VKalVrt - package for vertex reconstruction in ATLAS.

V. Kostioukhine *
CPPM, IN2P3-CNRS, Marseille, France

Abstract

A universal vertex reconstruction package VKalVrt is presented. It allows the use of different constraints, the reconstruction of cascade decays and has the possibility to use robust versions of the fitting functional to reduce the sensitivity to outlying measurements. Based on the Kalman filter approach the package is fast enough to deal with hundreds of tracks. Descriptions of basic features, used algorithms, calling sequences are given. Some results obtained with VKalVrt package in the primary and secondary vertex reconstruction are presented.

Draft 1.0

*On leave of absence from IHEP (Protvino, Russia)
1 Introduction

The vertex reconstruction is an important problem for high energy collider experiments. It has an impact on flavour physics, b-tagging performance, lifetime measurements. Due to a high energy of collision at LHC a multiplicity of tracks may reach several hundreds, so vertex reconstruction program must be fast. Also a possibility to reconstruct a cascade decay chain with unknown apriori topology is highly desirable. To be as precise as possible the vertex fitting procedure must use full information from track reconstruction (3D particles trajectories with complete error matrices).

The VKalVrt vertex fitting package is based on fast Kalman filtering method. The package allows fast and easy implementation of any type of the constraints for vertex fit in addition to already implemented ones. Charged and neutral tracks can be treated simultaneously. Cascade decays can be reconstructed consecutively. A possibility to use nonlinear robust fitting functionals is added by using a reweighting of track error matrices in each step of the fit. The package is written in FORTRAN and can be used in ATLSIM or as a standalone tool for CBNT analysis now. An interface to ATHENA framework is under construction.

At the beginning of the paper a simple way to reconstruct a common 3D intersection point of the selected tracks using VKalVrt package is described. This example can be a good starting point for using VKalVrt. Then basic features of the package with descriptions of used algorithms are presented. An information about package availability and usage is given at the end. And in Appendix B a complete description of package subroutines with parameters definition is presented.

2 How to start

In the simplest (but practical) case user needs to select a set of tracks and calculates a common vertex for these tracks. A minimal calling sequence to do this with VKalVrt is:

```c
real apar(5,NxTrk),aerr (15,NxTrk),par0(3,NxTrk),parfs(3,NxTrk),chi2tr (NxTrk)
real xyz0(3),vrtfit(3),pbot(3),covf(21),vrte(6),chi2
integer ich(NxTrk),iflag,icovv0,ierr,ntrk

Track selection code is here

-------------------------------------------------------------
do i=1,ntrk
    call VKalTrkPar(aOvert(i),zvert(i),CotThVert(i),phiVert(i),
                   ptinver(i),covert11(i),covert21(i),covert22(i),covert31(i),
                   convert32(i),convert33(i),convert41(i),convert42(i),convert43(i),
                   convert54(i),convert51(i),convert52(i),convert53(i),convert54(i),
                   convert55(i),ich(i),apar(i,i),aerr(i,i))
endo

EMAG=2.083 ! Field in the center of ATLAS
call PFITF(0,0,EMAG,vrt,e,0)
call CPPFEST(ntrk,xym,ich,apar,par0)
call CFITX(ich,icovv0,ntrk,ich,xym,par0,apar,aerr,
                   xyzfit,parfs,pbot,covf,chi2,chi2tr,ierr)
```

2
Track parameters in VKalVrt don’t coincide with those in ATLAS CBNT's (see Appendix A for the details) so VKalTrkPar makes a conversion from CBNT to VKalVrt representation for selected sample of tracks. Output parameters of the VKalTrkPar are ICH - charge, APAR(5) · track parameters, AERR(15) · error matrix of the track parameters. PRCFIT subroutine initializes the parameters needed for the fit and for using constraints. BMAG is a magnetic field value in Tesla and must have nonzero value. For ATLAS it is equal to 2.083 · magnetic field value at the center of inner detector. Assuming that XYZ0(3) is a starting point of the fit CFPEST subroutine does an estimation of tracks parameters at this point needed for the fit. And CFTT does the fit itself. A full description of the parameters for each subroutine is in Appendix B.

As a results of the calling sequence presented above a user obtains

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZFIT(3)</td>
<td>X, Y, Z coordinates of fitted vertex</td>
</tr>
<tr>
<td>PARFS(31...NTRK)</td>
<td>fitted parameters for each track</td>
</tr>
<tr>
<td>COVF(21)</td>
<td>error matrix for (X,Y,Z)<em>{fitted} or (X,Y,Z,Px,Py,Pz)</em>{fitted}</td>
</tr>
<tr>
<td>PTO/T(5)</td>
<td>momentum of the particle decayed at vertex</td>
</tr>
<tr>
<td>CHI2</td>
<td>(\chi^2) of the fit</td>
</tr>
<tr>
<td>CHI2TR(1...NTRK)</td>
<td>(\chi^2) for each track</td>
</tr>
</tbody>
</table>

For each track CFIT gives the fitted parameters \((\Theta, \phi, \frac{1}{R})\) at estimated vertex position where \(R^1\) is a radius of curvature of the track. A particle momentum can be calculated from this parameters in the following way

\[
P_{\perp} = 0.0029979246 \times B_{MAG} \times R
\]

\[
P_{x} = P_{\perp} \cdot \cos \phi
\]

\[
P_{y} = P_{\perp} \cdot \sin \phi
\]

\[
P_{z} = \frac{P_{\perp}}{\tan \Theta}
\]

This momentum doesn’t coincide with momentum calculated at the position of the closest approach of the trajectory to the beam as it is written in CBNT because of change of particle direction due to magnetic field on the way from beam position to fitted vertex. Moreover even an absolute value of momentum is different because track curvature obtained after application of vertex constraint (output of VKalVrt) is different from the initial one. VKalVrt gives improved track parameters (track momentum) at fitted vertex position. To obtain the improved track parameter at any other point along trajectory one may use CFNEWP subroutine from package (see Appendix B).

