Kinematic Fitting Algorithms and Lessons Learned from KWFIT

Paul Avery

Department of Physics, University of Florida, Gainesville, FL 32611-8440, USA

Abstract

I discuss the kinematic fitting program **KWFIT** and some general lessons learned in applying kinematic fitting to particle physics data. Though originally written for CLEO use, **KWFIT** is a general standalone program.

KWFIT encodes standard constraints involving vertexing, mass, energy, 4-momentum and back-to-back tracks. A powerful capability is its ability to build "virtual particles" from a set of daughters under a vertex constraint. I have recently added routines to optimally fit for lifetimes using all track and vertex information and a new module that allows the lifetime of unusual classes of decays such as $D^0 \to K_S\pi^0$ to be determined.

The large amount of accumulated experience with **KWFIT** points to certain lessons about track representation and sensitivity to errors.

Keywords: kinematic, fitting, vertex, constraint, algorithm, lifetime, kwfit, lagrange

1 Introduction: Kinematic Fitting

I discuss in this paper the kinematic fitting package **KWFIT**, various versions of which have been used in the CLEO experiment since 1990. Kinematic fitting can be defined as a mathematical procedure in which one uses the physical laws governing a particle interaction or decay to improve the measurements describing the process. For example, the fact that the three daughters in $D^+ \to K^-\pi^+\pi^+$ decay must come from a common space point can be used to improve the momentum vectors of the daughter particles, thus improving the mass and momentum resolution of the D^+. These resolution improvements translate to larger signal to background ratios, frequently elevating marginal signals to statistical significance. In recognition of its importance in data analysis, kinematic fitting is used in virtually all high energy physics experiments today.

2 Introduction to the KWFIT Package

The main characteristics of **KWFIT** are presented here; interested readers can find out more information – or download a copy – from the online documentation[1]. (Every listed routine is hyperlinked directly to the documentation describing it.) At the present time **KWFIT** is implemented in Fortran, but that is an unimportant limitation since my aim here is to discuss the algorithms and general implementation considerations. The language issue is temporary anyway since a C++ version is being designed, though it should be pointed out that the implementation in object-oriented languages needs to be done with great care because of the complexity of the results one wants to extract. Mathematics is easy, design is hard.

KWFIT is built around the following data entities: (1) tracks (charged, neutral, etc.), (2) a magnetic field (fixed, but arbitrarily oriented), (3) primary vertex position and dimensions and (4) initial 4-momentum of the primary vertex. The first two entities are required because manipulations of tracks moving through magnetic fields are fundamental operations. Their values can be entered through standard interfaces. Additional track input routines have been written to simplify the process of converting CLEO tracks to **KWFIT** format. These track input routines constitute the only experiment-specific part of **KWFIT** and they are isolated in a single subdirectory. Other fill routines can be added as needed, e.g., for particles in the HEPEVT track list.

Seven variables are generally needed to completely define a track[2]. **KWFIT** uses as its internal representation a point on its trajectory and its 4-momentum there, represented here as α, where $\alpha = (p_x, p_y, p_z, E, x, y, z)$. I call this the 'W' representation. I will comment on the track representation later when I discuss numerical issues.

3 Implementation of constraints

3.1 General mechanism

Constraints are implemented in **KWFIT** through the Lagrange multiplier (LM) mechanism, though some mathematical tricks are used to dramatically improve speed and to overcome subtle numerical instabilities. Consider a set of n tracks (a total of $7n$ parameters) which we represent by the row vector $\alpha^T = (\alpha_1^T, \alpha_2^T, \ldots \alpha_n^T)$. Initially, the track parameters have the unconstrained values α_0 and covariance matrix $\mathbf{V}_{\alpha 0}$, obtained by a track fit, for example. The r functions describing the constraint conditions can generally be written $\mathbf{H}(\alpha) = \mathbf{0}$, where $\mathbf{H} = (H_1, H_2, \ldots H_r)$. Expanding about a convenient point α_A yields the linearized equations $\mathbf{D}\delta\alpha + \mathbf{d} = \mathbf{0}$, where $D_{il} = \partial H_i/\partial\alpha_l$ and $\delta\alpha = \alpha - \alpha_A$. The χ^2 can then be written $\chi^2 = (\delta\alpha - \delta\alpha_0)^T \mathbf{V}_{\alpha 0}^{-1}(\delta\alpha - \delta\alpha_0) + 2\lambda^T(\mathbf{D}\delta\alpha + \mathbf{d})$, where λ is a vector of length r containing the lagrange multipliers. Minimizing the χ^2 with respect to $\delta\alpha$ and λ yields the solution[3] for $\delta\alpha$ and its covariance matrix,

