ll have $q = N$ constraints: one for the invariant mass and $N - 1$ for the requirement that the tracks should have a common point in space. The key equations to use are (12) and (13). It has to be noted, however, that for a large number of tracks this procedure implies the inversion of huge matrices and the computation of lots of partial derivatives. Therefore, for large $N$ other methods might be more appropriate. In the rest of this note the explicit formulae and partial derivatives needed for the fit of $N = 3$ tracks will be shown (the formulae for $N = 2$ is an straightforward reduction of the previous set).

The array of measurements, $\mathbf{y}$, will be made up by putting consecutively together the 3 sets of track parameters given at the same $z$-position$^6$ (let’s called Z0 for later). In an obvious notation:

$$ \mathbf{y} = (x_1, y_1, x'_1, y'_1, p_1, x_2, y_2, x'_2, y'_2, p_2, x_3, y_3, x'_3, y'_3, p_3) $$ (15)

The covariance matrix $\mathbf{C}_y$ is made up in the same way, and has a block-diagonal form. The constraint that the 3 tracks should add up to make a particle with invariant mass $M$ can be written in the following way:

$$ \sqrt{(p_1^2 + m_1^2)(p_2^2 + m_2^2) + \sqrt{(p_1^2 + m_1^2)(p_3^2 + m_3^2) + \sqrt{(p_2^2 + m_2^2)(p_3^2 + m_3^2) - $$ (16)

$$ p_1 p_2 \cos \theta_1 - p_1 p_3 \cos \theta_2 - p_2 p_3 \cos \theta_3 - \beta = 0 $$

Where $m_1, m_2, m_3$ are the masses of the daughter particles and:

$$ \beta = \frac{1}{2} \left( M^2 - m_1^2 - m_2^2 - m_3^2 \right) $$ (17)

$$ \cos \theta_1 = \frac{x'_1 x'_2 + y'_1 y'_2 + 1}{\sqrt{(x'_1)^2 + (y'_1)^2 + 1}\sqrt{(x'_2)^2 + (y'_2)^2 + 1}} $$

$$ \cos \theta_2 = \frac{x'_1 x'_3 + y'_1 y'_3 + 1}{\sqrt{(x'_1)^2 + (y'_1)^2 + 1}\sqrt{(x'_3)^2 + (y'_3)^2 + 1}} $$ (18)

$$ \cos \theta_3 = \frac{x'_2 x'_3 + y'_2 y'_3 + 1}{\sqrt{(x'_2)^2 + (y'_2)^2 + 1}\sqrt{(x'_3)^2 + (y'_3)^2 + 1}} $$

That the tracks must have a common point in space is expressed in the following equations:

$$ (x_2 - x_1)(y'_1 - y'_2) - (y_2 - y_1)(x'_1 - x'_2) = 0 $$ (19)

$$ (x_3 - x_1)(y'_1 - y'_3) - (y_3 - y_1)(x'_1 - x'_3) = 0 $$ (20)

Once the constraint equations are identified, the actual implementation of the fit in a real program is just a matter of computing lots of partial derivatives and inverting matrices. Although the fit procedure is clear, it will be very helpful to write down explicitly all the equations used in the Fortran code developed for the purpose, should any one read it in the future.

$^6$The “standard” LHCb reference frame is used here, with the $z$-coordinate running positive towards the increasing aperture of the spectrometer, and the $y$-coordinate running positive upwards.
3 Fortran Implementation of the fit.

To implement this fit a set of 5 Fortran routines were written (AxLM.cftfit, AxLM_itera, AxLM.evalu, AxLM.chisq and AxLM.nwcoy), and another one (axcbun23) for the interface to the analysis library of LHCb.

- *axcbun23* is the routine to be called to bundle 2 or 3 tracks with the invariant mass constraint of the mother particle. The arguments are the same as for *axcbndle*. No other action is to be taken by the user except substituting *axcbndle* by *axcbun23*.

- *AxLM.cftfit* is the main routine which drives the fit. Organises the input data in suitable arrays as outlined above. Takes control of the iteration loop to decide when the constraint equations are sufficiently well satisfied. If the fit is successful, calls the relevant routines to compute the $\chi^2$ and covariance matrix of the new parameters.

- *AxLM_itera* is the routine in charge of performing one iteration of the fit to improve measurements. It is, essentially, an implementation of equations (12) and (13).