Usually these four subroutines provide enough functionality for simple vertex reconstruction like \(V^0\) decay search or \(\gamma \rightarrow ee\) conversion search, simple primary vertex reconstruction. More complex example - cascade decay of \(B_s \rightarrow J/\psi(\rightarrow \mu\mu)\phi(\rightarrow K^+K^-)\) reconstruction can be found in Appendix C.

3 Algorithm

A goal of the full 3-dimensional vertex fit is to obtain the vertex position and track momenta at the vertex for all tracks participating in the fit as well as the corresponding error matrices.

\(^1\)For charged particles. For neutral particles see Appendix A
The input information for the fit consists of 5 parameter particle trajectory parametrization \( q_i = \{ \delta z, \delta Z, \Theta, \phi_i, \frac{1}{\rho} \} \) for 3D curved track in the magnetic field and its weight matrix\(^2 \) \( W_i \) at the reference point for each track. This helix "perige" parametrization of the charged particle trajectory was described in [1, 3] and Appendix A. This parametrization allows a simple extrapolation of particle trajectory to the vertex region in any part of the detector. If one assumes that the particle is born at the vertex \( \vec{V} \) then the trajectory parameters \( q_i \) can be expressed as a function of particle origin vertex \( \vec{V} \) and particle momentum at this vertex \( \vec{p}_i \): \( q'_i = T(\vec{V}, \vec{p}_i) \). A vertex fitting functional is then

\[
\chi^2 = \sum_{i=1}^{N_{tracks}} (q_i - T(\vec{V}, \vec{p}_i))^T W_i (q_i - T(\vec{V}, \vec{p}_i))
\]

(1)

here \( W_i \) is 5 \times 5 weight matrix from track fit. Then the goal of the fit is to find \( \vec{V} \) and \( \vec{p}_i \) which minimize (1). \( \vec{p}_i \) is the particle momentum at the vertex which is improved in comparison with the momentum obtained from the track reconstruction results \( q_i \) only.

This \( \chi^2 \) (1) can be linearized at some convenient point close to the vertex

\[
\chi^2 = \sum_{i=1}^{N_{tracks}} (\delta q_i - D_i \delta \vec{V} - E_i \delta \vec{p}_i)^T W_i (\delta q_i - D_i \delta \vec{V} - E_i \delta \vec{p}_i)
\]

(2)

here \( D_i = \frac{\partial T(\vec{V}, \vec{p}_i)}{\partial \vec{V}} \) and \( E_i = \frac{\partial T(\vec{V}, \vec{p}_i)}{\partial \vec{p}_i} \) are matrices of derivatives. The exact expressions for derivative matrices can be found in [3].

A fast method to find a solution of the minimization problem for (2) based on a specific matrix structure of (2) was proposed in [1, 3]. One may demonstrate that this method is completely the same as can be obtained using a Kalman filter based approach\(^2\). In this approach a step of Kalman filter is a recalculation of already found vertex position due to one track addition. The Kalman filter system equation in this case is

\[
\vec{V}_i = \vec{V}_{i-1}
\]

(3)

and the measurement equation is

\[
q_i = T(\vec{V}_i, \vec{p}_i) + \epsilon_i = c_{0i} + D_i \vec{V}_i + E_i \vec{p}_i + \epsilon_i
\]

(4)

near some point of linearization. \( \epsilon_i \) here is a process noise. Kalman filter method with (3),(4) and the "fast" method for finding the solution of (2) give exactly the same formulas for vertex position, updated track parameters and error matrices. One may use for calculation any of the two methods which is more convenient for given problem.

If initial estimation of vertex position (point of functional linearization) is far from the fitted vertex, the track parameters should be extrapolated to the fitted point, all derivatives recalculated and fitting procedure repeated. The track parameters error matrix should also

\(^2\)5-element vector \( apar(..., i) \) in the example above

\(^3\)15-element vector \( aerr(..., i) \) in the example above gives a track error matrix - the inverse of \( W_i \)
be translated to the new vertex position. Current version of VKalVrt uses a simple extrapolation method for charged particle in constant magnetic field without multiple scattering for error matrix translation. Particle trajectory can be calculated both in constant magnetic field approximation and using an exact ATLAS magnetic field.

4 Constraints

The constraints are included into the program by using a Lagrange multipliers method. A constraint itself is a function

\[ A_j(\vec{V}, \vec{p}_1, ..., \vec{p}_n) = \text{const} \]  

(5)

and is added to the fitting functional as

\[ \chi^2 = \chi_0^2 + \sum_{j=1}^{N_{\text{const}}} \lambda_j \cdot A_j^2 \]  

(6)

Here \( \chi_0^2 \) is a usual vertex fitting functional (2) without constraints, \( \lambda_j \) is a Lagrange multiplier and \( j \) is a constraint number. \( A_j^2(\ldots) \) can be linearized around some point \((\vec{V}_0, \vec{p}_0)\) so one has

\[ \chi^2 = \chi_0^2 + \sum_{j=1}^{N_{\text{const}}} \lambda_j \cdot (A_j^2 = \chi_0^2 + \sum_{j=1}^{N_{\text{const}}} \lambda_j \cdot (A_j^0 + H_j^T \delta V + \delta V^T H_j + F_{ij}^T \delta p_i + \delta p_i^T F_{ij}) \]  

(7)

where \( H_j = \frac{\partial A_j}{\partial \vec{V}} \), \( F_{ij} = \frac{\partial A_j}{\partial \vec{p}_i} \). \( A_j^0 \) is an exact value of \( A_j \) at the \((\vec{V}_0, \vec{p}_0)\) point, \( \delta \vec{V} = \vec{V} - \vec{V}_0 \) and \( \delta \vec{p}_i = \vec{p}_i - \vec{p}_0 \).