$$\begin{aligned} \delta\alpha &= \delta\alpha_0 - \mathbf{V}_{\alpha 0}\mathbf{D}^T\lambda \\ \lambda &= \mathbf{V}_D(\mathbf{D}\delta\alpha_0 + \mathbf{d}) \\ \mathbf{V}_\alpha &= \mathbf{V}_{\alpha 0} - \mathbf{V}_{\alpha 0}\mathbf{D}^T\mathbf{V}_D\mathbf{D}\mathbf{V}_{\alpha 0} \end{aligned} \tag{1}$$

where $\mathbf{V}_D = (\mathbf{D}\mathbf{V}_{\alpha 0}\mathbf{D}^T)^{-1}$ is the constraint covariance matrix and $\chi^2 = \lambda^T\mathbf{V}_D^{-1}\lambda = \lambda^T(\mathbf{D}\delta\alpha_0 + \mathbf{d})$. Note that the χ^2 can be written as a sum of r terms, one for each constraint.

KWFIT implements many constraints utilizing this mechanism, including (1) mass, (2) energy, (3) total momentum, (4) 3-momentum, (5) 4-momentum, (6) back-to-back pair (unboosted), (7) back-to-back pair (boosted), (8) fixed vertex, (9) known but fuzzy vertex, (10) unknown vertex, (11) lifetime, (12) double vertex (for $D^0 \to K_S\pi^0$) and (13) double vertex plus lifetime. Special 2-D versions of the single vertex constraints are also provided.

In general, the nonlinearities of even simple fits requires that the procedure above be applied iteratively until satisfactory convergence is achieved. Track parameters and their errors, covariance matrices, fit information and other quantities can be obtained through access routines.

3.2 The special case of vertex constraints

Vertex constraints introduce subtleties that can cause numerical problems. The subtleties, when properly accounted for, turn out to make the solution much faster and more stable numerically than what would be obtained by straightforwardly applying the LM procedure in the previous section.

First, vertex constraints add new variables, the vertex coordinates $\mathbf{x}$, so we insert them before the track quantities to create the extended parameter vector: $\tilde{\alpha}^T = (\mathbf{x}^T, \alpha_1^T, \alpha_2,^T \ldots \alpha_n^T)$. Similarly, we create the extended constraint matrix, $\tilde{\mathbf{D}} = (\mathbf{E}, \mathbf{D})$, where $E_{ij} = \partial H_i/\partial x_j$. For the case where the initial vertex is unknown, the vertex position never appears in the quadratic part of the χ^2 and the LM procedure in the previous section fails. A simple way to overcome this difficulty is to assign a large initial error to the vertex position and apply the LM algorithm. Unfortunately, the calculation of the vertex covariance matrix involves a subtraction of two very large matrices, a stability problem which even double precision does not solve.

The algorithm in **KWFIT** avoids this problem and gains significant execution speed by exploiting the special nature of vertex constraints. For each track there are two equations which relate that track's parameters with the vertex coordinates independently of the other tracks. If the tracks are initially uncorrelated, the $2n \times 2n$ matrix $\mathbf{V}_{\tilde{D}}^{-1} = \tilde{\mathbf{D}}\mathbf{V}_{\tilde{\alpha}0}\tilde{\mathbf{D}}^T$ has a special form that can be inverted with n 2×2 inversions and a single 3×3 inversion[3]. Further matrix manipulations eliminate the instability caused by subtracting two large matrices when determining the covariance matrix of the vertex coordinates $\mathbf{x}$. It can easily be shown that this method is a Kalman algorithm in which the tracks are the measurements and the vertex coordinates are the estimated parameters[4].

4 Virtual Particles

In many physics analyses we wish to merge a set of particles so that the new "virtual" particle and its covariance matrix replace the original particles in subsequent analyses. Consider, for example, the decay chain $\overline{B}^0 \to D^{*+}\pi^-,\ D^{*+} \to D^0\pi^+,\ D^0 \to K^-\pi^+$. We could take the following steps: (1) combine $K^-\pi^+$ to form a D^0, (2) merge the D^0 with a π^+ to make a D^{*+} and (3) combine the D^{*+} with a π^- to make a $\overline{B}^0$. At every step the merging process must produce a virtual particle that is the best approximation to the reconstructed particle.

KWFIT implements an efficient and unusual algorithm[5] for building virtual particles based on the vertex constraint. The daughter tracks are first vertexed using one of the standard vertex constraint routines (unknown point, known but fuzzy point, or fixed point). The updated tracks are then swum to the vertex and the 4-momentum is calculated by summing the 4-momenta of the tracks.

