CBM-SOFT-note-2006-001
I3HP-FutureDAQ-note-2006-001
9 January 2006

Primary vertex fit based on the Kalman filter

S. Gorbunov[1] and I. Kisel[2]

[1]DESY Zeuthen, Platanenallee 6, 15738 Zeuthen, Germany
[2]Kirchhoff Institute for Physics, University of Heidelberg, 69120 Heidelberg, Germany

Abstract

The note describes a Kalman filter based reconstruction routine for fitting the primary vertex. Results of tests of the vertex routine are presented and discussed. The source code is given in the Appendix.

1 Introduction

The goal of a vertex fit is to obtain the vertex position and associated covariance matrix using a set of track estimates and their covariance matrices. The proposed algorithm is aimed for the primary vertex fit problem which is characterized by high track multiplicity and absence of additional physical constraints on the vertex tracks. The algorithm is based on the extended Kalman filter method.

2 The extended Kalman filter method

The Kalman filter method [1] is intended for finding the optimum estimation $\mathbf{r}$ of an unknown vector $\mathbf{r}^t$ according to the known measurements $\mathbf{m}_k,\ k = 1 \dots n$ of the vector $\mathbf{r}^t$.

The Kalman filter starts with a certain initial approximation $\mathbf{r} = \mathbf{r}_0$ and refines the vector $\mathbf{r}$, consecutively adding one measurement after the other. The optimum value is attained after the addition of the last measurement.

In the general case the unknown vector $\mathbf{r}^t$ is changed from one measurement to the next. Here we describe a simplified version of the Kalman filter, where the vector $\mathbf{r}^t$ is identical for all measurements.

The measurement $\mathbf{m}_k$ linearly depends on $\mathbf{r}^t$:

$$\mathbf{m}_k = H_k \mathbf{r}^t + \boldsymbol{\eta}_k \,, \tag{1}$$

where $\boldsymbol{\eta}_k$ — an error of the k-th measurement. The error is unbiased and its covariance matrix V_k is known:

$$\begin{aligned} < \boldsymbol{\eta}_k > &= \mathbf{0} \,, \\ < \boldsymbol{\eta}_k \cdot \boldsymbol{\eta}_k^T > &= V_k \,. \end{aligned} \tag{2}$$

The algorithm of fitting consists of two stages:

1. Initialization. Choose an approximate value of vector $\mathbf{r}_0$. Its covariance matrix is set to $C_0 = \mathrm{I} \cdot \mathrm{inf}$, where inf denotes a large positive number.

2. Filtration. For each measurement $\mathbf{m}_k$ a vector $\mathbf{r}_k$, which is the optimum estimation of the vector $\mathbf{r}^t$ according to the first k measurements, is calculated:

$$\begin{aligned} S_k &= (V_k + H_k C_k H_k^T)^{-1} , \\ K_k &= C_k H_k^T S_k , \\ \boldsymbol{\zeta}_k &= \mathbf{m}_k - H_k \mathbf{r}_{k-1} , \\ \mathbf{r}_k &= \mathbf{r}_{k-1} + K_k \boldsymbol{\zeta}_k , \\ C_k &= C_{k-1} - K_k H_k C_k , \\ \chi_k^2 &= \chi_{k-1}^2 + \boldsymbol{\zeta}_k^T S_k \boldsymbol{\zeta}_k . \end{aligned} \tag{3}$$

Here $\mathbf{r}_{k-1}$, C_{k-1} — the optimum estimation, obtained at the previous step and the error covariance matrix; $\mathbf{m}_k$, V_k — the k-th measurement and its covariance matrix; the matrix H_k — the model of measurement; the matrix K_k is the so-called gain matrix; the vector $\boldsymbol{\zeta}_k$ denotes residual; the value χ_k^2 is the total χ^2-deviation of the obtained estimation $\mathbf{r}_k$ from the measurements $\mathbf{m}_1, \ldots \mathbf{m}_k$.

The vector $\mathbf{r}_n$ obtained after the filtration of the last measurement is the desired optimal estimation with the covariance matrix C_n.

