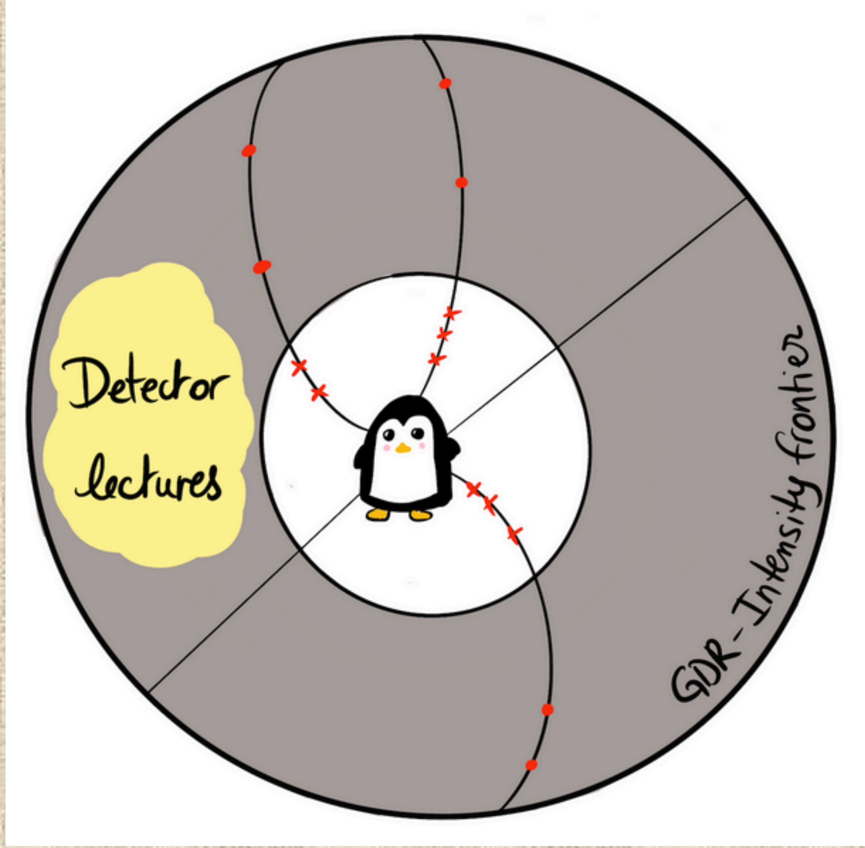
Algorithms for trajectography

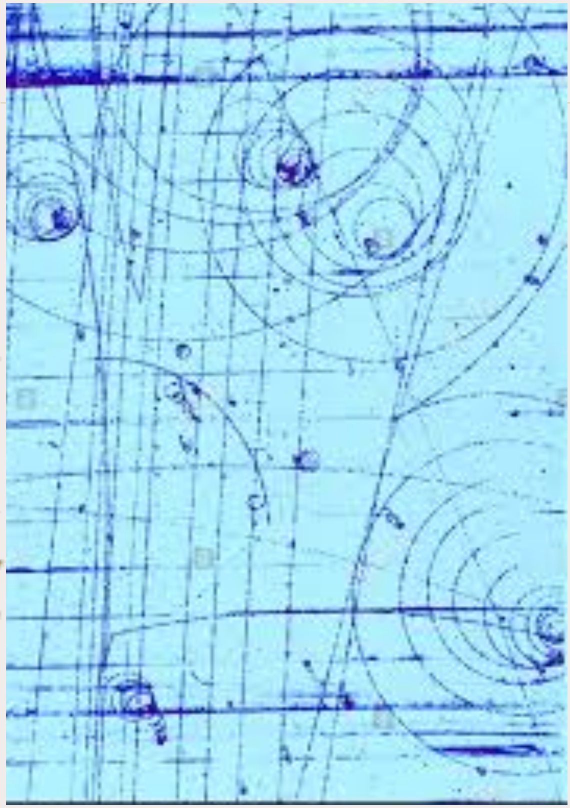


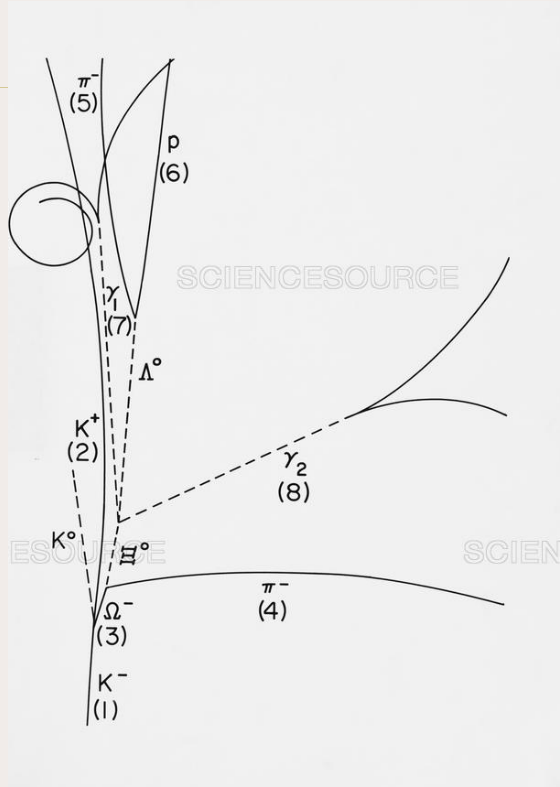
P. Billoir, LPNHE Paris nov. 2021

main topics

- track finding
- track fitting
- progressive approach to Kalman Filter
- trajectory in a magnetic field
- vertex finding/fitting
- alignment/calibration

the good old times (bubble chambers)





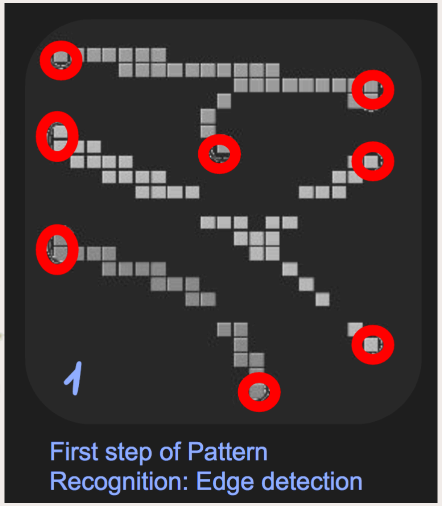
pattern recognition…
by hand
(sophisticated) track fit
by computer

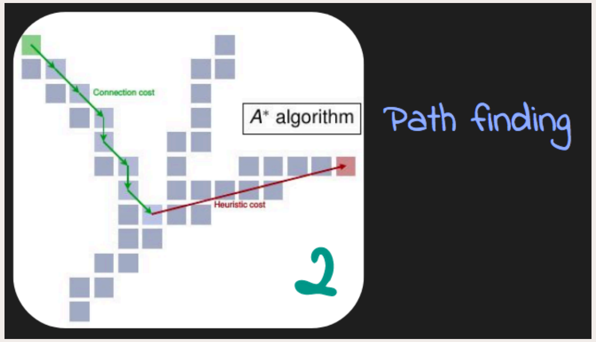
already there: particle identification (density of bubbles)

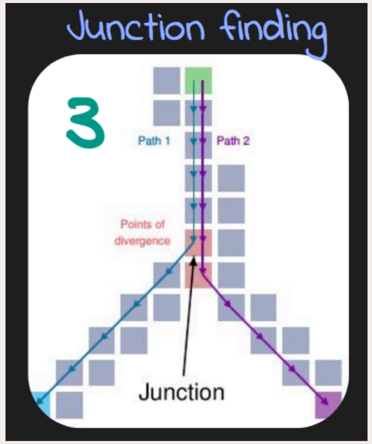
in the 70’s: more or less automatic scanning of pictures
but in the same time: bubble chambers are replaced by electronic detectors: spark chambers, wire chambers,…

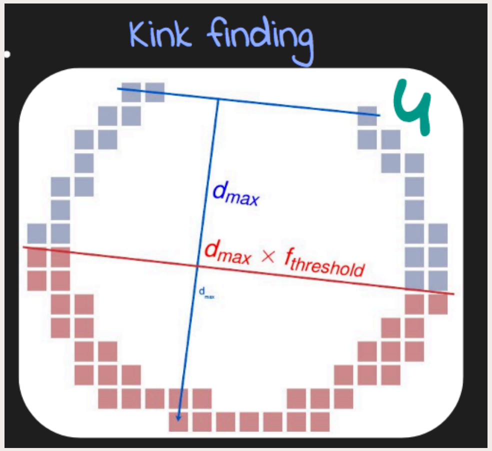
"microscopic" pattern recognition example of TREx

M. Haigh, P.Denner, for DUNE





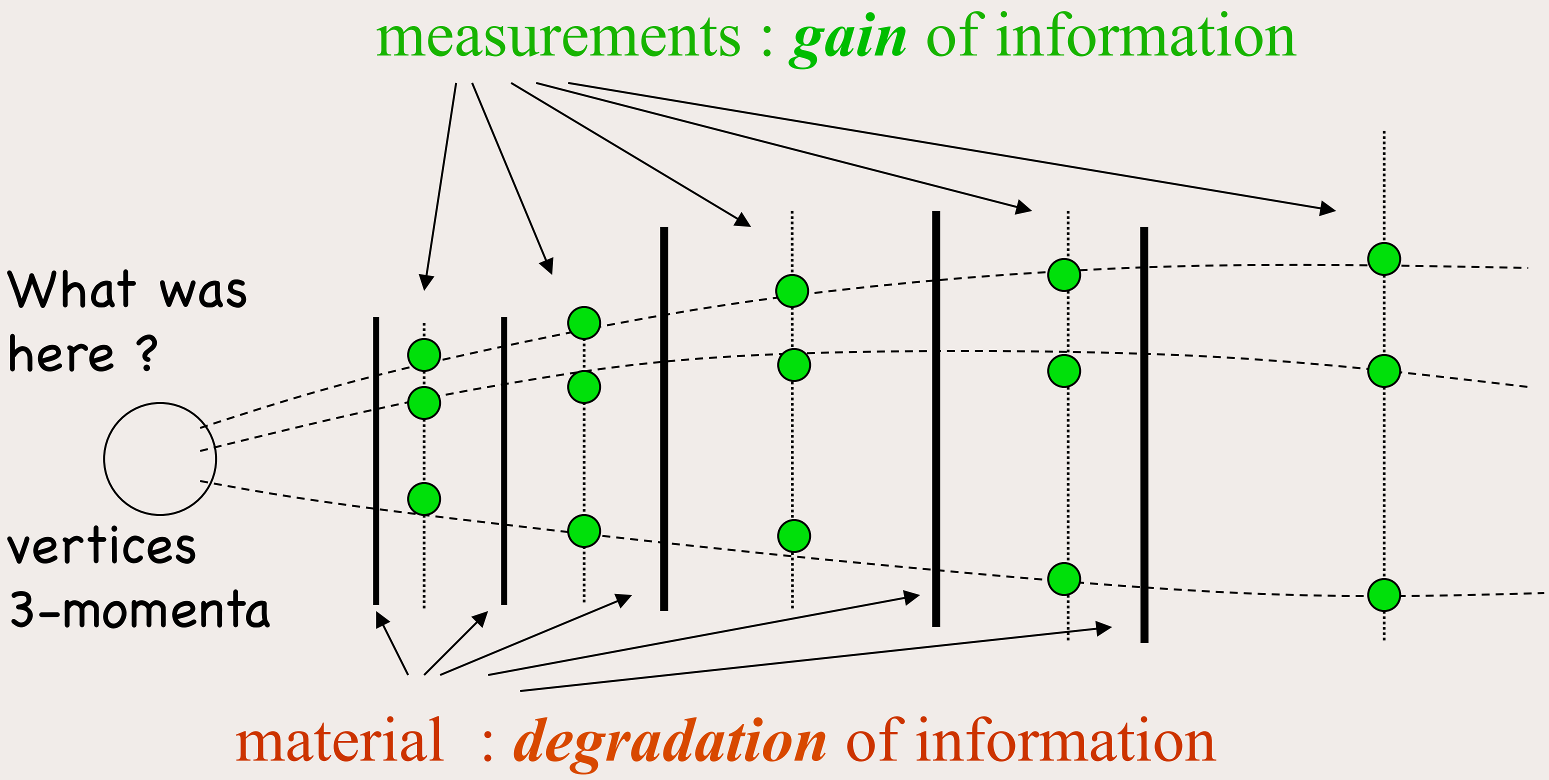




neutrino experiment:
rare events, few tracks, but complex topology

aim: find a precise description of all details

What do we want ?



