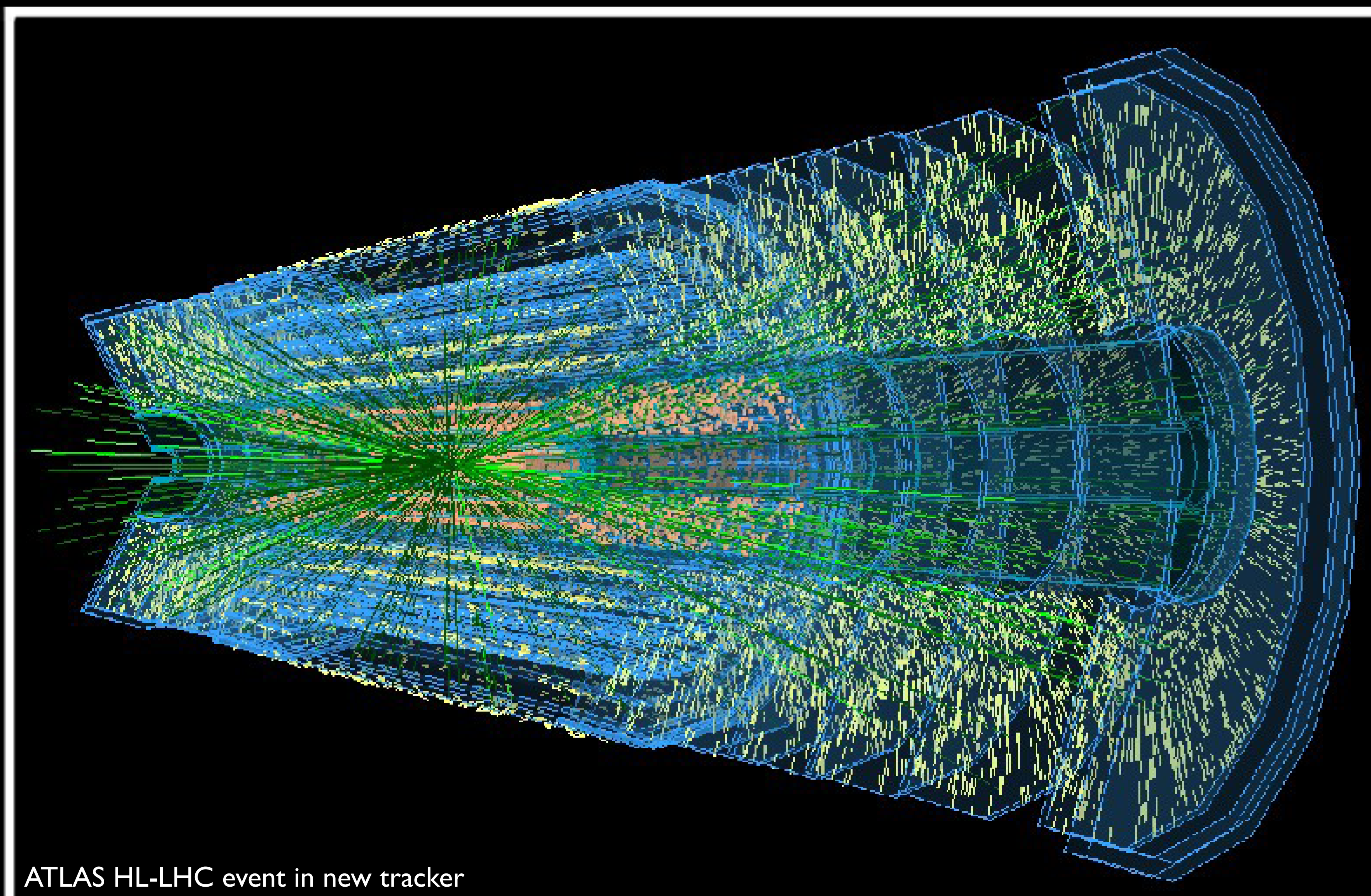
Tracking at the LHC (Part 3):
Concepts for Track Reconstruction

