CBNote 285

A New Multi-Vertex Fitter and Updated Vertex-Locater Information

Mark Lakata
October 23, 1995 - Version 1.00
`http://nsdssp.lbl.gov:8000/lakata/tcver3.html`

Abstract

A new vertex fitting package was written to replace the current fitting routines. The new routines do multi-vertex fitting, suited for displaced K_S vertices, by fitting a new helix track to the SVX/PWC/JDC hits, with the constraint that the tracks pass through a common point. The old routines, by contrast, fit only one vertex (if at all) and did not fit the helices to the hits, but rather fitted a vertex to the helices. The new routines offer improved resolution and efficiency, especially in events with shorter tracks and 4-prong vertices.

Contents

1 Introduction

The first generation of vertex fitting was done by two routines: TCVERT and TCVRTX. TCVERT decided which tracks to use in the fit and then called TCVRTX to do the actual fitting. A new version of TCVERT was written, called TCVER2 for use with the kinematic fitter. However, TCVER2 does not make any attempt at finding the best combination of tracks to pair together; it simply makes all $2^N - N - 1$ combinations of N tracks, and passes each combination to TCVRTX for fitting. If the fit converges and passes χ^2 cuts, then the vertex is kept. In an analysis of displaced vertices, such as for $K_S \to \pi^+\pi^-$, one would like to have just one good configuration of pairings.

Thus TCVER3 was born. TCVER3 is best suited for looking for displaced vertices, and returns only those vertices of the best configuration. This allows one to make a cut on the number of prongs at each vertex of an event. For instance, for $p\overline{p} \to K^\pm \pi^\mp K_S, K_S \to \pi^+\pi^-$ typical has two separated vertices, the primary vertex near (0,0,0) and the secondary K_S decay vertex located a few cm away from (0,0,0). In this case, a good event would be called (2,2) since there are two vertices, each with two tracks. This is distinguished from (4) which means one vertex with four tracks. The notation is that of the crystal barrel event display (CBDISP V1.10/07 or newer).

Another serious problem with the old vertex fitter was that the positions of the vertices were only indirectly constrained by the actual raw data hits. Helix tracks were fitted to the raw data hits, resulting in 5-parameter helices. These 5-parameter helices were then used as the raw data input to a 3-parameter vertex fit. The error matrix does not constrain the vertex fit sufficiently enough, and the result is that a parameter such as the initial track direction can be modified by the fit much more than it should be. TCVRHX addresses this problem by fitting the raw data hits directly, with 3-parameter helices that pass through a 3-parameter common point.

The event display (CBDISP) was modified to automatically display multiple vertices. Use `JD:VERT` to display the vertices and their associated tracks. Also, the number of prongs of each vertex is listed, thus "`0,2,2`" would mean that there was one neutral vertex and two 2-prong vertices in the event.

2 1-2-3 Instructions

1. Configure cboff to redo the vertex tracking. This involves putting the `'VERT'` word after the `CHAM` card in the cboff steering file, along with `'RTRK'`. Thus the minimal line in the cboff steering file for produced data would be:

   ```
   CHAM 'TRAK' 'RAWS' 'PATT' 'CIRC' 'HELX' 'VERT' 'RTRK' 'GPWC'
   ```

   You can substitute `GVXT` for `GPWC` if necessary.

2. Add these cards to your locater steering cards (or create new file if you don't already have a locater steering file). Note that there are other things you can put in this file too, see sections 3,4.

   ```
   VERT 3
   VFIT 2
   ```

3. Set Fortran Unit 81 to point to the locater steering file, and Fortran Unit 99 to point to the cboff steering file. For example, on UNIX:

   ```
   setenv FORT81 /user/analysis/locater.steer
   setenv FORT99 /user/analysis/cboff.steer
   ```

4. (optional) From your USER routine, call a routine such as TCVRHY (see section 5) to make cuts on the events based on the types of vertices found, and a routine such as TCVRHZ to decode some of the vertex information (see section 6).

5. (optional) If you concerned about CPU time limitations, and your data still contains all the locater banks (such as **TCHT**, **TCTK**, **TCHX** etc.) (*This is not true for production data*) then you can save some CPU time and leave out the intermediate cards, for example.

   ```
   CHAM 'TRAK' 'VERT' 'RTRK'
   ```

   However, the official crystal barrel produced data only contains the **TCTR** bank, so full retracking must be redone. Another idea is to first skim the production data (without retracking) and looking for events with the desired prong and PED multiplicities, and then to fully retrack (with vertex fitting) the skimmed data on a second pass.

3 TCVER3, Vertex Configuration Generator

Call Arguments: none
Output: bank structure at TCVX. See fig 5 and section 9
Called from: TCTRAK

TCVER3 attempts to build up a vertex configuration starting by pairing individual tracks, and then merging vertices if necessary. Here is a description of the algorithm.

1. Make two tables of positively charged tracks and negatively charged tracks. Use only tracks that have no helix error, and which are at least **V3MH** SVX/PWC/JDC hits long. **T3MH** can be changed, and the default value is 3.