How to build the best estimator of the physical quantities ?

the ingredients

what is supposed to be known

- nature and precision of the measurements
- nature and magnitude of the "noises" in the matter
  (secondary interactions, multiple scattering, continuous energy loss)
- equation of propagation (magnetic field)

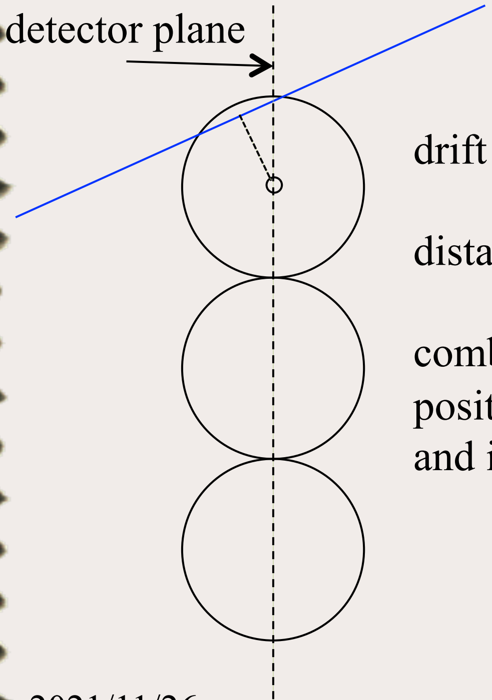
Remarks: the nature of the particle (e,μ,π, etc) may be unknown; the points above may depend on the mass hypothesis

to be done

- grouping the local "hits" into track candidates (pattern recognition)
- fitting the parameters at origin (just after production)
  if needed: iteration to solve the ambiguities
- inter/extrapolating to other detectors (RICH, muon chambers,…)
- if possible: information for particle identification (dE/dx,…)
- finding primary/secondary vertices: topology and final fit

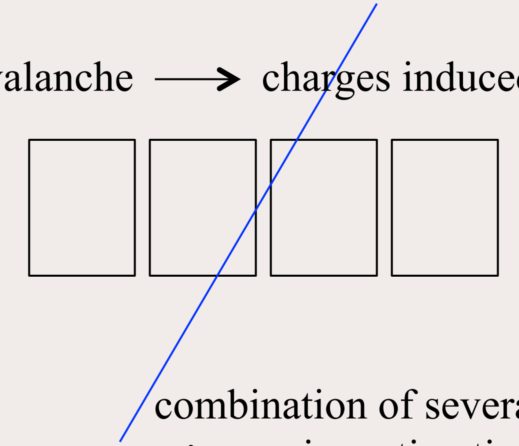
local measurements ("hits")

ideally: one or two coordinate(s) of a point on a surface
practically: often an indirect measurement (e.g. a drift time) or a combination of elemantary signals (a « cluster »)



drift time
↓
distance to axis
↓
combination of position in plane and incidence angle

avalanche → charges induced in pads



combination of several amplitudes
→ precise estimation of coordinate (much better than pad size)

pattern recognition vs final track fit

- aim of patt. rec.: find *association* of hits. The precision needed is the power of *separation* between hits, not the error on their position.

- the final track fit should give the *best estimator,* using a precise estimation of the positions of hits and the error on them, and the full covariance matrices of the track parameters.

- in practice, these tasks may interfere, and the whole procedure may be a more or less intricate combination of *finding* and *fitting* steps

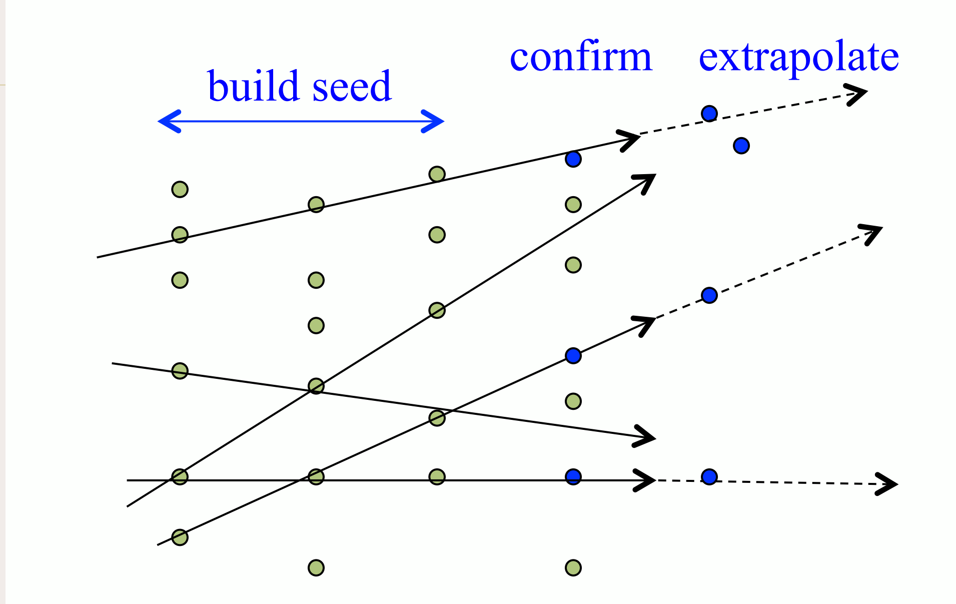
Note: in many cases, the limiting factor is not the hit measurement error, but the noise (mainly multiple scattering). *Do not be more royalist than the king !*

patt. rec. 1: extending tracks from seeds

general principle: build seeds from a few shells, extrapolate to next shells as long as compatible hits are found

tune criteria to:

- accept a new point
- confirm the track



- very flexible strategy (choice of shells for seeding, shell ordering,…)
- each new hit may be used to update the track parameters ➔ better extrapolation
- may consists in successive passes, iterations, etc
- may need much tuning to optimize the trade-off between efficiency/ghost rate/speed

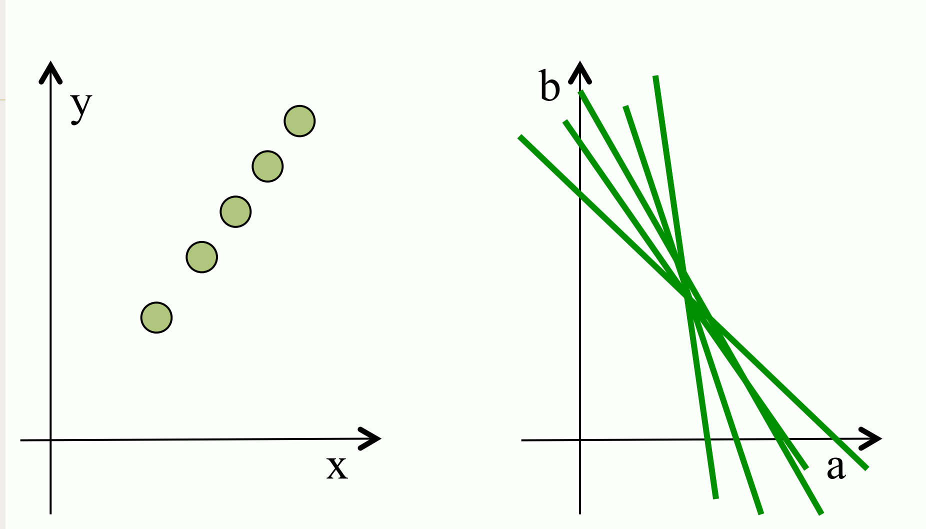
patt. rec. 2: Hough transform

first level: detect aligned points;

straight line $y = ax+b$
x,y ➔ 1 line im (a,b) plane

aligned points ➔ accumulation in (a,b) plane

alternative form
$\rho \cos(\theta-\theta_0) = \rho_0$
(avoids singularity for vertical lines)

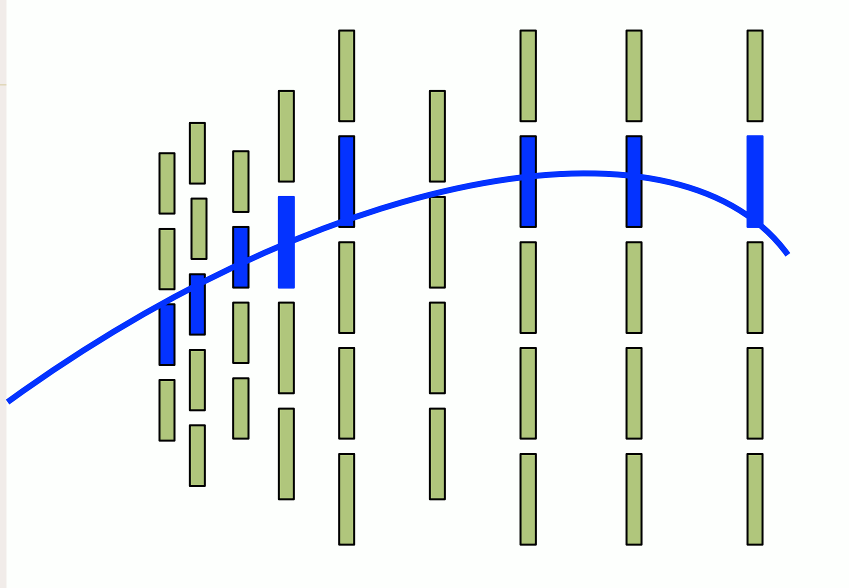


generalization: detection of curves in nD space described by a *simple* combination of *few* parameters (if many of them: huge number of pixels needed)

practical implementation: simple computation + large memory
OK for parallel computing with many « small » CPUs (FPGA, GPU,…)
flexibility: possibility of zoom in a restricted zone of large counting
but: high sensitivity to noise (ghosts)

patt. rec. 3: sample of routes

- simulate trajectories of tracks of physical interest
- define the pattern of hits for each one
- collect enough patterns to cover the wanted phase space (e.g. p_t > min)
- run time: flag the « filled » routes (flexible strategy to define the criteria of « filling »)



- OK for parallel computing with many small CPUs
- do not need any parameterization of trajectories
- large memory needed
- may produce multiple counting, ambiguities, ghosts

pattern recognition in brief

- no universal solution: the procedure has to be adapted to the layout of the experiment
- in most cases, it consists of parallelizable sub-algorithms and more global cleaning steps (rejection of poor candidates, resolution of ambiguities)
- the best method is often a combination of different algorithms in successive steps
- the pattern recognition may internally use some track fitting procedures for a more precise discrimination and extrapolation. In general, the fit may be simplified
- *machine learning may help to optimize the strategy*

basic tool for track fitting :

Kalman Filter (*progressive* method)

found in many textbooks… (here : Wikipedia)

Predict

Predicted (*a priori*) state estimate	$\hat{\mathbf{x}}_{k\|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1\|k-1} + \mathbf{B}_k \mathbf{u}_k$
Predicted (*a priori*) estimate covariance	$\mathbf{P}_{k\|k-1} = \mathbf{F}_k \mathbf{P}_{k-1\|k-1} \mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k$

Update

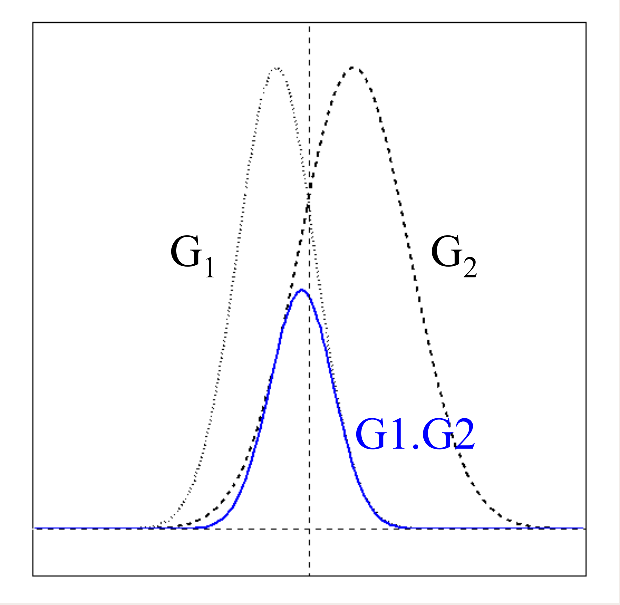
Innovation or measurement residual	$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k\|k-1}$
Innovation (or residual) covariance	$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k\|k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k$
Optimal Kalman gain	$\mathbf{K}_k = \mathbf{P}_{k\|k-1} \mathbf{H}_k^{\mathrm{T}} \mathbf{S}_k^{-1}$
Updated (*a posteriori*) state estimate	$\hat{\mathbf{x}}_{k\|k} = \hat{\mathbf{x}}_{k\|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$
Updated (*a posteriori*) estimate covariance	$\mathbf{P}_{k\|k} = (I - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k\|k-1}$

+ even more complicated expression for the "smoothing"

we will present something equivalent (and hopefully more intuitive !) and try to go further

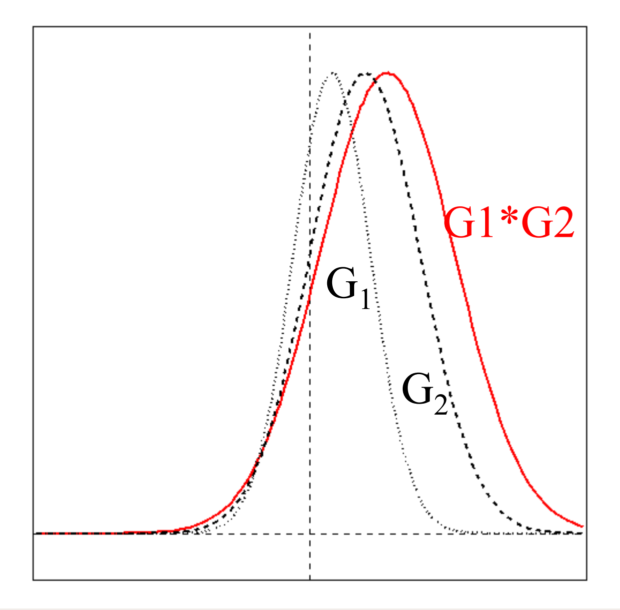
playing with gaussians
(or other distributions...)