Lectures given at the University of Freiburg

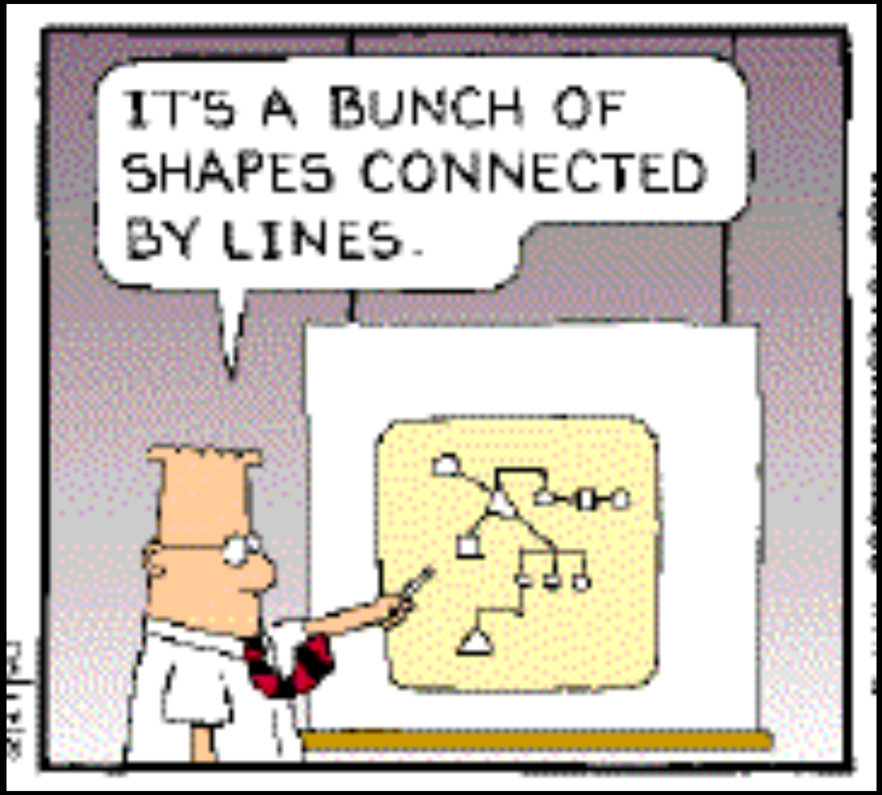
Markus Elsing, 12-13.April 2016



ATLAS HL-LHC event in new tracker

Introduction

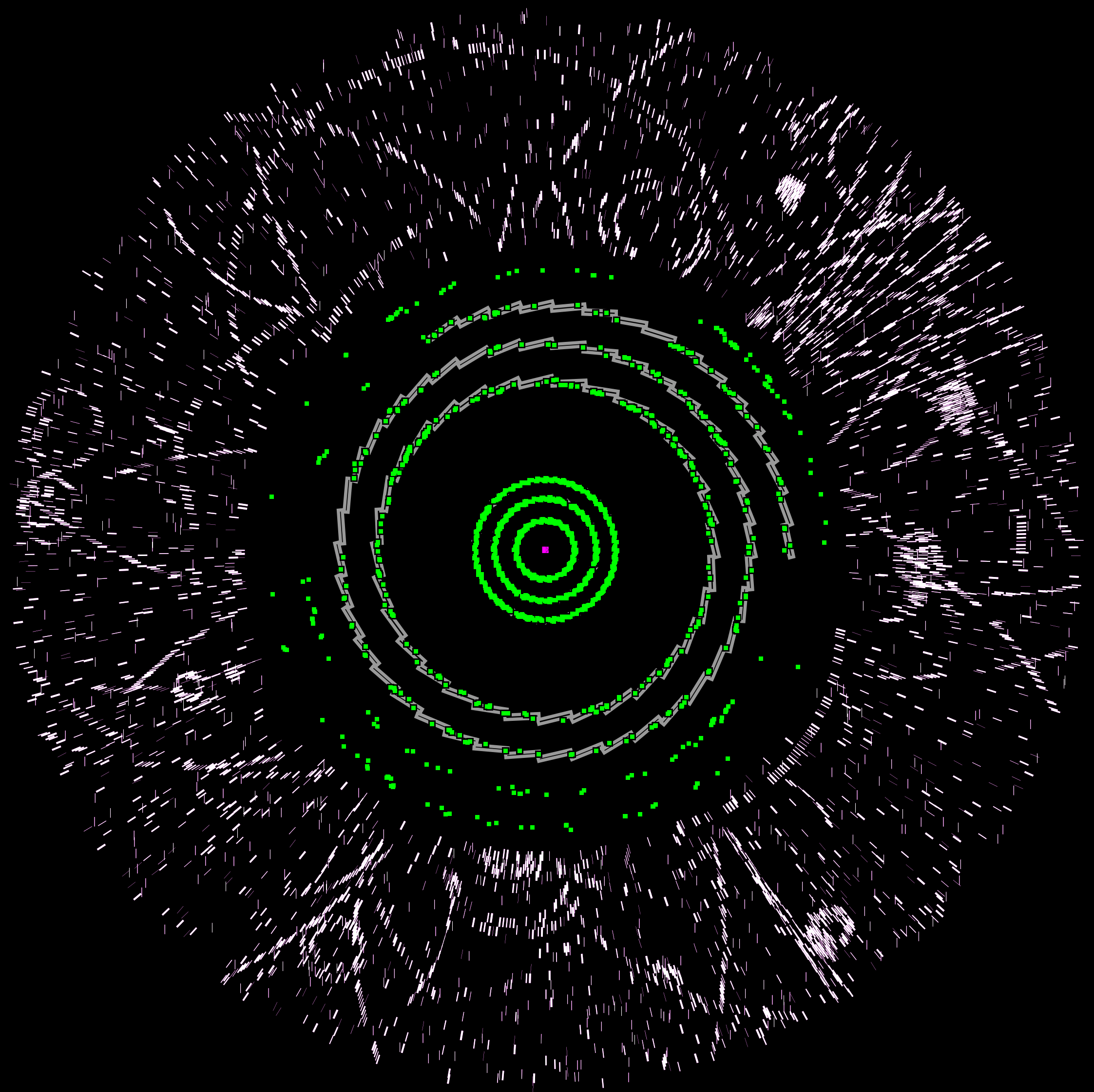
- in this lecture I will discuss the most complex and CPU consuming aspect of event reconstruction at the LHC
  - ➡ finding trajectories (tracks) of charged particles produced in p-p collisions
- will have to introduce various techniques for
  - ➡ pattern recognition, detector geometry, track fitting, extrapolation ...
  - ➡ including mathematical concepts and aspects of software design



... so why does it matter ?

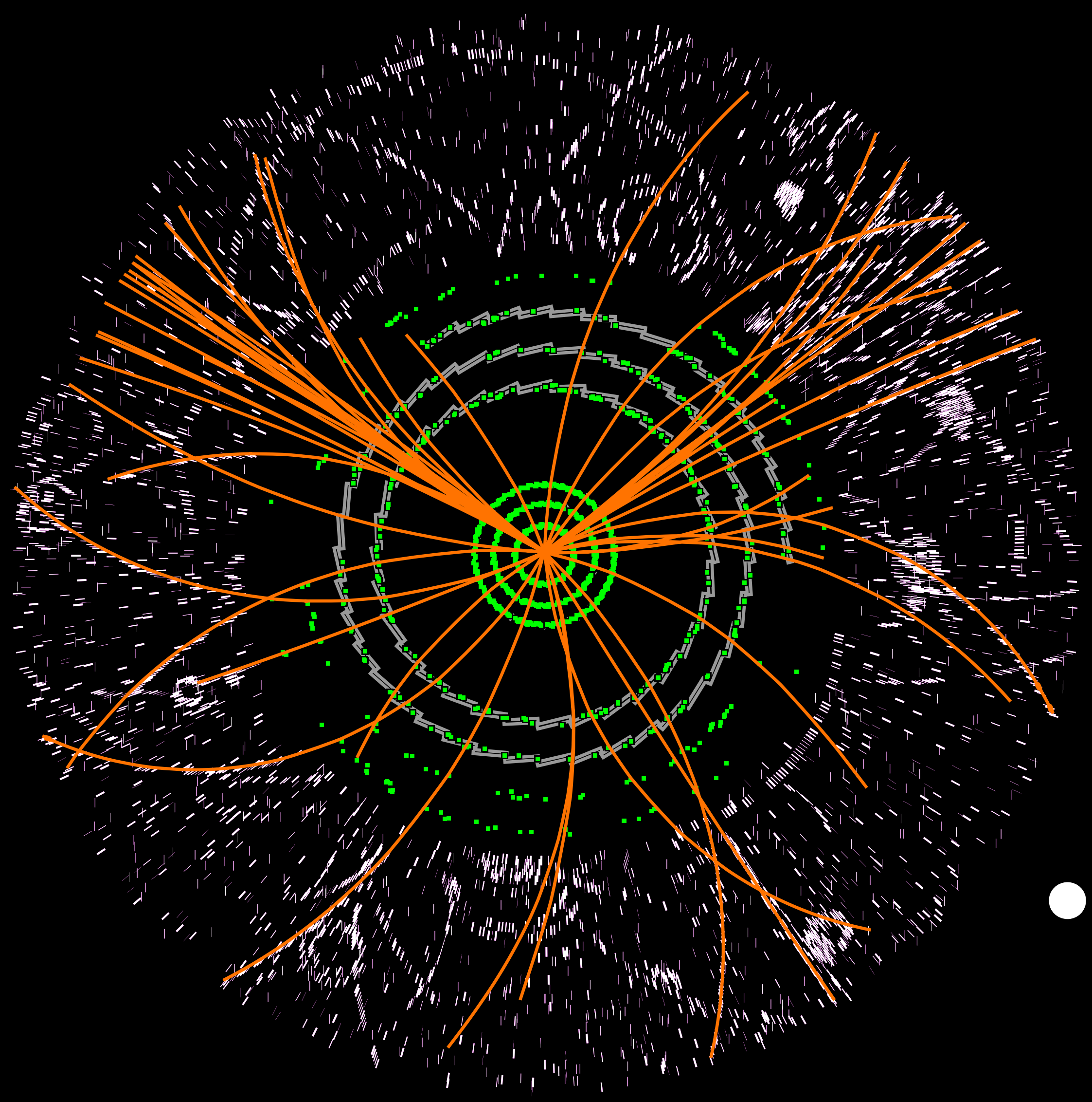
The Tracking Problem

- particles produce in a p-p interaction leave a cloud of hits in the detector



The Tracking Problem

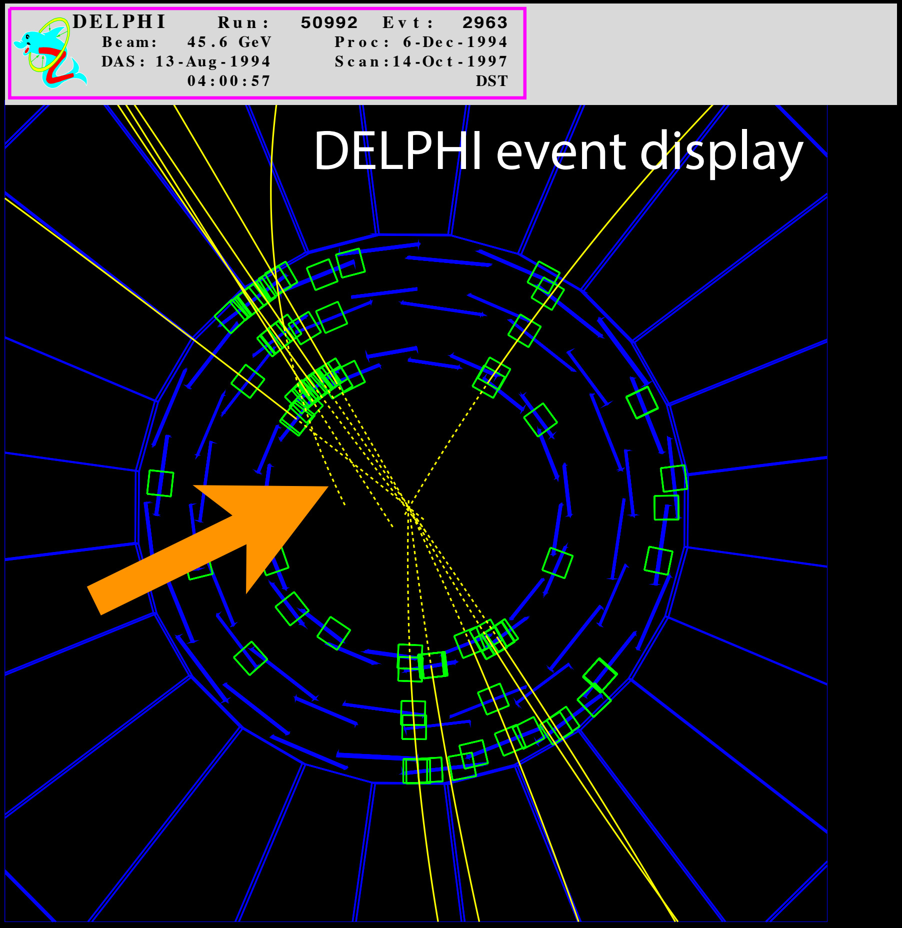
- particles produce in a p-p interaction leave a cloud of hits in the detector

