An Object Oriented Kalman Filter Tracking Algorithm

David Nathan Brown

Lawrence Berkeley National Lab
Dave_Brown@lbl.gov

We present an object oriented formulation of the extended Kalman filter algorithm for track fitting and finding. The implementation makes independent fits in both track directions from which optimal results can be derived at any point. This approach is mathematically equivalent to the conventional asymmetric configuration while allowing simpler object behavior. The parametric, geometric, and kinematic functions of track fitting are put into separate objects, reducing the object complexity. This formulation allows an extension to treat non-Gaussian measurement errors and process noise. This algorithm will be used for track fitting with the BaBar experiment at the SLAC B-factory[1].

Key words: Kalman filter;Track fit;Maximum Likelihood;

1 Introduction

The extended Kalman filter algorithm is the most popular tool for track fitting in HEP experiments today [2]. Its success stems from both its computational simplicity and its flexibility; by breaking down track fitting into a sequence of mathematically simple steps, it can successfully model complex tracking devices without the inversion of large matrices. Producing complex collective behavior from a sequence of simple steps allows a good match to object oriented programming languages. In this paper we describe a Kalman track fit implemented as a C++ class library.

2 Class Structure

The central class is the KalSite. This describes an 'occurrence' at a fixed position (site) along a track. The occurrence can be a measurement, passage through material, or passage through a magnetic field irregularity. KalSite is

Preprint submitted to Elsevier Preprint *25 April 1997*

a pure virtual base class with daughters (KalHit, KalMaterial, and KalBend) for each type of occurrence. KalSites methods allow manipulation of track parameters and covariance matrices as parametric (linear algebra) objects. Methods describing the physical and geometric significance of the parameters reside in other classes.

The KalSite interface provide three basic functions:

– Construction. This takes as input an estimate of the local track parameters at the site, plus the specific occurrence (IE the measurement, piece of material, or field integral). This computes and caches the linearization of the occurrence's effect on the track parameters, expanded about the input parameters.
– Processing. This takes as input a set of track parameters and returns an updated version of same accounting for the (linearized) effect of the occurrence.
– Updating. This takes as input a set of track parameters which represent the current best estimate at the site. The KalSite cache is re-linearized around this estimate.

In addition to caching the occurrence linearization, KalSites also cache the result of their processing (local fit parameters and chisquared). The cache separately stores the linearization and results for processing in the outward (direction of positive particle energy loss) and inward (opposite) directions, as these are not degenerate for KalHit or KalBend. Outward and Inward are defined by the Trajectory class described below.

The track itself is represented by the KalTrack class, which is mostly an ordered collection of KalSite objects. The KalTrack constructor takes as input an estimate of the track trajectory. From that constructs its KalSites using the list of measurements, the material model, and the field map. The KalSite objects are ordered according to the flight distance along the initial track at which they occur. The KalTrack is fit by chaining calls to the 'process' method of the KalSites, starting with the initial parameter estimate (with an inflated covariance). Inward and outward fits are performed separately. A track fit can be improved by passing the resultant local optimal fit parameters to the KalSite 'update' methods, then repeating the fit. Re-fitting can be performed iteratively for given convergence criteria. If necessary, a rescan of the material and field map along the resultant trajectory can be performed.

The Trajectory class describes the geometric properties of the track (position, direction, and curvature) as a function of flight distance along the track. A trajectory may be simple (a helix) or compound (a piecewise helix). More information on Trajectories can be found in [3].

The kinematic interpretation of the track is given by the magnetic field map object. The map defines the track momentum given the position, direction, and curvature of its trajectory. Non-uniformities of the field may be integrated over a trajectory range, providing the input occurrence to construct a KalBend object.

The material model is represented by a cooperating set of classes which describe the position, shape, size, and composition of all relevant matter in the tracking volume. Given a Trajectory, the model identifies those elements through which the track passes. Given a particle mass and momentum, an intersection can describe the material effects (multiple scattering and energy loss) which it induces. The intersection provides the mean, RMS, and statistical distribution (sum of Gaussians) of the effect.

3 Mathematical Implementation

KalSites implement the weighted mean formalism of the extended Kalman filter. The 'process' method performs a step in the procedure called 'filtering' in Fruewirth's notation. Fit results contain both the covariance and inverse covariance matrices. KalSites act directly only on that matrix to which the effect of their occurrence is additive (KalHit processes the inverse covariance, KalMaterial the covariance, KalBend leaves both unchanged). The directly processed matrix is then inverted to give its homologue. This trick maintains the computational accuracy of the results during chained processing of the sites. This implementation requires the same number of matrix inversions and multiplies to obtain a full fit of the track (optimal results at both ends) as the gain matrix formalism. Once fully fit, optimal parameters can be obtained anywhere along the track at the cost of at most one matrix inversion.

This implementation allows computing the change in chisquared for adding or removing a hit anywhere along the track at the cost of one matrix inversion. The chisquared change is computed as the chisquared contribution of averaging the opposite-direction cached fit results of the KalSites on either side of the hit being added (removed), including (removing) the effect of that hit. If the hit is to be added (removed) permanently, the full fit can be recomputed with exactly half as many computational steps as required for the original full fit. The combination of a fast chisquared calculation plus a relatively fast refit makes this implementation an attractive tool for pattern recognition.

4 Maximum Likelihood Extension

The hit resolution of tracking detectors as well as multiple scattering and energy loss effects are typically non-Gaussian. Because the Kalman filter uses a linear estimation algorithm, it is not possible to exploit knowledge of the non-Gaussian parent beyond its RMS in the fit. This is a general property of all 'progressive' methods.

To attempt an improved tracking algorithm in the presence of these effects, we have produced maximum likelihood based track fit. This uses a perturbative sequence of KalTrack fits of the same track to model non-Gaussian effects. This algorithm is applicable when the measurement or process noise parent distributions are known and can be expressed as a sum of Gaussians, $P(x) = \Sigma_i G(\bar{x}_i, \sigma_i, A_i : x)$, with $A_i >> A_{i+1}$. These parent distributions are associated with KalSite objects through their occurrence.

The likelihood function for the track parameters can then be written as the product of these parents, with appropriate linear algebra terms. Expanding this product gives a weighted sum of simple linear relations, each extracting a particular set of component Gaussians from every KalSite. Each linear relation can be solved as an individual Kalman fit. Typically the number of terms in this sum is enormous, but a good approximate solution can be formed as the weighted sum of the fit results of the first few terms.

To evaluate the performance of this algorithm, we used a simulation of data from the BaBar silicon tracker, which includes realistic non-Gaussian hit errors and material effects. We used as a tracking figure of merit the true resolution on a primary vertex derived from the track fit results. An improvement of roughly 10% on the vertex resolution using the maximum likelihood track fit over the Kalman fit is observed, at a cost of roughly twice the compute time. A more detailed description of this maximum likelihood method will be presented in a future publication.

References

[1] BaBar Technical Design Report, SLAC-REP-950457 (1995)

[2] P. Billoir, NIM A225 (1984) 352
R. Fruewirth, NIM A262 (1987) 444
J. Boudreau, 'The SLT Class Library',
http://cactus.phyast.pitt.edu/ joe/slt.html

[3] Steve Schaffner, talk given in this session.