combination of ***independent*** measurements
product of p.d.f. = addition of *informations*



for gaussians:
$G_1(\mu_1,\sigma_1).G_1(\mu_2,\sigma_2) = G_1(\mu',\sigma')$
$\mu' = (\mu_1/\sigma_1^2+\mu_2/\sigma_2^2) / (1/\sigma_1^2+1/\sigma_2^2)$
$1/\sigma'^2 = 1/\sigma_1^2 + 1/\sigma_2^2$

combination of ***independent*** errors
convolution of p.d.f. = addition of *noises*



for gaussians:
$G_1(\mu_1,\sigma_1)*G_1(\mu_2,\sigma_2) = G_1(\mu'',\sigma'')$
$\mu'' = \mu_1+\mu_2$
$\sigma''^2 = \sigma_1^2 + \sigma_2^2$

gaussians in nD space

$G(\mathbf{x}) = K \exp\left(-\sum W_{ij}(x_i-\mu_i)(x_j-\mu_j)/2\right)$ $K^2 = \det(W)/(2\pi)^n$

covariance matrix $C = W^{-1}$

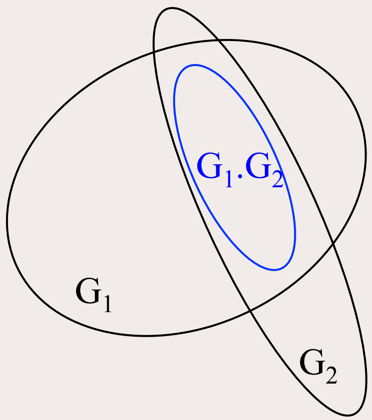
combining gaussians:

product: $(\boldsymbol{\mu}_1, W_1) . (\boldsymbol{\mu}_2, W_2) \rightarrow (W_1+W_2)^{-1}.(W_1\boldsymbol{\mu}_1+W_2\boldsymbol{\mu}_2)$, W_1+W_2

(« barycenter » , addition of weight matrices)

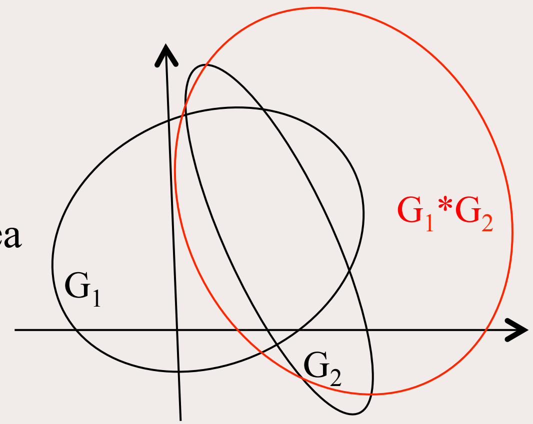
convolution: $(\boldsymbol{\mu}_1, W_1) * (\boldsymbol{\mu}_2, W_2) \rightarrow \boldsymbol{\mu}_1+\boldsymbol{\mu}_2$, $(W_1^{-1}+W_2^{-1})^{-1}$

(addition of biases, addition of covariance matrices)

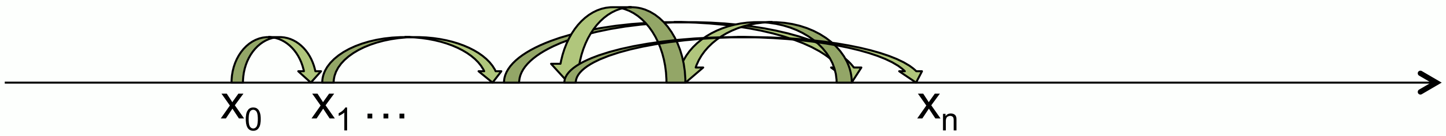


1σ contours

quantitatively:
information = 1/area
(1/volume in nD)



a 1-parameter problem
where is/was the flea ?



a flea moves by jumps on x axis; initial position : x_0
at each time step (independently):
- measurement (precision σ)
- jump (standard deviation τ)

what is the “best” estimator of the position x_0 ? x_n ?

intuitively :
- if $\sigma \ll \tau$: the instant one; the other ones are spoiled by the jumps
- if $\tau\sqrt{n} \ll \sigma/\sqrt{n}$ (that is $n\tau \ll \sigma$): the average of n measurements
- intermediate case: not obvious; truncated mean ? truncated weighted mean ?

- the best linear estimator should be a weighted combination of the measurements

How to evaluate the weights ?

The *heavy* optimal solution

One wants to estimate x_0 , accounting for the correlations between successive measurements:

$x_0^{mes} = x_0 + \varepsilon_0$
$x_1^{mes} = x_0 + \eta_1 + \varepsilon_1$
$x_2^{mes} = x_0 + \eta_1 + \eta_2 + \varepsilon_2$
…

ε_k : meas. error at time k ; η_k : jump at time k

covariance matrix C of the deviations $\Delta x_k = x_k^{mes} - x_0$:

$$\begin{matrix} \sigma^2 & 0 & 0 & 0 & \dots \\ 0 & \sigma^2+\tau^2 & \tau^2 & \tau^2 & \dots \\ 0 & \tau^2 & \sigma^2+2\tau^2 & 2\tau^2 & \dots \\ 0 & \tau^2 & 2\tau^2 & \sigma^2+3\tau^2 & 3\tau^2 \dots \\ & \dots\dots & & & \end{matrix}$$

$\chi^2 = \Sigma\,(C^{-1})_{ij}\,\Delta x_i\,\Delta x_j \;\rightarrow\; x_0^{fit} = \Sigma_j\,(C^{-1})_{ij}\,x_i^{mes}$

with n measurements: matrix (n×n) to be inverted

A better option: the progressive method

The measurement informations are included one at a time, and the degradation (jump) is accounted for at each step

the key point: at each step, one has to combine two ***independent*** informations:
- the optimal combination of all previous measurements
- the measurement at this time: this gives the optimal combination of previous + this one

then, this new combination undergoes the next jump, so it is degraded: the error after the jump is the quadratic addition of the error before and the jump itself, which are ***independent***

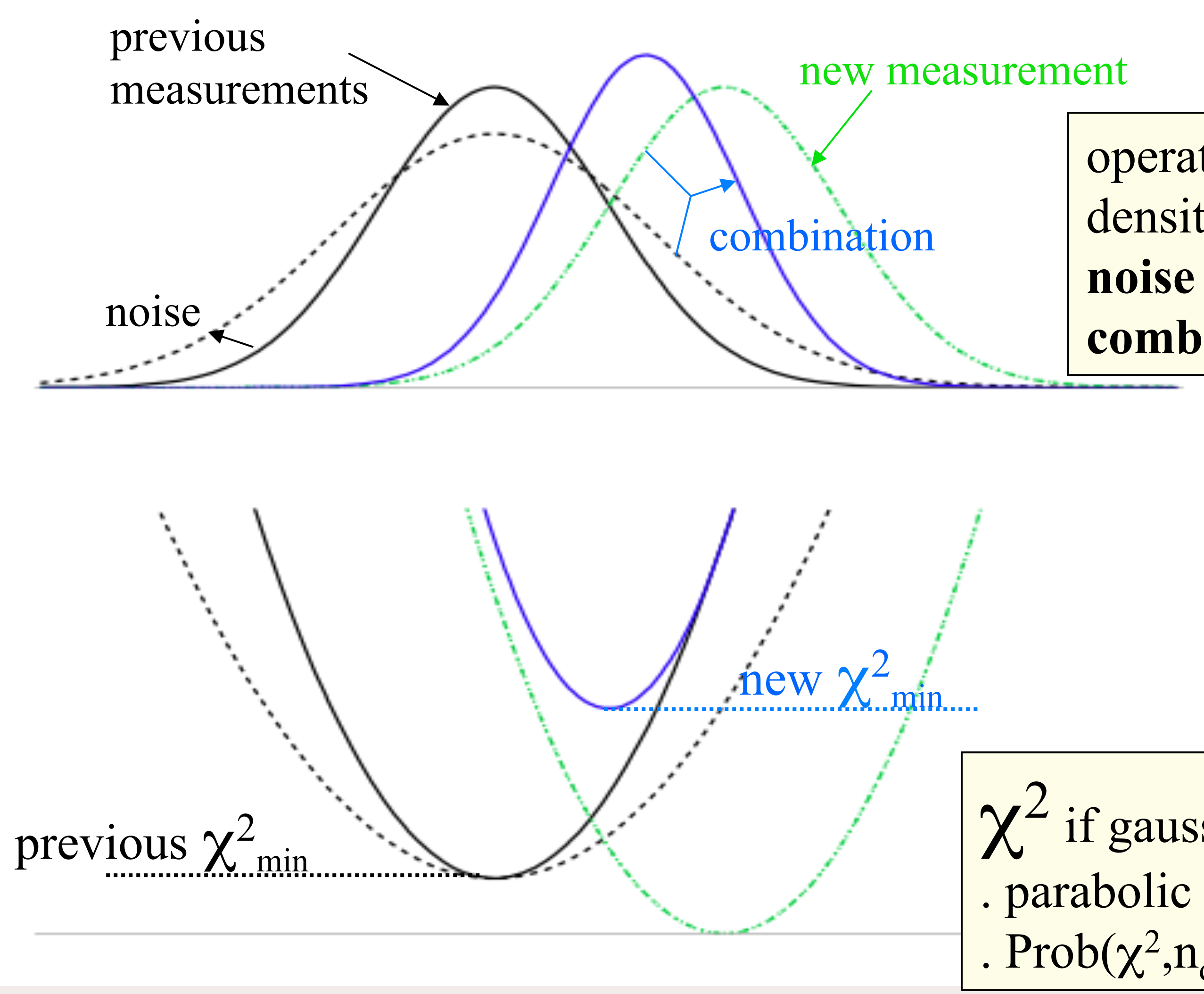
combining independent measurements (*adding informations*)
$(x', \sigma') + (x'', \sigma'') \Rightarrow (w'x'+w''x'')/(w'+w'')$ with $w'=1/\sigma'^2$, $w''=1/\sigma''^2$
combining independent errors: σ' and $\sigma'' \Rightarrow (\sigma'^2 + \sigma''^2)^{1/2}$

at each step: a χ^2 may be updated

with n steps: the number of operations is ***proportional to n***

recipe for the best estimate of the initial state:
- **start from the last point**
- **go backwards, down to the first one**

one step of the progressive estimator



operations on the density of probability
noise : *convolution*
combination : *product*

χ^2 if gaussian errors:
. parabolic shape
. Prob(χ^2,n_{deg}) is exploitable

more (almost for free)

• final position x_n :
forward filter (same procedure, going from 0 to n)

• intermediate position x_k (interpolation) : starting from both ends towards point k, combine *independant* backward and forward estimators $X_{n\to k}$ and $X_{0\to k}$.
x_k^{mes} may be omitted or included in one of them
(équivalent to the "smoother" in the kalmanian jargon)

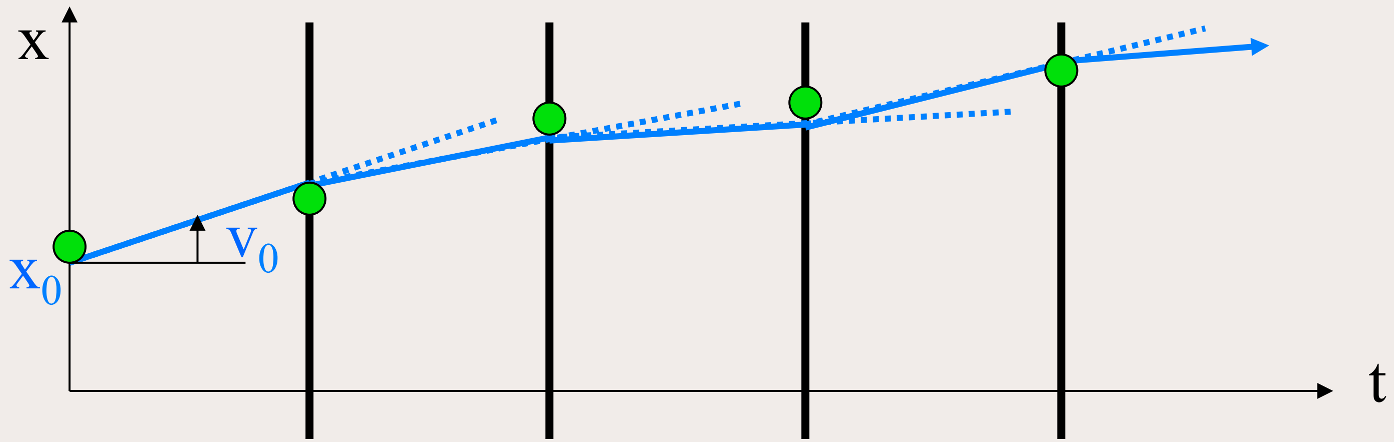
• **compatibility criterion** : the variance of $x_k^{interp}(w/o\ x_k^{mes}) - x_k^{mes}$ is $V(interp) + \sigma^2$

• **abnormal** jump detected by comparing $X_{n\to k} - X_{0\to k}$ to the predicted variance

in brief : with the forward filter and the backward filter (keeping the intermediate results) one can obtain all that

But: *if one point is modified (e.g. one measurement added or removed), all following steps have to be redone).* For example: if working on-the fly (incorporating measurements in real time), the backward filter would be heavy … but probably useless

linear problem with 2 parameters (movement with "noisy" speed)



initial conditions: x_0 , v_0 (to be estimated)