A solution of (7) can be obtained in the following way \( \vec{V} = \vec{V}_0 + \vec{V}_1, \vec{p}_i = \vec{p}_0 + \vec{p}_1i \), where \( \vec{V}_0, \vec{p}_0i \) is a solution of the corresponding problem without constraints \( \chi^2 = \chi_0^2 \). Then a second part \( \vec{V}_1 \) of the solution with constraints can be obtained in the following way

\[ \vec{V}_1 = -\sum_{j=1}^{N_{\text{const}}} \lambda_j \cdot C_{\text{cov}} \cdot L_j \]

\[ \lambda_j = M_j \cdot \left( \sum_i F_{ij}^T C_i^{-1} F_{ij} + L_j^T \cdot C_{\text{cov}} \cdot L_j \right)^{-1} \]

\[ L_j = \sum_i B_i C_i^{-1} F_{ij} - H_j \]

\[ M_j = A_j^2 - H_j^T \cdot \vec{V}_0 - \sum_i F_{ij}^T \vec{p}_0i \]

\[ C_{\text{cov}} = (K - \sum_i B_i C_i^{-1} B_i^T)^{-1} \]

\[ K = \sum_i D_i^T W_i D_i, \quad B_i = D_i^T W_i E_i, \quad C_i = E_i^T W_i E_i \]

The corrections to tracks momenta \( \vec{p}_1i \) can be obtained in the same way.

Currently 6 different constraints are implemented and can be specified in the input to CFIT.
<table>
<thead>
<tr>
<th>Description of the constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Invariant mass of particle decayed at the fitted vertex or mass of any $m$-particles</td>
</tr>
<tr>
<td>subsystem out of $N$ produced at the vertex particles is equal to $M$</td>
</tr>
<tr>
<td>2 Particle decayed at the fitted vertex come from given vertex $V_0$</td>
</tr>
<tr>
<td>3 Polar angle $\Theta$ of momentum of particle decayed at the fitted vertex is equal to the</td>
</tr>
<tr>
<td>polar angle $\Theta$ of $V_{fitted} - V_0$ where $V_0$ is a given (e.g. primary) vertex. This</td>
</tr>
<tr>
<td>means that particle trajectory is passing through given vertex in $Z - R$ plane but it is</td>
</tr>
<tr>
<td>free in $R - \phi$ plane. This constraint is useful when track measurement errors in $Z</td>
</tr>
<tr>
<td>direction are much higher than errors in $R - \phi$ plane.</td>
</tr>
<tr>
<td>4 Combination of the constraints 1 and 2</td>
</tr>
<tr>
<td>5 Combination of the constraints 1 and 3</td>
</tr>
<tr>
<td>6 Fitted vertex should be close to the given one (e.g. primary) $V_0$</td>
</tr>
</tbody>
</table>

The constraint 6 is not a real one because it is implemented as a $\chi^2$ type term in the fitting functional. It is called "constraint" following a common practice.

Any other constraint which can be expressed as a matrix of derivatives at the vertex position may be easily implemented in the code of the VKalVrt.

5 Robust functionals

The estimation of the vertex can be made less sensitive to the presence of the tracks from other vertices in the selected set of tracks by using a more robust version of the Kalman filter. An M-estimator method is used in VKalVrt. The basic idea of the method is a modification of the objective function such that outliers (measurements with big residuals) have less influence on the estimate. This can be achieved by downweighting the tracks with big residuals. Usually $\chi^2 = \sum_{i=1}^{N} \frac{x_i^2}{2}$ is substituted by $\chi^2_{\text{robust}} = \sum_{i=1}^{N} \Psi(x_i)$ with special selection of function $\Psi$.

The residuals can be calculated only after the vertex position estimation so procedure must be iterative. After successful fit the track error matrices should be scaled down according to obtained residuals and vertex position estimation procedure should be repeated. The iterations stop when the vertex position is stable.

Currently in addition to standard "$\chi^2$ type" functional the six robust functionals[5] are implemented in VKalVrt. They are listed in table below. For the first three of them ($IROB = 1 \div 3$) the outliers weights go to zero so that they don’t contribute at all to the solution. The last 3 functionals ($IROB = 4 \div 6$) limit the influence of the outliers by limiting the corresponding derivatives[5].
<table>
<thead>
<tr>
<th>IROB</th>
<th>Function $\Psi(x)$</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Standard $\chi^2 = \frac{x^2}{2}$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Geman-McClure function $\frac{x^2/2}{1 + x^2}$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Welsch function $\frac{c^2}{2} \left(1 - \exp\left(-\frac{x^2}{c^2}\right)\right)$</td>
<td>2.9846</td>
</tr>
<tr>
<td>3</td>
<td>Cauchy function $\frac{c^2}{2} \cdot \log \left(1 + \frac{x^2}{c^2}\right)$</td>
<td>2.3849</td>
</tr>
<tr>
<td>4</td>
<td>L1-L2 function $2 \cdot \left(\sqrt{1 + \frac{x^2}{2}} - 1\right)$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Fair function $c^2 \cdot \left(\frac{</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>Huber function $\frac{x^2}{2}$ if $</td>
<td>x</td>
</tr>
</tbody>
</table>

In all functionals in the table $x = \frac{z_{\text{meas}} - z_{\text{fit}}}{\sigma}$ and $c$ is a tuning constant selected so that functional has 95% efficiency\(^4\) on normally (without outliers) distributed data.

