Alignment with tracks fitted with a Kalman filter

LHC Alignment Workshop, 25/06/2007

Wouter Hulsbergen (CERN/LBD)

introduction

- good reasons to use **same track model** in calibration and reconstruction
  - track model and calibration are not independent
  - consistency is more important than correctness!
- practically all modern experiments use a Kalman filter for track fitting
  - one important advantage is efficiency in dealing with multiple scattering
- it has been said that Kalman filter track fit is unsuitable for alignment
  - tracks that come out of the K-filter usually have incomplete covariance matrix
- in this talk, I'll discuss in reasonable detail
  - an alternative formulation of the minimum chisquare formalism for alignment
  - how to make the output of the Kalman filter suitable for alignment
  - how to include vertex and mass constraints
- this is all 'theory': I have no real results to present!

minimum chisquare fit

- define a track chisquare as

$$\chi^2 = \sum_{\text{hits i}} \left(\frac{m_i - h_i(x)}{\sigma_i} \right)^2$$

where

  - m → measurement, σ → measurement error
  - x → track parameters, usually 5
  - h → measurement model

- we can also write this in a matrix notation

$$\chi^2 = r^T V^{-1} r$$

  - r = m – h(x) → residual vector
  - V → measurement covariance matrix (usually diagonal)

- the 'least squares estimator' is the value for x that minimizes chisquare

minimum chisquare fit (II)

- the condition that the chisquare is minimal wrt ‘x’ is

$$\boxed{0 \equiv \frac{\mathrm{d}\chi^2}{\mathrm{d}x} = -2H^T V^{-1} r}$$

H=dh(x)/dx

N equations

usually non-linear in x

- solution can be obtained by linearizing the measurement model
  - start with some value $x^{(0)}$, calculate first derivative
  - calculate also second derivative (neglect d^2r/dx^2)

$$\boxed{\frac{\mathrm{d}^2\chi^2}{\mathrm{d}x^2} = 2H^T V^{-1} H}$$

NxN matrix

  - obtain new estimate of parameters with

'Newton-Raphson'

$$\boxed{x^{(1)} = x^{(0)} - \left(\frac{\mathrm{d}^2\chi^2}{\mathrm{d}x^2}^{(0)} \right)^{-1} \frac{\mathrm{d}\chi^2}{\mathrm{d}x}^{(0)}}$$

$$\boxed{\mathrm{Cov}(x) = 2 \left(\frac{\mathrm{d}^2\chi^2}{\mathrm{d}x^2}^{(0)} \right)^{-1}}$$

- if h(x) is not a linear model (H is not constant): use iterations

chisquare minimization for alignment

- suppose now, that we have
  - a sample of independently reconstructed tracks
  - a set of calibration constants 'alpha' common to the tracks
- we would like to minimize a total chisquare

$$\boxed{\chi^2 = \sum_{\text{tracks j}} \left(r^T V^{-1} r \right)_j}$$

with respect to both alpha and all track parameters

- following procedure outlined on previous slides. two scenarios:

1. minimize for x and alpha simultaneously on large sample of tracks
   - unpractical, because too many parameters
2. minimize every track to x first, then alpha on a large sample of tracks
   - keep track of dependence of x on alpha through total derivative

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}\alpha} = \frac{\partial}{\partial \alpha} + \frac{\partial x}{\partial \alpha} \frac{\partial}{\partial x}}$$

chisquare minimization for alignment

- calculate dx/dα from requirement that track chisquare remains minimal

$$0 = \frac{\mathrm{d}}{\mathrm{d}\alpha}\frac{\partial \chi^2}{\partial x} = \frac{\partial^2 \chi^2}{\partial \alpha \partial x} + \frac{\mathrm{d}x}{\mathrm{d}\alpha}\frac{\partial^2 \chi^2}{\partial x \partial x} \quad \Rightarrow \quad \frac{\mathrm{d}x}{\mathrm{d}\alpha} = -\frac{\partial^2 \chi^2}{\partial \alpha \partial x}\left(\frac{\partial^2 \chi^2}{\partial x \partial x}\right)^{-1}$$