- *AxLM_evalu* evaluates the constraint equations (6) after each step to decide when to finish up.

- *AxLM_chisq* computes the $\chi^2$ after completion of the fit according to equation (14).

- *AxLM_nwcoy* translates the improved set of 15 parameters (and the 15x15 cov. matrix) to the 6 parameters set of the final track.

In the following, the partial derivatives and the formulae used in routines *AxLM_itera* and *AxLM_nwcoy* will be explicitly written down. The other routines contain more or less straightforward material. For matrix multiplication and inversion, the standard package of CERNLIB routines was used. I should perhaps note that no special effort was dedicated to get a fast implementation of the algorithm. For example, the fact that the input covariance matrix of parameters has a block-diagonal form and is symmetric, is not exploited in any way to speed up the calculus.

3.1 Formulae used in routine *AxLM_itera*.

To begin with, the constraint equations are rewritten in the uniform notation of the “true” values $\eta$:

\begin{align*}
    f_1(\eta) &= (\eta_6 - \eta_1)(\eta_4 - \eta_9) - (\eta_7 - \eta_2)(\eta_3 - \eta_8) = 0 \quad (21) \\
    f_2(\eta) &= (\eta_{11} - \eta_1)(\eta_4 - \eta_{14}) - (\eta_{12} - \eta_2)(\eta_3 - \eta_{13}) = 0 \quad (22) \\
    f_3(\eta) &= \sqrt{(\eta_8^2 + m_1^2)(\eta_{10}^2 + m_2^2)} + \sqrt{(\eta_9^2 + m_1^2)(\eta_{15}^2 + m_3^2)} + \sqrt{(\eta_{10}^2 + m_2^2)(\eta_{15}^2 + m_3^2)} - \\
    &- \eta_5\eta_{10}\cos\theta_1 - \eta_5\eta_{15}\cos\theta_2 - \eta_{10}\eta_{15}\cos\theta_3 - \beta = 0 \quad (23)
\end{align*}

To get the matrix $B$ one needs to compute the the partial derivatives of the constraint equations. In the same way it is done in routine *AxLM_itera*, intermediate variables and
expressions will be used to abbreviate the notation.

\[
\begin{align*}
P12 &= (\eta_5^2 + m_1^2)(\eta_1^2 + m_2^2) & S12 &= \eta_3\eta_8 + \eta_1\eta_9 + 1 \\
P13 &= (\eta_5^2 + m_1^2)(\eta_2^2 + m_3^2) & S13 &= \eta_3\eta_{13} + \eta_1\eta_{14} + 1 \\
P23 &= (\eta_{10}^2 + m_2^2)(\eta_{15}^2 + m_3^2) & S23 &= \eta_8\eta_{13} + \eta_1\eta_{14} + 1 \\
\end{align*}
\]

(24)

\[
\begin{align*}
S11 &= \eta_3^2 + \eta_1^2 + 1 \\
S22 &= \eta_8^2 + \eta_9^2 + 1 \\
S33 &= \eta_{13}^2 + \eta_{14}^2 + 1
\end{align*}
\]

(25)

\[\cos \theta_1, \cos \theta_2 \text{ and } \cos \theta_3 \text{ are now written as:}\]

\[
\begin{align*}
\cos \theta_1 &= (S12)/\sqrt{(S11)(S22)} \\
\cos \theta_2 &= (S13)/\sqrt{(S11)(S33)} \\
\cos \theta_3 &= (S23)/\sqrt{(S22)(S33)}
\end{align*}
\]

(26)

Partial derivatives to be used later:

\[
\begin{align*}
\partial_5 \left( \sqrt{P12} \right) &= \eta_5 \sqrt{\eta_1^2 + m_2^2}/(\eta_5^2 + m_1^2) \\
\partial_{10} \left( \sqrt{P12} \right) &= \eta_{10} \sqrt{\eta_2^2 + m_3^2}/(\eta_{10}^2 + m_2^2) \\
\partial_5 \left( \sqrt{P13} \right) &= \eta_5 \sqrt{\eta_1^2 + m_3^2}/(\eta_5^2 + m_1^2) \\
\partial_{15} \left( \sqrt{P13} \right) &= \eta_{15} \sqrt{\eta_2^2 + m_3^2}/(\eta_{15}^2 + m_2^2) \\
\partial_{10} \left( \sqrt{P23} \right) &= \eta_{10} \sqrt{\eta_2^2 + m_3^2}/(\eta_{10}^2 + m_2^2) \\
\partial_{15} \left( \sqrt{P23} \right) &= \eta_{15} \sqrt{\eta_2^2 + m_3^2}/(\eta_{15}^2 + m_2^2)
\end{align*}
\]

