Rigorous integration of K-filter track fit into alignment procedure

LHC Alignment Workshop, 16/06/2009

Wouter Hulsbergen (NIKHEF)

Outline
- motivation
- master formulas for min. χ^2 alignment
- K-filter in alignment
- use of vertices and mass constraints

WH, arXiv:0810.2241, NIMA600:471-477 (2008)

Contents close to talk in previous alignment workshop, but now actually used for detector algorithm in LHCb

motivation

- practically all modern experiments use a Kalman filter for track fitting
  - application to track and vertex fitting described by Fruhwirth, 1989
  - most important advantage is efficiency in dealing with multiple scattering

- good reasons to use **same track model** in calibration and reconstruction
  - track model and calibration are not independent
  - often consistency is more important than correctness!

- however, K-filter track fit is not 'out-of-the-box' suitable for closed-form (millipede-like) alignment procedure
  - tracks that come out of the K-filter have an incomplete covariance matrix
  - will explain again why that is important
  - will then show how it can be fixed

summary of alignment formalism

- Millipede / closed form method / global X ^{2} method

"minimize a total χ^2

$$\chi^2(\alpha, \{x_i\}) = \sum_{\text{tracks } i} \chi_i^2(\alpha, x_i)$$

with respect to alignment parameters and all track parameters"

- leads to coupled set of non-linear equations

$$\frac{\partial \sum_i \chi_i^2}{\partial \alpha} = 0 \qquad \text{and} \qquad \forall_i \; \frac{\partial \chi_i^2}{\partial x_i} = 0$$

- solve with 'newton-raphson' (linearization)

summary of alignment formalism

- track parameters can be eliminated, leading to 'reduced' linear problem

“solve linearized system $\left.\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2}\right|_{\alpha_0} \Delta\alpha = -\left.\frac{\mathrm{d}\chi^2}{\mathrm{d}\alpha}\right|_{\alpha_0}$ for $\boldsymbol{\Delta\alpha}$”

- master equations for the 'total' derivatives

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

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

covariance matrix for (biased) residuals (usually called $\boldsymbol{R}$)

- ingredients for each track
  - vector of residuals $\boldsymbol{r}$
  - measurement covariance matrix $\boldsymbol{V}$ (diagonal)
  - derivatives of residuals to track parameters $\boldsymbol{H}$
  - **track parameter covariance matrix $\boldsymbol{C}$**
  - derivatives of residuals to alignment parameters $\boldsymbol{\partial r/\partial\alpha}$

notation from Fruhwirth

summary of alignment formalism

- track parameters can be eliminated, leading to 'reduced' linear problem

“solve linearized system $\left.\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2}\right|_{\alpha_0} \Delta\alpha = -\left.\frac{\mathrm{d}\chi^2}{\mathrm{d}\alpha}\right|_{\alpha_0}$ for $\boldsymbol{\Delta\alpha}$”

- master equations for the 'total' derivatives

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

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

this term is exactly 0 if track fit at minimum X^2,
i.e. if you take the residuals from a fitted track

summary of alignment formalism

- Millipede / closed form method / global X^2 method

“solve linearized system
$$\left.\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2}\right|_{\alpha_0} \Delta\alpha = -\left.\frac{\mathrm{d}\chi^2}{\mathrm{d}\alpha}\right|_{\alpha_0}$$

- master equations for the derivatives

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

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

- this 'simplication' (which it isn't) is essential for what follows: it means you do not need to add explicitely the information from
  - residuals from scattering angles
  - hits in external detector
  - vertex constraints

$\Bigg\}$ constraints already contained in 'active' residuals and in track covariance C

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:
ϑ-hat=0

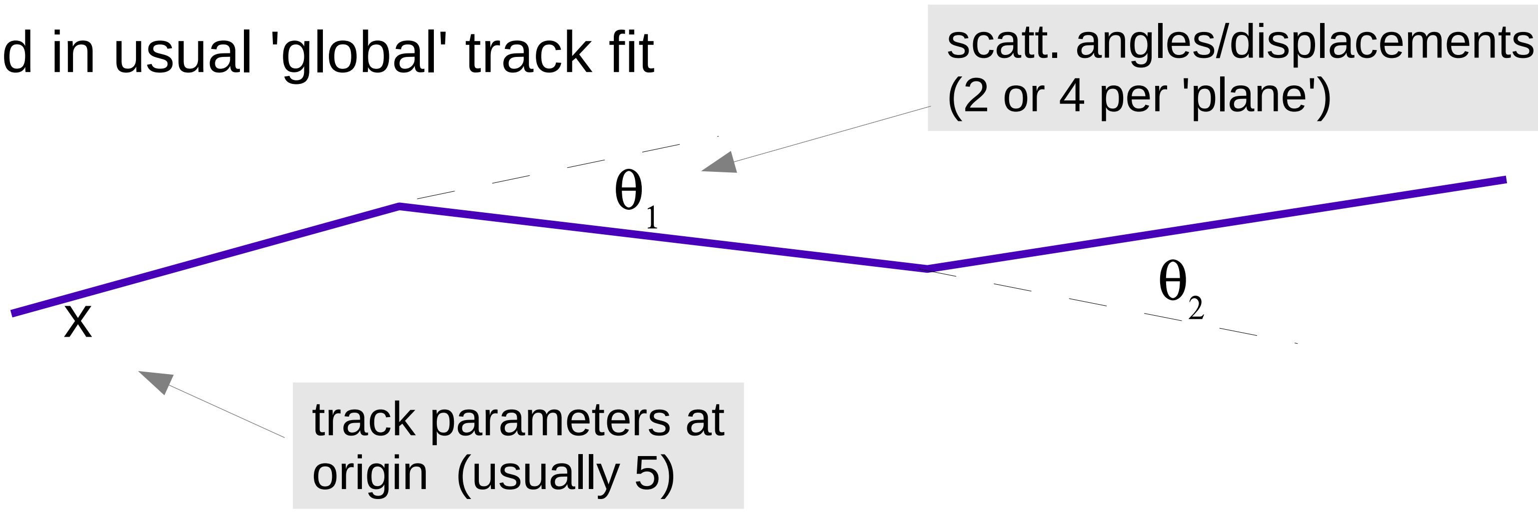
$$\chi^2 = \sum_{\text{hits } i} \frac{(m_i - h_i(x,\theta))^2}{V_{ii}} + \sum_{\text{scat.angles } j} \frac{(\hat{\theta}_j - \theta_j)^2}{\Theta_{jj}}$$