All robust functionals are constructed so that $\lim \Psi(x) = \frac{x^2}{2} = \chi^2$ what means that near minimum they behave exactly like usual $\chi^2$ functional. VKalVrt package returns as "$\chi^2$ of the solution" a value of functional at the minimum. Then for the robust functional the "$\chi^2$ of the solution" is a usual $\chi^2$ at small values and some monotonic function of the solution quality at high values. Monotony of the "robust $\chi^2$" means that it can be used for quality cut of $\chi^2 < A$ type. The only problem is the cut constant $A$ which must be selected empirically and not based on usual "probability less than 1(0.1..%)%" criterion.

6 Magnetic field

VKalVrt has a possibility to use an exact nonuniform magnetic field of ATLAS detector in vertex search. The tracks are translated to the approximate vertex position by using a modified GEANT3 subroutine GRKUTA [6] and a 3D map of ATLAS magnetic field. VKalVrt has two options:

1. 3D table of ATLAS magnetic field is taken from single file\(^5\) (fast option for standalone CBNT analysis);

2. A complete ATLAS magnetic field description package is used[10].

\(^4\)Efficiency here is defined as a ratio $\frac{\sigma^2_{\text{rob,x}}}{\sigma^2_{\text{obs,x}}}$

\(^5\)This file can be received from author
At the new point the tracks derivatives are calculated using only Z component of the magnetic field which is dominant everywhere inside ATLAS inner detector. An accuracy of the derivatives obtained is enough for precise vertex position estimation. Due to speed reason a default mode in VKalVrt is a work with constant magnetic field. More information can be found in Appendix B.

7 Where to find VKalVrt

VKalVrt package is available on CVS in offline/atrecon/VKalVrt. The include files are in offline/atrecon/include/atrecon/vkalcdes. Also on CVS in the directory offline/atrecon/VKalVrt/examples one may find some examples of using VKalVrt. Corresponding explanations may be found in README file available there. Currently there are 3 different examples:

- Using of VKalVrt in ATLSIM environment for primary and inclusive secondary vertices reconstruction;
- Using of VKalVrt in standalone mode for CBNT analysis with the same functionality as above;
- Using of VKalVrt also in standalone mode for $B_s \rightarrow J/\psi(\rightarrow \mu^+\mu^-)\phi(\rightarrow K^+K^-)$ decay reconstruction (CBNT based).

7.1 Example: using of VKalVrt in ATLSIM

ATLSIM version of VKalVrt provides the following functionality:

- Reconstruction of primary interaction and pile-up vertices. Selection is based on the reconstructed information only. No MC information is used. See [7] for the details of the algorithm.
- Reconstruction of common secondary vertices inside jets (either ATLFAST or reconstructed in calorimeter).
- Creation of the KVRTPRIM block in standard CBNT with vertices from VKalVrt and xKalman.

By default VKalVrt uses calorimeter based jets for secondary vertices reconstruction. If such jets are absent VKalVrt uses ATLFAST jets. For the moment VKalVrt tries to reconstruct one inclusive secondary vertex inside jet (b-tagging inspired mode of work) and provide information needed for vertex based b-tagging for each found vertex. VKalVrt has the following data cards for CBNT module:

```
Basic     =1(def) write vertexing information
NVertF    =10(def) maximum number of vertices
VKalV     =1(def) write VKal vertices
XKalV     =1(def) write XKalman vertices
```
The following information is present in KVRTPIM block in CBNT.

- IVTYP =1 VKalVrt primary or pileup vertex (Primary is the first in the list, the rest are pileups)
- IVTYP =2 Xkalman vertex
- IVTYP =11 VKalVrt secondary vertex
- VRTPX - X coordinate of vertex
- VRTPY - Y coordinate of vertex
- VRTPZ - Z coordinate of vertex
- VPCOXX - error matrix
- VPCOXY - error matrix
- VPCOZZ - error matrix
- VRTPT - Sum of all vertex track Pt if IVTYP≠11
- VRTNT - Invariant mass of all tracks in vertex if IVTYP=11
- VRATIO =0 Number of tracks in vertex
  \[ \frac{E_{\text{vertex}}}{E_{\text{tracks}}} \]
  for IVTYP=1,2
  \[ \text{jet number} \]
  for IVTYP=11
- JETVRT =0 for IVTYP=1,2
- JETVRT =jet number from JETSINFO block for IVTYP=11

Calling of VKalVrt from ATLSIM is illustrated in XKAL.KUMAC file available in offline/atrecon/VKalVrt/examples. In this example ATLFAST jets are used for secondary vertices search. To produce them one needs to call ATLFASTFROMATRECON(0) befor e(!!!) calling VKalVrt. In this example the reconstructed information is written in CBNT and can be used for vertex based b-tagging. The corresponding explanations are in README file.

### 7.2 Example: standalone VKalVrt

![Histograms](image)

Figure 1: **Accuracy of primary vertex reconstruction obtained for WH(400 GeV) events [7]**

Two examples of VKalVrt usage for CBNT analysis are provided with the package. One example is a functional analog of the ATLSIM example and reconstructs primary vertices and
inclusive secondary vertices inside ATLFAST jets based on the information written in CBNT. The similar code was used in [7, 8, 9] for secondary vertex based b-tagging studies. The quality of reconstruction is illustrated in fig.1,2.