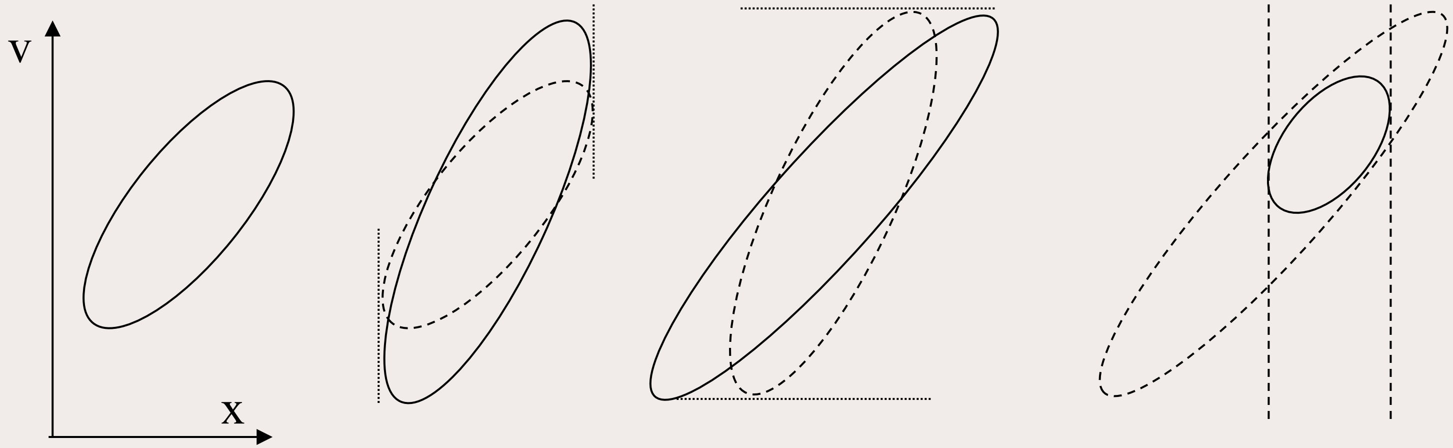
at each time step Δt :

- measurement of x (error ε_k , variance σ^2)
- random variation ζ_k of v_k (variance ρ^2)
- displacement $v_k.\Delta t$

$x_k^{mes} = x_0 + (v_0 + \zeta_1)\,\Delta t + (v_0 + \zeta_1 + \zeta_2)\,\Delta t + \ldots + \varepsilon_k$

→ *correlation* (x_k^{mes}, x_j^{mes}) *through the* ζ_i

progressive fit: one step "on-the fly"
in the (x,v) plane



previous measurements:
state vector $p_o(x_o,v_o)$
cov. matrix $C_o(p_o)$

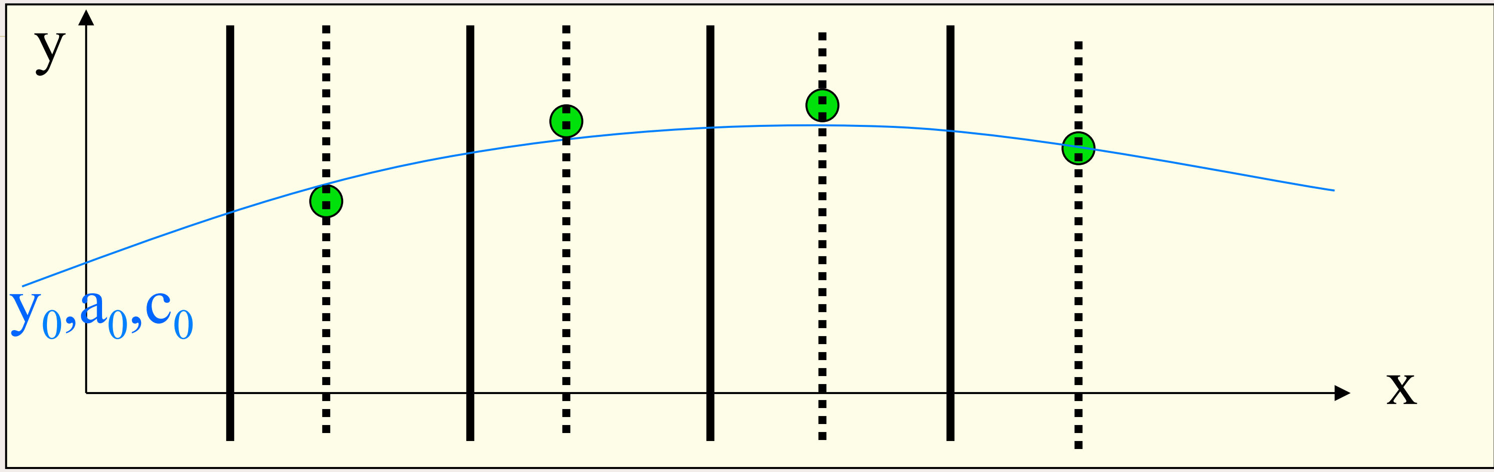
noise on v
$C_b = C_o + B$
$B = \begin{pmatrix} 0 & 0 \\ 0 & \rho^2 \end{pmatrix}$
degraded information

propagation
$p'(x_o + v_o\Delta t, v_o)$
$C' = D.C_b.D^t$
$D = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}$
$W' = (D^t)^{-1}.W.D$

combination with a new measurement p_n
$W' = (C')^{-1}$, $W_n = (C_n)^{-1}$
$(W' + W_n)\, p'' = W'\, p' + W_n p_n$
$C''(p'') = (W' + W_n)^{-1}$
gained information

problem with 3 parameters

(simplified planar trajectograph)



parabolic approximation: $y(x) = y_0 + a_0\, x + c_0\, x^2/2$

physical parameters : y_0 , p_y/p_x , q/p ($c = 0.3B\, q/p$)

the measurements are linear functions of the parameters

→ same formalism, with C,W,D as (3x3) matrices

if non negligible energy loss ΔE: introduce Δc in the propagation step

noise : multiple scattering (affects a) ; fluctuation of ΔE (affects c)

NB: to evaluate the noise terms, E (that is c) needs to be evaluated

linear approximation

In real world : no exact linear model

possible solution:

- choose convenient parameters **p** (e.g. cartesian ou cylindrical coord.)

- define ***lines/surfaces*** (planes, cylinders,…) for ***measurements*** and ***material*** (the noise in a thin slice of material may be described by a matrix C_b with a correlation between position and direction)

- define a ***reference trajectory*** $\mathbf{T_{ref}}$ close to the true one (from patt. rec. or preliminar fit)

- propagate the ***deviations*** **δp** of **p** from $\mathbf{T_{ref}}$ in the *linear approximation:*
  $D_{S\rightarrow S'} = \partial(\boldsymbol{\delta}\mathbf{p'}) / \partial(\boldsymbol{\delta}\mathbf{p}) = \partial\mathbf{p'}/\partial\mathbf{p}$ (jacobian matrix)

- apply the KF formalism; if needed, modify $\mathbf{T_{ref}}$ and iterate if the **δp** are too large *(it is also possible to change T_{ref} at some steps)*

a (false) technical problem: how to begin ?

at start: insufficient information to define p_0, and get inversible C_0, W_0
example : the first measurement is x or a linear combination linéaire of x and v → W has a 0 eigenvalue (the p.d.f. is a stripe; p_0 is degenerate along this stripe)

practically, the elementary matrix operations (convolution, propagation, product) are always possible :

• *convolution* : $(W^{-1}+C)^{-1} = (1+WC)^{-1}.W$
1+WC is *inversible* in the useful cases

• *propagation :* $W' = (D^{-1})^t.W.(D^{-1})$

• *product* : if W_1 and/or W_2 is singular, the system $(W_1+W_2)\,p = W_1p_1+W_2p_2$ has a solution which does not depend on the choice of p_1 and p_2 on the axis of the stripes
extreme case : parallel stripes : p is undefined, and the result in again a stripe

one can use the weight matrices in all steps

usual method with the standard KF (using covariance matrices); start with large values in C.
but: possible problems of precision

general case: 3D trajectory in Bfield
(5 parameters)

which parameters ?
it depends on the geometry of the tracking system

Examples:

- fixed target or endcap in a collider:
  surfaces: planes perpendicular to the beam (fixed z)
  - position: x,y
  - direction: θ(or η) and ϕ, or direction cosines c_x,c_y, or slopes t_x=dx/dz, t_y=dy/dz
  - *signed* curvature (q/R or q/p_t ou q/p)
- barrel in a collider, with **B** along z :
  surfaces: cylinders (e.g. beam pipe + concentric shells) :
  - position (angle Φ, z)
  - direction (angles θ, ϕ)
  - curvature (q/R or q/p_t ou q/p)

procedure: same as before, with 5-vectors for the state, 5x5 matrices for W,C,D

“simple” measurement/noise

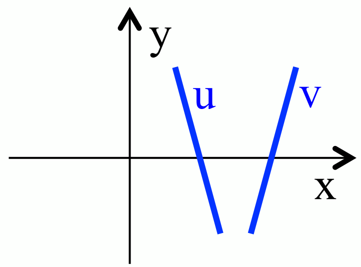
measurement of one coordinate, e.g. x:
$p_{meas} = (x_{meas}, 0,0,0,0)$ $W_{meas} = \mathrm{diag}(1/\sigma^2, 0,0,0,0)$

measurement of two coordinates x,y:
$p_{meas} = (x_{meas}, y_{meas}, 0,0,0)$ $W_{meas} = \mathrm{diag}(1/\sigma_x^2, 1/\sigma_y^2, 0,0,0)$

scattering in a surface:
C_{ms} = (2x2) submatrix on t_x, t_y (includes correlation)

scattering in a layer:
C_{ms} = (4x4) submatrix on x, y, t_x, t_y (includes correlations)

"oblique" measurements



« stereo » measurement

- a combination is measured, e.g. $u = ax+by$ (stereo)

$w_u = 1/\sigma_u^2$ ("weight" of the u measurement)

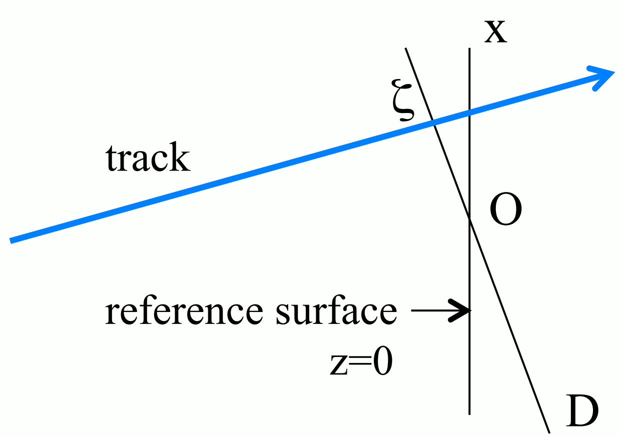
contribution to χ^2 : "stripe" in the (x,y) plane

$w_u(u^{mes}-ax-by)^2 = (x-x^{mes},y-y^{mes})^t\, W\, (x-x^{mes},y-y^{mes})$

x^{mes},y^{mes} : any point such that $ax^{mes} + by^{mes} = u^{mes}$

$W = (a,b).w_u.(a,b)^t = 1/\sigma_u^2\, (a^2\ ab\ ,\ ab\ b^2)$ (matrix of rank 1)

- measurement in a detector which is *oblique* w.r.t the reference surface



trajectory of slope $a = dx/dz$

measuring ζ (with error σ) in D

amounts to measure $y = \zeta\,(\lambda+\mu a)$ with

errror $|\lambda+\mu a|.\sigma$

λ,μ : constants depending on geometry

note: a is known at this stage (at least roughly)

general formulation for several measurements in the same detector:

contribution to $\chi^2 = (\mathbf{p}-\mathbf{p}^{mes})^t\, W_p\, (\mathbf{p}-\mathbf{p}^{mes})$ with $W_p = M^t W_m M$

W_m : weight matrix of the measurements **m** ; M: dependence **dm/dp**

exogenous measurements

some informations from non-trajectographic detectors may be injected at some stages on the filter:

examples:

• E measured in a calorimeter may be injected in the initial state of the backward filter as an estimator of q/p (if the matching and the sign q are inambiguous…)

• ΔE mesured as a γ energy in a calorimeter may be injected at an intermediate point or the trajectory (more delicate, but may be very useful for electrons…)

not everything is gaussian in real world…

two kinds of "non-gaussianity"

- "short range" : e.g. measurement with uniform distribution in an interval
  smoothed by convolution (gaussian limit for large numbers)
- "with long tails": the gaussian limit may fail

practically, for charged particles :

- non-linearity in the propagation → distortion of the p.d.f.
- multiple sattering : low probability of a diffusion at large angle (à la Rutherford)
- energy loss:
  . ΔE through ionisation is almost déterministic, with small fluctuations
  . more violent occurrences : δ-rays, and above all bremstrahlung (**major problem for electrons**)

If the gaussian approximation fails, what to do ?

God's algorithm

5-vector **p** to describe the state of the particle on a surface

chaining elementary operations on the p.d.f. F(**p**) :

• **measurement** (local) : *multiplication* by $f^{meas}(m(\mathbf{p}))$

• **noise** (local) : *convolution* with $f^{noise}(\mathbf{p})$

• **propagation** : *changement of variables* $F(\mathbf{p}) \rightarrow F^{pr}(\mathbf{p}^{pr}(\mathbf{p}))$:

obvious difficulty: computing power needed for functions in a 5D space!