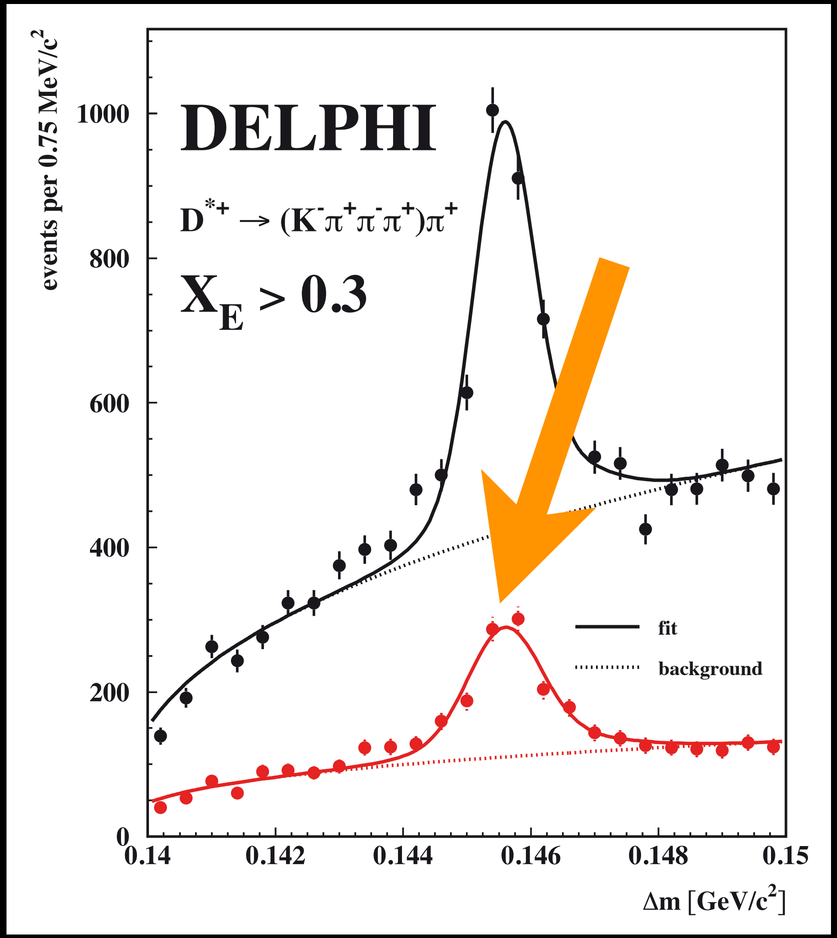


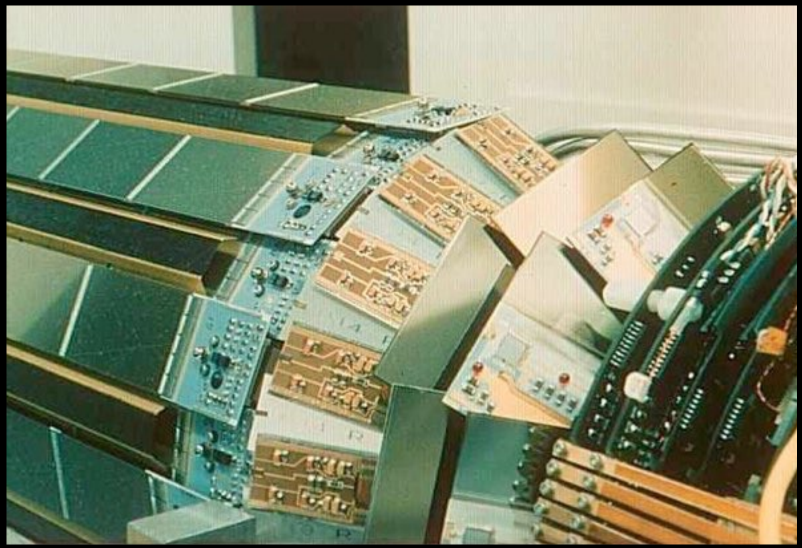
- tracking software is used to reconstruct their trajectories

Role of Tracking Software

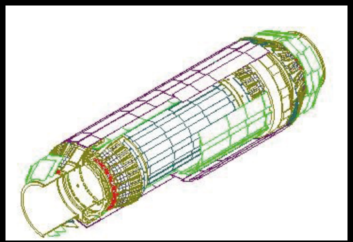
- **optimal** tracking software
  - ➡ required to fully explore performance of detector
- **example**: DELPHI Experiment at LEP
  - ➡ silicon vertex detector upgrade
    - initially not used in tracking to resolve dense jets
    - pattern mistakes in jet-chamber limit performance