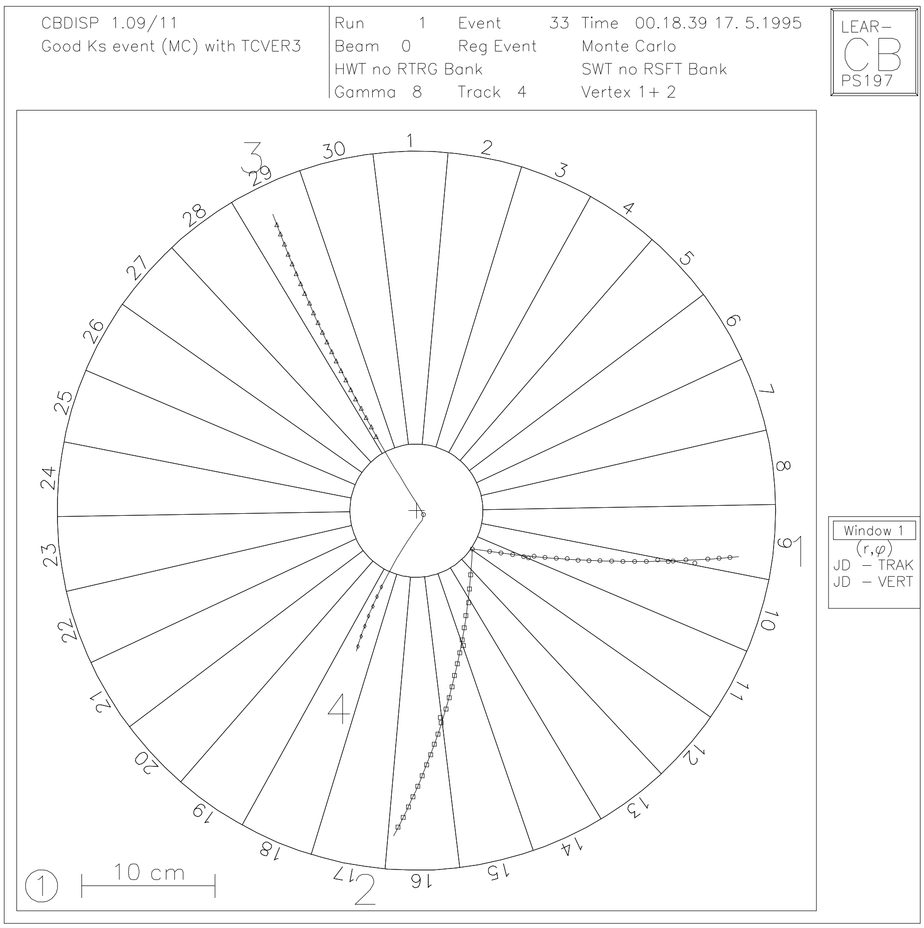

Figure 1: Example of 2 double vertex fit, in the channel $K_s K^+ \pi^-$

2. Pair each + track to a – track, and attempt to fit a common vertex, with a call to TCVRTX. This results in a list of vertices and the tracks connected to them.

3. Make all allowable configurations of vertices, using the above list of vertices, such that no track is used more than once in a configuration.

4. Pick the configuration that has the lowest reduced χ^2, that is the minimum value of

$$\chi^2/n = \frac{\sum_{\text{vertices}} \chi^2}{\sum_{\text{vertices}} N_{df}}$$

   Now we have a list of the best 2-prong vertices.

   optional step: Let $\vec{r}_{ij}$ = difference of positions of any two vertices i and j and $\vec{p}_i$ = the total momentum of all tracks at vertex i, which assumes that the vertex is the result of a neutral particle decay into charged particles. If $\vec{r}_{ij} \cdot \vec{p}_i > 0$, that can mean that the neutral particle that decayed at vertex i is moving towards (rather than away from) vertex j. Since this is an unlikely event and likely to be a wrong combination of tracks, a χ^2 penalty of `V3BA` is added to the configuration. `V3BA` can be changed, and the default value is 100.

5. For the remaining unmatched tracks, fit a pseudo-vertex to each single track (best guess of the vertex).

6. At this point, we have a list of 2-prong and 1-prong vertices (v_i) which includes every track.

7. Now attempt to combine these vertices together. For each vertex (v_i), try to attach every other vertex ($v_j, j \neq i$) to it.

   (a) Find the distance between the vertices v_i and v_j. Reject those that are greater than `V3DI` cm apart. `V3DI` can be changed, and the default is 3.

$$\vec{r}_{ij} = \vec{v}_i - \vec{v}_j$$

$$\text{Reject if} \quad |\vec{r}_{ij}| > \texttt{V3DI} \quad \text{cm}$$

   (b) Calculate the χ^2 for the hypothesis that $v_i \equiv v_j$. Reject those pairings with < `V3CL` % CL. `V3CL` can be changed, and the default is 0.01.

$$CL = \texttt{PROB}(\chi^2, N_{df} = 3)$$

$$\chi^2 = \vec{r}_{ij}(\mathcal{C}_i + \mathcal{C}_j)^{-1}\vec{r}_{ij}$$

$$\mathcal{C}_i = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix}_i$$

(c) Repeat the above two steps for all the vertices v_j.

(d) Once we have a list of vertices $(v_i, v_{j_1}, v_{j_2}, \ldots)$ that could possibly go together, attempt to fit all the tracks to a common vertex.

8. Drop all unused vertices, and keep the good ones in **TCVX**, update **TCTR** banks.

The end result is that every track will be paired to one and only one vertex. These vertices will have 1 or more tracks attached to them. Exception: Global tracking adds one "neutral" vertex, with error code 1000, which is by default (0,0,0), and used as the vertex for all PEDs. Thus every event has 1 "neutral" vertex (regardless if there are PEDs or not) and zero or more "charged" vertices. See figure 1 for a good example of a two vertex fit.

Controlling TCVER3

To use vertex fitting at all, in the cboff steering card file (Fortran Unit 99) use the **'VERT'** option with the **CHAM** card. Then use **VERT 3** to use TCVER3.

These are the steering cards for LOCATER and CBOFF that involve TCVER3. The one card that is probably most important is **V3MH**, the minimum number of hits on a track. The default value of 3 is the minimum from the helix fit, but in reality a value of 5-7 is more reasonable. I have used 7 in the comparison tests, as it seems to cleanup noisy events better, but keep in mind that it also cuts down on the solid angle acceptance. With 2-prong, long-track triggered data, there is no reason that this value be smaller than 7.