(27)

Partial derivatives of the cosines:

\[
\begin{align*}
\partial_3(\cos \theta_1) &= \frac{\eta_8 - \eta_3}{\sqrt{(S11)(S22)}} \\
\partial_4(\cos \theta_1) &= \frac{\eta_9 - \eta_4}{\sqrt{(S11)(S22)}} \\
\partial_5(\cos \theta_1) &= \frac{\eta_5 - \eta_8}{\sqrt{(S11)(S22)}} \\
\partial_3(\cos \theta_2) &= \frac{\eta_3 - \eta_3}{\sqrt{(S11)(S33)}} \\
\partial_4(\cos \theta_2) &= \frac{\eta_1 - \eta_3}{\sqrt{(S11)(S33)}} \\
\partial_5(\cos \theta_2) &= \frac{\eta_3 - \eta_{13}}{\sqrt{(S11)(S33)}} \\
\partial_3(\cos \theta_3) &= \frac{\eta_8 - \eta_3}{\sqrt{(S22)(S33)}} \\
\partial_4(\cos \theta_3) &= \frac{\eta_9 - \eta_3}{\sqrt{(S22)(S33)}} \\
\partial_5(\cos \theta_3) &= \frac{\eta_8 - \eta_{13}}{\sqrt{(S22)(S33)}} \\
\end{align*}
\]

(28)
Finally, the partial derivatives of the constraint functions.

\[
\begin{align*}
\partial_1(f_1) &= \eta_9 - \eta_4 & \partial_6(f_1) &= -\partial_1(f_1) \\
\partial_2(f_1) &= \eta_3 - \eta_8 & \partial_7(f_1) &= -\partial_2(f_1) \\
\partial_3(f_1) &= \eta_2 - \eta_7 & \partial_8(f_1) &= -\partial_3(f_1) \\
\partial_4(f_1) &= \eta_6 - \eta_1 & \partial_9(f_1) &= -\partial_4(f_1) \\
\partial_{10}(f_1) &= -\eta_5 \eta_10 \partial_5(\cos \theta_1) - \eta_5 \eta_15 \partial_3(\cos \theta_2) \\
\partial_{11}(f_1) &= -\eta_5 \eta_10 \partial_4(\cos \theta_1) - \eta_5 \eta_15 \partial_4(\cos \theta_2) \\
\partial_{12}(f_1) &= -\eta_5 \eta_10 \partial_6(\cos \theta_1) - \eta_10 \eta_15 \partial_6(\cos \theta_3) \\
\partial_{13}(f_1) &= -\eta_5 \eta_15 \partial_13(\cos \theta_1) - \eta_10 \eta_15 \partial_13(\cos \theta_3) \\
\partial_{14}(f_1) &= -\eta_5 \eta_15 \partial_14(\cos \theta_1) - \eta_10 \eta_15 \partial_14(\cos \theta_3) \\
\partial_{15}(f_1) &= \partial_5 \left(\sqrt{P12}\right) + \partial_5 \left(\sqrt{P13}\right) - \eta_10 \cos \theta_1 - \eta_15 \cos \theta_2 \\
\partial_{16}(f_1) &= \partial_{10} \left(\sqrt{P12}\right) + \partial_{10} \left(\sqrt{P23}\right) - \eta_1 \cos \theta_1 - \eta_15 \cos \theta_2 \\
\partial_{17}(f_1) &= \partial_{15} \left(\sqrt{P13}\right) + \partial_{15} \left(\sqrt{P23}\right) - \eta_1 \cos \theta_2 - \eta_10 \cos \theta_1
\end{align*}
\]

(29)

(30)

(31)

3.2 Formulae used in routine \textit{AxLM\_nu4cov}.

After the fitting procedure has converged, routine \textit{AxLM\_item} returns a set of 15 improved parameters (for the 3 tracks) and a 15x15 covariance matrix. The task of \textit{AxLM\_nu4cov} is to translate all this to the final 6 parameters of the mother track and its 6x6 covariance matrix.