variance of ϑ-hat
(function of type and momentum)

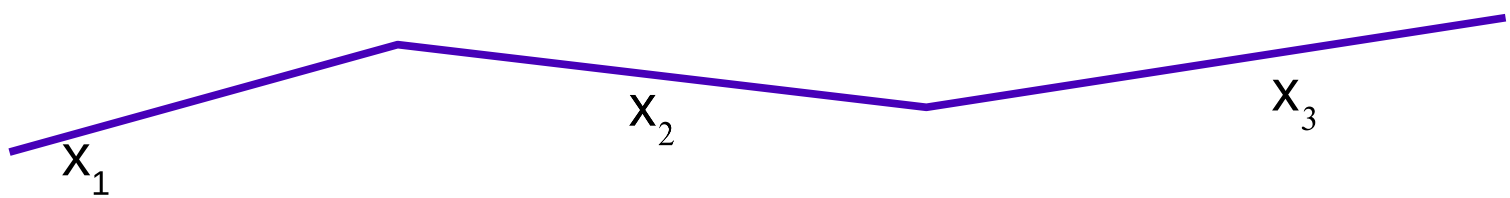
  - in the Kalman fit, it looks different, but it is essentially the same
- easiest way to propagate into alignment formalism: change the symbols
  - **x**: track parameters, including multiple scattering angles
  - **m**: measurement vector, including ϑ-hat
  - **V**: covariance matrix for the measurements, including Θ
  - **r**: residual vector, including residuals for scattering angles
- master formulas for alignment chisquare minimization do not change

track models: 'global' versus 'kalman'

- model used in usual '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 $\boldsymbol{C}$ is incomplete: covariance matrix computed for every state vector $\boldsymbol{x}_i$ but correlations are missing

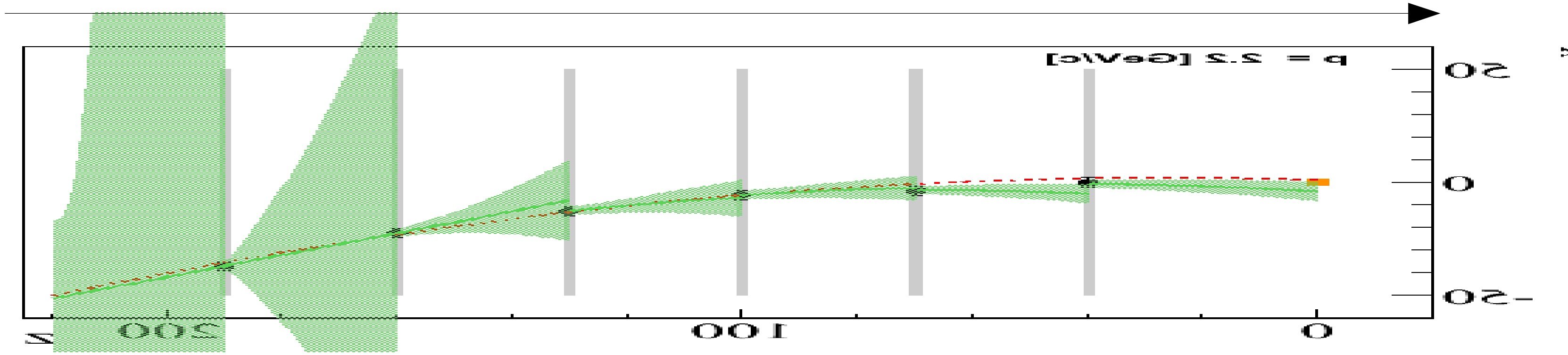
$$C = \begin{pmatrix} C_1 & C_{1,2} & C_{1,3} & \cdot & \cdot & C_{1,n} \\ & C_2 & C_{2,3} & \cdot & \cdot & C_{2,n} \\ & & C_3 & \cdot & \cdot & C_{3,n} \\ & & & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot \\ & & & & & C_n \end{pmatrix}$$

missing in standard K-fit

- off-diagonal elements essential for closed-form alignment procedure
  - will just give you a flavour of what it takes to compute them
  - for details see arXiv:0810.2241