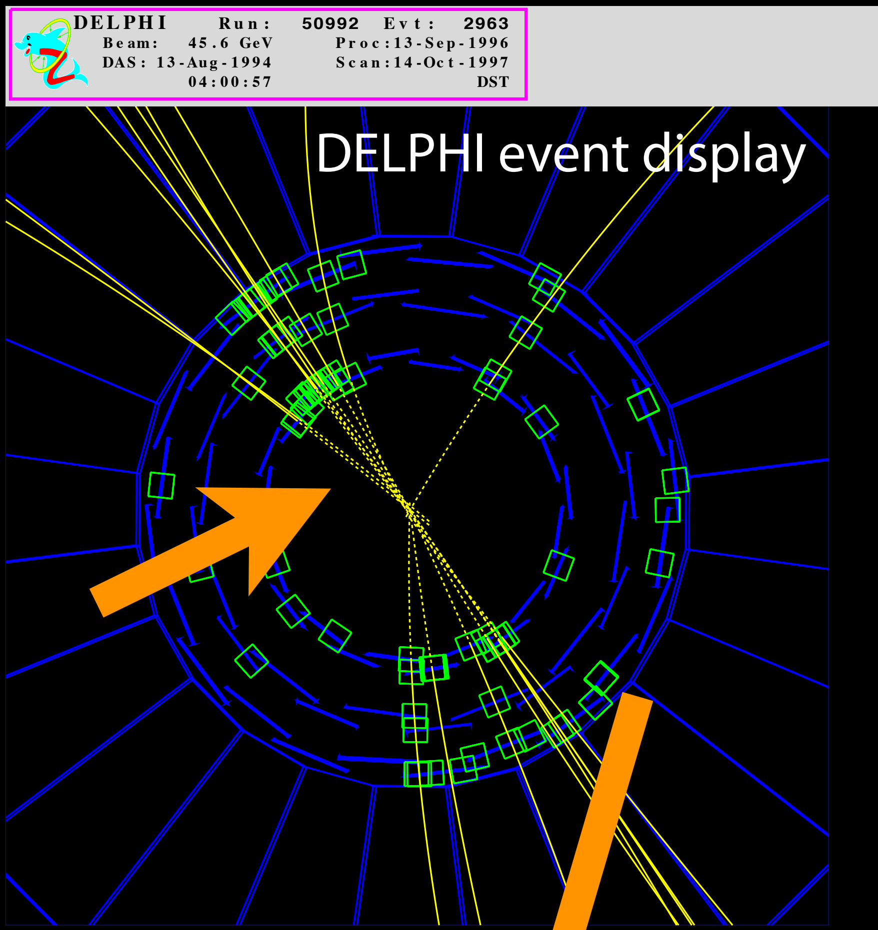


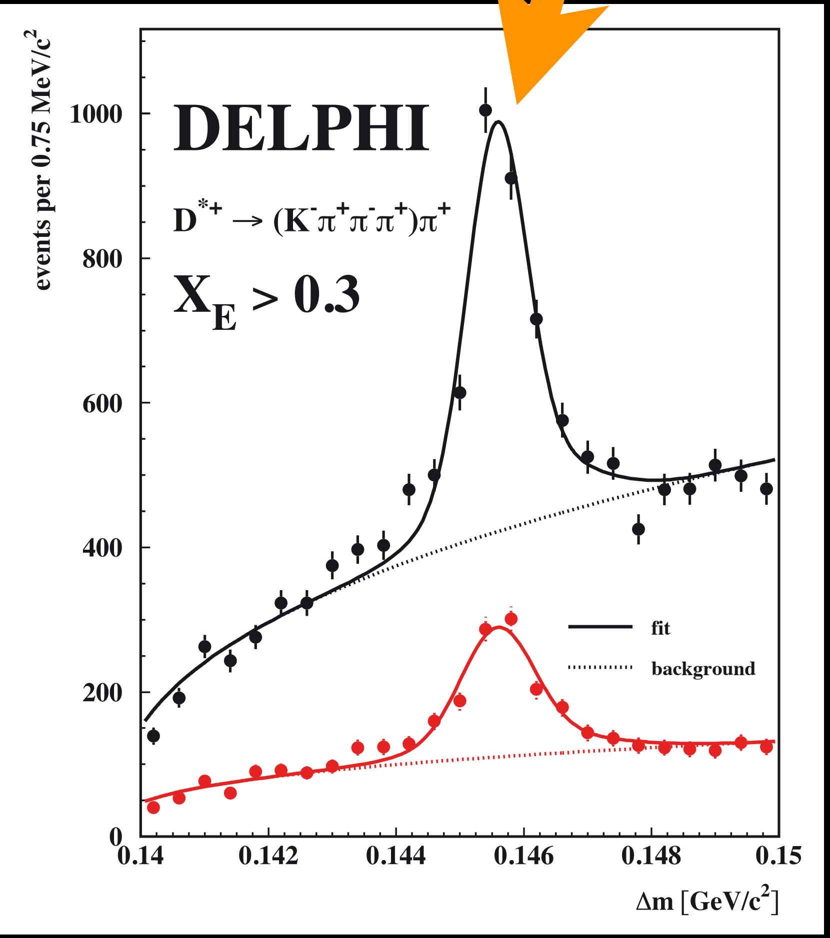
DELPHI vertex detector



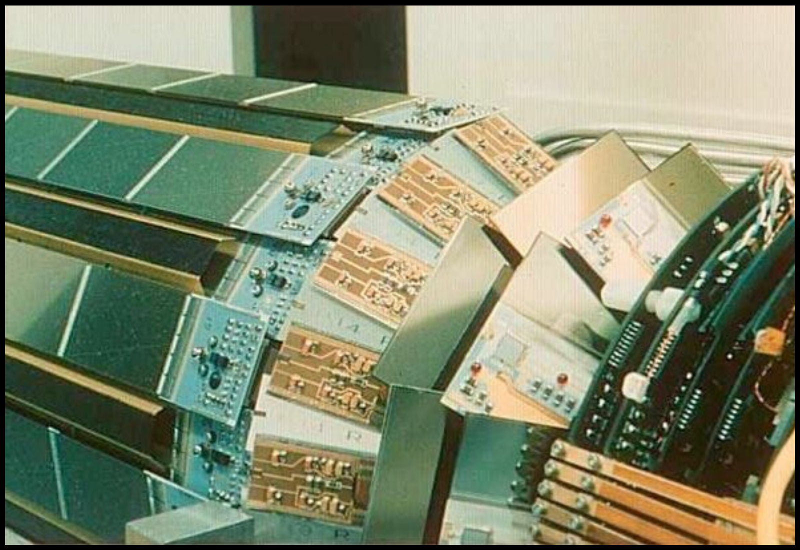
Role of Tracking Software

- **optimal** tracking software
  - ➡ required to fully explore performance of detector
- **example**: DELPHI Experiment at LEP
  - ➡ silicon vertex detector upgrade
    - initially not used in tracking to resolve dense jets
    - pattern mistakes in jet-chamber limit performance
  - ➡ 1994: redesign of tracking software
    - start track finding in vertex detector
  - ➡ **factor ~ 2.5 more D* signal** after reprocessing

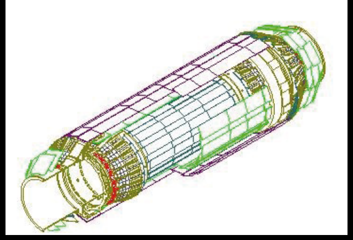




(M.E. et al)