- now calculate 'total derivatives' of chisquare to alpha

$$\frac{\mathrm{d}\chi^2}{\mathrm{d}\alpha} = 2\sum_{\text{tracks}} \frac{\partial r}{\partial \alpha}^T V^{-1}\left(V - HCH^T\right)V^{-1}\, r$$

$$\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2} = 2\sum_{\text{tracks}} \frac{\partial r}{\partial \alpha}^T V^{-1}\left(V - HCH^T\right)V^{-1}\frac{\partial r}{\partial \alpha}$$

C = Cov(x)

- these formulas give the least squares estimator for alpha
- same result as in Blobel and Kleinwort (2002), Bruckman et al (2005), etc

minimum chisquare condition is 'local'

- it seems as if derivative to one parameter depends on each hit on track

$$\frac{d\chi^2}{d\alpha} = 2 \sum_{\text{tracks}} \frac{\partial r}{\partial \alpha}^T V^{-1} \left(V - HCH^T \right) V^{-1} r$$

this matrix correlates derivatives for module 'i' with hits in module 'j'

- however, if the track chisquare is at its minimum

$$H^T V^{-1} r = 0 \quad \Rightarrow \quad \frac{d\chi^2}{d\alpha} = 2 \sum_{\text{tracks}} \frac{\partial r}{\partial \alpha}^T V^{-1} r$$

diagonal

- hence, the first derivative is 'local': only partial derivatives count
- why is this relevant? if there are other contributions to X^2, e.g.
  - multiple scattering constraints
  - hits in a reference system
  - vertex constraints

then we do not need to include those in the residual vector 'r'

Including multiple coulomb scattering

- in a global track fit:
  - scattering angles explicitely included in track model
  - chisquare gets extra terms to constrain scattering angle

expected angle: $\hat{\vartheta}=0$

$$\chi^2 = \sum_{\text{hits } i} \frac{(m_i - h_i(x,\theta))^2}{V_{ii}} + \sum_{\text{scat.angles } j} \frac{(\hat{\theta}_j - \theta_j)^2}{\Theta_{jj}}$$

variance of $\hat{\vartheta}$ (function of type and momentum)

  - in the Kalman fit, it looks different, but it is essentially the same

- easiest way to propagate into alignment formalism: change the symbols
  - **x**: track parameters, including multiple scattering angles
  - **m**: measurement vector, including $\hat{\vartheta}$
  - **V**: covariance matrix for the measurements, including Θ
  - **r**: residual vector, including residuals for scattering angles

- master formulas for alignment chisquare minimization do not change

summarizing the formalism

- master equations for the derivatives

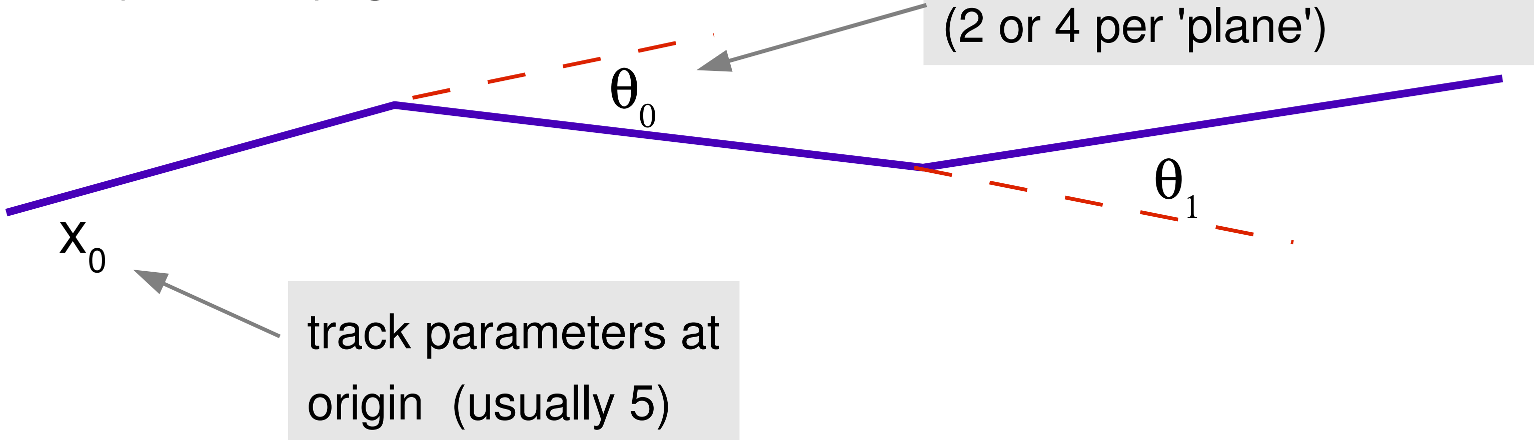
$$\frac{\mathrm{d}\chi^2}{\mathrm{d}\alpha} = 2 \sum_{\text{tracks}} \frac{\partial r}{\partial \alpha}^T V^{-1} r$$