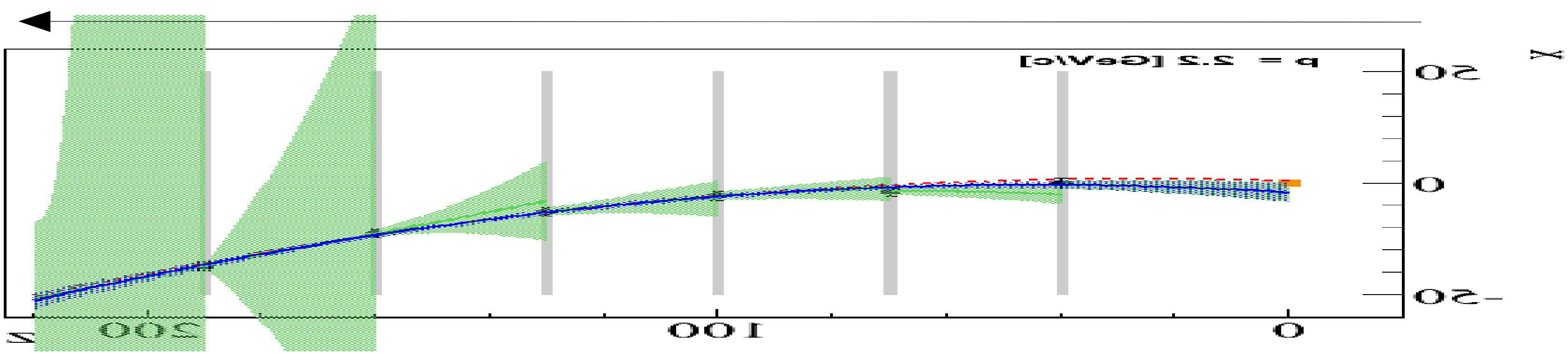
kalman fit reminder

- step 1: filtering



filtered states $\boldsymbol{x}_k$ with $\boldsymbol{C}_k$ contain information from all *preceding* measurements

- step 2: smoothing
  - update states at earlier measurements with what happened later in the filter



smoothed states $\boldsymbol{x}_k^{(n)}$ with $\boldsymbol{C}_k^{(n)}$ contain information from *all* measurements

covariance of smoothed states

- covariance matrix of smoothed state computed with (e.g. Fruhwirth 1989)

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

with *smoother gain matrix*

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

- my tiny contribution: covariance between neighbouring smoothed states

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

- correlation between any two states computed recursively

$$C_{k-1,l}^{n} \;=\; A_{k-1} C_{k,l}^{n} \qquad k \leq l$$

that's it! that's all!

- note: entire smoothing is 'trivial' if there is no process noise (scattering)
  - in that case $A_{k-1} \;=\; \left(F_{k-1}\right)^{-1}$
  - only single independent state, all states '100%' correlated

intermezzo: bi-drectional fit versus smoother gain

- two approaches to smoothing in K-filter
  - smoother gain formalism: propagate downstream information recursively by computing how much information was added (e.g. Fruhwirth)
  - bi-directional fit: run filter in 2 directions, use weighted average (Brown and Roberts?)
- bi-directional fit more popular these days
  - faster, allows for interpolation, allows for 'smoothing on demand'

intermezzo: bi-drectional fit versus smoother gain

- two approaches to smoothing in K-filter
  - smoother gain formalism: propagate downstream information recursively by computing how much information was added (e.g. Fruhwirth)
  - bi-directional fit: run filter in 2 directions, use weighted average (Brown and Roberts?)
- bi-directional fit more popular these days
  - faster, allows for interpolation, allows for 'smoothing on demand'
- but … even for linear fit, results not exactly identical
  - this is because of that 'nagging' feature of K-filter: it relies on a 'seed'
    - seed: finite cov. matrix even before any measurements are processed
    - in bi-directional fit, seed information added on both sides
    - with smoother gain, only on one side

intermezzo: bi-drectional fit versus smoother gain

- two approaches to smoothing in K-filter
  - smoother gain formalism: propagate downstream information recursively by computing how much information was added (e.g. Fruhwirth)
  - bi-directional fit: run filter in 2 directions, use weighted average (Brown and Roberts?)
- bi-directional fit more popular these days
  - faster, allows for interpolation, allows for 'smoothing on demand'
- but … even for linear fit, results not exactly identical
  - this is because of that 'nagging' feature of K-filter: it relies on a 'seed'
    - seed: finite cov. matrix even before any measurements are processed
    - in bi-directional fit, seed information added on both sides
    - with smoother gain, only on one side
- formulas on previous slides really only work for smoother gain formalism
  - diagonal of C (computed in fit) must be **consistent** with off-diagonal
    - otherwise, run in all sorts of precision problems with correlations
  - luckily LHCb track fit implements both smoothing procedures