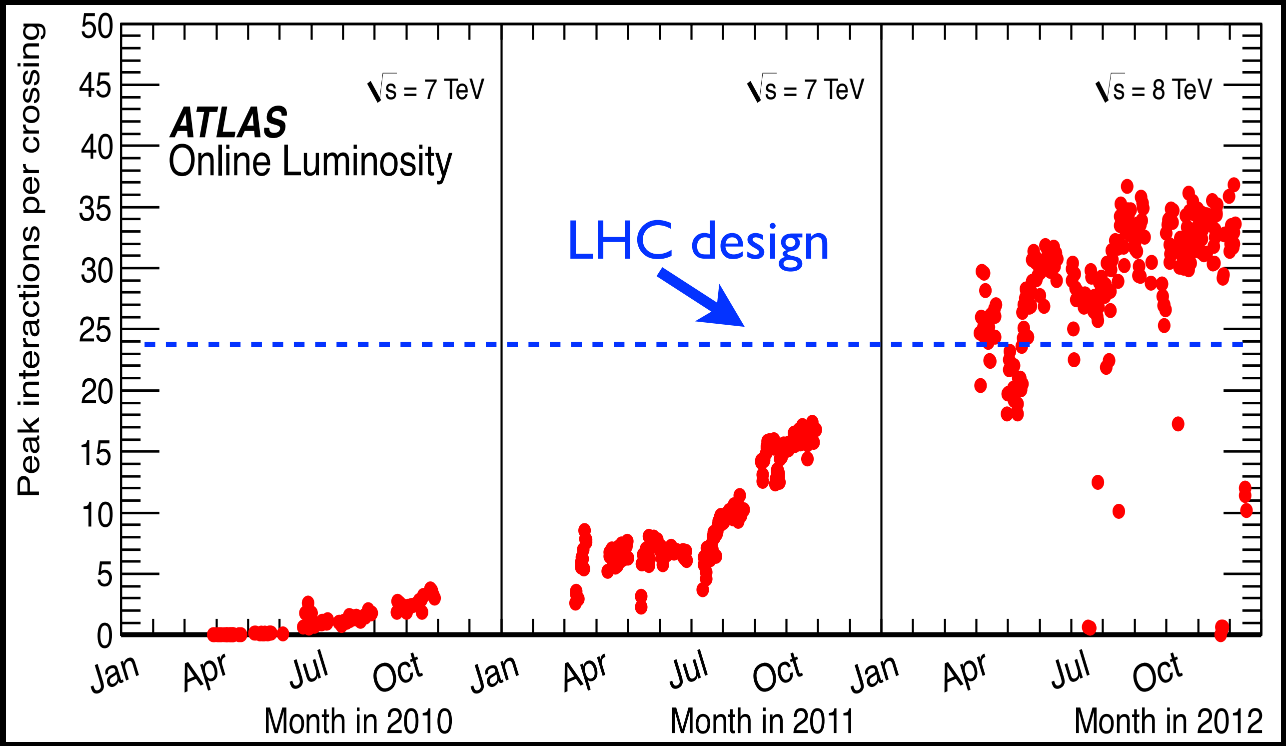


DELPHI vertex detector



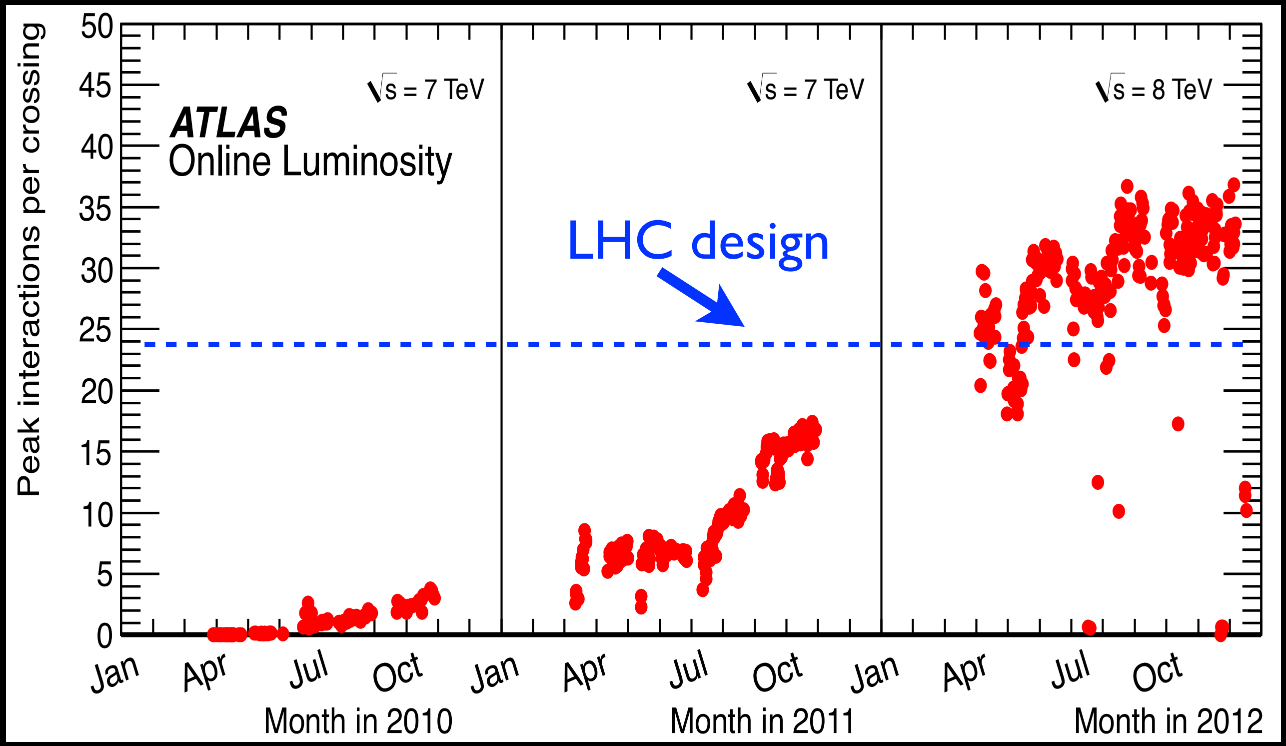
Tracking at the LHC ?

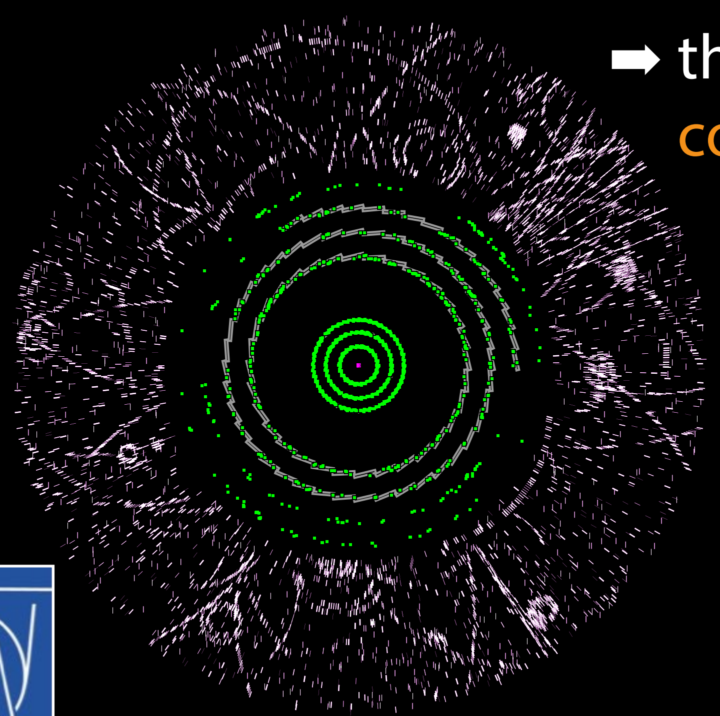
- reminder:
  - ➡ LHC is a high luminosity machine
    - proton bunches collide every 25 (50) nsec in experiments
    - each time > 20 p-p interactions are observed ! (event pileup)
  - ➡ our detectors see hits from particles produced by all > 20 p-p interactions
    - ~100 particles per p-p interaction
    - each charged particle leaves ~50 hits



Tracking at the LHC ?

- reminder:
  - ➡ LHC is a high luminosity machine
    - proton bunches collide every 25 (50) nsec in experiments
    - each time > 20 p-p interactions are observed ! (event pileup)
  - ➡ our detectors see hits from particles produced by all > 20 p-p interactions
    - ~100 particles per p-p interaction
    - each charged particle leaves ~50 hits

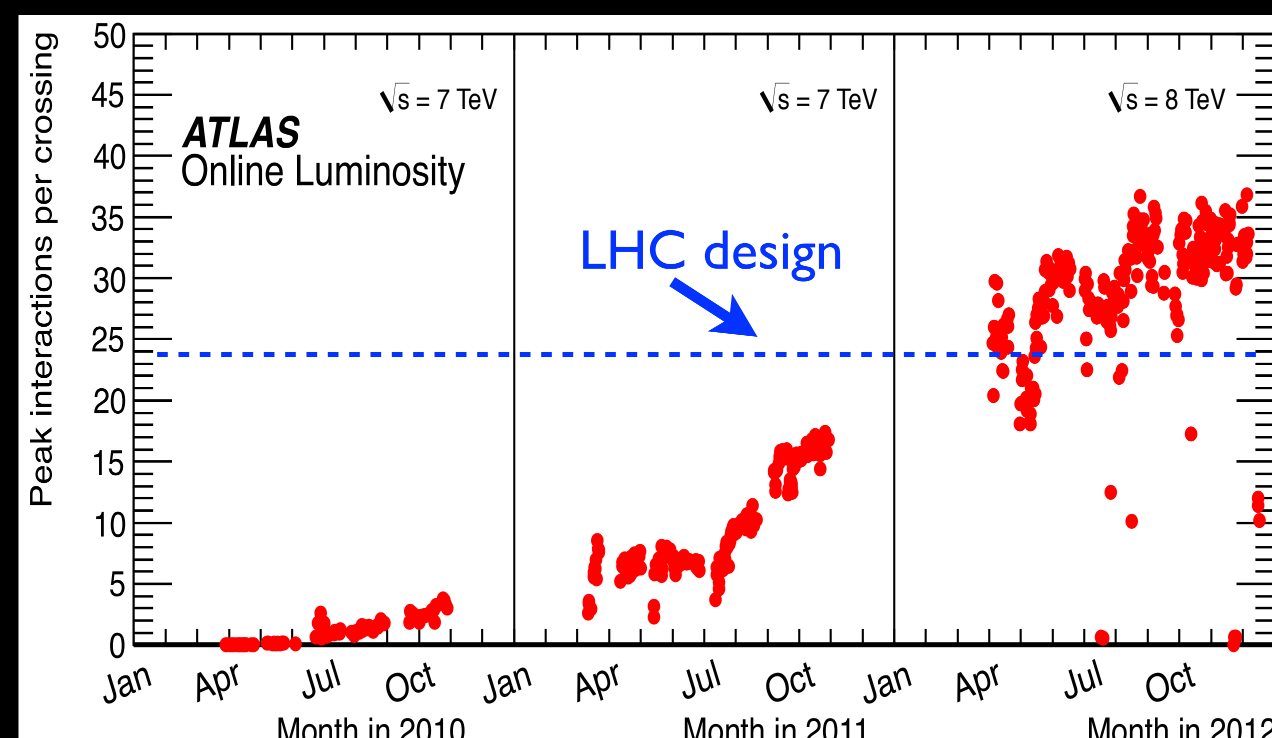


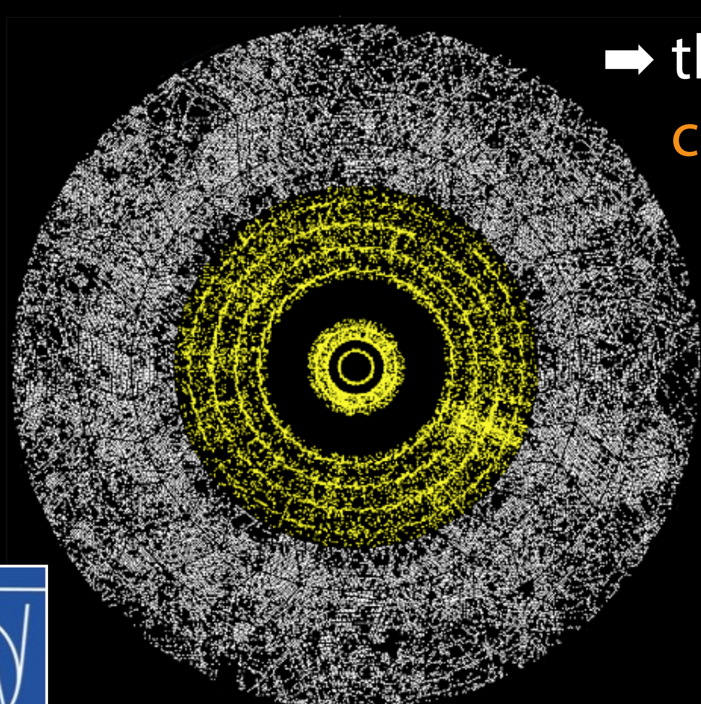


➡ this is how 1 pp collisions looks like

Tracking at the LHC ?