$$\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2} = 2 \sum_{\text{tracks}} \frac{\partial r}{\partial \alpha}^T V^{-1} \overbrace{\left(V - HCH^T\right)}^{\substack{\text{covariance matrix} \\ \text{for (biased) residuals} \\ \text{(usually called R)}}} V^{-1} \frac{\partial r}{\partial \alpha}$$

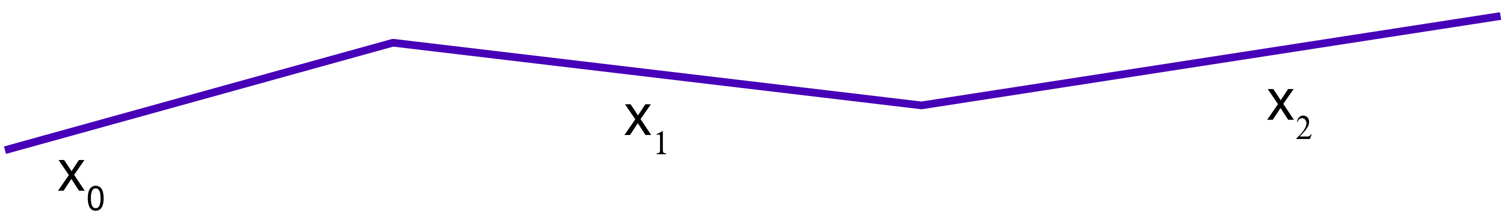
- ingredients
  - residuals **r**
  - measurement covariance matrix **V** (diagonal)
  - derivatives of residuals to track parameters **H**
  - track covariance matrix **C**
  - derivatives of residuals to alignment parameters **∂r/∂α**
- this is **nothing new**, but you might still like this write-up: Bocci and Hulsbergen, ATL-INDET-PUB-2007-009.

track models: 'global' versus 'kalman'

- model used in (ATLAS) 'global' track fit



- model used in usual 'Kalman-filter' track fit



- these models are not necessarily different: they should represent similar trajectories (otherwise, one of them is probably not optimal)
- these models are also not bound to the fitting method
  - we could write down a K-filter with the global track fit model and vice versa
  - it would just be rather inefficient to do so

track fitting: 'global' versus 'kalman'

- global fit method
  - covariance matrix of all track parameters calculated
  - used for alignment in e.g. MILLIPEDE, Atlas' 'Global Chisquare'
- Kalman filter
  - track model chosen such that not all track parameter correlations need to be calculated
  - global covariance matrix **C** is incomplete: covariance matrix computed for every state vector $\mathbf{x}_i$ but correlations are missing
  - problem for application of closed-form alignment procedure
- challenge: calculate the missing parts
  - hope that it isn't too hard
  - hope that it isn't too (CPU) time consuming: matrix C can be very large

calculation of 'global' covariance C in Kalman filter

- math isn't more difficult than K-filter itself, but a bit hard to explain unless you are already familiar with Fruhwirth's notation
  - will still sketch calculation and ingredients
  - since you'll probably get lost anyway, I'll rush through it
- strategy
  - step 1: covariance matrix of neighbouring states after 'prediction step'
  - step 2: covariance matrix of neighbouring states after 'smoother step'
  - step 3: extend to non-neighbouring states

$$C = \begin{pmatrix} 0 & 1,2 & 3 & 3 & 3 & 3 \\ & 0 & 1,2 & 3 & 3 & 3 \\ & & 0 & 1,2 & 3 & 3 \\ & & & 0 & 1,2 & 3 \\ & & & & 0 & 1,2 \\ & & & & & 0 \end{pmatrix}$$