implementation and complexity

- summarizing: to use K-filter tracks in alignment
  - use smoother gain formalism for smoothing, at least to compute cov-matrices
  - store smoother gain matrix
  - use recursive formula to compute full matrix $\boldsymbol{C}$ for tracks in alignment
- complexity
  - thanks to recursive formula, complexity is modest
  - every off-diagonal element in matrix $\boldsymbol{C}$ is result of inner product of two $\boldsymbol{5D}$ vectors
  - of course, matrix $\boldsymbol{C}$ is BIG: dimension is $(5N)^2$
- surprisingly enough, time consumption not a big deal in LHCb
  - O(1 ms) per track (about ~30 hits per track)
  - comparable to time of single track fit
  - thanks to highly optimized matrix algebra (ROOT::Math::SMatrix)

application in LHCb

- traditional approach: 'loca'l alignment, followed by 'global' alignment
  - problem: weak modes → how do you make the sub systems 'consistent'?
- now use standard LHCb K-filter track fit in alignment
  - can align all parts of tracking system (VELO,TT,IT,OT,MUON)
    - simultaneously, or one-at-a-time, or using one system as reference
    - at any granularity
  - residuals, track derivatives all computed by track fit
    - proper scattering model, proper magnetic field integration
    - only extra ingredient is alignment derivatives: ~10 lines of code!
- several 'external constraints' are implemented
  - penalty terms using reference (survey) geometry, lagrange constraints, eigenvalue cuts
- most important advantage: because we use the standard track fit
  - we align with the track model used in physics
  - there is little extra alignment code → less mistakes, maintenance

intermezzo: normalizing eigenvalues

- 'eigenvalue analysis' very useful to identify weak modes
  - small eigenvalue means 'statistically' poorly constrained mode
  - but what is small? how do you compare translations and rotations?
- proposal by A. Bocci and WH (ATL-INDET-PUB-2007-009)
  - introduce a 'rescaling' of linear system with coordinate errors

original system: $Ax = b$

$$A \equiv \frac{1}{2} \frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2} \qquad b \equiv -\frac{1}{2} \frac{\mathrm{d}\chi^2}{\mathrm{d}\alpha}$$

rescaled system: $(S A S^T)\,(S^{T^{-1}} x) \;=\; S b$

intermezzo: normalizing eigenvalues

- 'eigenvalue analysis' very useful to identify weak modes
  - small eigenvalue means 'statistically' poorly constrained mode
  - but what is small? how do you compare translations and rotations?
- proposal by A. Bocci and WH (ATL-INDET-PUB-2007-009)
  - introduce a 'rescaling' of linear system with coordinate errors

original system: $$Ax = b$$

$$A \equiv \frac{1}{2}\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2} \qquad b \equiv -\frac{1}{2}\frac{\mathrm{d}\chi^2}{\mathrm{d}\alpha}$$

rescaled system: $$(SAS^T)\,(S^{T^{-1}}x) = Sb$$

  - choose diagonal scaling matrix $$S_{ii} = \sqrt{\frac{n_X}{A_{ii}}}$$

number of hits contributing to this parameter

diagonal of A

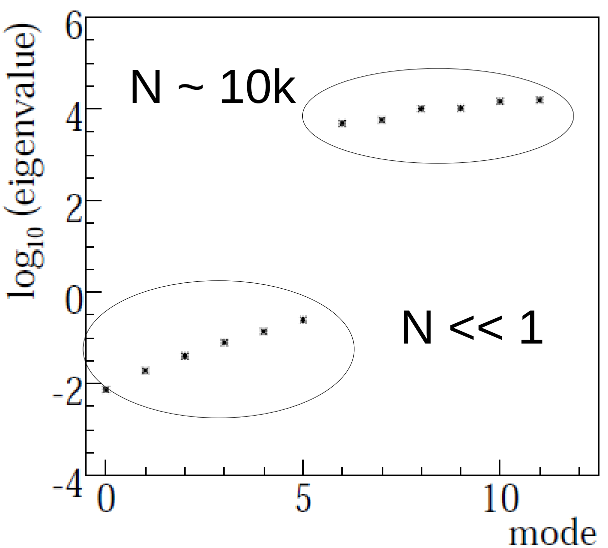
- with this normalization **eigenvalue** to good approximation equal to **number of hits contribution to alignment mode**
  - this gives absolute scale to eigenvalues

validating the implementation

- even 'trivial' algebra may require lot's of code to implement
  - exploited two methods to check implementation
- a) compute covariance matrix 'R' of track residuals numerically
  - move each measured coordinate 'i' by distance 'epsilon' perpendicular to track
  - row 'i' of covariance matrix follows from
  - where 'δr' is the change in residual

$$R_{ij} = -\frac{\delta r_j^{(i)}}{\delta r_i^{(i)}} R_{ii},$$

- b) study eigenvalues: if you see the global weak modes, you must have done things right!



graph shows eigenvalues for alignment of two LHCb Velo halves without reference system

- 12 alignment parameters
- 6 global unconstrained dofs
- about 10k tracks + primary vertices
- scaling recipee from Bocci and Hulsbergen

note that the small eigenvalues are not numerically zero: that's mostly because of the Kalman seed!