In the case the measurements $\mathbf{m}_k$ nonlinearly depend on $\mathbf{r}^t$, it is necessary to linearize the model of measurement. As a point of linearization a certain vector $\mathbf{r}^0$ is selected:

$$\mathbf{m}_k(\mathbf{r}^t) = \mathbf{h}_k(\mathbf{r}^t) + \boldsymbol{\eta}_k \approx \mathbf{h}_k(\mathbf{r}^0) + H_k(\mathbf{r}^t - \mathbf{r}^0) + \boldsymbol{\eta}_k , \tag{4}$$

where H_k is the Jacobian of $\mathbf{h}_k(\mathbf{r})$ at $\mathbf{r}^0$:

$$H_{k\,(ij)} = \left. \frac{\partial \mathbf{h}_k(\mathbf{r})_{\,(i)}}{\partial \mathbf{r}_{\,(j)}} \right|_{\mathbf{r}=\mathbf{r}^0} . \tag{5}$$

The Kalman filter with nonlinear measurement model is called the *extended* Kalman filter. Equations of filtration for the extended Kalman filter are the same as in the ordinary case (3) with an exception for the residual $\boldsymbol{\zeta}_k$, which is calculated according to the formula:

$$\boldsymbol{\zeta}_k = \mathbf{m}_k - (\mathbf{h}_k(\mathbf{r}^0) + H_k(\mathbf{r}_{k-1} - \mathbf{r}^0)) . \tag{6}$$

As can be seen, the result of the fit depends on the point of linearization $\mathbf{r}^0$. Taking the current solution $\mathbf{r}_{k-1}$ as the point of linearization, the extended Kalman filter coincides with the usual one (6). But in order to get more robust and precise results, the fitting procedure must be repeated several times, using the optimal estimation $\mathbf{r}_n$ as the linearization point for the next iteration.

3 Primary vertex fit

The fit of primary vertex consists in determining the position of the vertex using the track estimates obtained after the track fitter. Each track estimate is considered as an independent measurement of the vertex position [2, 3, 4]. The vertex fit is achieved by the extended Kalman filter, described in the previous section.

Let us denote:

$\mathbf{r} = (x_v,\ y_v,\ z_v)^T$ — the vertex position;

$a_k, b_k, (q/p)_k$ — the directions and the inverse momentum of the k-th track, originating from the vertex $\mathbf{r}$;

$\mathbf{m}_k = (x_k, y_k, t_{xk}, t_{yk}, (q/p)_k)^T$ — the k-th track estimate, parametrized at a certain z_{ref};

V_k — the covariance matrix of the k-th track estimate;

$\mathbf{h}_k(\mathbf{r}, a_k, b_k, (q/p)_k)$ — parameters of the k-th track, extrapolated from z_v to z_{ref}:

$$\mathbf{h}_k(\mathbf{r}, a_k, b_k, (q/p)_k) \quad = \quad \begin{pmatrix} x_v + a_k \cdot (z_{ref} - z_v) + O((z_{ref} - z_v)^2) \\ y_v + b_k \cdot (z_{ref} - z_v) + O((z_{ref} - z_v)^2) \\ a_k \\ b_k \\ (q/p)_k \end{pmatrix} . \tag{7}$$

Here the term $O((z_{ref} - z_v)^2)$ describes the deviation of the track from a straight line in a magnetic field (see details in [5]).

The measurement model is:

$$\begin{aligned} \mathbf{m}_k &= \mathbf{h}_k(\mathbf{r}^t, a_k^t, b_k^t, (q/p)_k^t) + \boldsymbol{\eta}_k \,, \\ < \boldsymbol{\eta}_k \cdot \boldsymbol{\eta}_k^T > &= V_k \,. \end{aligned} \tag{8}$$

The measurement model is nonlinear in z_v, a_k, b_k (7) and, therefore, should be linearized at a certain values (z_v^0, a_k^0, b_k^0). Let us extrapolate the track estimates to z_v^0 (setting $z_{ref} = z_v^0$). The Jacobian H_k of the vector $\mathbf{h}_k$ at z_v^0 is:

$$H_k \quad = \quad \begin{pmatrix} 1 & 0 & -a_k^0 & 0 & 0 & 0 \\ 0 & 1 & -b_k^0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} . \tag{9}$$

The linearised measurement model, according to (4), is:

$$\mathbf{m}_k(\mathbf{r}) \quad \approx \quad \begin{pmatrix} a_k^0 \cdot z_v^0 \\ b_k^0 \cdot z_v^0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & -a_k^0 & 0 & 0 & 0 \\ 0 & 1 & -b_k^0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_v \\ y_v \\ z_v \\ a_k \\ b_k \\ (q/p)_k \end{pmatrix} + \boldsymbol{\eta}_k \,. \tag{10}$$

In the linearized model of measurement:

- only the x and y components of $\mathbf{m}_k$ depend on the vertex parameters $\mathbf{r}_k$;
- these components do not depend on the parameters $(a_k, b_k, (q/p)_k)$ of the vertex track;
- the parameters of the k-th vertex track are measured only by the k-th track estimate $\mathbf{m}_k$.