Let’s represent by the vector \( \mathbf{x} = (x, y, z, x', y', p) \) the 6 parameter set of the mother track\(^7\) and by the matrix \( \mathbf{T} \) the Jacobian of the transformation between variables \( \mathbf{x} \) and \( \eta \). In this notation, the covariance matrices are related by the following expression:

\[
\mathbf{C}_x = \mathbf{T} \mathbf{C}_\eta \mathbf{T}^t
\]

(32)

As before, to abbreviate writing the following variables are introduced:

\[
\begin{align*}
\text{pz}1 &= \eta_5/\sqrt{(S11)} & \text{pz}2 &= \eta_{10}/\sqrt{(S22)} & \text{pz}3 &= \eta_{15}/\sqrt{(S33)} \\
\text{spz} &= \text{pz}1 + \text{pz}2 + \text{pz}3 \\
\text{spz}1 &= \eta_3 \text{pz}1 + \eta_8 \text{pz}2 + \eta_{13} \text{pz}3 \\
\text{spz}2 &= \eta_4 \text{pz}1 + \eta_9 \text{pz}2 + \eta_{14} \text{pz}3
\end{align*}
\]

(33)

(34)

\(^7\)The first 3 parameters of \( \mathbf{x} \) will contain in fact the coordinates of the fitted decay vertex, and it is bounded to be better than the result of a normal vertex fit because the extra condition of the invariant mass was imposed.
And the corresponding partial derivatives:

\[
\begin{align*}
\partial_2(pz1) &= -\eta_3(pz1)/(S11) \\
\partial_3(pz1) &= -\eta_4(pz1)/(S11) \\
\partial_5(pz1) &= 1/(S11) \\
\partial_6(pz1) &= -\eta_6(pz2)/(S22) \\
\partial_7(pz1) &= -\eta_6(pz2)/(S22) \\
\partial_8(pz1) &= 1/(S22)
\end{align*}
\]

The variables \( \mathbf{x} \) in terms of \( \eta \) are:

\[
\begin{align*}
x_1 &= x = \eta_1 + \eta_3(\eta_6 - \eta_1)/(\eta_3 - \eta_8) \\
x_2 &= y = \eta_2 + \eta_4(\eta_6 - \eta_1)/(\eta_3 - \eta_8) \\
x_3 &= z = Z0 + (\eta_6 - \eta_1)/(\eta_3 - \eta_8) \\
x_4 &= x' = (s1pz)/(s0pz) \\
x_5 &= y' = (s2pz)/(s0pz) \\
x_6 &= p = \sqrt{(s1pz)^2 + (s2pz)^2 + (s0pz)^2}
\end{align*}
\]

\( Z0 \) is the z-position at which the initial track parameters and covariance matrices are given as an input to the fit\(^8\). And to finish, the the non-zero partial derivatives to fill matrix \( \mathbf{T} \) are:

\[
\begin{align*}
\partial_1(x_1) &= 1 - \eta_3/(\eta_3 - \eta_8) \\
\partial_3(x_1) &= -1/(\eta_3 - \eta_8) \\
\partial_6(x_1) &= \eta_3/(\eta_3 - \eta_8) \\
\partial_8(x_1) &= \eta_3(\eta_6 - \eta_1)/(\eta_3 - \eta_8)^2 \\
\partial_9(x_1) &= -\partial_1(x_2) \\
\partial_1(x_2) &= -\eta_4/(\eta_3 - \eta_8) \\
\partial_3(x_2) &= -\eta_4(\eta_6 - \eta_1)/(\eta_3 - \eta_8)^2 \\
\partial_6(x_2) &= -\partial_1(x_2) \\
\partial_8(x_2) &= -\partial_2(x_2) = 1 \\
\partial_9(x_2) &= -\partial_2(x_2)
\end{align*}
\]

\[
\begin{align*}
\partial_3(x_4) &= \frac{1}{(s0pz)} \left[ (pz1) + \eta_3 \partial_3(pz1) - \frac{(s1pz) \partial_3(pz1)}{(s0pz)} \right] \\
\partial_8(x_4) &= \frac{1}{(s0pz)} \left[ (pz2) + \eta_8 \partial_8(pz2) - \frac{(s1pz) \partial_8(pz2)}{(s0pz)} \right] \\
\partial_{13}(x_4) &= \frac{1}{(s0pz)} \left[ (pz3) + \eta_{13} \partial_{13}(pz3) - \frac{(s1pz) \partial_{13}(pz3)}{(s0pz)} \right]
\end{align*}
\]