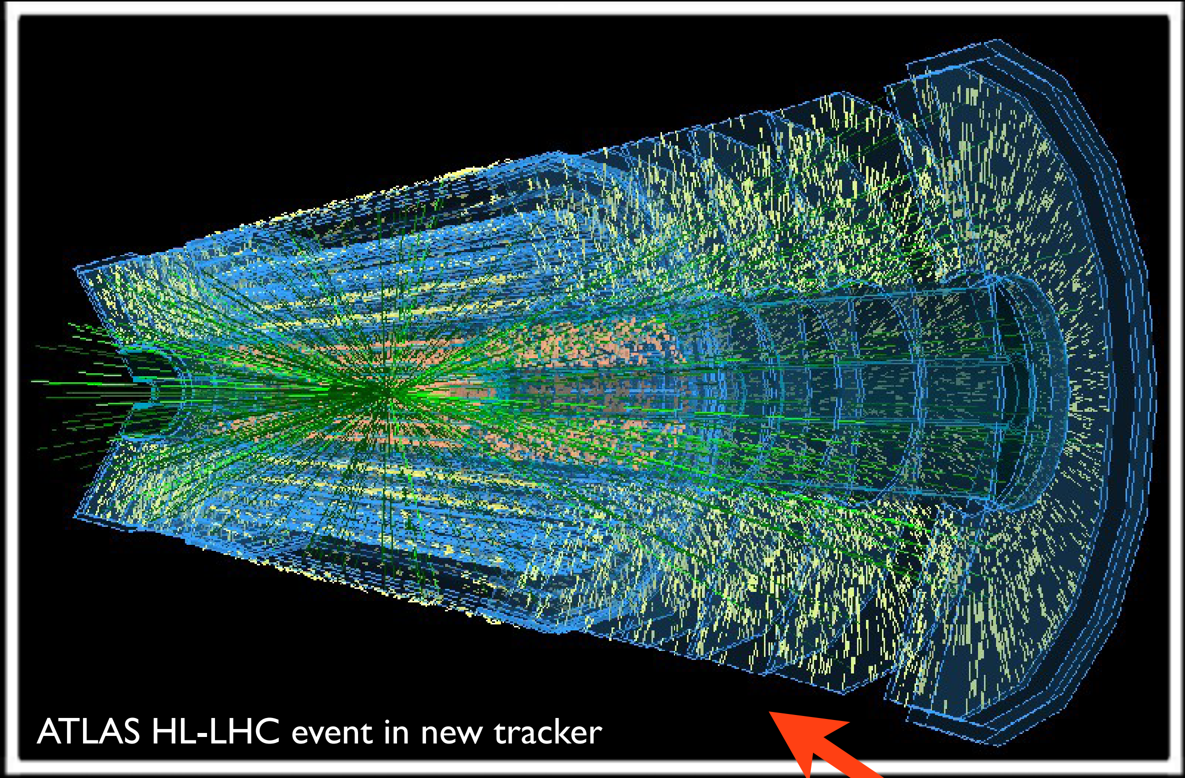
- reminder:
  - ➡ LHC is a high luminosity machine
    - proton bunches collide every 25 (50) nsec in experiments
    - each time > 20 p-p interactions are observed ! (event pileup)
  - ➡ our detectors see hits from particles produced by all > 20 p-p interactions
    - ~100 particles per p-p interaction
    - each charged particle leaves ~50 hits





➡ this is how 1 pp collisions looks like
- now imagine 50 of them overlapping
- task of tracking software is to resolve the mess ...

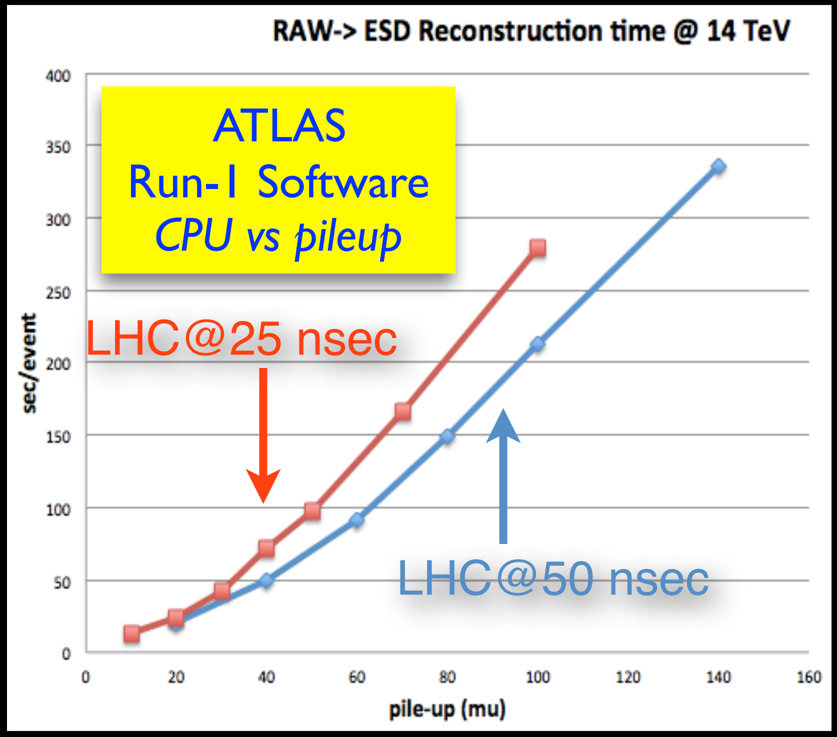
Tracking at the LHC ?



- **track reconstruction**
  - ➡ combinatorial problem grows with pileup
  - ➡ naturally resource driver (CPU/memory)
- **the million dollar question:**
  - ➡ how to reconstruct LH-LHC events within resources ? (pileup ~ 140-200)

event display
from title page

- **more than 10 years of R&D on LHC tracking software**
  - ➡ we knew that tracking at the LHC is going to be challenging
    - building on techniques developed for previous experiments
  - ➡ processor technologies will change in the future
    - need to rethink some of the design decisions we did
    - adapt software to explore modern CPUs: threading, data locality...



Outline of Part 3

- charged particle trajectories and extrapolation
  - ➡ trajectory representations and trajectory following in a realistic detector
  - ➡ detector description, navigation and simulation toolkits
- track fitting
  - ➡ classical least square track fit and a Kalman filter track fit
  - ➡ examples for advanced techniques
- track finding
  - ➡ search strategies, Hough transforms, progressive track finding, ambiguity solution
- the ATLAS track reconstruction (as an example)