Thus, the values $a_k, b_k, (q/p)_k$ do not influence the measurement of the vertex position $\mathbf{r}_k$ with the track estimate $\mathbf{m}_k$ and, therefore, there is no need to fit these values at the k-th step of the Kalman filter.

The vertex fit is achieved according to eq. (3) of the Kalman filter with the state vector $\mathbf{r} = (x_v, y_v, z_v)^T$ and the measurement model H_k^{xy}:

$$\begin{aligned}
H_k^{xy} &= \begin{pmatrix} 1 & 0 & -a_k^0 \\ 0 & 1 & -b_k^0 \end{pmatrix}, \\
S_k &= (V_k^{xy} + H_k^{xy} C_k H_k^{xyT})^{-1}, \\
K_k &= C_k H_k^{xyT} S_k, \\
\boldsymbol{\zeta}_k &= \mathbf{m}_k^{xy} - H_k^{xy}(\mathbf{r}_{k-1} - \begin{pmatrix} 0 \\ 0 \\ z_v^0 \end{pmatrix}), \\
\mathbf{r}_k &= \mathbf{r}_{k-1} + K_k \boldsymbol{\zeta}_k, \\
C_k &= C_{k-1} - K_k H_k^{xy} C_k, \\
\chi_k^2 &= \chi_{k-1}^2 + \boldsymbol{\zeta}_k^T S_k \boldsymbol{\zeta}_k.
\end{aligned} \tag{11}$$

Here the two-dimensional measurement $\mathbf{m}_k^{xy}$ and the matrix V_k^{xy} denote the x and y components of the track estimate $\mathbf{m}_k$ and the corresponding part of its covariance matrix V_k.

The directions a_k^0, b_k^0 in eq. (11) are determined by fitting the track $\mathbf{m}_k$ to the vertex guess $\mathbf{r}^0$ used in the linearization. For that, one step of filtration with the vertex (x_v^0, y_v^0) as the measurement with the null matrix of errors is performed:

$$\begin{aligned}
\tilde{H}_k &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \\
\tilde{K}_k &= V_k \tilde{H}_k^T (\mathrm{O} + \tilde{H}_k V_k \tilde{H}_k^T)^{-1}, \\
\tilde{\boldsymbol{\zeta}}_k &= (x_v^0, y_v^0)^T - \tilde{H}_k \mathbf{m}_k, \\
\mathbf{m}_k^0 &= \mathbf{m}_k + \tilde{K}_k \tilde{\boldsymbol{\zeta}}_k, \\
a_k^0 &= \mathbf{m}_{k\,(2)}^0, \\
b_k^0 &= \mathbf{m}_{k\,(3)}^0.
\end{aligned} \tag{12}$$

Here $\mathbf{m}_k^0$ denotes the track fitted to the $\mathbf{r}^0$ vertex from which the directions a_k^0, b_k^0 are taken. Note, that there is no need to calculate the momentum $(q/p)_k^0$ of the vertex track.

Additionally, the χ^2 distance between the track estimate and the vertex guess $\mathbf{r}^0$ is calculated in order to reject wrong tracks:

$$\chi_k^2 = \tilde{\boldsymbol{\zeta}}_k^T (C^0 + \tilde{H}_k V_k \tilde{H}_k^T)^{-1} \tilde{\boldsymbol{\zeta}}_k, \tag{13}$$

where C^0 is the covariance matrix of the vertex guess $\mathbf{r}^0$.