Steers	Card	Default	Description
cboff	**CHAM 'VERT'**	off	Turns on vertex fitting
locater	**VERT** n	1	Selects which vertex combinatorics to do 1 = TCVERT One vertex fit 2 = TCVER2 CBKFIT vertex fit 3 = TCVER3 Multivertex fit
	V3CL CL_{min}	0.01	Minimum confidence level to fit two vertices together [0.0,1.0]
	V3DI d_{max}	3.0	Maximum distance between vertices before attempting merge (cm)
	V3BA χ^2_{pen}	100.0	Introverted vertex pair χ^2 penalty
	V3MH n_{hit}	3	Minimum number of hits on track. If less, then totally ignore track.

4 TCVRHX, Full Helix & Vertex Fitting

Call Arguments: `CALL TCVRHX(NTRKS,NTRAK,ITCVT,IERR,XGUES)`

`INTEGER NTRKS` **- number of tracks in `NTRK()`**
`INTEGER NTRK()` **- array of TCTK track numbers.**
`INTEGER ITCVT` **- bank pointer to empty vertex bank.**
`INTEGER IERR*` **- return value. 0 if no problems.**
`REAL XGUES(3)` **- initial guess of vertex.**

Output: bank structure at TCVT. See section 9

Called from: TCVER3

The current vertex fitter TCVRTX uses the results of the helix fit as the raw input to a fit. Thus the vertex is fitted against the constraints of the helix parameters (5 values per track) rather than against the JDC/PWC/SVX hits. In general, this leads to a systematic error caused by the particular parameterization method of the helices. For example, one of the helix parameters, ψ_0, gives the initial angle of the center of circle, which is equal to the initial angle of the momentum up to a constant of $\pi/2$. This quantity is highly constrained by the hits, especially the hits in the outer layers, since they have a big "lever-arm" on the track.

However, without the hits as inputs to the fit, values like ϕ_0 begin to wander, more than they should. For short tracks, this value can change by several degrees, thus moving the fitted track far from the original hits. This is clearly unsatisfactory.

The new vertex fitting routine is based on the helix fitting routine. Instead of fitting several tracks independently, each with 5 helix parameters, the new fit fits all tracks simultaneously, each with 3 helix parameters, plus 3 more parameters for the location of the vertex. Thus the number of fit parameters goes from $5N$ to $3 + 3N$, which is smaller for $N \geq 2$. The fit is done by iterating the method of Lagrange multipliers.[1].

If only one track is attempted in the fit, then assign a vertex position that is either

- the closest approach to $(0, 0, 0)$ IF the hit information is missing or if the first hit is at JDC layer 3 or before.
- the position of the first hit of the track otherwise.

Controlling TCVRHX

To use vertex fitting at all, in the cboff steering card file (Fortran Unit 99) use the `'VERT'` option with the `CHAM` card. Then use `VERT 3` and `VFIT 2` to use TCVRHX. TCVRHX **only works with** TCVER3**!**

[1] Brandt, Siegmund, **Statistical and Computational Methods in Data Analysis**

These are the steering cards for LOCATER that involve TCVRHX. Some of the parameters from TCVRTX are also used ... see the source code if you really care.

steers	Card	Default	Description
locater	`VFIT` n	1	Selects which vertex fit to do 1 = TCVRTX Fits to helix parameters 2 = TCVRHX Fits to wire hits
	`VHCT` $(\chi^2/n)_{thresh}$	0.001	Maximum change in χ^2/n for convergence
	`VHER` f	1.0	Hit error estimate scaling factor
	`VHGC` $(\chi^2/n)_{greatest}$	3.0	Greatest χ^2/n at end of fit
	`VHMC` $(\chi^2/n)_{greatest}$	50.0	Greatest χ^2/n during fit.

5 TCVRHY, User Aide Routine 1

Call Arguments: `CALL TCVRHY(NVERT,NPRONG,IERR)`

`INTEGER NVERT` **- number of vertices (not counting neutral)**
`INTEGER NPRONG(10)` **- array of prongness of vertices.**
`INTEGER IERR` **- number of vertices with error code greater than 100.**

This routine returns an array, where each element gives the number of vertices with that prongness. For example, a two vertex event, each with two tracks, would have `NPRONG(2)=2` and the rest of `NPRONG(x)=0`. A one vertex event with four tracks would have `NPRONG(4)=1`. Note `NVERT` $= \sum_{I=1}^{10}$ `I` $*$ `NPRONG(I)`.

For example, in USER:

```
      SUBROUTINE USER

      INTEGER NVERT,NPRONG(10),IERR

      CALL TCVRHY(NVERT,NPRONG,IERR)
      IF (NVERT.EQ.2.AND.NPRONG(2).EQ.2.AND.IERR.EQ.0) THEN
*
* do something interesting with the event
*         ...
      ENDIF
      RETURN
      END
```

6 TCVRHZ, User Aide Routine 2

Call Arguments: `CALL TCVRHZ(NPRONG,IVERT,E,PX,PY,PZ,CHRG,MASS, X,T,RCHI,ITCVT,IERR)`

input:

	`NPRONG`	**- how many prongs desired**
	`IVERT`	**- Ith vertex with that many prongs**

output:

	`E(10)`	**- energies of outgoing tracks (`1..NPRONG`)**
	`PX(10)`	**- momentum x (`1..NPRONG`)**
	`PY(10)`	**- momentum y (`1..NPRONG`)**
	`PZ(10)`	**- momentum z (`1..NPRONG`)**
	`CHRG(10)`	**- charges (`1..NPRONG`)**
	`MASS`	**- invariant mass of vertex**
	`X(3)`	**- (x, y, z) position of vertex**
	`T(4)`	**- (p_x, p_y, p_z, e) momentum of vertex**
	`RCHI`	**- Reduced χ^2/n of vertex**
	`ITCVT`	**- pointer to vertex zebra bank**
	`IERR`	**- return code**