efficiently dealing with vertex constraints

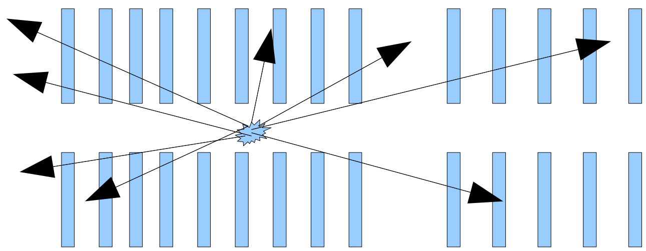
- vertex and mass constraints are useful for constraining alignment degrees of freedom that are poorly constrained by single tracks
  - multi-track constraints effectively connect parts of detector that are never traversed simultaneously by single track
  - e.g. elliptical distortions, 'clocking' effect in central detectors
- 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
- if global covariance matrix of track parameters is known, we can do this much more efficiently

efficiently dealing with vertex constraints (II)

- vertex fit
  - input: track parameters 'at start of track' with covariance matrix
  - output: new track parameters + covariance + correlations for all tracks
- 'propagate' new constraint back to other parameters using matrix C
  - in global fit: propagate to scattering angles
  - in kalman fit: propagate to all other states along track
- results in
  - 'super' matrix ***C*** that represents cov matrix of all tracks simultaneously
  - full covariance for all residuals on all tracks, including inter-track correlations
  - 'updated' residuals for all tracks
- advantage: fast and simple, no dedicated track fits needed
  - see arXiv:0810.2241
- used in LHCb in two ways
  - link two halves of VELO detector by using primary vertices
  - fix weak mode in curvature by using J/psi mass constraint

using primary vertex constraints in LHCb

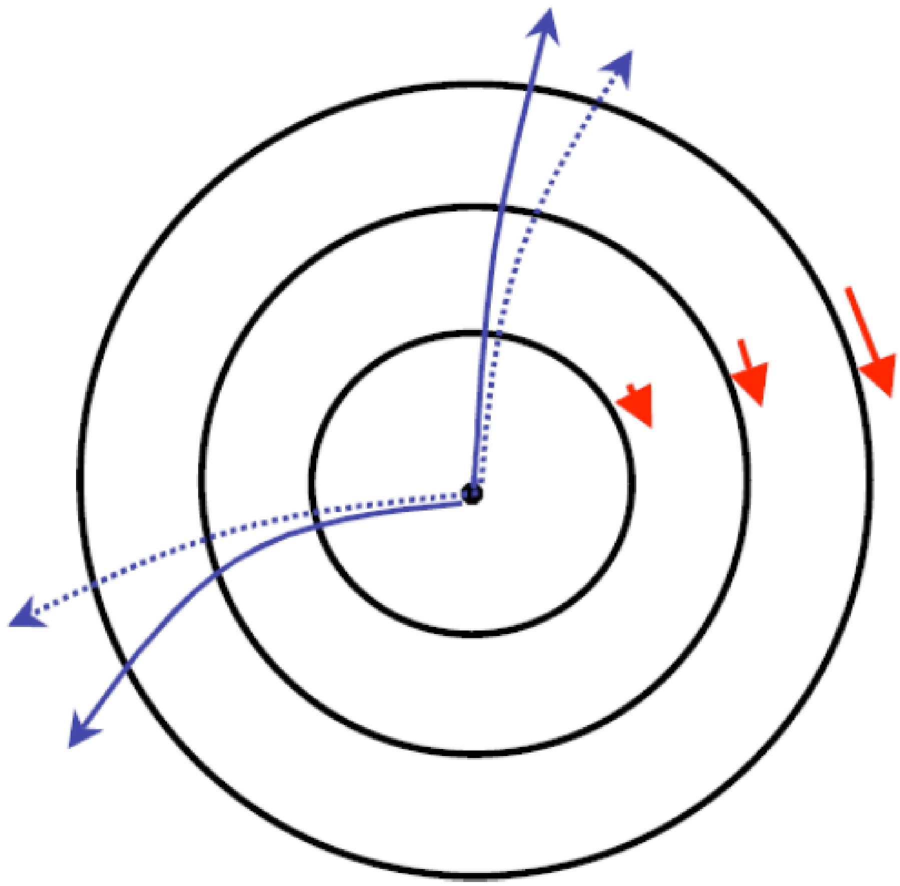
- primary vertex constraints very useful to constrain LHCb VELO
  - links left and right half of system
  - links forward and backward part of system