The complete algorithm is following: `Appendix`

- First approximation of the vertex $\mathbf{r} = (x_v^0, y_v^0, z_v^0)$, C. `032-045`
- A few iterations of the vertex fit: `047-156`
  1. Initialization: `051-065`
     - Store vertex from previous iteration: $\mathbf{r}^0 = \mathbf{r}$, $C^0 = C$. `051-056`
     - The initial vertex state vector is taken from the previous iteration $\mathbf{r} = \mathbf{r}^0$.
     - The initial covariance matrix is set to $C = \mathrm{I} \cdot \inf$. `058-062`

   - Initialize the vertex χ^2 and number of degrees of freedom. 064-065
2. Filtering (repeats for every track estimate): 067-154
   - Extrapolate the track estimate $\mathbf{m}_k$ to z_v^0. 071
   - Calculation of χ^2-distance of the k-th track estimate from the vertex $\mathbf{r}^0$ (13). 079-085
   - Calculation of parameters of the k-th vertex track a_k^0, b_k^0 (12). 087-095
   - Update the state vector $\mathbf{r}$ with the Kalman filter formalism (11). 098-152
3. Repeat filtering for the next track.

- Make the next iteration of the fitting routine.
- Store $\mathbf{r}$, C. 159-163

A special feature of the presented algorithm is the extrapolation of the track estimates to the vertex linearization point $\mathbf{r}^0$. This approach makes it possible to fit the vertex without including the track parameters into the vertex state vector (11) and to maximally simplify the calculations. In particular, only two operations of division are performed for each track, while in the standard approach [1] two-fold inversion of a 5×5 matrix is required.

The source code of the algorithm is presented in the Appendix. In order to reduce the calculations, all symmetrical matrices are stored as vectors in the low-triangle form $A = \{A_{(00)}, A_{(01)}, A_{(11)}, A_{(20)}, A_{(21)}, A_{(22)}, A_{(30)}, \ldots\}$.

4 Performance

The algorithm has been implemented for the CBM experiment [6].

Table 1 gives the precision of the primary vertex reconstruction obtained from 10000 events of central Au+Au collisions at 25 AGeV. The algorithm provides a very high accuracy: the residuals of the x_v and y_v positions of the primary vertex are less than 1 μm, and the z_v position is reconstructed with an accuracy better than 4 μm. The normalized residuals (pulls) are close to unity. A little increase of the pulls is probably due to including some secondary tracks into the primary vertex fit.

Parameter	Resolution (μm)	Pull
x_v	0.67	1.08
y_v	0.64	1.11
z_v	3.62	1.10

Table 1: Residuals and normalised residuals (pulls) of the primary vertex parameters obtained from 10000 events of central Au+Au collisions at 25 AGeV

The total number of reconstructed tracks used in the primary vertex fitting routine (Fig. 1) is quite large. In order to investigate the dependence of the vertex resolution on this number, the primary vertex fitting routine has been applied to smaller sub-sets of tracks (Fig. 2). Such $\sim 1/\sqrt{N}$ type behavior allows a significant speedup of the fitting routine in case the maximal precision of the primary vertex is not necessary. This will be especially important for online event selection where time consumptions are crucial.

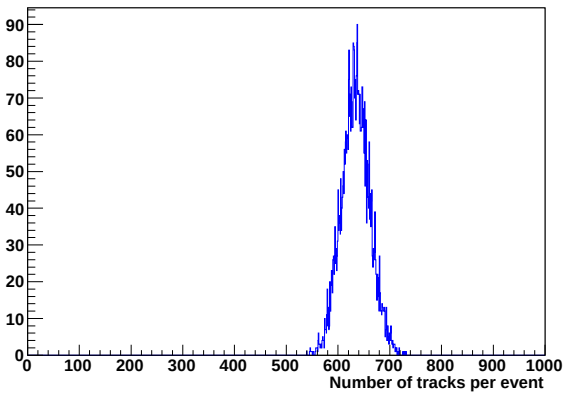

Figure 1: Number of reconstructed tracks per event used by the primary vertex fit algorithm for central Au+Au collisions at 25 AGeV

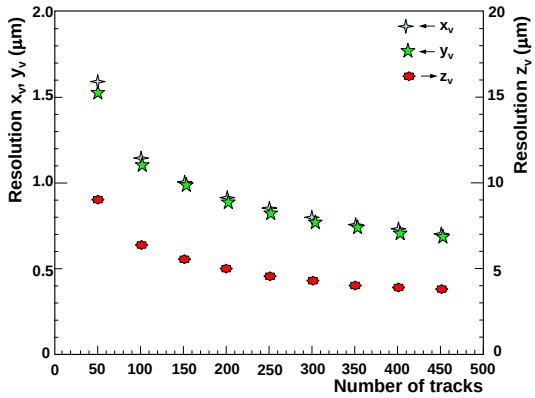

Figure 2: Primary vertex position resolutions versus number of tracks used in the primary vertex fit (the scale for z_v is shown on the right side)

5 Conclusion

A primary vertex reconstruction routine based on the Kalman filter method has been developed for the CBM experiment. The algorithm shows a high accuracy and is suitable for further implementation at the trigger level.