But : "On trouve avec le Ciel des accommodements" *(Tartuffe)*

the gaussian sum

principle: approximation of F(**p**) , f^{meas} et f^{noise} by a sum of *gaussian functions*

$F(\mathbf{p}) = \Sigma\, \alpha_i\, G_i\,(\mathbf{p})$ with $G_i(\mathbf{p}) = C_i \exp\left(- (\mathbf{p}-\mathbf{p}_i)^t\, W_i\, (\mathbf{p}-\mathbf{p}_i) / 2\right)$

- works well in many cases for f^{meas} et f^{noise} (function of 1 variable)
- F is defined and positive everywhere if all $\alpha_I > 0$, and it vanishes at infinity
- the operations (product, convolution, linear propagation) are easy and give again a sum of gaussians

product : $(\mathbf{p}_1,W_1)\times(\mathbf{p}_2,W_2) = \left((W_1+W_2)^{-1}(W_1\mathbf{p}_1+W_2\mathbf{p}_2)\,,\, W_1+W_2\right)$

convolution : $(\mathbf{p}_1,W_1)*(\mathbf{p}_2,W_2) = \left(\mathbf{p}_1+\mathbf{p}_2\,,\,(W_1^{-1}+W_2^{-1})^{-1}\right)$

But : the number of components increases multiplicatively

possible remedies:

- suppress components of low amplitude
- merge nearby components into one

→ *to be optimized for each case, depending on the final impact on physics results*

in practice: used mainly for electron trajectories

propagation: the Runge-Kutta integration method

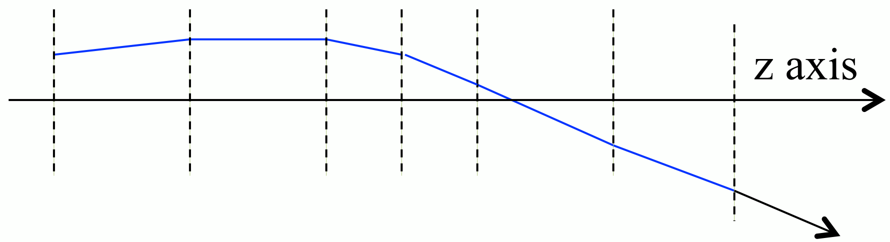
generic problem: $y' = f(t, y), \quad y(t_0) = y_0$ solved by steps h in t

1 step order 4

$$\left.\begin{aligned} k_1 &= f\left(t_n, y_n\right) \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\ k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\ k_4 &= f\left(t_n + h, y_n + hk_3\right) \end{aligned}\right\} \; y_{n+1} = y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

steps along z axis in a magnetic field

RK applied to the state vector (x, y, t_x, t_y)



$$\begin{aligned} \frac{\mathrm{dx}}{\mathrm{dz}} &= t_x \\ \frac{\mathrm{dy}}{\mathrm{dz}} &= t_y \\ \frac{\mathrm{dt_x}}{\mathrm{dz}} &= c\frac{q}{p}\sqrt{1 + t_x^2 + t_y^2}\left(t_x t_y B_x - (1 + t_x^2)B_y + t_y B_z\right) \\ \frac{\mathrm{dt_y}}{\mathrm{dz}} &= c\frac{q}{p}\sqrt{1 + t_x^2 + t_y^2}\left((1 + t_y^2)B_x - t_x t_y B_y - t_x B_z\right) \end{aligned}$$

parameterized propagation

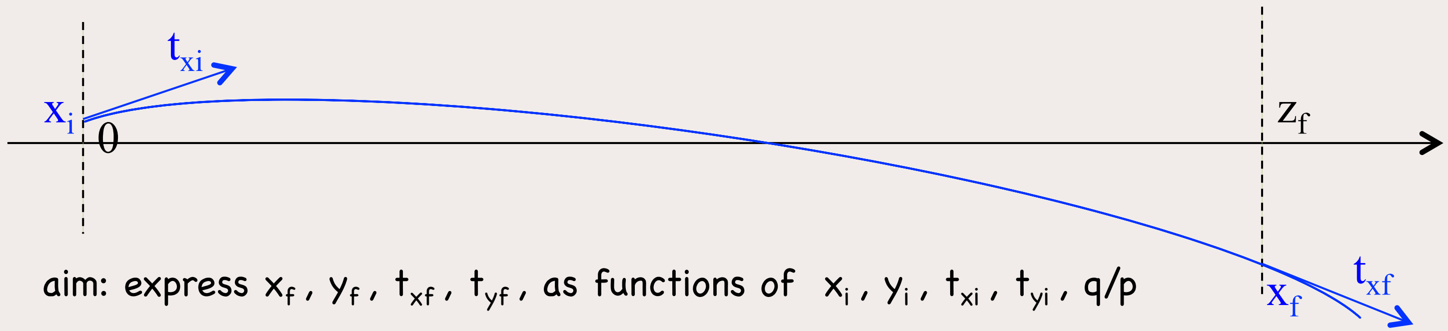
idea: instead of using RK extrapolation for every track, precompute formulae to get a faster execution
principle:

- chose a few reference surfaces that will contain « nodes » of the Kalman Filter.
- to go from the initial surface Σ_i to the final one Σ_f , express the state vector $\mathbf{S}_f$ on Σ_f through analytical of tabulated functions of the components of the state vector $\mathbf{S}_i$ on Σ_i

guiding criteria

- at infinite momentum, the trajectory is a straight line
- so, we can try an expansion in powers of q/p of $\Delta\mathbf{S}_f$, the difference between $\mathbf{S}_f$ and the straight line extrapolation
- the precision should be small compared to the other sources of error (mainly multiple scattering)
- the phase space may be reduced for trajectories close to the origin (particles for physics analysis)

first example in the « endcap » description (x, y, t_x, t_y, q/p at fixed z): propagate ***from*** $\boldsymbol{z_i=0}$ ***to*** $\boldsymbol{z_f}$
- t_x and t_y are bounded by the acceptance ;
- x_i and y_i are small, so terms at first order in x_i,y_i are sufficient



aim: express x_f , y_f , t_{xf} , t_{yf} , as functions of x_i , y_i , t_{xi} , t_{yi} , q/p

explicit formulae

$$\Delta \mathbf{S_f} = \sum_k \mathbf{A}_k(t_{xi}, t_{yi})(q/p)^k + \sum_k \left(x_i\, \mathbf{B}_k(t_{xi}, t_{yi}) + y_i\, \mathbf{C}_k(t_{xi}, t_{yi})\right)(q/p)^k$$

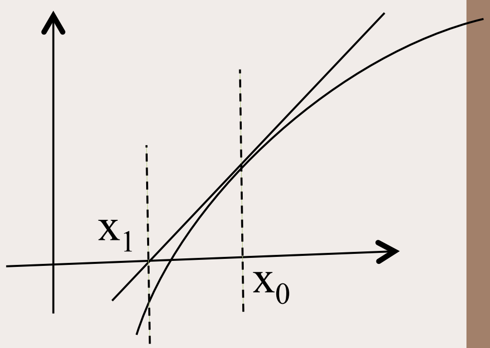
this gives 4 expansions (for x_f . x_f , t_{xf} , t_{xf}), assuming p to be constant, e.g. for x_f :

$$x_f = x_i + z_f t_{xi} + \sum_k A_k^x(t_{xi}, t_{yi})(q/p)^k + \sum_k \left(x_i\, B_k^x(t_{xi}, t_{yi}) + y_i\, C_k^x(t_{xi}, t_{yi})\right)(q/p)^k$$

the coefficients A,B,C may be tabulated or expressed as analytic functions of t_{xi} , t_{yi}

byproducts

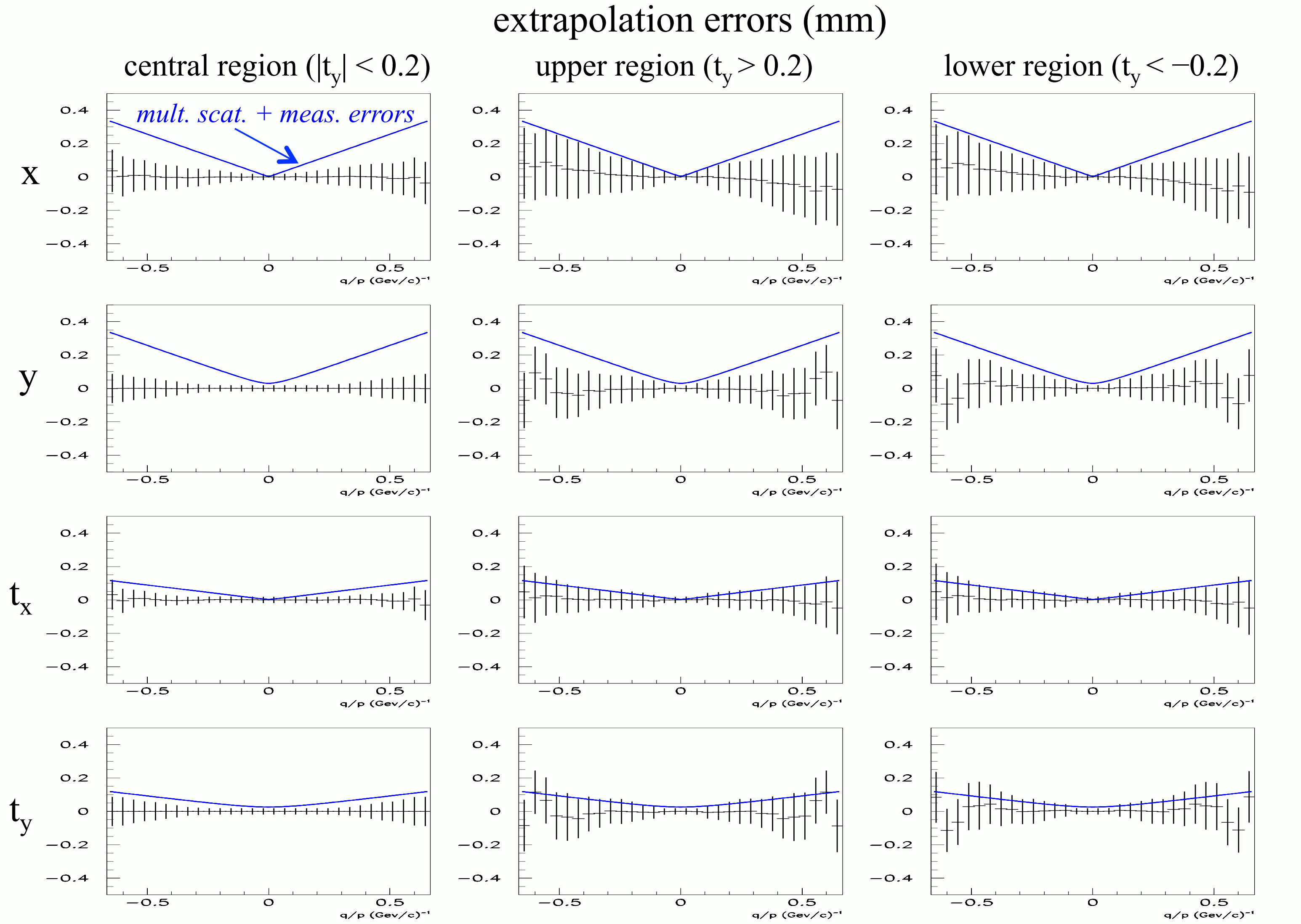
- jacobian matrix D: straightforward derivatives w.r.t. x_i, y_i, q/p , easy for t_{xi}, t_{yi}
- ***reverse propagation*** with the **Newton-Raphson** method:
  starting from S_f , we want to find S_i such that $S_i \rightarrow S_f$
  if S_i^0 is a good approximation, and $S_i^0 \rightarrow S_f^0$, then $S_f \approx S_f^0 + D.(S_i - S_i^0)$
  so $S_i \approx S_i^0 - D^{-1}. (S_f - S_f^0)$
  that is: we just need a direct propagation + a linear transform
  if needed: iterate (the convergence is ***very*** *fast)*



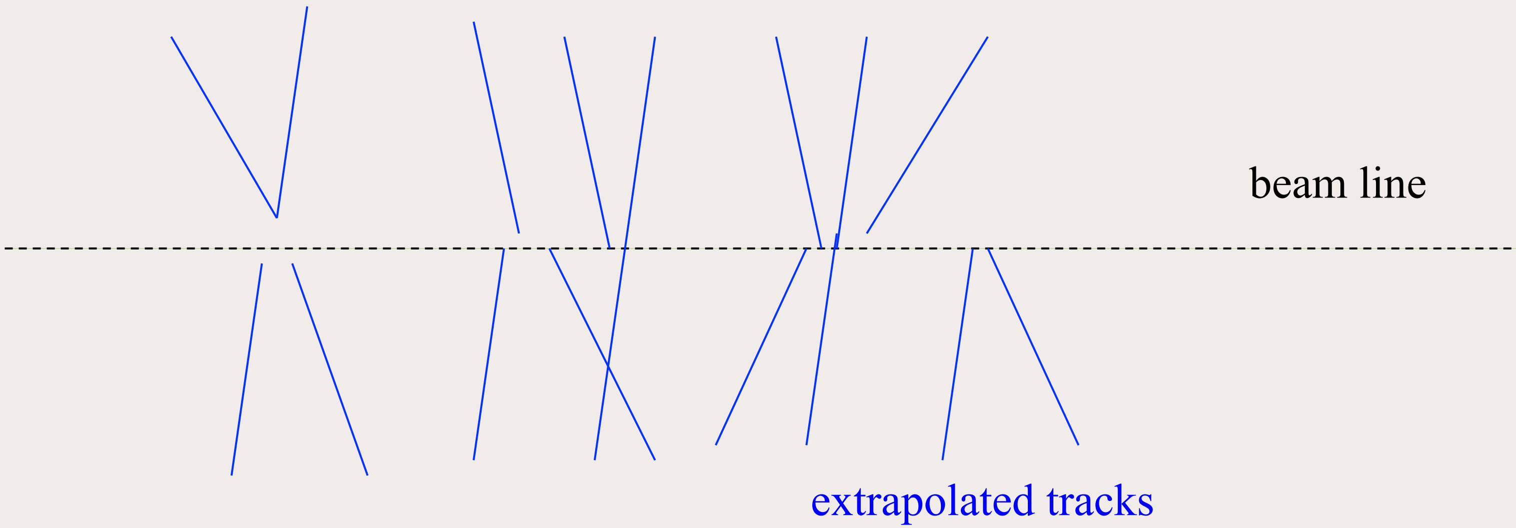
- propagation from z_i to z_f with $z_i \neq 0$: $z_i \rightarrow 0$ then $0 \rightarrow z_f$
  jacobian matrix $D_{if} = D_{0f}^{-1}.D_{i0}$

possible implementation: choose a few « main surfaces » for the full formulae and complement by short range extrapolation (1 step of RK or simpler local parameterization)

application to LHCb (from TT to T1, 5 m through the magnet)



vertexing
pattern recognition

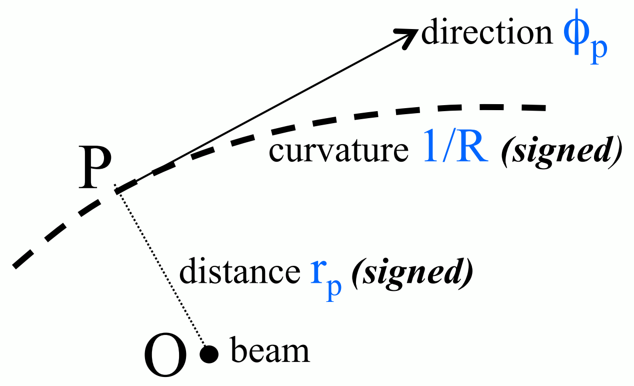


- find seeds for primary vertices (e.g. clusters in z of closest approach)
- detect short lived decays to build secondary vertices (tracks with significant impact parameter)

the "perigee" parameters (here: barrel coordinates)

idea: extrapolate the full track information at a point close to the vertex
➔ the next operations will be *local*

xy projection (beam along z axis)