Second example is a program for a cascade \( B_s \rightarrow J/\psi(\rightarrow \mu^+ \mu^-) \phi(\rightarrow K^+ K^-) \) decay reconstruction with constraints. More details and explanations about this example can be found in Appendix C. Both examples can be compiled and linked by \textit{make examp} (\textit{make examp1}). After running of the examples some histograms will be present in vertex.his PAW histogram file.

![Histogram](image)

Figure 2: \( \pi^+ \pi^- \) invariant mass distribution obtained during inclusive secondary vertices reconstruction in WH(400 GeV) events [7]

8 Conclusions

A universal package VKalVrt for vertex reconstruction in ATLAS is presented. It can be used for primary and secondary vertices reconstruction with application of different constraints and robust versions of fitting functionals. Code is fast enough to deal with many tracks vertex reconstruction problem. The primary vertex reconstruction takes \( \sim 1 \, ms \) for WH(400 GeV) event and \( \sim 3 \, ms \) for complete primary+pileup vertices reconstruction at Pentium III (1000 MHz). The VKalVrt package provides many possibilities for different physics studies: different error matrices calculations for resolution studies, track extrapolation from/to any point, track impact parameters calculation with errors with respect to any point.

Package is available in CVS together with some working examples illustrating its use for primary and secondary vertex reconstruction. It was successfully used for b-tagging with secondary vertices studies [8, 9].
References


[6] "GEANT Detector Description and Simulation Tool" *W5013 CERN Program Library Long Writeup*


[9] V. Kostiukhine, J. Levêque, A. Rozanov and J.B. de Vivie "Physics impact of the change of the pixel b-layer longitudinal pitch from 300 to 400 microns" *ATL-INDET-2003-016*

[10] [http://atlas.web.cern.ch/Atlas/groups/muon/magfield/mag_page1.html](http://atlas.web.cern.ch/Atlas/groups/muon/magfield/mag_page1.html) and references on this page
Appendix A: Track parametrization

VKalVrt uses the following 6 parameters taken at the point of the closest approach to Z-axis for track description:

1. \( \delta e \) - transverse (X,Y) impact parameter;
2. \( \delta Z \) - Z coordinate;
3. \( \Theta \) - polar angle of the trajectory with respect to Z-axis;
4. \( \phi \) - azimuthal angle of the trajectory;
5. \(-Q/R\) - curvature of track with sign, determined by charge;
6. \( Q \) - charge.

Usually first 5 parameters are kept in array with 5 elements inside VKalVrt and charge is treated separately. The special charge variable is needed only to distinguish neutral tracks from charged because charge information is present in the sign of the curvature for charged tracks. The corresponding error matrix is symmetric and is kept in array with 15 elements. Radius of track is connected with track \( P_1 \) by the following formula \( \frac{1}{R} = \frac{0.0029979246 \cdot B_{MAG}}{P_1} \). Magnetic field here should be expressed in Tesla. For the case of neutral track identified by \( Q = 0 \) fifth parameter is defined as \( \frac{1}{R} = \frac{0.0029979246 \cdot B_{MAG}}{P_\perp} \) also with the only difference that \( R \) is always positive. The "curvature" \( \frac{1}{R} \) for neutral track can't be simply put to 0 because usually such track is produced as a summed track after multiparticle decay vertex fit (e.g. \( J/\psi \) track after \( \mu^+\mu^- \) vertex fit). For such track the momentum is well defined and strongly correlated with other track parameters. For correct using of reconstructed neutral track in subsequent vertex fits with constraints (e.g. cascade decay reconstruction or primary vertex fit) full information about the track must be kept. This is done by using of nonzero "curvature" for neutral track.

Error matrix for these 5 track parameters (except for charge) is kept in symmetric form as an array with 15 elements inside VKalVrt.

The track parameters and error matrix can be obtained from particle momentum and known particle production vertex with XYZTRP subroutine (see Appendix B).

A particle momentum can be calculated from track parameters in the following way:

\[
P_\perp = 0.0029979246 \cdot B_{MAG} \cdot R
\]

\[
P_x = P_\perp \cdot \cos \phi
\]

\[
P_y = P_\perp \cdot \sin \phi
\]

\[
P_z = \frac{P_\perp}{\tan \Theta}
\]
Appendix B: Package subroutines

Everywhere in this appendix (REAL*4)I means an input parameter of REAL*4 type and (REAL*4)O means an output parameter of REAL*4 type.

Subroutine VKalTrkPar(a0vert(i), zvert(i), CotThVert(i), phivert(i), @ ptinvvert(i), covert11(i), covert21(i), covert22(i), covert31(i), @ covert32(i), covert33(i), covert41(i), covert42(i), covert43(i), @ covert44(i), covert51(i), covert52(i), covert53(i), covert54(i), @ covert55(i), ich, apar, aerr)

VKalTrkPar is a subroutine for conversion of track parameters and error matrix from usual ATLAS to VKalVrt representation.

ICH track charge (Integer) O
APAR(5) VKalVrt track parameters (see Appendix A) (Real*4) O
AERR(15) track error matrix (Real*4) O

Another parameters are the usual ATLAS track parametrization (see ATLAS CBNT description http://droussea.home.cern.ch/droussea/cbnt/cbnt.html)

Subroutine PRCFIT(ntrk, wm, wmfit, BMAG, vrt, vrtE, irob)

Subroutine PRCFIT sets up different parameters needed for vertex reconstruction and calculation of the constraints used. In general the parameters not used in the constraints may have arbitrary values. This is a user responsibility to guarantee the correct values of the parameters which are really used!