\(^8\)See the paragraph before equation (15).
\[
\begin{align*}
\partial_1(x_4) &= \frac{1}{(s0pz)} \left[ \eta_3 \partial_1(pz1) - \frac{(s1pz) \partial_1(pz1)}{(s0pz)} \right] \\
\partial_2(x_4) &= \frac{1}{(s0pz)} \left[ \eta_5 \partial_2(pz1) - \frac{(s1pz) \partial_2(pz1)}{(s0pz)} \right] \\
\partial_3(x_4) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_3(pz2) - \frac{(s1pz) \partial_3(pz2)}{(s0pz)} \right] \\
\partial_4(x_4) &= \frac{1}{(s0pz)} \left[ \eta_8 \partial_4(pz2) - \frac{(s1pz) \partial_4(pz2)}{(s0pz)} \right] \\
\partial_5(x_4) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_5(pz2) - \frac{(s1pz) \partial_5(pz2)}{(s0pz)} \right] \\
\partial_6(x_4) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_6(pz2) - \frac{(s1pz) \partial_6(pz2)}{(s0pz)} \right]
\end{align*}
\]

\[
\begin{align*}
\partial_1(x_5) &= \frac{1}{(s0pz)} \left[ (pz1) + \eta_1 \partial_1(pz1) - \frac{(s2pz) \partial_1(pz1)}{(s0pz)} \right] \\
\partial_2(x_5) &= \frac{1}{(s0pz)} \left[ (pz2) + \eta_9 \partial_2(pz2) - \frac{(s2pz) \partial_2(pz2)}{(s0pz)} \right] \\
\partial_3(x_5) &= \frac{1}{(s0pz)} \left[ (pz3) + \eta_9 \partial_3(pz3) - \frac{(s2pz) \partial_3(pz3)}{(s0pz)} \right]
\end{align*}
\]

\[
\begin{align*}
\partial_4(x_5) &= \frac{1}{(s0pz)} \left[ \eta_4 \partial_4(pz1) - \frac{(s2pz) \partial_4(pz1)}{(s0pz)} \right] \\
\partial_5(x_5) &= \frac{1}{(s0pz)} \left[ \eta_4 \partial_5(pz1) - \frac{(s2pz) \partial_5(pz1)}{(s0pz)} \right] \\
\partial_6(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_6(pz2) - \frac{(s2pz) \partial_6(pz2)}{(s0pz)} \right] \\
\partial_7(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_7(pz2) - \frac{(s2pz) \partial_7(pz2)}{(s0pz)} \right] \\
\partial_8(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_8(pz2) - \frac{(s2pz) \partial_8(pz2)}{(s0pz)} \right] \\
\partial_9(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_9(pz2) - \frac{(s2pz) \partial_9(pz2)}{(s0pz)} \right] \\
\partial_{10}(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_{10}(pz2) - \frac{(s2pz) \partial_{10}(pz2)}{(s0pz)} \right] \\
\partial_{11}(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_{11}(pz2) - \frac{(s2pz) \partial_{11}(pz2)}{(s0pz)} \right] \\
\partial_{12}(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_{12}(pz2) - \frac{(s2pz) \partial_{12}(pz2)}{(s0pz)} \right] \\
\partial_{13}(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_{13}(pz2) - \frac{(s2pz) \partial_{13}(pz2)}{(s0pz)} \right] \\
\partial_{14}(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_{14}(pz2) - \frac{(s2pz) \partial_{14}(pz2)}{(s0pz)} \right] \\
\partial_{15}(x_5) &= \frac{1}{(s0pz)} \left[ \eta_9 \partial_{15}(pz2) - \frac{(s2pz) \partial_{15}(pz2)}{(s0pz)} \right]
\end{align*}
\]
\[ \frac{1}{x_6} \left\{ \partial_3(pz1) \left[ (s0pz) + (s1pz)\eta_3 + (s2pz)\eta_4 + (s1pz)(pz1) \right] \right\} \] (42)