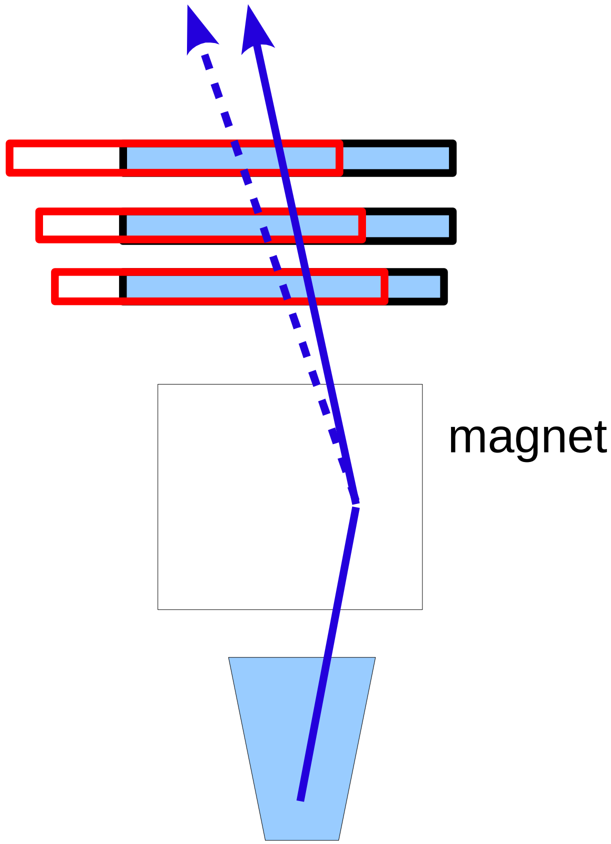


- use reconstructed primary vertices in alignment
  - O(20) tracks per vertex
  - can combine information from >1000 hits!
  - CPU is an issue
    - computing correlation between 1000 track states is expensive!
    - solve by artificially reducing number of tracks per fitted vertex

using a mass constraint

inconvenient weak mode in both central and forward detectors: curvature bias



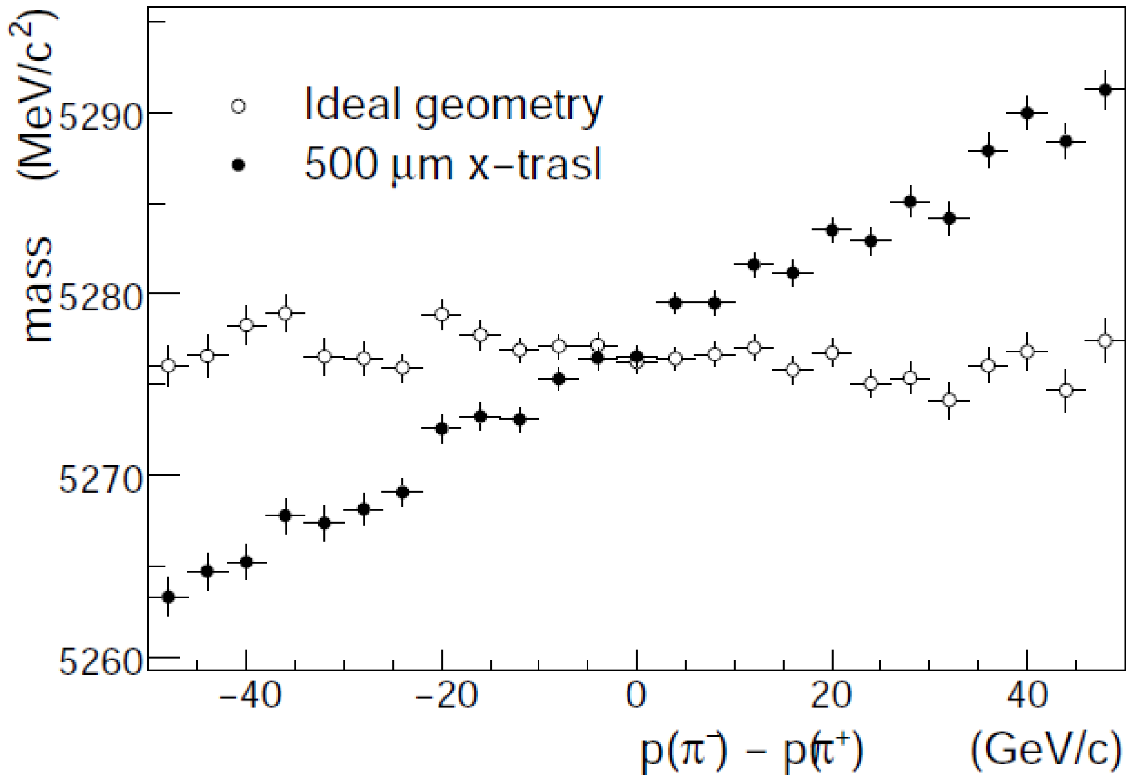


central detector: 'clocking'
(figure taken from Alison 2008)

forward detector: shearing or rotation wrt magnet axis

using mass constraint (II)

- easy to see by plotting two-prong mass as function of momentum (or p_T) difference