NTRK number of tracks (Integer) I
WM(1:NTRK) particle masses (if < 0 - track doesn't participate in mass constraint) (Real*4) I
WMFIT mass for mass constraint (Real*4) I
BMAG magnetic field (Tesla). Must be present in all cases. Default in ATLAS is 2.083 (Real*4) I
VRT(3) point for directional constraint (usually primary vertex) (Real*4) I
VRTE(6) error matrix for VRT (for IFLAG=6) (Real*4) I
ILOB Robustness 0 = $\chi^2$, > 0 - robust functional (see Section 5 for details) (Integer) I

Subroutine CFPEST(ntrk, vrt, ich, par, par0)

Subroutine CFPEST gives a primary estimation of track parameters at the starting point of the fit

NTRK number of tracks (Integer) I
VRT(3) initial estimation of vertex position (Real*4) I
ICH(1:NTRK) particle charges (Integer) I
PAR(5,1:NTRK) track parameters (Real*4) I
PAR0(3,1:NTRK) estimations of track parameters in vertex VRT(3) (Real*4) O
Subroutine CFIT(iflag, ifcovv0, ntrk, ich, xyz0, par0, apar, aerr, @ xyzfit, parfs, ptot, covf, chi2, chi2tr, ierr)

*CFIT* is a main subroutine of the package. It calculates an estimation of common vertex position of the track sample with different constraints and fitting algorithms. Subroutine also calculates the track parameters constrained on passing through found vertex and different errors. Special argument IFCOVV0 is used to select a type of the returned error matrix because calculation of (X,Y,X,Px,Py,Pz) error matrix for the large number of participating tracks (primary vertex reconstruction ...) is time consuming and generally not needed. Error matrix for (X,Y,Z) is usually enough in most cases.

**IFLAG** type of the fit (Integer) I

<table>
<thead>
<tr>
<th>Value</th>
<th>Action</th>
<th>Required information in PRCFIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>simple vertex fit</td>
<td>WMFIT, WM(1:NTRK)</td>
</tr>
<tr>
<td>1</td>
<td>vertex fit with mass constraint</td>
<td>VRT(3)</td>
</tr>
<tr>
<td>2</td>
<td>V0 track pass through primary vertex</td>
<td>VRT(3)</td>
</tr>
<tr>
<td>3</td>
<td>Z constraint</td>
<td>WMFIT, WM(1:NTRK), VRT(3)</td>
</tr>
<tr>
<td>4</td>
<td>1-2</td>
<td>WMFIT, WM(1:NTRK), VRT(3)</td>
</tr>
<tr>
<td>5</td>
<td>1-3</td>
<td>WMFIT, WM(1:NTRK), VRT(3)</td>
</tr>
<tr>
<td>6</td>
<td>Fitted vertex should be close to given one</td>
<td>VRT(3), VRTE(6)</td>
</tr>
</tbody>
</table>

**IFCOVV0** type of the returned error matrix (Integer) I

<table>
<thead>
<tr>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Only vertex (X,Y,Z) error matrix (6 par)</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>Full (X,Y,Z,Px,Py,Pz) error matrix (21 par)</td>
</tr>
</tbody>
</table>

**NTRK** number of tracks (Integer) I

**ICH(1:NTRK)** particle charges (Integer) I

**XYZ0(3)** starting vertex position (Real*4) I

**PAR0(3, 1:NTRK)** estimation of track parameter at vertex XYZ0 (Real*4) I

**APAR(6, 1:NTRK)** track parameters (Real*4) I

**AERR(5, 1:NTRK)** error matrix for track parameters (Real*4) I

**XYZFIT(3)** fitted vertex position (Real*4) O

**PARFS(3, 1:NTRK)** obtained track parameters at vertex XYZFIT (Real*4) O

**PTOT(3)** obtained total momentum vector (Real*4) O

**COVF(21)** error matrix for (X,Y,Z) or (X,Y,Z,Px,Py,Pz) (Real*4) O

**CHI2** $\chi^2$ of vertex fit (for IROB > 0 see Section 5) (Real*4) O

**CHI2TR(1:NTRK)** Contribution of each track to $\chi^2$ of the fit (Real*4) O

**1ERR**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>O'k</td>
</tr>
<tr>
<td>-1</td>
<td>Bad fit $\chi^2 &gt; 1000000$</td>
</tr>
<tr>
<td>-2</td>
<td>Fit without error matrix</td>
</tr>
<tr>
<td>-3</td>
<td>No fit at all</td>
</tr>
<tr>
<td>-4</td>
<td>Noninvertible error matrix for IFLAG=6</td>
</tr>
</tbody>
</table>
Subroutine XYZTRP(ICH,VRT0,PV0,COV1,PAR,ERR,BMAG,IERR)