This routine returns various pieces of information (given above) for a particular vertex. The user should call TCVRHY first to determine how many of a certain vertex there are, e.g. how many 2-prong vertices there are. Then the user can call TCVRHZ with that prong value (e.g. 2), and ask for the first, second, third, etc. vertex with that prong value. For example, to extra the information for a two vertex event, each vertex having two prongs, use this example code:

```
      INTEGER NVERT,NPRONG(10),IERR
      INTEGER ITCVT
      REAL E(10),PX(10),PY(10),PZ(10),CHRG(10),MASS,X(3),CHI,T(4)

      CALL TCVRHY(NVERT,NPRONG,IERR)
      IF (NVERT.EQ.2.AND.NPRONG(2).EQ.2.AND.IERR.EQ.0) THEN
         DO I=1,2
*
* the '2' argument is to indicate that we want a 2 prong vertex
* the 'I' argument is to indicate which one (first or second)
*
           CALL TCVRHZ(2,I,E,PX,PY,PZ,CHRG,MASS,X,T,CHI,
     &                 ITCVT,IERR)
*
* do something here ...
*
           ...
         ENDDO
      ENDIF
```

7 TCVRTX, Modified Old Software

There was one modification to this routine. The definition of the parameter, **IVRTX**, was extended. If the value is positive, then it represents the number of the vertex bank to use, e.g. 1 to N. If the value is negative, then the absolute value is a ZEBRA pointer to a **TCVT** bank. This modification is totally backward compatible with TCVERT and TCVER2, which use the first case, and is also compatible with TCVER3 which uses the second case.

8 Comparisons

K_S Monte Carlo, Mass Resolution

10000 $k_s K^+ \pi^-$ Monte Carlo events were analyzed, by plotting the invariant masses of found 2-prong vertices. A gaussian plus linear polynomial was fitted to the $\pi^+\pi^-$ invariant mass. TCVRTX and TCVRHX were each separately used along with TCVER3, and the results given in the table below. Note that the width of the peak is significantly reduced from 11.7 MeV to 10.6 MeV, an improvement of 9%. See figure 2. The values for background and SNR are given assuming that one makes a 1 sigma cut on the K_S mass.

Also, the reconstructed vertex position was compared to the MC generated vertex position, by measuring the distance (δr) between the two in the xy plane, in the z direction and also in xyz space. Since the distance distributions have long tails, a better comparison is made with the log of the distance. See figure 3 and the table below. The best improvement is in the xy plane, where the mean of the $\log(\delta r_{xy})$ dropped from -1.39 to -1.66, which corresponds to the δr_{xy} distances of .25 cm and .19 cm respectively. See figure 3.

	TCVRTX (old)	TCVRHX (new)
Events in K_S peak	2900 ± 80	3270 ± 90
Width (MeV)	11.7 ± 0.3	10.6 ± 0.3
Mean (MeV)	496.4 ± 0.3	496.3 ± 0.2
Background (1 sigma cut)	270 ± 40	250 ± 70
SNR (1 sigma cut)	7.3 ± 1.2	8.9 ± 2.3
$\overline{\log(\delta r_{xy})}$	-1.39	-1.66
$\sim \delta r_{xy}$	0.25 cm	0.19 cm
$\overline{\log(\delta r_z)}$	-1.12	-1.26
$\sim \delta r_z$	0.33 cm	0.29 cm
$\overline{\log(\delta r_{xyz})}$	-0.59	-0.77
$\sim \delta r_{xyz}$	0.55 cm	0.46 cm

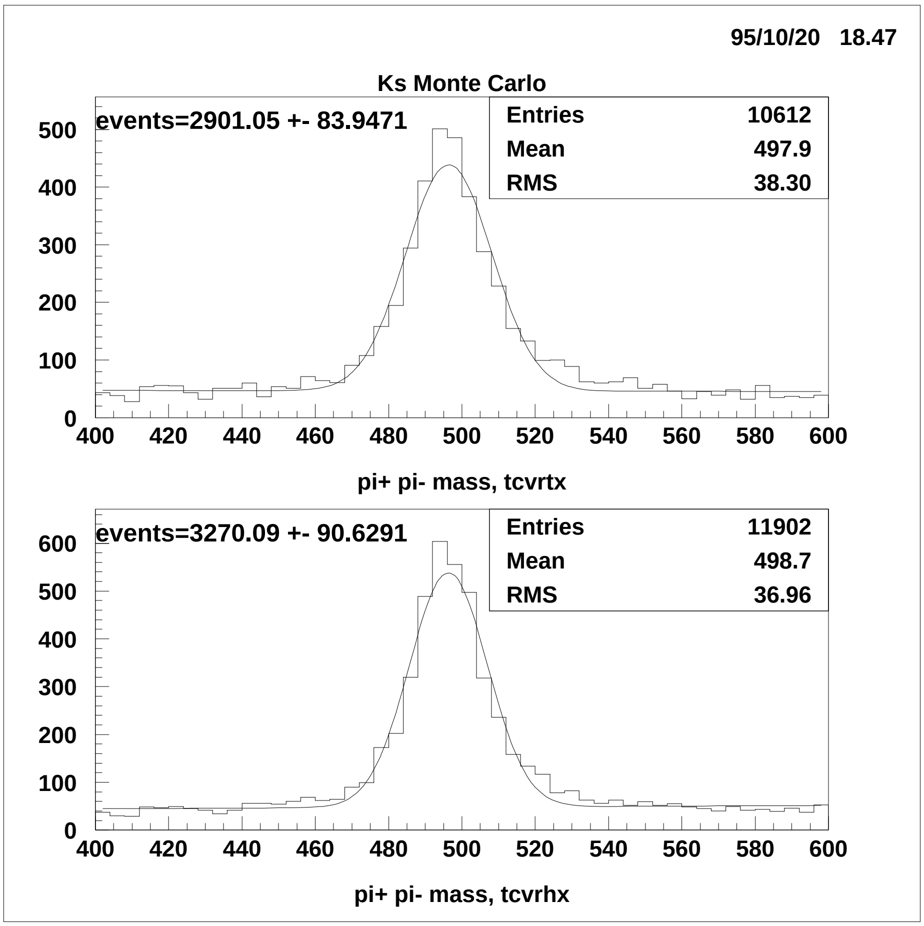

Figure 2: K_S peak in $\pi^+\pi^-$ invariant mass. (a) TCVRTX (old fitter) (b) TCVRHX (new fitter), fitted with a gaussian plus linear function.