6 Acknowledgements

We acknowledge the support of the European Community-Research Infrastructure Activity under the FP6 "Structuring the European Research Area" programme (HadronPhysics, contract number RII3-CT-2004-506078).

References

[1] R. Frühwirth et al., Data analysis techniques for high-energy physics, Second edition, Cambridge Univ. Press (2000).

[2] P. Billoir and S. Qian, Fast vertex fitting with a local parametrization of tracks, Nucl. Instr. and Meth. A311 (1992) 139-150.

[3] R. Frühwirth, P. Kubinec, W. Mitaroff and M. Regler, Vertex reconstruction and track bundling at the LEP collider using robust algorithms, Comp. Phys. Commun. 96 (1996) 189-208.

[4] D. Emeliyanov, I. Kisel, S. Masciocchi, M. Sang and Yu. Vassiliev, Primary vertex reconstruction by Rover, HERA-B note 00-139, 2000.

[5] S. Gorbunov and I. Kisel, An analytic formula for track extrapolation in an inhomogeneous magnetic field, CBM-SOFT-note-2005-001, GSI, Darmstadt, 18 March 2005.

[6] CBM Collaboration, Compressed Baryonic Matter Experiment. Technical Status Report, GSI, Darmstadt, 2005, (`http://www.gsi.de/onTEAM/dokumente/public/DOC-2005-Feb-447_e.html`).