Subroutine *XYZTRP* converts the result of the vertex fit (vertex + total momentum) to the standard track parametrization (*ε*,*Z*,*θ*,*φ*,*l*/R). Obtained track with parameters in standard form may be used as input for another vertex fit.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICH</td>
<td>particle charge</td>
<td>Integer 1</td>
</tr>
<tr>
<td>VRT0(3)</td>
<td>particle decay vertex position (X,Y,Z)</td>
<td>Real*4 I</td>
</tr>
<tr>
<td>PV0(3)</td>
<td>particle momentum (Px,Py,Pz)</td>
<td>Real*4 I</td>
</tr>
<tr>
<td>COV1(21)</td>
<td>error matrix for (X,Y,Z,Px,Py,Pz)</td>
<td>Real*4 I</td>
</tr>
<tr>
<td>PAR(6)</td>
<td>track parameters in standard form</td>
<td>Real*4 O</td>
</tr>
<tr>
<td>ERR(15)</td>
<td>track error matrix</td>
<td>Real*4 O</td>
</tr>
<tr>
<td>BMAG</td>
<td>magnetic field (in Tesla)</td>
<td>Real*4 I</td>
</tr>
<tr>
<td>IERR</td>
<td>error flag</td>
<td>Integer O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>O’k</td>
</tr>
<tr>
<td>≠ 0</td>
<td>Wrong error matrix</td>
</tr>
</tbody>
</table>

Subroutine CFNEWP(ICH,PARO,VRT,S,PARN,PER)

Subroutine **CFNEWP** calculates the trajectory parameters with respect to a new reference point.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICH</td>
<td>particle charge</td>
<td>Integer 1</td>
</tr>
<tr>
<td>PARO(5)</td>
<td>track parameters with respect to (0,0,0)</td>
<td>Real*4 I</td>
</tr>
<tr>
<td>VRT(3)</td>
<td>new reference point (vertex)</td>
<td>Real*4 I</td>
</tr>
<tr>
<td>S</td>
<td>curved distance between old and new perigee</td>
<td>Real*4 O</td>
</tr>
<tr>
<td>PARN(5)</td>
<td>track parameters with respect to VRT point</td>
<td>Real*4 O</td>
</tr>
<tr>
<td>PER(3)</td>
<td>new perigee with respect to VRT point</td>
<td>Real*4 O</td>
</tr>
</tbody>
</table>

Subroutine CFERPR(ICH,PAR,VRT,S,ERO,ERR)

Subroutine **CFERPR** calculates the error matrix for new track parameters obtained with **CFNEWP** subroutine with respect to a new reference point.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICH</td>
<td>particle charge</td>
<td>Integer 1</td>
</tr>
<tr>
<td>PAR(5)</td>
<td>track parameters with respect to (0,0,0)</td>
<td>Real*4 I</td>
</tr>
<tr>
<td>VRT(3)</td>
<td>new reference point (vertex)</td>
<td>Real*4 I</td>
</tr>
<tr>
<td>S</td>
<td>curved distance between old and new perigee</td>
<td>Real*4 O</td>
</tr>
<tr>
<td>ERO(15)</td>
<td>error matrix for old track parameters</td>
<td>Double*4 I</td>
</tr>
<tr>
<td>ERR(15)</td>
<td>error matrix for new track parameters</td>
<td>Double*4 O</td>
</tr>
</tbody>
</table>
Subroutine CFIMP(ich,ifl,par,err,vrt,vcov,rmp,rcov,sign)

Subroutine CFIMP calculates 3-dimensional impact parameter of track with respect to any vertex.

ICH  particle charges  (Integer) I
IFL(+1,0,−1)  Add or subtract6 vertex errors contribution to impact errors  (Integer) I
PAR(5)  track parameters in standard form  (Real*4) I
ERR(15)  track error matrix  (Real*4) I
VRT(3)  vertex position (X,Y,Z)  (Real*4) I
VCOV(6)  vertex covariance matrix  (Real*4) I
COV(21)  error matrix for (X,Y,Z,Px,Py,Pz)  (Real*4) I
RIMP(2)  impact parameters in (ΔRxy,Z) form  (Real*4) O
RCOV(15)  error matrix for impact parameters  (Real*4) O
SIGN  3D impact significance from RIMP and RCOV  (Real*4) O

Subroutine FITERM(nt,ermm)

Subroutine FITERM calculates full error matrix for
(X,Y,Z,Px1,Px1,PzΩ,...,PzN,PyN,PzN) with size (3 * Nτ + 3) * (2 * Tτ + 4) / 2 in symmetric form.

NT  number of tracks  (Integer) I
ERMM((3 * NT + 3) * (3 * NT + 4) / 2)  Full error matrix for vertex+tracks  (Real*4) O

Subroutine CALCPTOT(ntrk,parfs,ptot,totmass,mass)

Subroutine CALCPTOT calculates 4-momentum and invariant mass of particle system after vertex fit.

NTRK  number of tracks  (Integer) I
PARFS(3,1:NTRK)  Track parameter after vertex fit  (Real*4) I
PTOT(4)  4-momentum of particle system  (Real*4) O
TOTMASS  Mass of particle system  (Real*4) O
MASS  Mass of particles  (Real*4) I

6Vertex contribution must be added to track errors if track didn’t participate in vertex fit and subtracted if the track was included into the vertex fit. This is true if vertex was fitted with χ² method. For robust functionals situation is more complex.
Subroutine PRCMAG(kmag)

Subroutine \textit{PRCMAG} sets up a mode of magnetic field using in VKalVrt.

\[ K M A G \quad 0(\text{def}) - \text{constant magnetic field}, \ 1 - \text{exact magnetic field} \quad \text{(Integer)} \ i \]

Subroutine VKMAGFLD(x,y,z,fx,fy,fz)

Subroutine \textit{VKMAGFLD} returns an exact value of magnetic field at the given point inside ATLAS detector.