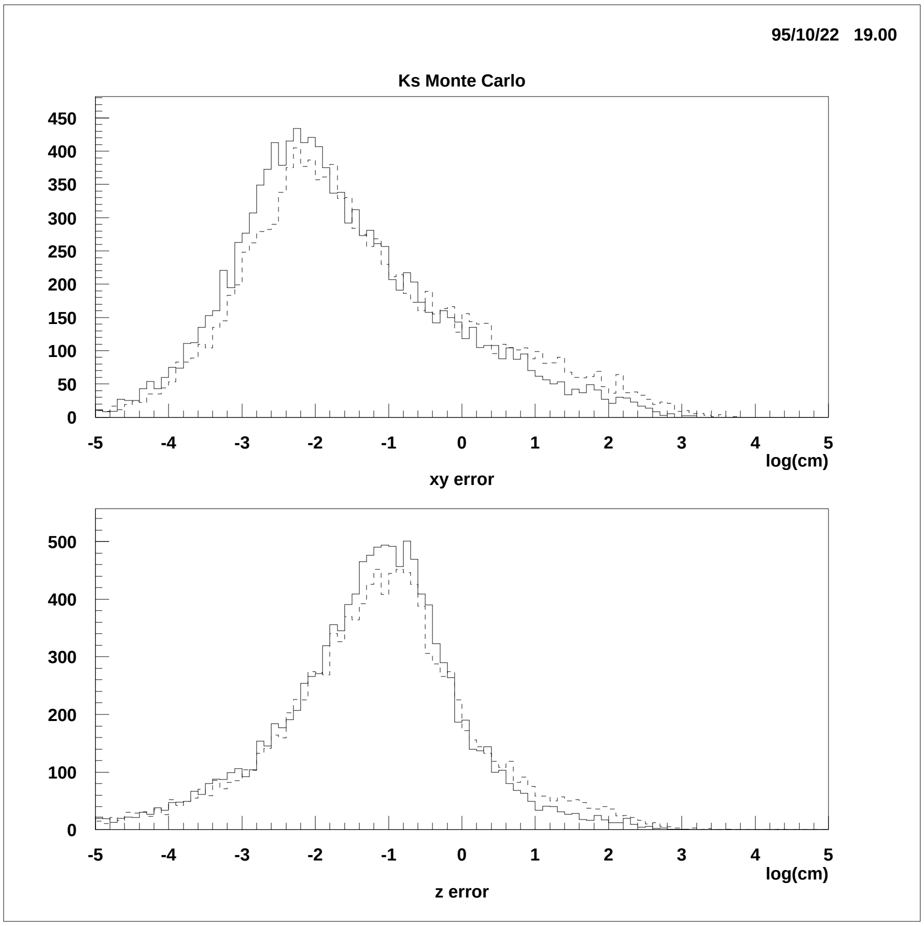

Figure 3: Vertex reconstruction error, log scale. (a) in xy plane (b) in z dimension. Solid line is TCVRHX, dashed line is TCVRTX.

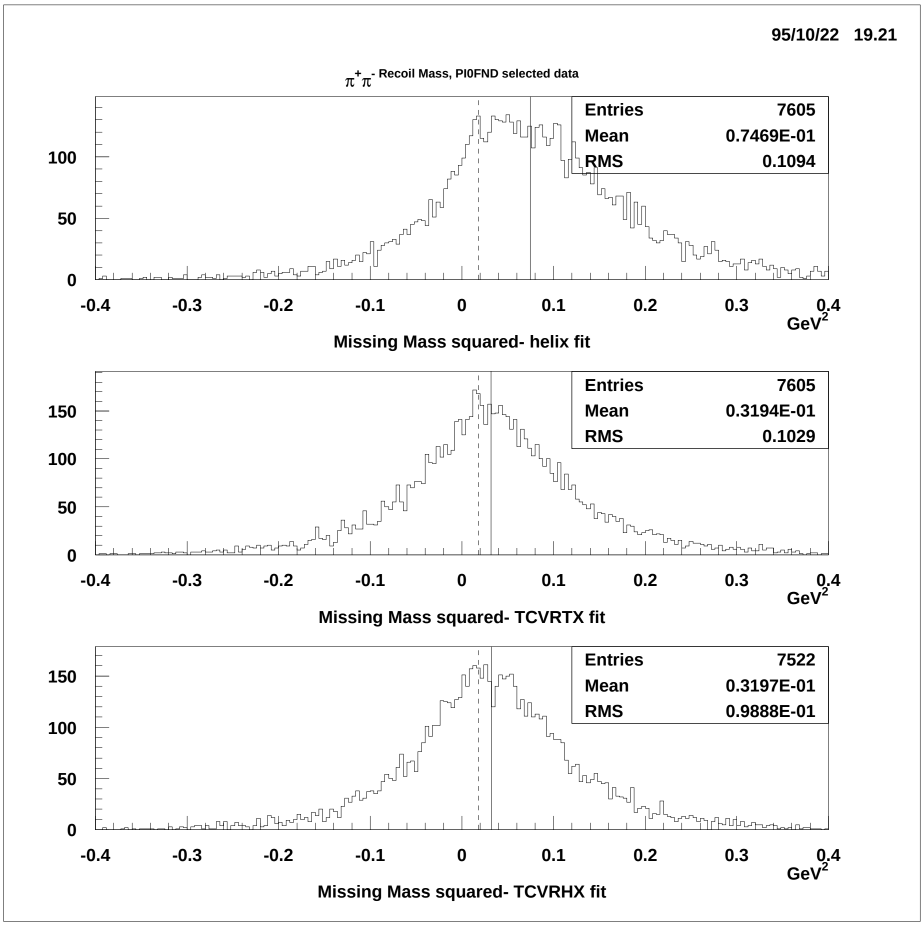

Figure 4: Missing mass squared histograms for (a) helix fit results, (b) TCVRTX fit results and (c) TCVRHX fit results. The dashed line represents $m^2_{\pi^0}$, while the solid line is the mean.

