Vertexing and Kinematic Fitting, Part II: Introduction to KWFIT

Lectures Given at SLAC
Aug. 5 – 7, 1998

Paul Avery
University of Florida

avery@phys.ufl.edu
http://www.phys.ufl.edu/~avery

Overview of plan

- **2nd of several lectures on kinematic fitting**
- **Focus in this lecture on real fitting examples using KWFIT**
- **Plan of lectures**
  - Lecture 1: Basic theory
  - Lecture 2: Introduction to the KWFIT fitting package
  - Lecture 3: Vertex fitting
  - Lecture 4: Building virtual particles
- **References**
  - **KWFIT**
    http://www.phys.ufl.edu/~avery/kwfit/ or
    http://w4.lns.cornell.edu/~avery/kwfit/
  - Several write-ups on fitting theory and constraints
    http://www.phys.ufl.edu/~avery/fitting.html

Quick Overview

- ***Third*** **generation of this software**
  - Used since 1990 for CLEO data analysis
- **Unified track list**
- **Kinematic constraints**
  - Vertex, mass, energy, 4-momentum, etc.
- **Build virtual particles using vertex constraints**
- **Many useful utility routines**
  - Transport through magnetic fields
  - Return errors for m, E, p, θ, ϕ, etc.
- **Experiment independent**
  - Experiment dependence limited to track filling routines
- **Fortran based**
- **Double precision only**
  - Needed because of covariance matrix calculations

KWFIT Tracks

- **Unified track list**
  - *All* particles stored in a single list

    QQ tracks
    Charged particles
    Photons
    π^0, K_S, Λ
    Virtual particles $\Rightarrow$ *D* and *B* mesons

  - Fill routines for each type (e/μ/π/K/p)

  - User sees particles as track indices. Each mass hypothesis is a separate track.

    CD track 1 $\Rightarrow$ KW track 2 (π) & KW track 3 (K)

• Track variables

- w(1-10) ⇒ The "*W*" track parameters

1	px
2	py
3	pz
4	E
5	x
6	y
7	z
8	pt
9	ptot
10	Q

- Representation greatly simplifies physics analysis and can be manipulated by a host of support routines.

- *Fitted* variables are 1 – 7. Why 7?

  5 helix parameters + mass, position along helix
  Can handle virtual particles

- Other consequences

  7×7 covariance matrix
  Vastly simpler math for implementing constraints

- **Track variables (available through access calls)**

w(10)	current track parameters
w0(10)	Unconstrained track parameters
Vw(7,7)	7×7 unconstrained covariance matrix
ext_position	Pointer to track position in original list
ext_origin	ID of track origin (e.g., CD, pi0, Ks, etc.)
lposition	TRUE if position info is available
lcovar	TRUE if covariance matrix available
lfixed_mass	TRUE if particle has fixed mass
mass	Mass used in 4-momentum

Kinematic fitting

- **Mechanism: Lagrange multipliers**

  Start with “unconstrained” parameters

  Linearize constraint equations

  Solve equations

  Update parameters

  Loop until $|\Delta\chi^2| < \varepsilon$ or too many iterations

  Update “current” parameters only (if requested)

- **Can check fit results *before* updating tracks**
  - Allows check of χ^2 to see if fit was good

- **Many constraints supported**
  - Mass
  - Energy
  - Vertex
  - Back-to-back (di-muon)
  - Total momentum
  - 4-momentum
  - 3-momentum
- **Many types of vertex constraints**
  - Unknown 3-D vertex
  - “Fuzzy” vertex, e.g., beam spot
  - Vertex lying on a plane
  - Vertex lying on a line
  - Fixed vertex
  - Single track consistent with “fuzzy” vertex
  - Single track consistent with fixed vertex

- **Functions to return track parameter errors**
  - Mass
  - Energy
  - Momentum
  - θ
  - ϕ
- **Many utilities**
  - Transport particles through magnetic fields (point, plane, cyl.)
  - Mass of 2,3,4 particles
  - Weighted average of 2 vertices, including χ^2
  - χ^2 that V_1 and V_2 are the same
  - L / σ between two vertices

Virtual particles

- **Build new KW track from *n* KW tracks**
  - Apply vertex constraint when building KW track
  - Vertexing requirement very flexible per input particle
  - Fast: only inverts n 2×2 and one 4×4 matrices
- **Fit decay sequences**