\[ \frac{1}{x_6} \left\{ \partial_1(pz1) \left[ (s0pz) + (s1pz)\eta_3 + (s2pz)\eta_4 + (s2pz)(pz1) \right] \right\} \]

\[ \frac{1}{x_6} \left\{ \partial_8(pz2) \left[ (s0pz) + (s1pz)\eta_8 + (s2pz)\eta_9 + (s1pz)(pz2) \right] \right\} \]

\[ \frac{1}{x_6} \left\{ \partial_9(pz2) \left[ (s0pz) + (s1pz)\eta_8 + (s2pz)\eta_9 + (s2pz)(pz2) \right] \right\} \]

\[ \frac{1}{x_6} \left\{ \partial_{13}(pz3) \left[ (s0pz) + (s1pz)\eta_{13} + (s2pz)\eta_{14} + (s1pz)(pz3) \right] \right\} \]

\[ \frac{1}{x_6} \left\{ \partial_{14}(pz3) \left[ (s0pz) + (s1pz)\eta_{13} + (s2pz)\eta_{14} + (s2pz)(pz3) \right] \right\} \]

\[ \frac{\partial_6(pz1)}{x_6} \left[ (s1pz)\eta_3 + (s2pz)\eta_4 + (s0pz) \right] \]

\[ \frac{\partial_9(pz2)}{x_6} \left[ (s1pz)\eta_8 + (s2pz)\eta_9 + (s0pz) \right] \]

\[ \frac{\partial_{15}(pz3)}{x_6} \left[ (s1pz)\eta_{13} + (s2pz)\eta_{14} + (s0pz) \right] \]

4 On the \( K^0_s \) vertex reconstruction.

Although it is not strictly related to the topic discussed in this note, the special treatment of the \( K^0_s \) vertex reconstruction performed for the studies of the Technical Proposal is also briefly described here.

The standard routine of vertex fitting in the \textit{Axlib} library, \textit{axvrtx}, makes a linear extrapolation of the tracks to estimate the \( z \)-position of the vertex, and then transports track parameters and covariance matrix to that estimated position to perform the fit. This is inappropriate when trying the reconstruct the vertex of a \( K^0_s \) which has decayed in the magnet region. Therefore, a modified version of \textit{axvrtx} has been developed for the \( K^0_s \) vertex fitting: \textit{axksvrtx}, which calls in turn some other routines.

- \textit{axksvrtx} is to be used in the same manner as \textit{axvrtx}. It fits the 2 top entries of the stack to a common vertex. It has, nevertheless, an extra output argument to return the invariant mass formed by the two tracks. This routine should be used instead of \textit{axemass} when searching for a \( K^0_s \).

- \textit{axkzestxy} is called from \textit{axksvrtx} to perform the \( z \)-estimation of the possible vertex formed by the 2 track candidates. If the first measurement of both tracks is outside the magnet region, it makes a linear extrapolation to look for the position of maximum approach. If the tracks are in the magnet region, uses only the Y-Z projection of the tracks to perform such extrapolation\(^9\).

\(^9\)Studies were made to see if some improvement could be expected by making use of the X-Z transport of the tracks according to the equations of motion in the LHCb field-map. The result was negative. This was studied by giving to the fitting procedure the true Montecarlo \( K^0_s \) decay vertex as the starting point instead of the estimation made by \textit{axkzestxy}, and no significant improvement was seen. Therefore, the conclusion was that a linear extrapolation based only in the non-bending projection (Y-Z) is sufficient as a first estimation of the \( z \)-vertex.
• `axkswim` is called from `axksvrtx` to make the transport of the 2 pion candidates to the estimated z-position given by `axkzestxy`. In the vertex region this routine simply calls the standard `axswim`, in other case calls the kalman-transport procedure used in routine `axtransport` developed for the kalman track fitting.

• `axkp2qpt` and `axkqpt2p` makes the conversion of the covariance matrix of the parameters used by the `Axlib` library to the ones used in the kalman-fitting-algorithm, and vice-versa\textsuperscript{10}.

The constrained fit implementation by `axcbun23` is already prepared to make the transport of the pions following the previous procedure when trying to bundle a $K_s^0$.

References


\textsuperscript{10}In the kalman-fitting package for LHCb there exists some routines to make this transformation, but only in one direction, not in both.