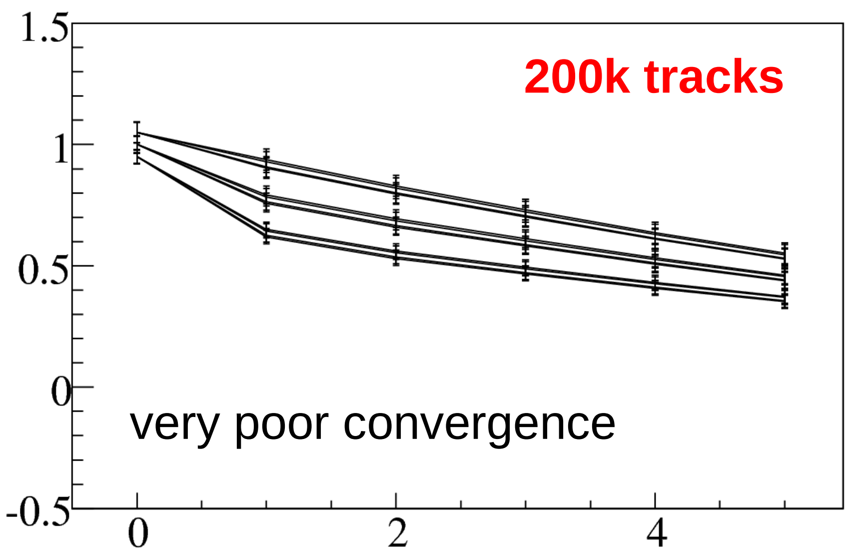
mean of B->pipi mass as function of pion momentum difference in LHCb

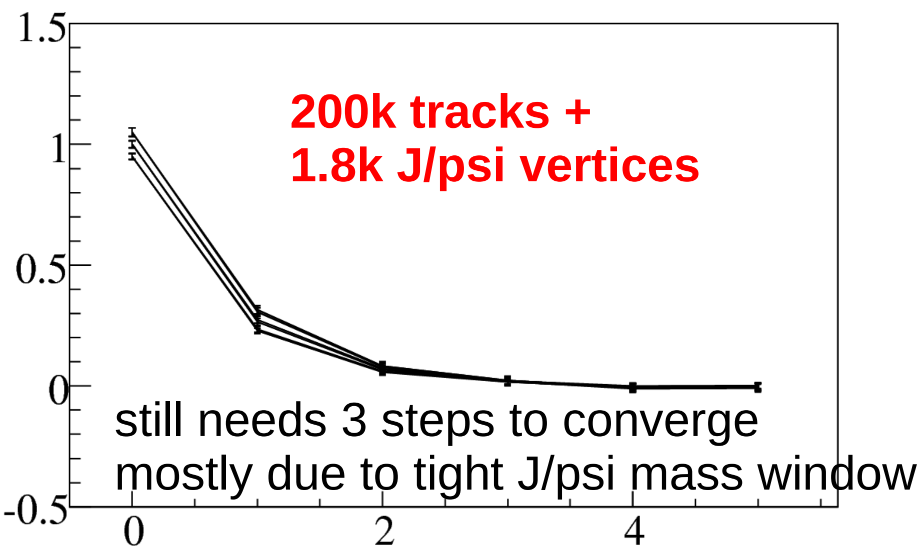
(from E. Simioni, PhD thesis)

- solution, at least in theory:
  - identify $J/\Psi \rightarrow \mu^+\mu^-$ decays (or $Z^0 \rightarrow \mu^+\mu^-$)
  - vertex with mass constraint
  - update track residuals with procedure outlined before

using a mass constraint (III)

- first test using simulated J/psi → mumu events
  - move LHCb T stations by ~1mm in x-coordinate (bending plane)
  - align with and without J/psi mass constraint vertex
- displacement in x of (12) OT layers as function of iteration





- main complication
  - signal in real data will not be as clean -->
  - (asymmetric) background under j/psi mass leads to bias

References

- Math
  - W. Hulsbergen, “The global covariance matrix of tracks fitted with a Kalman filter and an application in detector alignment”, NIMA600:471-477 (2008)
- Procedure
  - J. Amoraal et al, proceedings CHEP 2009
- Application
  - L. Nicolas et.al., “First studies of T-station alignment with simulated data”, CERN-LHCB-2008-065
  - L. Nicolas et al., “Alignment of LHCb tracking stations with tracks fitted with a Kalman filter”,CERN-LHCB-2008-065
  - results reported in several talks at this workshop

conclusions

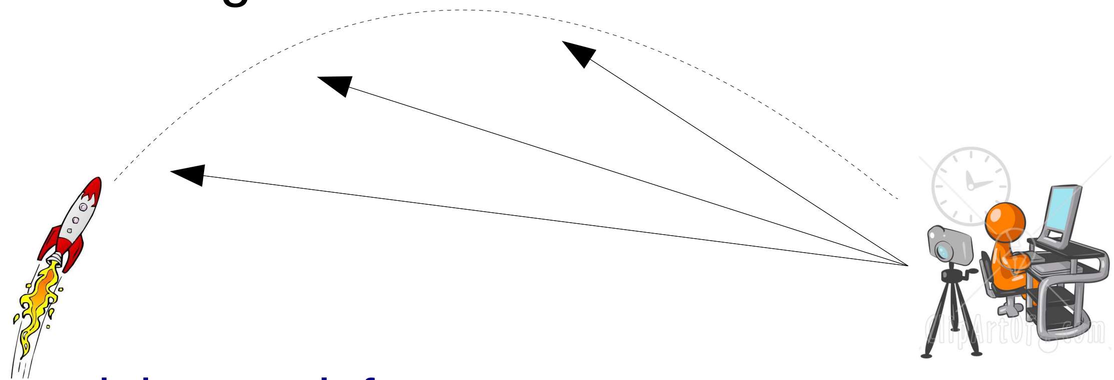
- advocated K-filter track for use in alignment
  - ease in dealing with multiple scattering
  - consistency: use same track model for calibration and physics analysis
- calculated small addition to standard K-filter formalism to compute the 'complete' covariance matrix
  - use smoother gain formalism
  - expressed in single recursive formula
  - each element is inner product of two 5D vectors
- used extensively in LHCb
  - alignment of all DOFs in LHCb tracking alignment (VELO, TT, IT, OT, MUON) with single tracks
  - once you have track fit, little extra math … most of the alignment code is just bookkeeping
- vertex and mass constraints can be added without refitting tracks if you have access to global track covariance matrix
  - illustrated with use of primary vertices and mass constraints