Appendix

```
001 /** CbmKF::FitPrimaryVertex routine performs primary vertex fit
002  *
003  * @author S.Gorbunov
004  * @author I.Kisel
005  *
006  * @version 2.0
007  * @since 26.12.05
008  *
009  * Geometrical fit of the primary vertex with the Kalman Filter method.
010  *
011  */
012
013 #include "CbmKF.h"
014 #include "CbmKFTrackInterface.h"
015 #include "CbmKFVertexInterface.h"
016 #include "CbmKFTrack.h"
017
018 ClassImp(CbmKF)
019
020 void CbmKF::FitPrimaryVertex( vector<CbmKFTrackInterface*> &Tracks,
021                               CbmKFVertexInterface &vtx             )
022 {
023   //* Constants
024
025   const Double_t CutChi2  = 3.5*3.5;
026   const Int_t   MaxIter = 3;
027
028   //* Vertex state vector and covariance matrix
029
030   Double_t r[3], *C = vtx.GetCovMatrix();
031
032   //* Initialize the vertex
033
034   for( Int_t i=0; i<6; i++ ) C[i]=0;
035   if( vTargets.empty() ){
036     r[0] = r[1] = r[2] = 0.;
037     C[0] = C[2] = 5.;
038     C[5] = 0.25;
039   }else{
040     CbmKFTube &t = vTargets[0];
041     r[0] = r[1] = 0.;
042     r[2] = t.z;
043     C[0] = C[2] = t.RR/3.5/3.5;
044     C[5] = (t.dz/2/3.5)*(t.dz/2/3.5);
045   }
046
047   //* Iterative fit of the vertex
048
049   for( Int_t iteration = 0; iteration < MaxIter; ++iteration ){
050
051     //* Store the vertex from the previous iteration
052
053     Double_t r0[3], C0[6];
054
```

```
055     for( Int_t i=0; i<3; i++ ) r0[i] = r[i];
056     for( Int_t i=0; i<6; i++ ) C0[i] = C[i];
057
058     //* Initialize the vertex covariance, Chi^2 & NDF
059
060     C[0] = 100.;
061     C[1] =   0.; C[2] = 100.;
062     C[3] =   0.; C[4] =   0.; C[5] = 100.;
063
064     vtx.GetRefNDF()  = -3;
065     vtx.GetRefChi2() =  0.;
066
067     for( vector<CbmKFTrackInterface*>::iterator itr  = Tracks.begin();
068                                                 itr != Tracks.end()  ; ++itr ){
069
070       CbmKFTrack T(**itr);
071       Extrapolate( T, r0[2] );
072
073       Double_t *m = T.GetTrack();
074       Double_t *V = T.GetCovMatrix();
075       Double_t a = 0, b = 0;
076       {
077         Double_t zeta[2] = { r0[0]-m[0],  r0[1]-m[1] };
078
079         //* Check the track Chi^2 deviation from the r0 vertex estimate
080
081         Double_t S[3] = { (C0[2]+V[2]), -(C0[1]+V[1]), (C0[0]+V[0]) };
082         Double_t    s =  S[2]*S[0] - S[1]*S[1];
083         Double_t chi2 =  zeta[0]*zeta[0]*S[0] + 2*zeta[0]*zeta[1]*S[1]
084                                                +   zeta[1]*zeta[1]*S[2];
085         if( chi2 > s*CutChi2 ) continue;
086
087         //* Fit of the vertex track slopes (a,b) to the r0 vertex estimate
088
089         s = V[0]*V[2] - V[1]*V[1];
090         if ( s < 1.E-20 ) continue;
091         s = 1./s;
092         a = m[2] + s*(   (  V[3]*V[2] - V[4]*V[1] )*zeta[0]
093                        + (- V[3]*V[1] + V[4]*V[0] )*zeta[1] );
094         b = m[3] + s*(   (  V[6]*V[2] - V[7]*V[1] )*zeta[0]
095                        + (- V[6]*V[1] + V[7]*V[0] )*zeta[1] );
096       }
097
098       //** Update the vertex (r,C) with the track estimate (m,V) :
099
100       //* Linearized measurement matrix H = { { 1, 0, -a}, { 0, 1, -b} };
101
102       //* Residual zeta (measured - estimated)
103
104       Double_t zeta[2] = { m[0] - ( r[0] - a*(r[2]-r0[2]) ),
105                            m[1] - ( r[1] - b*(r[2]-r0[2]) ) };
106       //* CHt = CH'
107
108       Double_t CHt[3][2] = { { C[0] - a*C[3], C[1] - b*C[3]},
109                              { C[1] - a*C[4], C[2] - b*C[4]},
110                              { C[3] - a*C[5], C[4] - b*C[5]} };
```

```
      //* S = (H*C*H' + V )^{-1}

      Double_t S[3] = { V[0] + CHt[0][0] - a*CHt[2][0],
                        V[1] + CHt[1][0] - b*CHt[2][0],
                        V[2] + CHt[1][1] - b*CHt[2][1] };
      //* Invert S
      {
        Double_t s = S[0]*S[2] - S[1]*S[1];
        if ( s < 1.E-20 ) continue;
        s = 1./s;
        Double_t S0 = S[0];
        S[0] =  s*S[2];
        S[1] = -s*S[1];
        S[2] =  s*S0;
      }

      //* Calculate Chi^2

      vtx.GetRefChi2()+=  zeta[0]*zeta[0]*S[0] + 2*zeta[0]*zeta[1]*S[1]
                                               +   zeta[1]*zeta[1]*S[2];
      vtx.GetRefNDF() += 2;

      //* Kalman gain K = CH'*S

      Double_t K[3][2];

      for( Int_t i=0; i<3; ++i ){  K[i][0] = CHt[i][0]*S[0] + CHt[i][1]*S[1] ;
                                   K[i][1] = CHt[i][0]*S[1] + CHt[i][1]*S[2] ; }

      //* New estimation of the vertex position r += K*zeta

      for( Int_t i=0; i<3; ++i ) r[i]+= K[i][0]*zeta[0] + K[i][1]*zeta[1];

      //* New covariance matrix C -= K*(CH')'

      C[0] -= K[0][0]*CHt[0][0] + K[0][1]*CHt[0][1];
      C[1] -= K[1][0]*CHt[0][0] + K[1][1]*CHt[0][1];
      C[2] -= K[1][0]*CHt[1][0] + K[1][1]*CHt[1][1];
      C[3] -= K[2][0]*CHt[0][0] + K[2][1]*CHt[0][1];
      C[4] -= K[2][0]*CHt[1][0] + K[2][1]*CHt[1][1];
      C[5] -= K[2][0]*CHt[2][0] + K[2][1]*CHt[2][1];

    }//* itr

  }//* end of iterations

  //* Copy the state vector to the output
  //* ( the covariance matrix has been used directly )

  vtx.GetRefX() = r[0];
  vtx.GetRefY() = r[1];
  vtx.GetRefZ() = r[2];
}
```