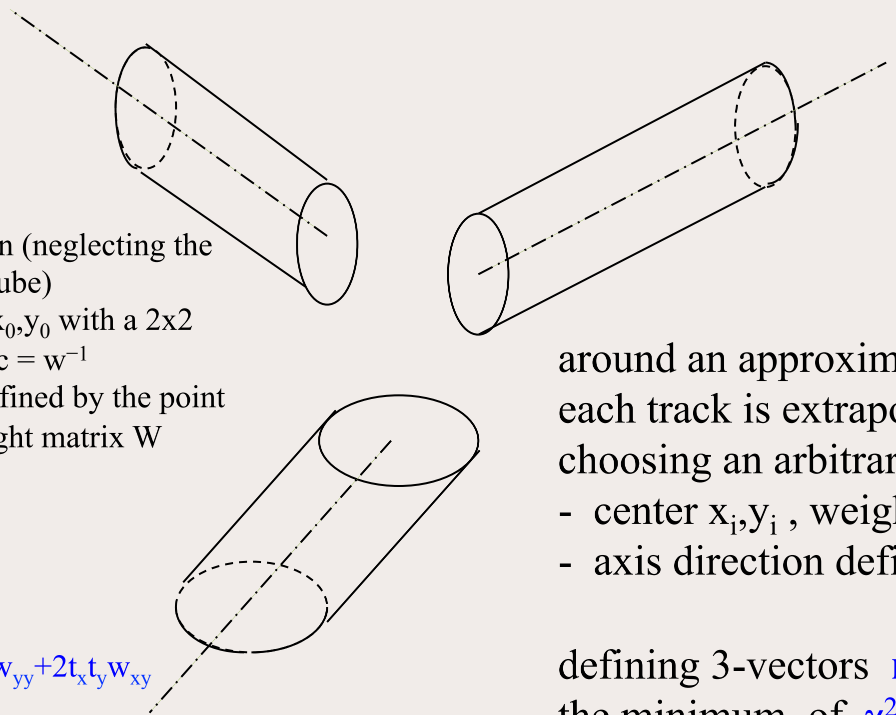
to complete the 3D description: z_p , θ_p

if **B** is along z axis:
$x = -r_P \sin\phi_p + s\cos\phi_p\ [- s^2/2R \sin\phi_p]$
$y = \ \ r_P \cos\phi_p + s\sin\phi_p\ [+s^2/2R \cos\phi_p]$
$z = z_P + s \cot\theta_p$
*(with s = **signed** distance from P in xy projection)*

sign convention: $r_P > 0$ if the track passes at the left of the origin
(terms [...] are negligible in general)

Advantages :
• smooth propagation from the fitted track param. to the perigee param. (the jacobian matrix D has no singularity with the consistent sign convention on r_p and q/R)
• short distance between perigee and vertex : linear approximation is valid; it may be used for any short lived decay. That is: the perigee params (and their covariance matrix) can be computed once.
• the perigee params have a physical meaning

“simple” vertexing
fitting vertex position



local approximation (neglecting the divergence of the tube)
at $z = z_0$: position x_0, y_0 with a 2x2 covariance matrix $c = w^{-1}$
the tube may be defined by the point x_0, y_0, z_0 with a weight matrix W

$W_{xx} = w_{xy}$
$W_{xy} = w_{xy}$
$W_{yy} = w_{yy}$
$W_{xz} = -t_x w_{xx}$
$W_{yz} = -t_y w_{yy}$
$W_{zz} = t_x^2 w_{xx} + t_y^2 w_{yy} + 2 t_x t_y w_{xy}$

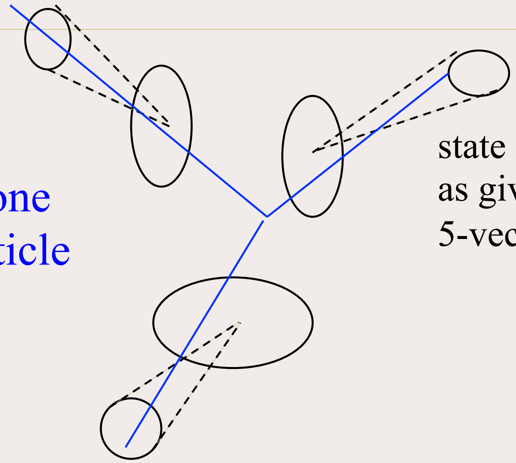
around an approximate position of the vertex:
each track is extrapolated as a « **tube** » of error
choosing an arbitrary position z_i :
- center x_i, y_i , weight matrix W (rank 2)
- axis direction defined by t_x , t_y

defining 3-vectors $\mathbf{r}_i = (x_i y_i z_i)$, $\mathbf{V} = (X,Y,Z)$
the minimum of $\chi^2 = \Sigma (\mathbf{V}-\mathbf{r}_i)^t W_i (\mathbf{V}-\mathbf{r}_i)$
gives the fitted position X,Y,Z of the vertex
(combination of the tubes)

the full vertex fit

aim: use the convergence of trajectories to improve their reonstruction
(add a virtual measurement and increase the lever arm)

vertex position + one
3-vector **p** per particle



state (position, direction, momentum)
as given by the track fit:
5-vector **q** + 5x5 covariance matrix

first trial: fit the position as before, and introduce this point as an additional measurement to all tracks.
not optimal: this position is correlated to the other measurements on the track

second trial: iterative procedure: adjust alternatively the vertex position and the $\mathbf{p}_i$ (3-momenta of the particles at the vertex) to fit the extrapolations to $\mathbf{q}_i$
possible but the convergence may be slow (zig-zag path)

the vertex fit as a hierarchical fit

"all in one" method: from a sample of *n* trajectoires ($\mathbf{q}_i$,W_i) at initial point (5n parameters) fit ***simultaneously*** 3n+3 parameters with the constraint of *convergence*:

- the position **V**(X,Y,Z) of a common origin
- the 3-momenta $\mathbf{p}_i$ of the particles *at this point* (or equivalently q/p_i , θ_i , ϕ_I)

tool : propagation function $\mathbf{q} = \mathbf{F}(\mathbf{V}, \mathbf{p})$ from vertex to initial point (simple if the initial point is close to the vertex, e.g. the perigee)

formulation with a global χ^2 :
find **V** and the $\mathbf{p}_i$ which minimize

$$\chi^2 = \Sigma \; (\mathbf{q}_i^{mes} - \mathbf{F}(\mathbf{V}, \mathbf{p}_i))^t \, W_i \, (\mathbf{q}_i^{mes} - \mathbf{F}(\mathbf{V}, \mathbf{p}_i))$$

a priori : problem in a space of dimension 3n+3
actually : *hierarchical* problem: 3 global param. + 3 particular param. for each track

$$\min(\chi^2) = \min|\mathbf{V} \left[\Sigma \; \min|p_i \, (\mathbf{q}_i^{mes} - \mathbf{F}(\mathbf{V}, \mathbf{p}_i))^t \, W_i \, (\mathbf{q}_i^{mes} - \mathbf{F}(\mathbf{V}, \mathbf{p}_i))\right]$$

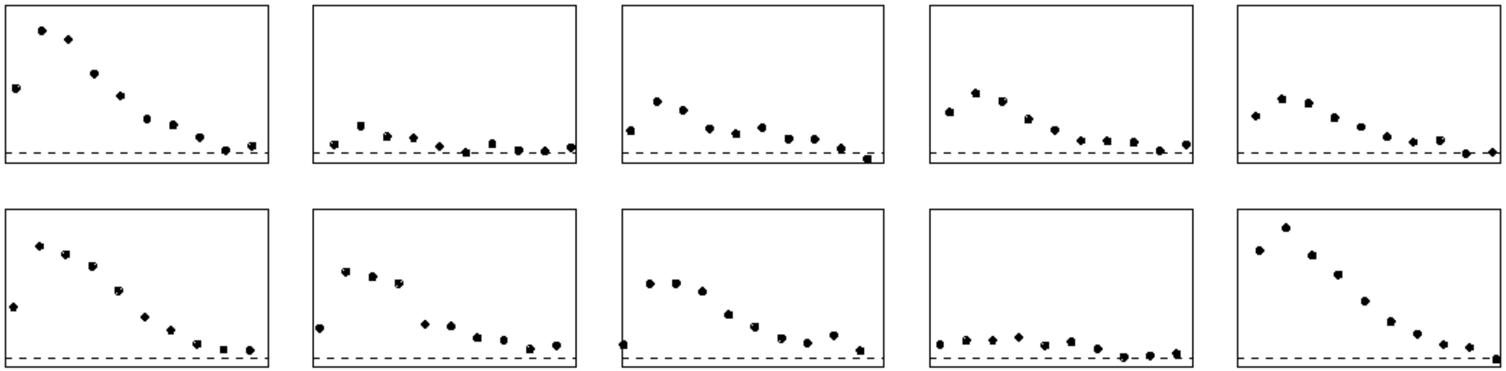
the "internal" et "external" minimizations have dimension 3

Note: the "nesting" remains valid without the gaussian approximation
that is: you can use e.g. `Minuit` with a `fcn` which itself calls n times `Minuit` (it works actually !)

other example of "hierarchical" fit (1)

sample of signals of the same shape, but with different amplitudes and dates :
$S(t) = A_i f(t-a_i)$; each one is measured at n times $t_k \rightarrow S_{ik}^{mes} = A_{ik} f(t_k - a_i) + \varepsilon^{mes}$
the shape is defined by *global* parameters p_1, p_2, … to be fitted

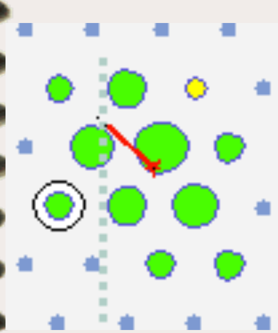
e.g. here $f(t) = 0$ for $t < 0$, $\exp(-p_1 t) - \exp(-p_2 t)$ for $t > 0$

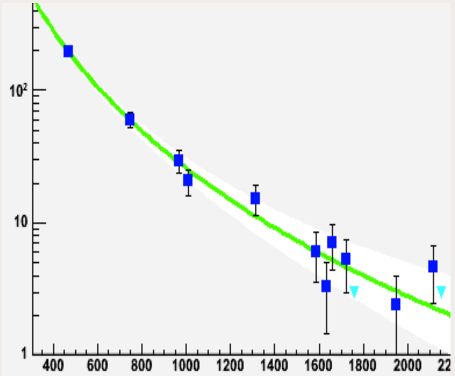


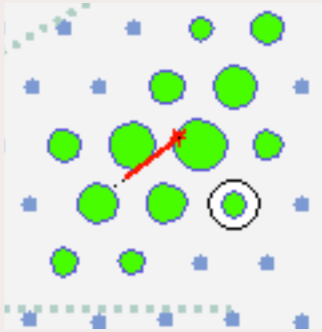
how to extract p_1 and p_2 from these measured signals ?