Trajectories and Extrapolation

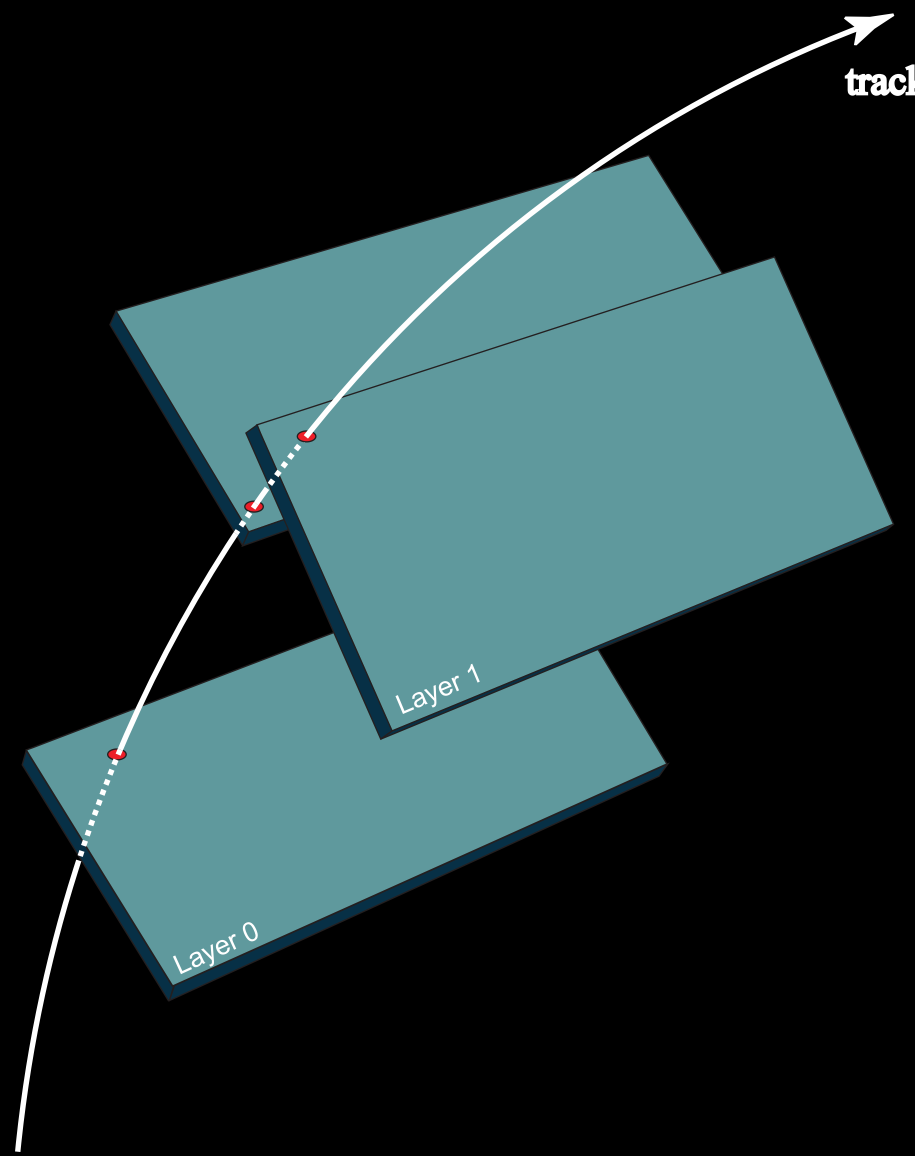
A Trajectory of a Charged Particle

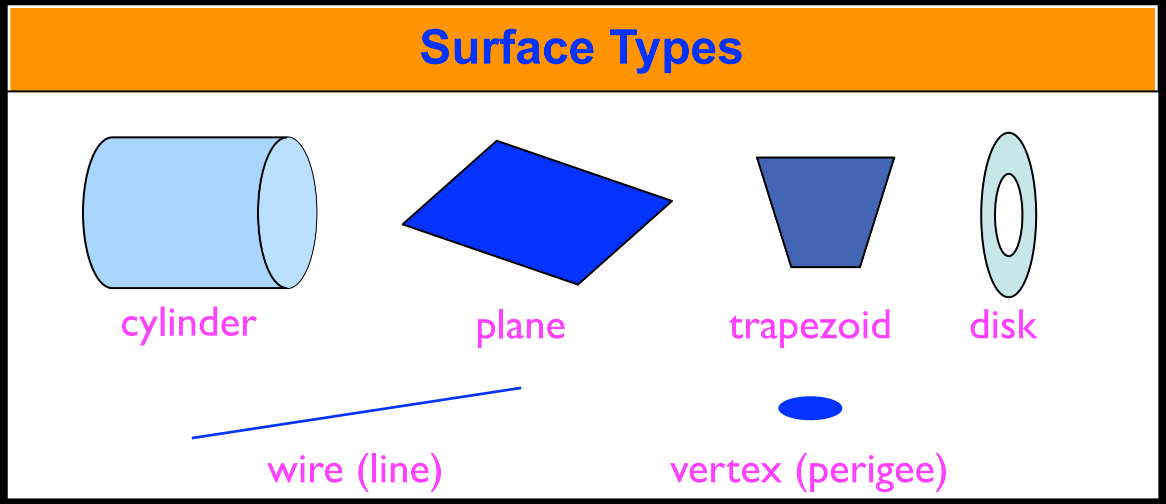
- ➡ in a solenoid B-field a charged particle trajectory is describing a **helix**
  - a circle in the plane perpendicular to the field (Rϕ)
  - a path (not a line) at constant polar angle (θ) in the Rz plane

- ➡ a trajectory in space is defined by **5 parameters**
  - the **local position** (l_1,l_2) on a plane, a cylinder, ..., on the surface or reference system
  - the **direction** in θ and ϕ plus the **curvature** Q/P_T

- ➡ ATLAS **choice**:

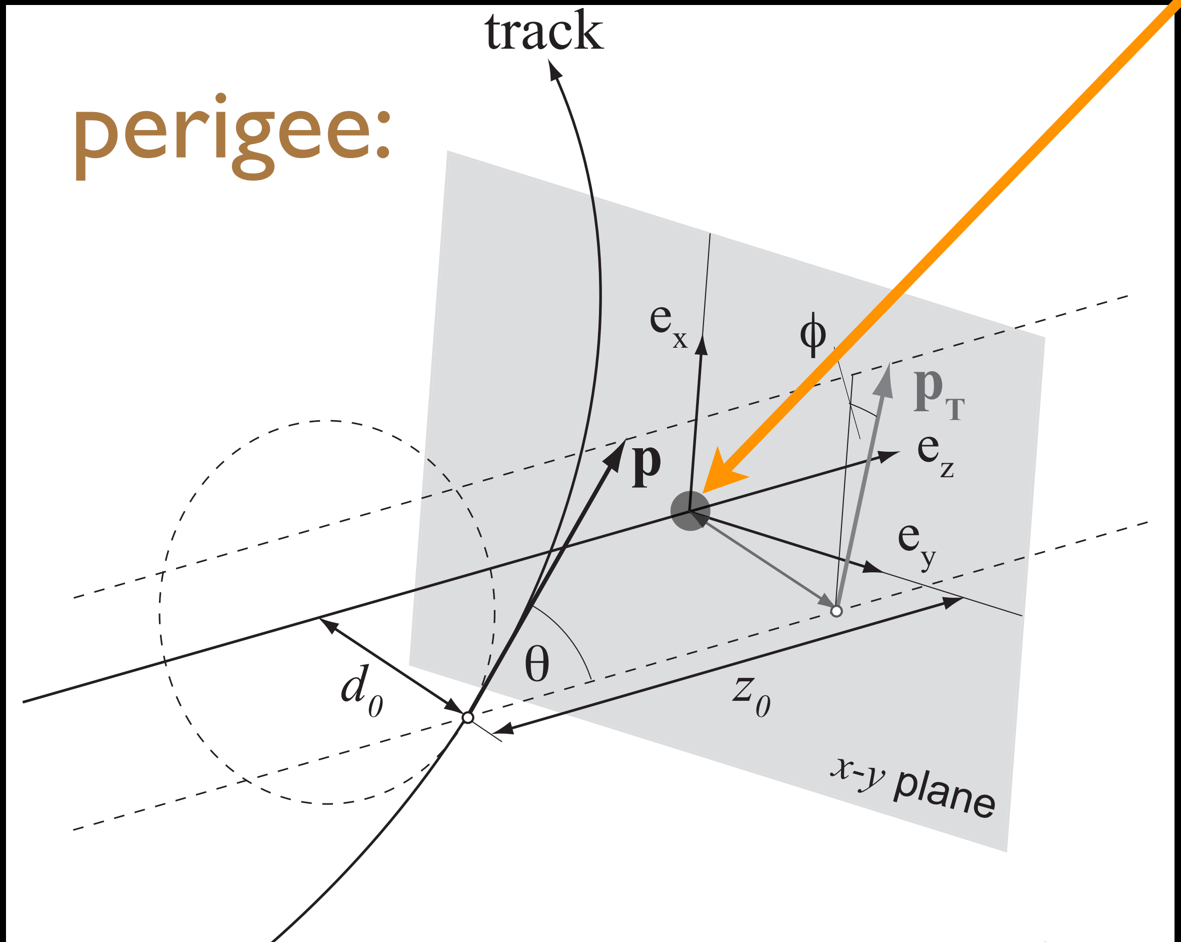
$$\vec{p} = (l_1, l_2, \theta, \phi, Q/P)$$