$\pi^+\pi^-\pi^0$ **Test**

June 1994 2-prong data (164517 events) were scanned for "Gold-plated $\pi +$ $\pi^-\pi^0$" events, by using these cuts:

1. Default production
2. Events must have 2 Prongs (88.5 %)
3. Events must have no type 13 PEDs (76.3 %))
4. Use only SMART type 0 PEDs, ignore the rest
5. Use only PEDs above 10 MeV, ignore the rest
6. Events must have 2 remainingg unmatched PEDs (11.7 %)
7.
   - (case A) Kinematic Fit of 10% CL or greater to the hypothesis $\pi^+\pi^-\pi^0$. (37.8 %)
   - (case B) PI0FND finds exactly one π^0. (79.0 %)

This selects 4896 events for case A and 7650 events for case B. The events are then processed again, with full reconstruction, including both types of vertex fitting, TCVRTX and TCVRHX. The momenta from the vertex bank are used to calculate the "missing mass" of the event, using just the charged tracks (or in other words the recoil mass of the two tracks). This should give a peak at the π^0 mass. Since the resolution is so poor in both cases, the *mass-squared* peak is smeared out into forbidden negative regions, yielding unphysical values of mass. However, the distributions peak near $m^2_{\pi^0}$ (see figure 4). The values of the square root of the mean and the RMS width are tabulated, and show that the old TCVRTX and new TCVRHX fits give similar results for both cases A and B (see table below). TCVRHX gives a small improvement in the RMS width at a slight efficiency cost.

The errors quoted for the means are

$$\sigma_{\sqrt{\mu}} = \frac{\sigma_{\text{RMS}}}{2\sqrt{\mu}\sqrt{N_{\text{entries}}}}$$

and are not necessarily one sigma gaussian errors. Errors in RMS width are roughly RMS/$\sqrt{N} \sim 0.001$.

		TCHELX	TCVRTX	TCVRHX
PI0FND	Events	7650	7605	7522
	$\sqrt{\text{Mean}}$ (MeV)	273 ± 2	179 ± 3	179 ± 3
	RMS (GeV^2)	0.110	0.103	0.099
CBFKIT	Events	4896	4889	4870
	$\sqrt{\text{Mean}}$ (MeV)	268 ± 2	159 ± 4	162 ± 4
	RMS (GeV^2)	0.088	0.081	0.081

9 Appendix A: Zebra Bank Details

The bank format structure has not been changed, except that now the **TCVT** bank may be filled with more than one vertex (TCVERT only outputted one vertex at most). As a summary, all the bank structures that are given in the locater manual are also given herein, plus the previously undocumented bank of **TCVH** is also presented. The bank structure is given in the following tables, and the layout is given graphically in fig 5.

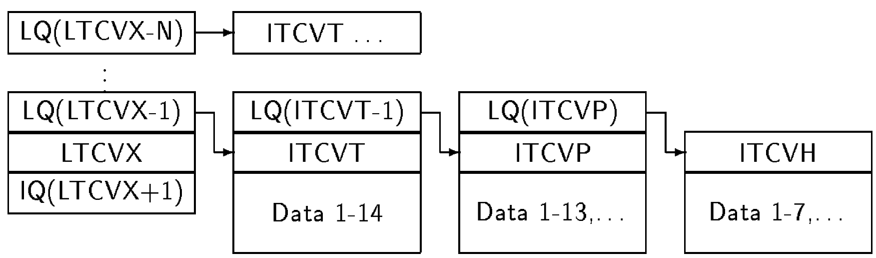

Figure 5: Layout of the bank structure containing all found vertices.

TCVX: The master vertex bank

Link	Variable	Descriptions
$\mathtt{ITCVT}_N =$	`LQ(LTCVX-N)`	link to N^{th} vertex
$\mathtt{ITCVT}_1 =$	`LQ(LTCVX-1)`	link to first vertex
	`IQ(LTCVX+1)`	number of vertices

TCVT: The parameters of the vertex

Link	Variable	Descriptions
$\texttt{ITCVT}_N$ =	LQ(LTCVX-N)	link to Nth vertex
ITCVP =	LQ(ITCVT-1)	link to ITCVP
	IQ(ITCVT+1)	number of tracks at vertex
	IQ(ITCVT+2)	number of neutrals at vertex
	IQ(ITCVT+3)	Error code
	IQ(ITCVT+4)	Number of degrees of freedom in fit
	Q(ITCVT+5)	x [cm] of vertex
	Q(ITCVT+6)	y [cm] of vertex
	Q(ITCVT+7)	z [cm] of vertex
	Q(ITCVT+8)	σ_{xx} [cm^2]
	Q(ITCVT+9)	σ_{xy} [cm^2]
	Q(ITCVT+10)	σ_{yy} [cm^2]
	Q(ITCVT+11)	σ_{xz} [cm^2]
	Q(ITCVT+12)	σ_{yz} [cm^2]
	Q(ITCVT+13)	σ_{zz} [cm^2]
	Q(ITCVT+14)	χ^2