\[
\begin{array}{ll}
X & \text{X coordinate of given point} \\
Y & \text{Y coordinate of given point} \\
Z & \text{Z coordinate of given point} \\
F_X & \text{X component of magnetic field at given point} \quad (k\text{gauss}) \\
F_Y & \text{Y component of magnetic field at given point} \quad (k\text{gauss}) \\
F_Z & \text{Z component of magnetic field at given point} \quad (k\text{gauss})
\end{array}
\]
Appendix C: $B_s \rightarrow J/\psi(\rightarrow \mu^+ \mu^-)\phi(\rightarrow K^+ K^-)$ decay reconstruction

This decay gives 2 cascade secondary vertices. At the first vertex $B_s$ decays into $J/\psi$ and 2 kaons and at the second one $J/\psi$ decays into 2 muons. The cascade $B_s$ decay reconstruction is then a two-step procedure. At the first step one should reconstruct $J/\psi$ decay vertex and obtain a neutral $J/\psi$ track. At the second step one should combine neutral $J/\psi$ track and two charged kaon tracks into one $B_s$ decay vertex. Both $J/\psi$ and $B_s$ vertices reconstruction may be done with mass constraints ($J/\psi, \phi$ masses) and $B_s$ vertex may be reconstructed with directional (primary vertex) constraint.

Let's assume that $k_1,k_2$ are the indices of kaon tracks and $m_1,m_2$ are the indices of muon tracks. The reconstruction starts from $J/\psi$ decay vertex fit with mass constraint.

call VKalTkPar(aOvert(ml),zvert(ml),CotThVert(m1),phivert(m1),
@ ptinvert(m1),convert11(m1),convert21(m1),convert31(m1),
@ convert32(m1),convert33(m1),convert41(m1),convert42(m1),convert43(m1),
@ convert44(m1),convert51(m1),convert52(m1),convert53(m1),convert54(m1),
@ convert55(m1),ich(m1),apar(1,1),aerr(1,1))
call VKalTkPar(aOvert(m2),zvert(m2),CotThVert(m2),phivert(m2),
@ ptinvert(m2),convert11(m2),convert21(m2),convert31(m2),
@ convert32(m2),convert33(m2),convert41(m2),convert42(m2),convert43(m2),
@ convert44(m2),convert51(m2),convert52(m2),convert53(m2),convert54(m2),
@ convert55(m2),ich(m2),apar(1,2),aerr(1,2))

wmfit=3.09688
wm(1)=0.10567
wm(2)=0.10567
call PRCFIT(2,wm,wmfit,BMAG,vtprpm,covprm,0)
call cfpext(2,vt,ich,apar.par0)
call CFIT(1,1,2,ich,vt,par0,apar,aerr,vrtfit,
@ parfs,ptot,covf,chi2res,chi2tr,ierr)

First parameter (=1) in CFIT call tells package to do the fit with mass constraint. To do it one needs the mass value for the constraint ($wmass = 3.09688$ is the mass of $J/\psi$ in this example) and masses for each particle used in the fit ($wm(1),wm(2) = 0.10567$ - muon mass here). As a result one has a reconstructed $J/\psi$ decay vertex. Invariant mass of two muon system with fitted parameters is exactly the $J/\psi$ one. To receive neutral $J/\psi$ track error matrix one needs full ($X,Y,Z,P_x,P_y,P_z$) fitted error matrix so second parameter in CFIT call is also set to 1.

Let's obtain now the neutral $J/\psi$ track parameters PARV0 and errors ERRV0:

call XYZTRP(0,vrtfit,ptot,covf,parv0,errv0,BMAG,ierr)

One can proceed with reconstruction of the 3 track decay vertex $B_s \rightarrow J/\psi K^+ K^-$

call VKalTkPar(aOvert(k1),zvert(k1),CotThVert(k1),phivert(k1),
@ ptinvert(k1),convert11(k1),convert21(k1),convert31(k1),
@ convert32(k1),convert33(k1),convert41(k1),convert42(k1),convert43(k1),
@ convert44(k1),convert51(k1),convert52(k1),convert53(k1),convert54(k1),
@ convert55(k1),ich(k1),apar(1,1),aerr(1,1))
call VKalTkPar(aOvert(k2),zvert(k2),CotThVert(im2),phivert(k2),
Bs decay vertex VRTFIT with covariance matrix COVF is ready. A constraint type 4 was used here what means that Bs track must pass through primary \textit{VRTPRM} vertex and mass of $K^+K^-$ system (1st and 2nd particles in the \textit{CFIT} call) must be equal to the mass of $\phi$ meson 1.0194 GeV/c$^2$. If one doesn’t want to use $\phi$ mass constraint then "\textit{call CFIT}(2,1,2...)" could be used. COVF(21) here is also a full $(V_x,V_y,V_z,P_x,P_y,P_z)$ covariance matrix (second parameter in CFTI is set to one). And if one needs an impact parameters of reconstructed $Bs$ track with respect to the primary vertex

\begin{verbatim}
call XYZIMP(0,vrtfit,ptot,covf,parv0,errv0,EMAG,ierr)
call CFIMP(0,1,parv0,errv0,vrtprm,covprm,RIMP,RCOV,SIGNI)
\end{verbatim}

Description of package subroutines used here and their parameters can be found in Appendix B. Some invariant mass distributions obtained with the code presented above are shown in fig.3.
Figure 3: $B_s \rightarrow J/\psi(\rightarrow \mu^+\mu^-)\phi(\rightarrow K^+K^-)$ vertices reconstruction. a) - $\phi \rightarrow K^+K^-$ decay reconstruction, b) - $J/\psi \rightarrow \mu^+\mu^-$ decay reconstruction, c) - $B_s$ decay reconstruction.