The calculation of the 7×7 covariance matrix requires much care because it must include (1) the 4×4 momentum contributions from the daughter tracks, (2) the 3×3 position piece that comes from the vertex fit, (3) the correlations introduced by the vertex constraint and (4) the errors and correlations arising from the swimming process. My derivation of the virtual particle covariance matrix fully exploits the Kalman structure of the vertex fitting algorithm to produce an efficient closed-form formula[5].

The algorithm implemented in **KWFIT** is actually more general because it permits three classes of tracks to be used simultaneously in building a virtual particle: (1) tracks which are used to determine the vertex; (2) tracks which are not used to determine the vertex but are forced to pass through the vertex determined from the class 1 tracks; and (3) tracks such as neutrals which have no position information but supply 4-momentum. The derivation of the covariance matrix is *much* more complicated than described above, but after extensive matrix manipulations the resulting formula becomes identical to the simple case, provided the $\mathbf{E}$ matrices are properly defined[5].

5 Optimal Determination of Lifetime

KWFIT was recently equipped with a new lifetime fitting algorithm[6] that optimally uses all track information and works for charged or neutral tracks for arbitrary distances. The algorithm uses the three equations of motion as the constraint conditions linking a track with a known starting point (fixed or fuzzy). The equations are functions of s/p, where s is the arc length from the starting point and p is the momentum. The algorithm exploits the trivial fact that $s/p = c\tau/m$, where $c\tau$ is the proper lifetime in distance units and m is the mass. The LM procedure is then used to determine $c\tau$, its error and correlations with other variables (two degrees of freedom).

Even more recently, an algorithm was added[1] to **KWFIT** that finds the lifetime of a particle decaying into two daughters, one of which has no position information, e.g. $D^0 \to K_S\pi^0$. The technique uses the fact that the neutral particle supplies 4-momentum which alters the trajectory of the other particle in such a way as to pass through a known starting point (fixed or fuzzy). The LM procedure then determines $c\tau$ plus its errors and correlations, with one degree of freedom.

6 Experience with Kinematic Fitting and KWFIT

Years of experience with kinematic fitting have brought many lessons. First, it has become clear that double precision should be the default for all floating point variables. There are many matrix operations and therefore inevitable roundoff errors, particularly for vertex constraints with one degree of freedom, where four terms must cancel to yield the $(\chi^2)^{-1/2}$ dependence near zero.

Second, it came as a surprise that many fits did not converge sufficiently with the default number of iterations. The criteria in effect now, which are adequate but can easily be changed through a set routine, is that convergence is achieved when the change in chisquare between two successive iterations satisfies $|\Delta\chi^2| < 0.001$ within 10 iterations.

Third, it appears that using the energy as one of the internal track variables causes some numerical problems. The fundamental reason is that the energy is usually highly correlated with the momentum components. Moreover, the mass constraint is very non-linear, and a particle formerly on the mass shell can easily be pulled off by applying a subsequent vertex constraint. These problems can be resolved by replacing energy by mass in the internal representation. In most cases mass is not strongly coupled to the momentum components (the correlation is zero for reconstructed tracks and gammas) and the mass constraint becomes linear. The change to mass will be made in the OO version, expected this year.

Finally, many people[7] find the idea of doing successive fits through a decay chain reprehensible, preferring instead to fit the entire chain at once. Unfortunately, a comprehensive fit provides only a single χ^2 to evaluate the fit quality, so valuable information is lost. However, the χ^2 calculation for r constraints can be written as a sum of r^2 terms, each of which provides fit quality information. This information is accessible from **KWFIT** and used to improve fit quality, though it has never, to my knowledge, been used. This area requires further research.

References

1 `http://www.phys.ufl.edu/~avery/kwfit/`

2 Seven variables are needed for particles at their decay point, e.g., $D^0 \to K^-\pi^+$. Of course, when 7 variables are defined for a reconstructed charged track, its covariance matrix is only of rank 5 since the mass is fixed and the location on its trajectory is ill-defined.

3 P. Avery, "Applied Fitting Theory VI: Formulas for Kinematic Fitting", CLEO CBX 98-37, `http://www.phys.ufl.edu/~avery/fitting/kinematic.ps.gz`

4 P. Avery, "Vertex Fitting", `http://www.phys.ufl.edu/~avery/fitting/kinfit_talk3.pdf`

5 P. Avery, "Applied Fitting Theory VII: Building Virtual Particles", CLEO CBX 98-38, `http://www.phys.ufl.edu/~avery/fitting/virtual.ps.gz`

6 P. Avery, "Directly Determining Lifetime Using a 3-D Fit", `http://www.phys.ufl.edu/~avery/fitting/lifetime.pdf`

7 You know who you are.