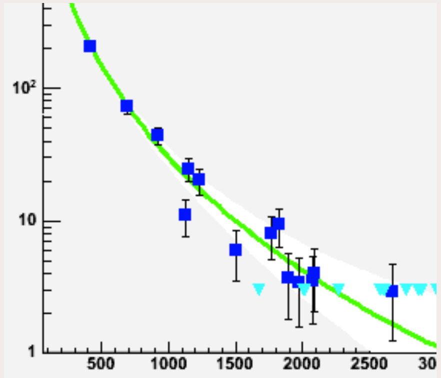
other example of "hierarchical" fit (2)

a set of events from the Surface Detector of AUGER (atmospheric showers)
signal in a tank at distance r_i from shower axis: $S_i = A_i f(r_i)$
- global parameters p,q for the shape, for example: $f(r) = 1/r^p(r+r_1)^q$
- individual parameters for each event: position (x_i,y_i) of the core, amplitude A_i









how to fit p,q from such data ?

linearization

if $\mathbf{V} \approx \mathbf{V}_0$ (vertex) and $\mathbf{p}_i \approx \mathbf{p}_{i0}$ (for every track) :

$\mathbf{q}_i = \mathbf{F}_i(\mathbf{V}, \mathbf{p}_i) \approx \mathbf{q}_{i0} + D_i.(\mathbf{V}-\mathbf{V}_0) + E_i.(\mathbf{p}_i-\mathbf{p}_{i0})$ *(short range propagation)*

E_i et D_i : (5×3) matrices, simple to compute if $\mathbf{q}_i$ is at the perigee

setting $\Delta\mathbf{q}_i = \mathbf{q}_i^{meas} - \mathbf{q}_{i0}$, on can fit $\boldsymbol{\delta}\mathbf{V} = \mathbf{V}-\mathbf{V}_0$ and the $\delta\mathbf{p}_i = \mathbf{p}_i-\mathbf{p}_{i0}$ to minimize

$$\chi^2 = \Sigma\, (\Delta\mathbf{q}_i - D_i\, \boldsymbol{\delta}\mathbf{V} - \mathbf{E_i}\, \delta\mathbf{p}_i)^t\, W_i\, (\Delta\mathbf{q}_i - D_i\, \boldsymbol{\delta}\mathbf{V} - \mathbf{E_i}\, \delta\mathbf{p}_i)$$

- *one* block of 3 equations on the full set of parameters:

$A\, \boldsymbol{\delta}\mathbf{V} + \Sigma\, B_i\, \delta\mathbf{p}_i = \mathbf{T}$ (1) with $A = \Sigma\, D_i^t\, W_i D_i$, $B_i = D_i^t\, W_i E_i$, $\mathbf{T} = \Sigma\, D_i^t\, W_i\, \Delta\mathbf{q}_i$

- *n* blocks de 3 equations on $\mathbf{V}$ and *one* $\mathbf{p}_i$:

$B_i^t\, \boldsymbol{\delta}\mathbf{V} + C_i\, \delta\mathbf{p}_i = \mathbf{U}_i$ (2) with $C_i = E_i^t\, W_i E_i$, $\mathbf{U} = \Sigma\, E_i^t\, W_i\, \Delta\mathbf{q}_i$

$$\begin{bmatrix} A & B_1 & B_2 & B_3 & \dots & & B_n \\ B_1^t & C_1 & 0 & \dots & & & 0 \\ B_2^t & 0 & C_2 & 0 & \dots & & 0 \\ B_3 t & 0 & 0 & C_3 & & 0 \dots & 0 \\ \dots & & \dots & & & & \dots \\ B_n^t & 0 & \dots & & & & C_n \end{bmatrix} \begin{bmatrix} \delta V \\ \delta\mathbf{p}_1 \\ \delta\mathbf{p}_2 \\ \delta\mathbf{p}3 \\ \dots \\ \delta\mathbf{p}_n \end{bmatrix} = \begin{bmatrix} \mathbf{T} \\ \mathbf{U}_1 \\ \mathbf{U}_2 \\ \mathbf{U}_3 \\ \dots \\ \mathbf{U}_n \end{bmatrix}$$

(sparse system by blocks 3x3)

resolution of the linear system

from equations (2) one can express the $\delta \mathbf{p}_i$ as functions of $\delta \mathbf{V}$

$$\delta \mathbf{p}_i = C_i^{-1} (U_i - B_i^t \, \delta \mathbf{V}) \qquad (3)$$

injecting these expressions in (1) one obtains an equation in $\delta \mathbf{V}$ only

$$(A - \Sigma B_i \, C_i^{-1} B_i^t) \, \delta \mathbf{V} = T - \Sigma B_i \, C_i^{-1} U_i \qquad (4)$$

(4) gives $\delta \mathbf{V}$ then each of the equations (3) gives $\delta \mathbf{p}_i$

as a bonus, we obtain also the full (3n+3)×(3n+3) covariance matrix …

$$\mathrm{cov}(\mathbf{V},\mathbf{V}) = (A - \Sigma B_i \, C_i^{-1} B_i^t)^{-1}$$

$$\mathrm{cov}(\mathbf{V},\mathbf{p}_i) = - \mathrm{cov}(\mathbf{V},\mathbf{V}) \, B_i \, C_i^{-1}$$

$$\mathrm{cov}(\mathbf{p}_i,\mathbf{p}_j) = \delta_{ij} \, C_i^{-1} + C_i^{-1} B_i^t \, \mathrm{cov}(\mathbf{V},\mathbf{V}) \, B_j \, C_j^{-1}$$

note that this procedure introduces correlations between the 3-momenta of all particles in the vertex, to be used in principle in the physics analysis ...

flexibility
(adding or removing one particle)

to add a track (fitted as $\mathbf{q}_{n+1}$, W_{n+1}):

- add a triplet of parameters $\delta\mathbf{p}_{n+1}$
- add in (1) $D_{n+1}^t W_{n+1} D_{n+1}$ to A, and one term $B_{n+1} = D_{n+1}^t W_{n+1} E_{n+1}$
- add in (2) one block of equations $B_{n+1}^t \,\delta\mathbf{V} + C_{n+1}\,\delta\mathbf{p}_{n+1} = \mathbf{U}_{n+1}$

taking as **starting values** the result of the fit with n particles ($\mathbf{V}_0$, $\mathbf{p}_{i0}$ for i=1…n):

$$(A+A_{n+1})\,\delta\mathbf{V} + \Sigma B_i\,\delta\mathbf{p}_i = \mathbf{T}_{n+1}$$

$$B_i^t\,\delta\mathbf{V} + C_i\,\delta\mathbf{p}_i = \mathbf{0} \quad \text{for } i=1\ldots n$$

$$B_{n+1}^t\,\delta\mathbf{V} + C_{n+1}\,\delta\mathbf{p}_{n+1} = \mathbf{U}_{n+1}$$

resolution:

$$(A - \Sigma B_i C_i^{-1} B_i^t + A_{n+1} - B_{n+1} C_{n+1}^{-1} B_{n+1}^t)\quad \delta\mathbf{V} = T_{n+1} - B_{n+1} C_{n+1}^{-1} U_{n+1}$$

only the terms in red are computed : fast procedure → many combinations may be tried

removing a track = adding it with a weight $-W_i$

Remark : the beam may be considered as a track to be added in a primary vertex (in general: very precise measurement of x,y, but z is undefined)

vertex fit with constraint(s)

examples:
- prompt or distant decay (neutral → +−) with mass hypothesis
- $\gamma \rightarrow e^+e^-$ with parallel tracks at the decay point;
in both cases: **p** points towards the main vertex (or just the beam line)
- more generally: combination of kinematical and geometrical constraints

Lagrange multipliers: universal tool

$\min|_{\mathbf{p}} (F(\mathbf{p}))$ with the constraint $C(\mathbf{p}) = 0 \;\Leftrightarrow\; \min|_{\mathbf{p},\boldsymbol{\lambda}} (F(\mathbf{p}) + \lambda C(\mathbf{p}))$

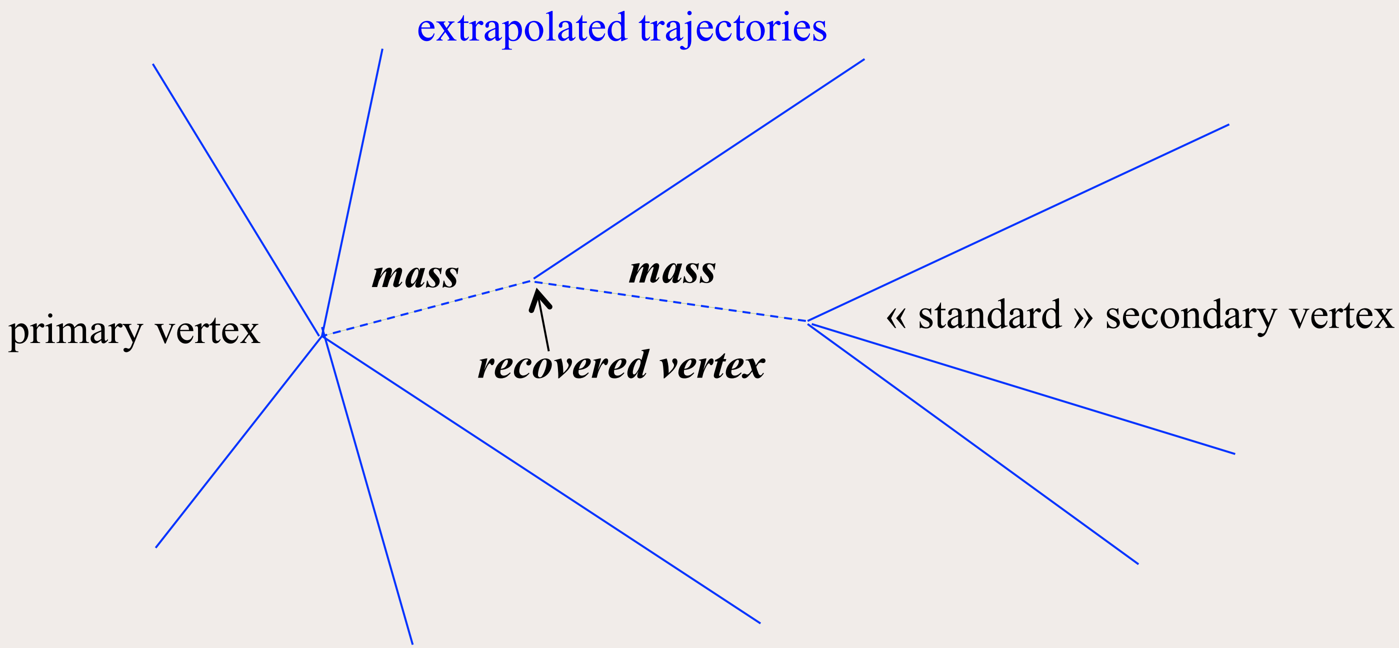
easy to solve in the following approximation *around the minimum* (or maximum) :
- the χ^2 or the log-likelihood is a quadratic function of the variations of parameters
- the constraint is linear

linear system → **p** as a function of λ , then elimination of λ with a linear equation

generalisation to several constraints:

$\min|_{\mathbf{p},\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2,\ldots} (F(\mathbf{p}) + \lambda_1 C_1(\mathbf{p}) + \lambda_2 C_2(\mathbf{p}) + \ldots)$

applying constraints to a decay tree
unseen region: indirect reconstruction



benefits:

- better reconstruction of 3-momenta and lifetimes ➔ *better precision on physics results*
- resolution of ambiguities on the topology of the event, if any

summary for track and vertex fit

- one can build a track fitting procedure by linking **elementary operations** on the local parameters trajectory (adding one measurement, adding one noise, propagation)
- when putting these operations in **order**, each step uses **independent** inputs
- in the **linear approximation** (almost always valid in useful cases), the steps are simple manipulations of 5-vectors and (5x5) matrices
- in the gaussian approximation one can define quality tests in terms of Prob(chi2), either for the global fit, or for a given point (detection of outliers)
- exogenous measurements may be injected at some steps (e.g. detectable energy losses)
- if needed, some non gaussian effects may be taken into account (esp. for electrons)
- the track fit may be coupled to the pattern recognition to refine prediction to a layer (a large variety de strategies are possible)
- the vertex fit may be achieved in a fast procedure (CPU time proportional to the number of tracks) with flexibility (adding or removing a track is easy)
- geometrical and physical constraints may be added to improve the final reconstruction: invariant masses, combination of connected vertices in a decay tree

procedures of alignment

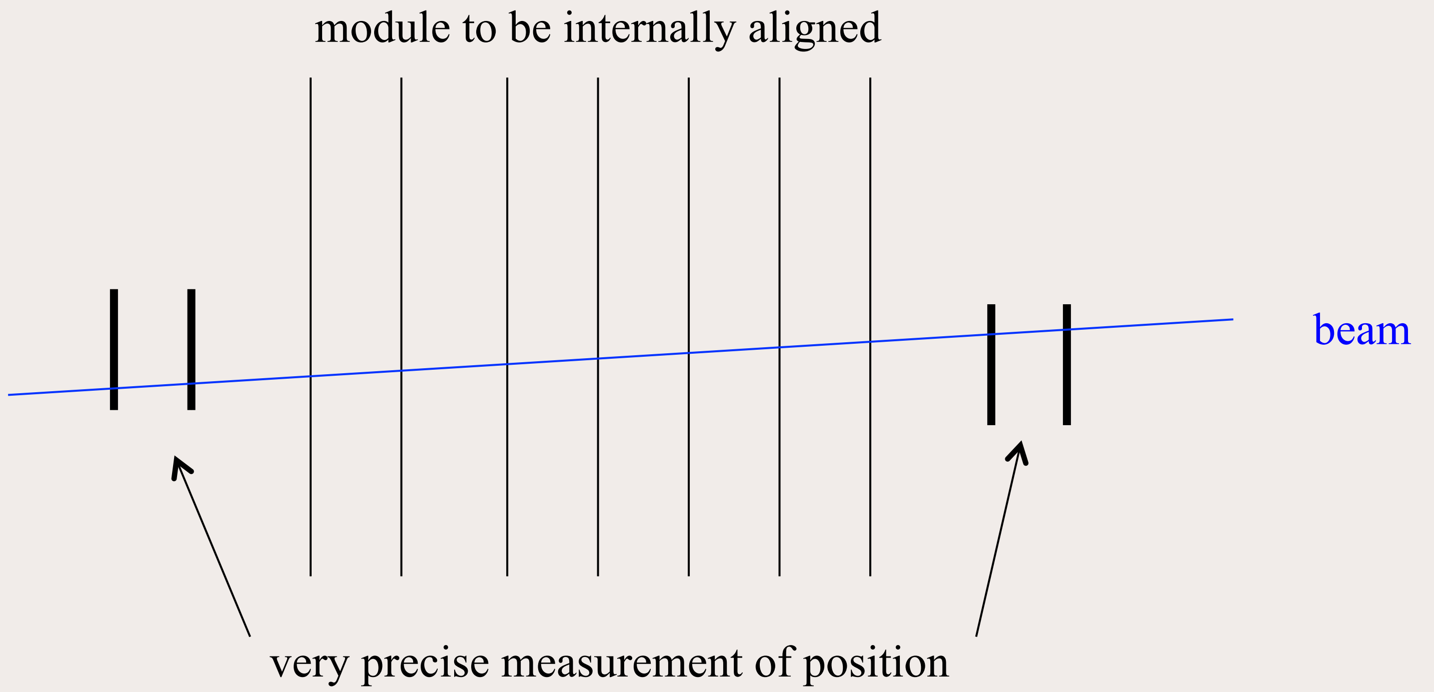
detector = assembly of elements supposed to be rigid
geometrical degrees of freedom for each element:
translation, rotation; *expansion, contraction ?*

first order: position of frames (« hardware » sensors)
second order: fine corrections (position of sensitive elements)
using signals from tracks (beam, cosmics, collision data)

calibration

determine shape of signals, biases, measurement errors
- simulation
- external inputs (beam, cosmics, point sources, pulses on elements of the electronic chain)
- internal
- with or without B field