TCVP: Improved parameters of tracks at the vertex

Link	Variable	Descriptions
ITCVP =	LQ(ITCVT-1)	link to ITCVP
ITCVH =	LQ(ITCVP)	link to ITCVH
	IQ(ITCVP+1)	Track number in TCTR
	IQ(ITCVP+2)	Quality Word
	Q(ITCVP+3)	Charge
	Q(ITCVP+4)	P_x [MeV/c]
	Q(ITCVP+5)	P_y [MeV/c]
	Q(ITCVP+6)	P_z [MeV/c]
	Q(ITCVP+7)	E [MeV]
	Q(ITCVP+8)	σ_{xx} [cm^2]
	Q(ITCVP+9)	σ_{xy} [cm^2]
	Q(ITCVP+10)	σ_{yy} [cm^2]
	Q(ITCVP+11)	σ_{xz} [cm^2]
	Q(ITCVP+12)	σ_{yz} [cm^2]
	Q(ITCVP+13)	σ_{xx} [cm^2]
	$\vdots$	1 to 13 repeated for other tracks

To access the Nth track, use `ITCVH = LQ(ITCVT-1)+LENVP*(N-1)`. `LENVP` is defined in `trkprm.inc` or with `+SEQ,TRKPRM`.

TCVH: improved helix parameters for the tracks at the vertex

See locater manual (CB Note 93,123) for explanation of helix parameters.

Link	Variable	Descriptions
`ITCVH =`	`LQ(ITCVP)`	link to ITCVH
	`Q(ITCVH+1)`	r_0
	`Q(ITCVH+2)`	z_0
	`Q(ITCVH+3)`	α
	`Q(ITCVH+4)`	$\tan\lambda$
	`Q(ITCVH+5)`	ψ_0
	`Q(ITCVH+6)`	s
	`Q(ITCVH+7)`	χ^2
	$\vdots$	1 to 7 repeated for other tracks

To access the Nth track, use `ITCVH = LQ(ITCVP)+LENVH*(N-1)`. `LENVH` is defined in `trkprm.inc` or with `+SEQ,TRKPRM`.

10 Appendix B: Details of TCVRHX

The parameters (see fig. 6) are:

- (x_v, y_v, z_v) The position of the vertex.
- α The curvature of the track. The momentum is `BMGCTC`$/\alpha$.
- ϕ_0 The position of the vertex w.r.t. the center of the circle.
- $\tan\lambda$ The z dependence.
- s The "sign" of the track. The true charge is $s\times$`BSGNTC` (a function of the sign of the magnetic field). The direction of motion around the circle is defined by s. If $s = -1$, the particle circulates counter-clockwise, while if $s = +1$, the particle circulates clockwise in the xy plane.
- β is the free parameter, that varies from 0 at the vertex to $+\infty$. In the fit, these are quantized as β_i for each hit i.

The helix parameterization used in TCVRHX is as follows:

$$x(\beta) = x_v - \frac{\cos\phi_0}{\alpha} + \frac{\cos(\phi_0 - s\beta)}{\alpha} \tag{1}$$

$$y(\beta) = y_v - \frac{\sin\phi_0}{\alpha} + \frac{\sin(\phi_0 - s\beta)}{\alpha} \tag{2}$$

$$z(\beta) = z_v + \frac{\tan\lambda}{\alpha}\beta \tag{3}$$

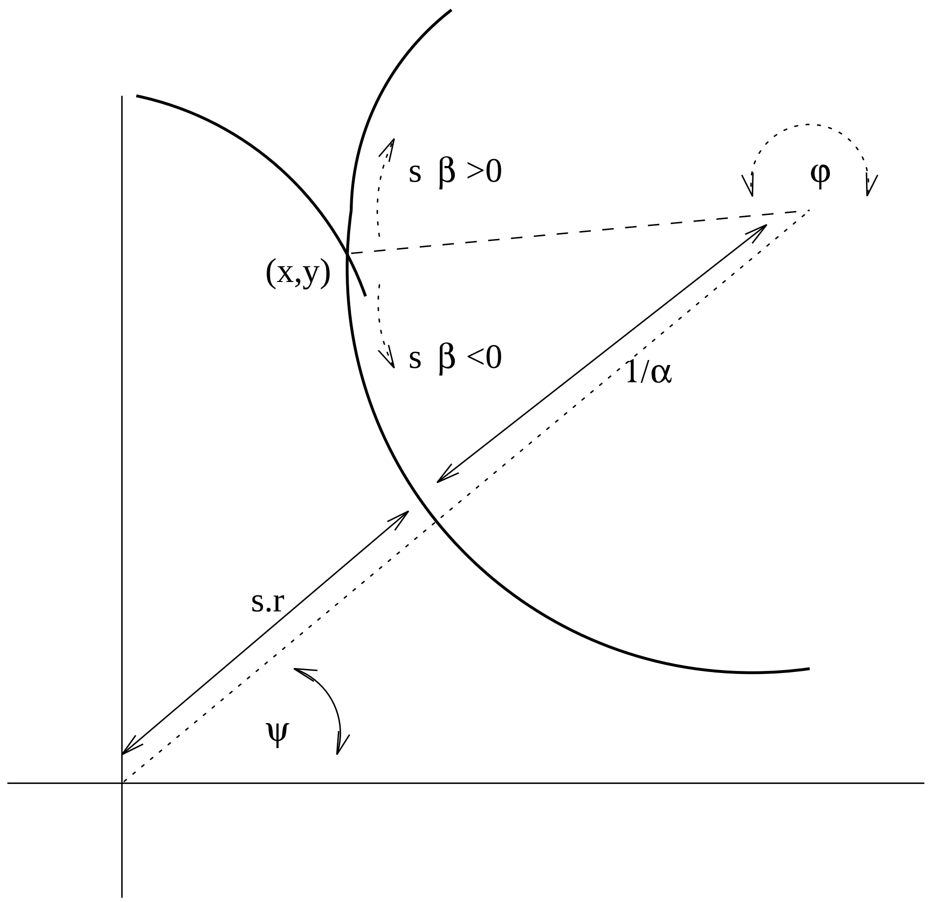

Figure 6: New vs. old helix parameterizations. Note, $\beta = 0$ at vertex, otherwise $\beta > 0$ always.