backup slides

linearizing Kalman track fit with reference trajectory

- for a **linear model** K-filter is a least-squares estimator
  - if seed covariance sufficiently large, result of **Kalman fit identical to global fit**
- however, this is not longer the case if the model is not linear
  - both (extended) K-filter and global fit use linearized model
  - results only identical if linearization performed about same reference solution
- in global fit: linearize around solution to previous iteration
  - compute derivatives using x_0
  - compute x_1, iterate
- naïve solution in extended Kalman fit: linearize around the *prediction*
  - I'll explain in a minute what I mean with that
  - this was for sure done in Hera-B, LHCb in early stages, and I think also in ATLAS

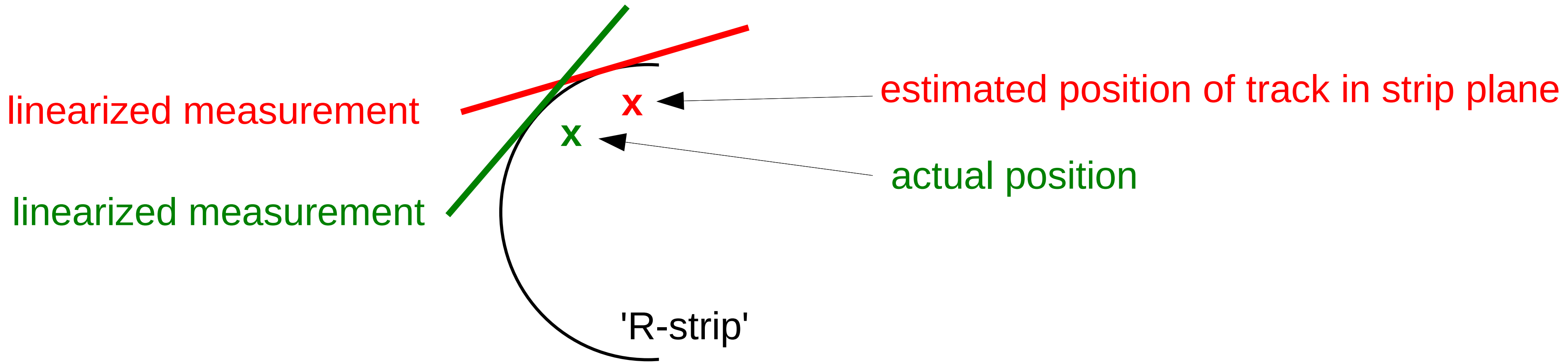
- suppose you are tracking a rocket with a K-filter



  - obviously, you need the result fast
    - so, you use a K-filter to add information as it comes
    - there is only a single iteration
  - to estimate derivatives at the next measurement, you use the best known trajectory, which is the prediction from the previous measurement
    - methods exist to deal more efficiently with non-linearities, but not important here
- my point: use case in HEP track fitting is entirely different
  - *track finding* before *track fitting*
  - already know the trajectory pretty well, or at least parts of it
  - trajectory that comes out of pattern reco is usually much better estimate of trajectory than trajectory after filtering a few hits

why naïve linearization fails

- example of non-linearity in LHCb track fit
  - LHCb tracker combines both x-y (T,TT) and r-phi geometry
  - track model is conventional xy model for a spectrometer (x,y,tx,ty,q/p)
  - velo R-strips are circles: projection strongly non-linear in this track model



- if estimate used for linearization is wrong, get entirely wrong derivatives
- in LHCb prediction can be VERY wrong
  - for the 'upstream' filter, the first R-hit may be filtered after extrapolating from 6 meters away, on other side of magnet
  - → basically no constraint in bending (xz) plane, and poor resolution in yz
- other important non-linearity: inhomogenous magnetic field

solution: linearize around 'reference' trajectory

- solution to this problem: don't linearize around the prediction but around a 'global' reference trajectory
  - in first iteration, take estimate of track parameters from pattern reco
    - compute propagation/measurement derivatives and residuals using reference
    - the fitted, linear track model is the **difference with reference trajectory**
  - in further iterations, use the 'smoothed' trajectory as reference
- note: this is exactly how you would compute derivatives in a global fit
    - **K-fit with reference is** again **identical to the global fit**
    - more stable than naïve linearization
- I have seen this approach first in Babar (see e.g. D. Brown, CHEP2000), but apparently, it was already used in Delphi, by Billoir and Fruhwirth

Including multiple coulomb scattering (II)

- one more look at the first derivative

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

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

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

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

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

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

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

using primary vertex constraints in LHCb