- matrix of 5x5 matrices
- diagonal entries come out of standard K-filter

step 1: covariance for 'filtered' state k-1 and 'predicted' state k

- kalman filter prediction (for linear models)

$$x_{k-1} \longrightarrow x_k^{k-1} = F_{k-1} x_{k-1}$$

filter state after hit k-1 contains all information of hits [1,...,k-1]

prediction at hit k

- cov. matrix for filtered state 'k-1' and prediction state 'k'

scattering ('noise') enters here

$$C = \begin{pmatrix} C_{k-1} & C_{k-1} F_{k-1}^T \\ F_{k-1} C_{k-1} & F_{k-1} C_{k-1} F_{k-1}^T + Q \end{pmatrix}$$

- this is trivial, except maybe the bit about the 'noise'

step 2: covariance of neigbouring smoothed states

- final result of the kalman filter consists of 'smoothed' states
  - state after information of all hits is processed
  - for alignment we need the correlation between smoothed states
  - Fruhwirth's notation for smoothed states: state $\mathbf{x}_k^n$, covariance matrix $\mathbf{C}_k^n$

- two strategies for 'smoothing'
  - smoothing formalism (see e.g. Fruhwirth, 1989)
  - bi-direction K-filter: runs filters in both directions and 'average'

  though latter is more popular now, we'll use former, but it doesn't matter

- suppose that we have a procedure to obtain the state at node 'k' after adding all remaining hits {k, ..., n}
  - how do we 'back-propagate' information from {k,...,n} to state k-1?
  - what happens to the covariance for states k-1 and k?

intermezzo: propagation formula

- suppose we have two observables ***(a,b)*** with covariance ***V***
- suppose we do something which makes that we know ***a*** better

$$a \longrightarrow \tilde{a} \qquad\qquad V_{aa} \longrightarrow \tilde{V}_{aa}$$

- we can propagate this knowledge to ***b*** using

$$\boxed{\begin{aligned} \tilde{b} &= b + V_{ab} V_{aa}^{-1} (\tilde{a} - a) \\ \tilde{V}_{bb} &= V_{bb} - V_{ba} V_{aa}^{-1} (V_{aa} - \tilde{V}_{aa}) V_{aa}^{-1} V_{ab} \\ \tilde{V}_{ab} &= \tilde{V}_{aa} V_{aa}^{-1} V_{ab} \end{aligned}}$$

- this is just another result of the least squares estimator
- formulas also work when ***a*** and ***b*** are vectors

step 2: covariance of neighbouring smoothed states (II)

- we apply the propagation formulas from the previous page to state 'k'
  - a = predicted state k,   a-tilde = smoothed state k
  - b = filtered state k-1
  - V_aa   = C_k^{k-1} --> covariance for predicted state k
  - V_aa-tilde = C_k^{n}   --> covariance for smoothed state k
- the result for the covariance matrix is

$$C_{k-1}^{n} \; = \; C_{k-1} \; + \; A_{k-1} \left(C_k^n - C_k^{k-1} \right) A_{k-1}^T$$

cov. matrix for
state k-1
(see e.g. Fruhwirth)

$$\boxed{C_{k-1,k}^{n} \; = \; A_{k-1} C_k^n}$$

correlation
(my notation)

- where I used the definition of the *smoother gain matrix* (see Fruhwirth)

$$\boxed{A_{k-1} \; = \; C_{k-1} F_{k-1}^T \left(C_k^{k-1} \right)^{-1}}$$

step 3: covariance for all smoothed states

- so, we calculated the correlation between two neighbouring states
  - 1st 'off-diagonal' in the global covariance matrix C
  - how do we calculate the correlation between other states?
- consider states k-2 and k
  - correlation can only occur *through* state k-1
  - then it takes the following form (not entirely trivial)

$$C^n_{k-2,k} = C^n_{k-2,k-1} \left(C^n_{k-1}\right)^{-1} C^n_{k-1,k}$$

- now consider the next diagonal

$$C^n_{k-3,k} = C^n_{k-3,k-2} \left(C^n_{k-2}\right)^{-1} C^n_{k-2,k-1} \left(C^n_{k-1}\right)^{-1} C^n_{k-1,k}$$