  Example: fit decay sequence shown below (measured particles shown in boldface) by combining particles starting at the bottom and building up the chain:

$$\bar{B}^0 \to D^{*+} \rho^-$$

$$D^{*+} \to D^0 \boldsymbol{\pi}^+$$

$$\rho^- \to \boldsymbol{\pi}^- \boldsymbol{\pi}^0$$

$$D^0 \to \mathbf{K}^- \boldsymbol{\pi}^+$$

Simple Example

$$D^0 \rightarrow K^- \pi^+$$

```
      subroutine anal1

*   ******************************
*   Called at beginning of job
*   ******************************

*     Initialization of kwfit. Clear everything.
      call kset_init

      return
      end
```

```
      subroutine anal2

*   ******************************
*   Called once per run
*   ******************************

      implicit none

      integer I
      logical ltesla
      double precision beam_pos(3), beam_sig(3)
      double precision bfield_mag, bfield_dir(3)

      data bfield_mag/-1.497/
      data bfield_dir/0., 0., 1./
      data ltesla/.TRUE./
*   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

*     Set beam position & width
*     Need to do this in case you make vertices weighted by beam spot
      call kset_beam_position(beam_pos, beam_sig)

*     Set B field magnitude and direction
      call kset_bfield(bfield_mag, bfield_dir, ltesla)

      return
      end
```

```
      subroutine anal3

*   ******************************
*   Called once per event
*   ******************************

      implicit none

*     Externals
      double precision kget_track_mass
      external         kget_track_mass

*     Local variables
      integer I, type, error, update, num_d0, list_d0(2), option_d0(2)
      integer ipi, iK, npi, nK, list_pi(100), list_K(100)
      logical lcovar, ldedx, lvtx
      double precision w_K(10), w_pi(10), w_d0(10), chisq
      double precision chisq_d0, chisq_mass, mass_d0
      double precision vtx(3), Vvtx(3,3)
*   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

*     Clear the kwfit track list every event
      call kset_clear
```

```
*     Make list of pions and kaons from CD tracks. We want
*     the covariance matrix built but no dE/dx correction is
*     necessary because the tracks have already been Kalman fit.
*     The list of kwfit tracks is returned in list_pi and list_K
      lcovar = .TRUE.
      ldedx = .FALSE.

*     These are the only calls that depend on CLEO information

      type = 3      !Pions
      call kfil_track_cd_all(type, ldedx, lcovar, npi, list_pi, error)

      type = 4      !Kaons
      call kfil_track_cd_all(typr, lcovar, lcovar, nK, list_K, error)
```

```
*     Loop over K, pi lists and build D0 4-momenta
      do iK=1,nK
        call kget_track_param(list_K(iK), w_K)
        do ipi=1,npi
          call kget_track_param(list_pi(ipi), w_pi)
          mass_d0 = kutl_mass2(w_k, w_pi)

*     Find the vertex of the K-pi pair.
*        vtx(3)    = returned vertex
*        Vvtx(3,3) = returned vertex covariance matrix
          num_d0 = 2                            !2 tracks
          list_d0(1) = list_K(iK)               !Pion
          list_d0(2) = list_pi(ipi)             !Kaon
          update = 0                            !Do not update input tracks
          lvtx = .FALSE.                        !Compute vertex from scratch
          call kvtx_unknown(num_d0, list_d0, update, lvtx,
     *                      vtx, Vvtx, chisq, error)
```

```
*     If the chisq is OK, update the input tracks. The update
*     causes the track parameters to be adjusted in such a
*     way as to make them pass through the new vertex point.
*     The covariance matrices of the tracks are not changed.

         if(chisq .lt. 10.) then
           call kfit_update_tracks

*     Move the tracks and covariance matrices to the new vertex point
           direct = 0          !Move in nearest direction
           do I=1,num_d0
           call ktrk_move_point_bend(list_do(I), vtx, direct, error)
           enddo

           call kget_param(list_d0(1), w_pi)    !Get pion track info
           call kget_param(list_d0(2), w_K)     !Get kaon track info
           call kutl_sum2(w_d0, w_pi, k_K)      !Compute D0 track info
         endif
```

```
*     Alternatively, you can build a D0 particle with a vertex
*     constraint and a full covariance matrix. The D0 track
*     parameters and covariance matrix are evaluated at the
*     vertex point.

*     Create track slot
          call kfil_track_create(kd0)

*     Build the D0 virtual particle
          option_d0(1) = 2     !Use pion to determine vertex
          option_d0(2) = 2     !Use kaon to determine vertex
          update = 0           !Do not update input tracks
          lvtx = .FALSE.       !Find vertex from scratch
          call kvir_vertex_unknown(num_d0, list_d0, option_d0,
     *                             update, lvtx, vtx,
     *                             chisq_d0, kd0, error)

*     Check the Kpi mass. If OK, apply a mass constraint forcing
*     the D0 to have the correct mass. The idea is to improve
*     the D0 track parameters so that the D0 can be used in
*     subsequent fits.
          mass_d0 = kget_track_mass(kd0)
          if(abs(mass_d0 - 1.8654) .lt. 0.010) then
            update = 2
            call kfit_mass(kd0, 1.865, update, chisq_mass, error)
          endif
```

```
  enddo
enddo

return
end
```