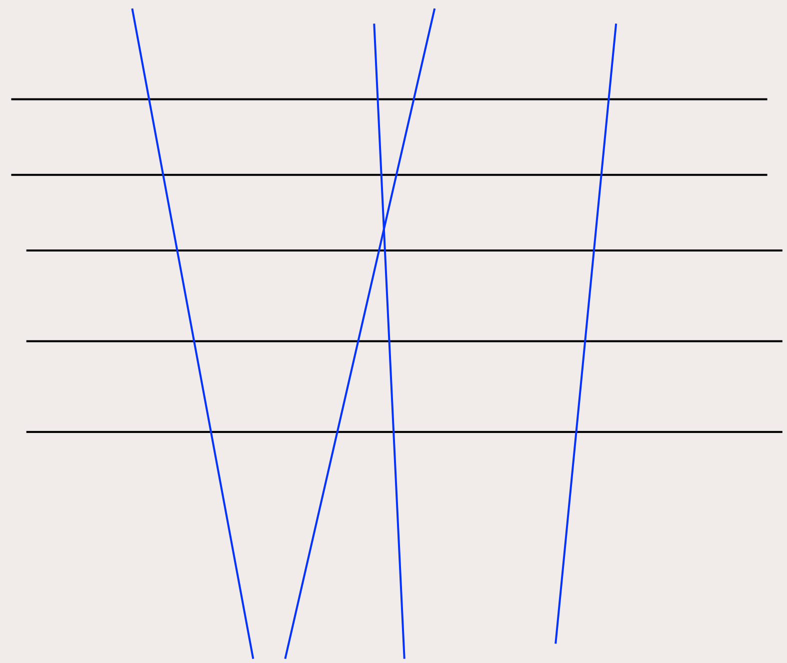
1. beam + external hodoscope



the module should be moveable: impossible for big detectors

2. cosmic rays

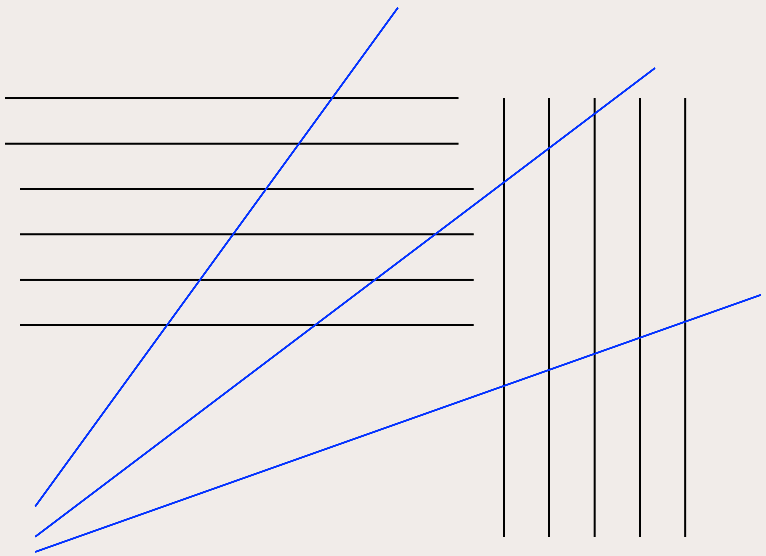
mainly vertical (esp. in underground places) ; random impact position



may connect different modules of a big detector
*no hodoscope ! **weak modes** may exist*

3. internal track sample

large statistics, real time data, but useful tracks come mainly from origin



momentum dependent: again ***weak modes***

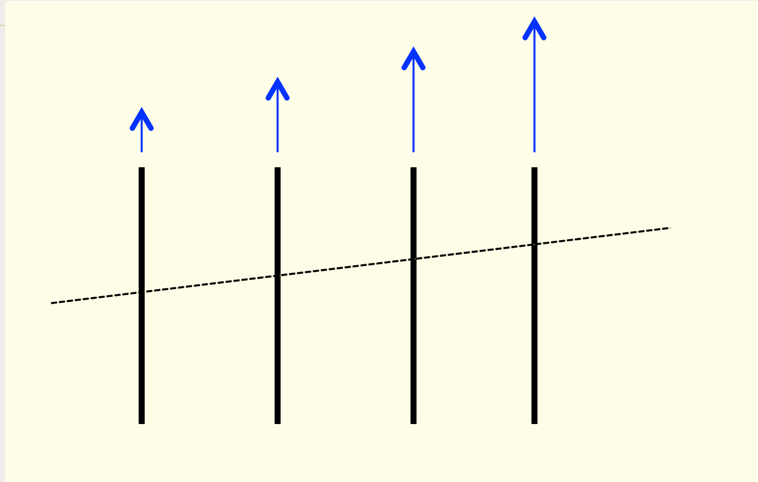
just for fun

when using a sample of tracks to make an alignment, you have to adjust:

- a few global parameters (the geometrical ones you want to obtain)
- individual parameters for each track (position, direction + curvature if magnetic field)

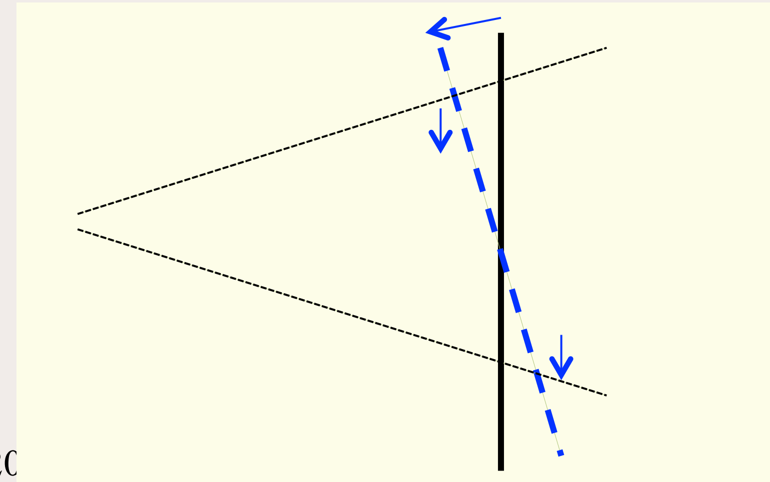
this is again a hierachical fit !

examples of « weak modes » in an internal geometric alignment



translation of all planes by a linear function of z:

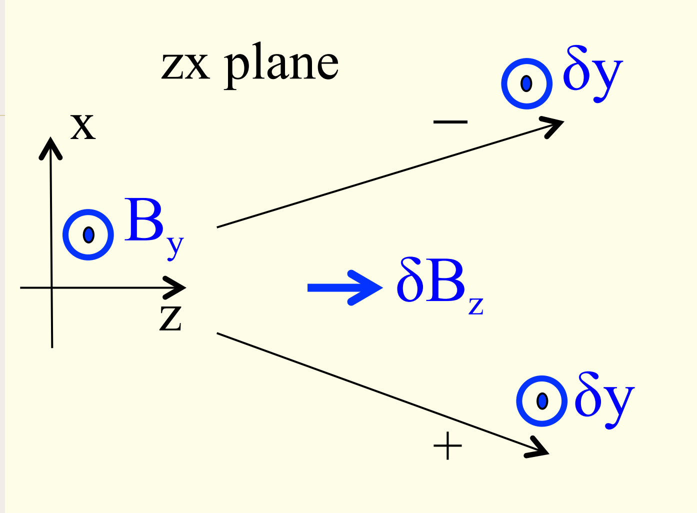
- *if no curvature: exactly compensated by a change of slope*
- *if curvature: compensated at first order*



for a sample of *divergent* tracks
a small **rotation** is equivalent (in average) to a **translation** along the other axis
some combination of them is weakly constrained

examples of correlation between alignment and field map

in a field mainly along y axis:
positive/negative particles get in average negative/positive x, t_x: the x>0 and x<0 have opposite average charge populations

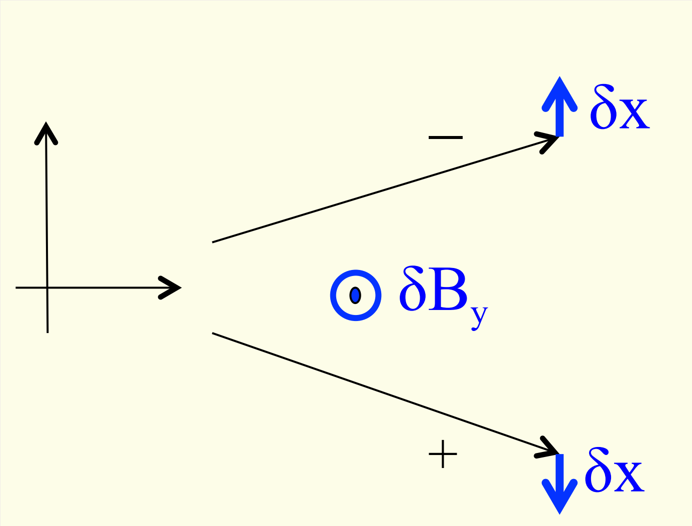


- a positive δB_z pushes both signs upwards

partially compensated by pushing a z-plane downwards (negative Δy)

- an increase of $|B_y|$ increases the divergence

partially compensated by pushing a z-plane towards negative z
if separate alignment of x-sides: pushing them in opposite directions



an alignment by tracks may give different results depending on the range of momenta

perspectives for the future

active development in various fields:

- machine learning
- real time reconstruction: parallel/local computation
- use timing information (progresses in hardware)

backup

a problem of precision (LHCb)

trying to implement the Kalman Filter included in PrPixelTracking (Velo)
in single precision on a GPU:

- *discrepancies between the GPU and the CPU results, and between them and the weight/information algorithm, when applied to the same data*
- *more precisely: the discrepancies (on fitted position/slope, covariance matrix, chi2) decrease with the number of points in the track*
- *agreement between all versions in double precision, and between single and double with the weight formalism*
- *the discrepancies increase with the initial value given to cov(Tx,Tx) and cov(Ty,Ty) at the beginning of the loop over points*

origin and solution of the problem

```
// compute the prediction
const float dz = zhit - z;
const float predx = x + dz * tx;

const float dz_t_covTxTx = dz * covTxTx;
const float predcovXTx = covXTx + dz_t_covTxTx;
const float dx_t_covXTx = dz * covXTx;

const float predcovXX = covXX + 2 * dx_t_covXTx + dz * dz_t_covTxTx;
const float predcovTxTx = covTxTx;
// compute the gain matrix
const float R = 1.0 / (1.0 / whit + predcovXX);
const float Kx = predcovXX * R;
const float KTx = predcovXTx * R;
// update the state vector
const float r = xhit - predx;
x = predx + Kx * r;
tx = tx + KTx * r;
// update the covariance matrix. we can write it in many ways ...
covXX /*= predcovXX  - Kx * predcovXX */ = (1 - Kx) * predcovXX;
covXTx /*= predcovXTx - predcovXX * predcovXTx / R */ = (1 - Kx) * predcovXT
covTxTx = predcovTxTx - KTx * predcovXTx;
// return the chi2
return r * r * R;
```

at first point $C_{xx} = \sigma^2$, $C_{TxTx} = \mathbf{Big}$
(in this code: **Big** *= 1)*
the loop (pred, upd, noise) begins at the ***second*** point with a ***nearly singular*** predicted covariance :
$C'_{xx} = \sigma^2 + \mathbf{Big}^2 \Delta z^2$
$C'_{xTx} = \mathbf{Big}\, \Delta z$, $C'_{TxTx} = \mathbf{Big}$
the « gain » business mixes **Big** and real quantities ➔ rounding errors !

here: making **Big** ➔ ∞ in the results after updating at point 2:
$x = x_2$ $Tx = (x_2 - x_1)/\Delta z$
$cov = (\sigma^2, \sigma^2/\Delta z, 2\sigma^2/\Delta z^2)$
$\chi^2 = 0$
the KF machinery was useless in the first step !

conclusion: do not rely blindly on black boxes
put your eyes inside, and put your hands if needed