- looks horrible enough, but we can reuse what we have already calculated

$$C^n_{k-3,k} = C^n_{k-3,k-2} \left(C^n_{k-2}\right)^{-1} C^n_{k-2,k}$$

final result

- recursive expressions for all diagonals in the matrix C

$$\boxed{C^n_{k-1,l} \;=\; A_{k-1} C^n_{k,l} \qquad k \leq l}$$

  - this is one multiplication of two 5x5 matrices for every off-diagonal 5x5 matrix

- requires 'smoother gain matrix' at every node

$$A_{k-1} \;=\; C_{k-1} F^T_{k-1} \left(C^{k-1}_k \right)^{-1} \;=\; \left(F_{k-1} \right)^{-1} \left(C^{k-1}_k - Q_k \right) \left(C^{k-1}_k \right)^{-1}$$

- to compute this matrix you need to have access to
  - all transport matrices (F)
  - all noise matrices (Q)
  - either the (forward) predicted result or the filtered result

- lucky in LHCb: default track fit keeps all this information with track

implementation for LHCb

- implemented calculation of matrix C in a Gaudi tool
  - it operates on 'fitted' tracks, using information stored in the K-filter nodes
- CPU time consumption
  - calculation not complicated, but CPU intensive
    - LHCb tracks have typically 50 hits
    - (symmetric) matrix C has typically ~ 30000 entries
  - surprisingly enough, time consumption not a big deal
    - O(1 ms) per track
    - relatively little compared to track fit itself
    - thanks to highly optimized matrix algebra (ROOT::Math::SMatrix)
- next step: actually use in LHCb's alignment framework

efficiently dealing with vertex constraints

- vertex and mass constraints are useful for constraining alignment degrees of freedom that are poorly constrained by single tracks
  - e.g. elliptical distortions, 'clocking' effect in central detectors
  - multi-track constraints effectively connect parts of detector that are never traversed simultaneously by single track
- usual way of including such constraints is with dedicated track fits
  - tracks fits that fit two tracks simultaneously, using common parameters for track origin
  - track fits that include a 'point' constraint from a vertex determined with other tracks
- however, if the global covariance matrix of the track parameters is available, we can do these this more efficiently

efficiently dealing with vertex constraints (II)

- assume you have a vertex fit that
  - takes track parameters 'at origin' with covariance as input
  - gives back new track parameters + covariance for all tracks
- using formulas on slide 15, 'propagate' this to other track parameters
  - in global fit: propagate to scattering angles
  - in kalman fit: propagate to all other states along track
- this allows to calculate
  - 'updated' residuals for all tracks
  - full covariance for all residuals on all tracks
- advantage: fast and simple, no dedicated track fits needed
- see also ATL-INDET-PUB-2007-009 (formula's only, no application yet)

conclusions

- calculated complete covariance matrix for K-filter tracks
- assuming that
  - we would like to use the standard K-filter track fit for alignment
  - we care about multiple scattering
  - we care about correlations between residuals (closed-form, a la MILLIPEDE)

  then it is good to know that this is possible, at least on paper
- even if you do not care about these things, the result is still useful because it can also be used to add vertex constraints to the problem
  - interesting both for 'closed-form' and 'iterative' alignment procedure
  - interesting both with and without multiple scattering on the track

backup slides

Including multiple coulomb scattering (II)

- one more look at the first derivative

$$\frac{d\chi^2}{d\alpha} = 2\sum_{\text{tracks}} \frac{\partial r}{\partial \alpha}^T V^{-1}\left(V - HCH^T\right) V^{-1}\, r$$

residuals for scattering angles are here!

- do we really need to deal with the scattering angles explicitely? not if we use that the track is at minimum chisquare

$$\frac{d\chi^2}{d\alpha} = 2\sum_{\text{tracks}} \frac{\partial r}{\partial \alpha}^T V^{-1}\, r$$

because V is diagonal and only 'hits' depend on alpha, only hit residuals remain

- in other words: make sure you use the right formula for the first derivative; otherwise, things become really complicated