Here are some relationships between this parameterization and the standard TCHELX helix parameterization.

$$\Psi_0 = \psi_0 - s\frac{\pi}{2} = \mathtt{ATAN2}(y_v - \frac{\sin\phi_0}{\alpha}, x_v - \frac{\cos\phi_0}{\alpha})$$

$$r_0 = s\left[\sqrt{(x_v - \frac{\cos\phi_0}{\alpha})^2 + (y_v - \frac{\sin\phi_0}{\alpha})^2} - 1/\alpha\right]$$

$$z_0 = z_v + \frac{s\tan\lambda}{\alpha}\left(\Psi_0 - \phi_0 + \pi\right)_{(-\pi,\pi)sheet}$$

The routine TCVRHX uses the same fitting algorithm of TCHELX, except that the number of fitted parameters and input variables is different.

	TCHELX	TCVRHX
Input variables	$3N$ Values	$3\sum_{i,tracks}^{M} N_i$ Values
Fitted parameters	5 Values	$3+3M$ Values
...for 1 track	5	–
...for 2 tracks	–	9
...for 4 tracks	–	15
(Y Vector)	$(r_0, z_0, \alpha, \tan\lambda, \psi_0)$	$(x_v, y_v, z_v, \alpha^1, \tan\lambda^1, \phi_0^1, \alpha^2, \tan\lambda^2, \phi_0^2, \ldots)$
Degrees of Freedom	$3N-5$	$3\left(\sum_{i,tracks}^{M} N_i - 1 - M\right)$
...for 1 track	~ 64	–
...for 2 tracks	–	~ 129
...for 4 tracks	–	~ 261

Where N = number of hits on a track and M = number of tracks at the vertex.

The equations of constraint and derivatives of them with respect to the various input and output variables are given below.

$$C_1 = \frac{1}{\alpha}(\cos^2\beta_i + \sin^2\beta_i - 1) \quad (4)$$

$$C_2 = x_v - z_i + \frac{\tan\lambda\beta_i}{\alpha} \quad (5)$$

$$where\ \beta_i = s\left(\phi_0 - \mathtt{ATAN2}(\alpha(y_i - y_v) + \sin\phi_0, \alpha(x_i - x_v) + \cos\phi_0)\right) \quad (6)$$

The derivatives with respect to the hits (dx_i, dy_i, dz_i) and to the fitted helices $(dx_v, dy_v, dz_v, d\alpha, d\tan\lambda, d\phi_0)$ are:

$$\frac{dC_1}{dx_i} = 2\cos\beta_i \quad (7)$$

$$\frac{dC_1}{dy_i} = 2\sin\beta_i \quad (8)$$

$$
\begin{aligned}
\frac{dC_1}{dz_i} &= 0 && (9)\\
\frac{dC_2}{dx_i} &= \frac{s\tan\lambda\sin\beta_i}{\cos^2\beta_i+\sin^2\beta_i} && (10)\\
\frac{dC_2}{dy_i} &= \frac{-s\tan\lambda\cos\beta_i}{\cos^2\beta_i+\sin^2\beta_i} && (11)\\
\frac{dC_2}{dz_i} &= -1 && (12)
\end{aligned}
$$

$$
\begin{aligned}
\frac{dC_1}{dx_v} &= -\frac{dC_1}{dx_i} && (13)\\
\frac{dC_1}{dy_v} &= -\frac{dC_1}{dy_i} && (14)\\
\frac{dC_1}{dz_v} &= -\frac{dC_1}{dz_i} && (15)\\
\frac{dC_1}{d\alpha} &= \frac{2}{\alpha}\left[\cos\beta_i(x_i-x_v)+\sin\beta_i(y_i-y_v)\right]-\frac{1}{\alpha}C_1 && (16)\\
\frac{dC_1}{d\tan\lambda} &= 0 && (17)\\
\frac{dC_1}{d\phi_0} &= \frac{2}{\alpha}(sin\beta_i\cos\phi_0-\cos\beta_i\sin\phi_0) && (18)\\
\frac{dC_2}{dx_v} &= -\frac{dC_2}{dx_i} && (19)\\
\frac{dC_2}{dy_v} &= -\frac{dC_2}{dy_i} && (20)\\
\frac{dC_2}{dz_v} &= -\frac{dC_2}{dz_i} && (21)\\
\frac{dC_2}{d\alpha} &= \tan\lambda\left(\frac{\frac{\beta_i}{\alpha^2}-\frac{s}{\alpha}[\cos\beta_i(y_i-y_v)-\sin\beta_i(x_i-x_v)]}{\cos^2\beta_i+\sin^2\beta_i}\right) && (22)\\
\frac{dC_2}{d\tan\lambda} &= \frac{\beta_i}{\alpha} && (23)\\
\frac{dC_2}{d\phi_0} &= -\frac{s\tan\lambda}{\alpha}\left(\frac{\cos\beta_i\cos\phi_0+\sin\beta_i\sin\phi}{\cos^2\beta_i+\sin^2\beta_i}-1\right) && (24)\\
& && (25)
\end{aligned}
$$

11 Bibliography

1. Brandt, Siegmund, *Datenanlyse* (German) or *Statistical and Computational Methods in Data Analysis* (English translation), 1st ed. 1970, 2nd ed. 1976., 3rd ed. 1992.

2. Meyer, Curtis, *Chamber Reconstruction Software*, CB-Note 93/Revised.