The Perigee Parameterisation

- helix representation w.r.t. a vertex

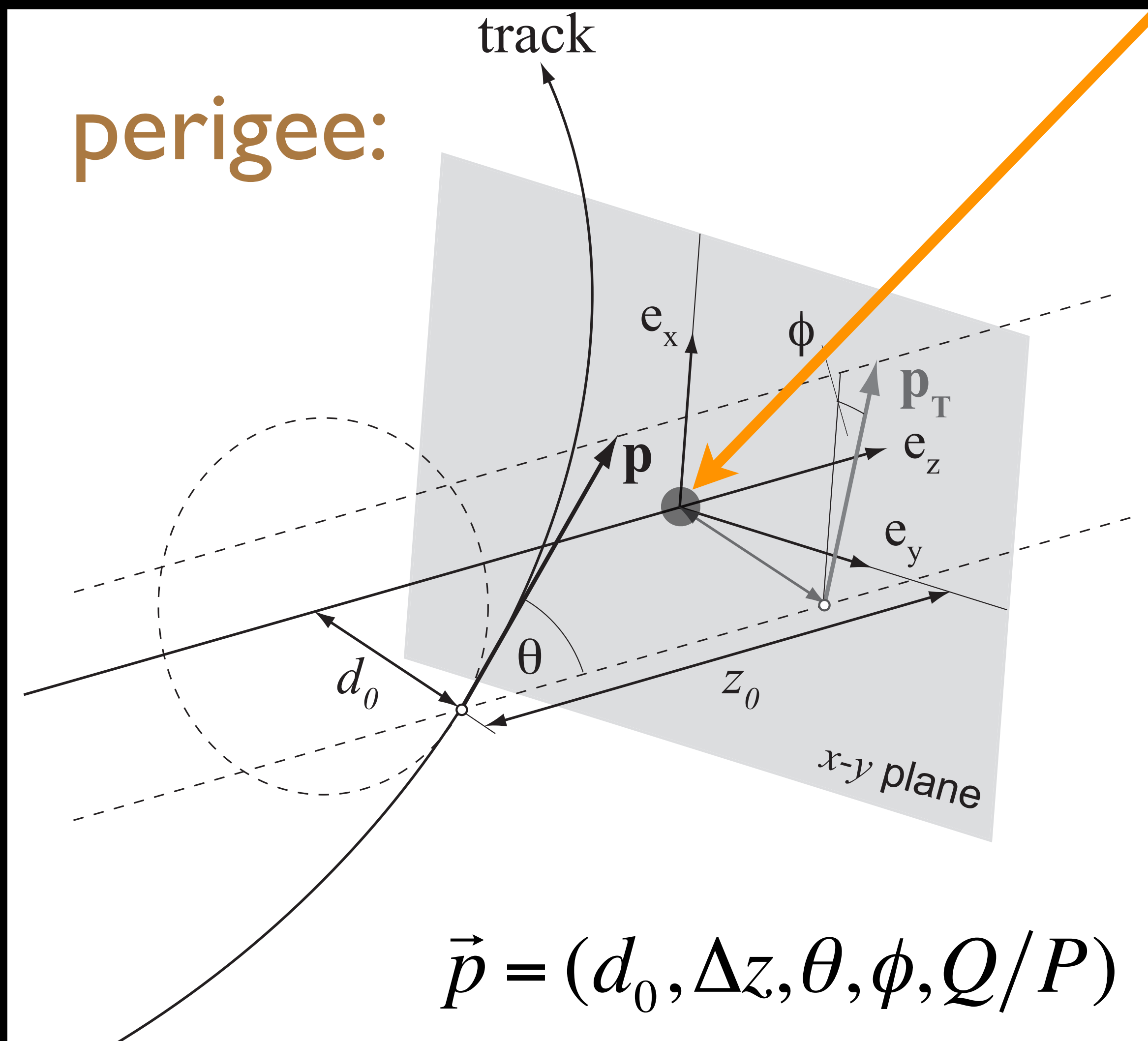


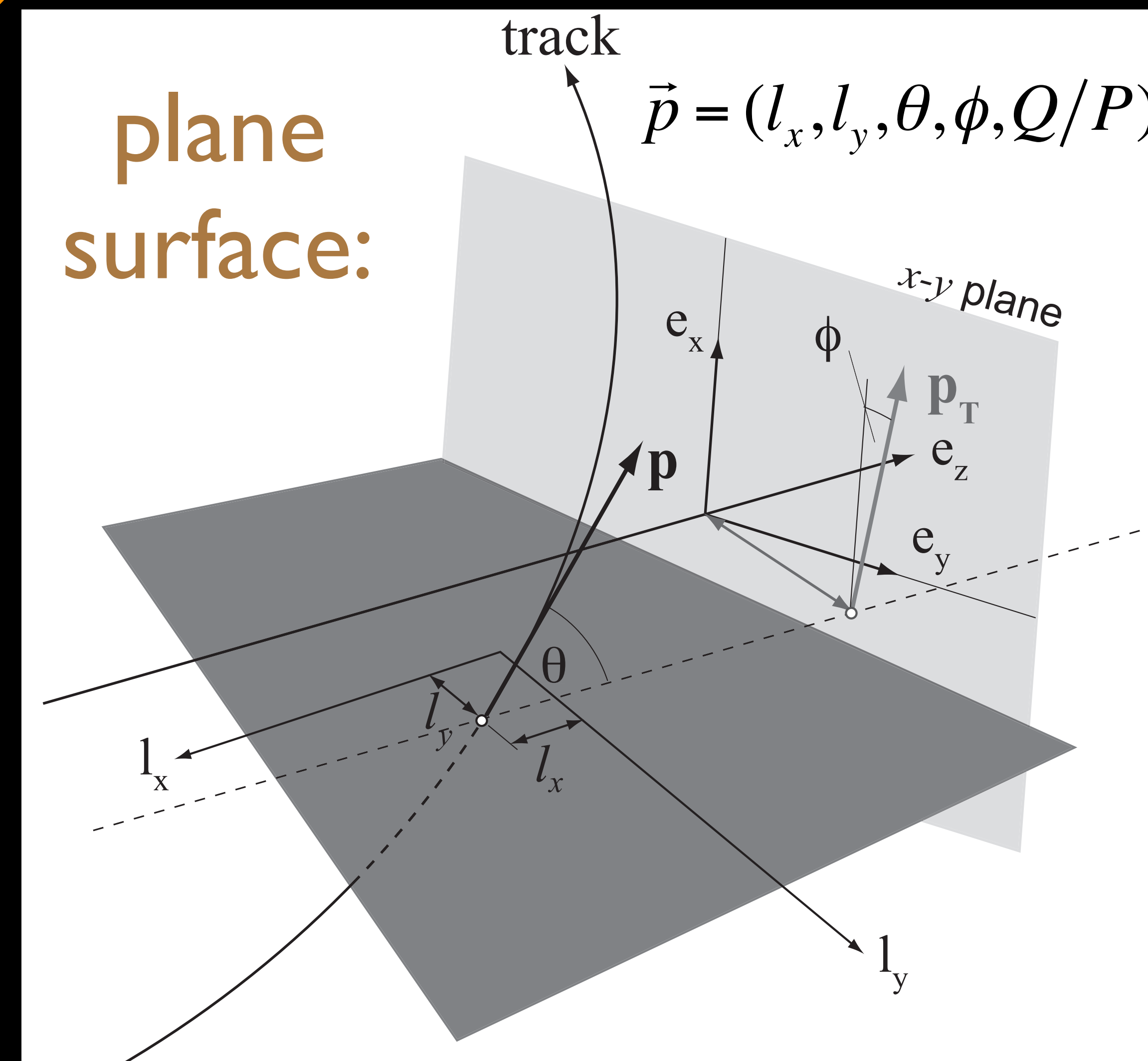
$$\vec{p} = (d_0, \Delta z, \theta, \phi, Q/P)$$

- commonly used
  - ➡ e.g. to express track parameters near the production vertex
  - ➡ alternative: e.g. on plane surface

The Perigee Parameterisation

- helix representation w.r.t. a vertex





- commonly used
  - ➡ e.g. to express track parameters near the production vertex
  - ➡ alternative: e.g. on plane surface

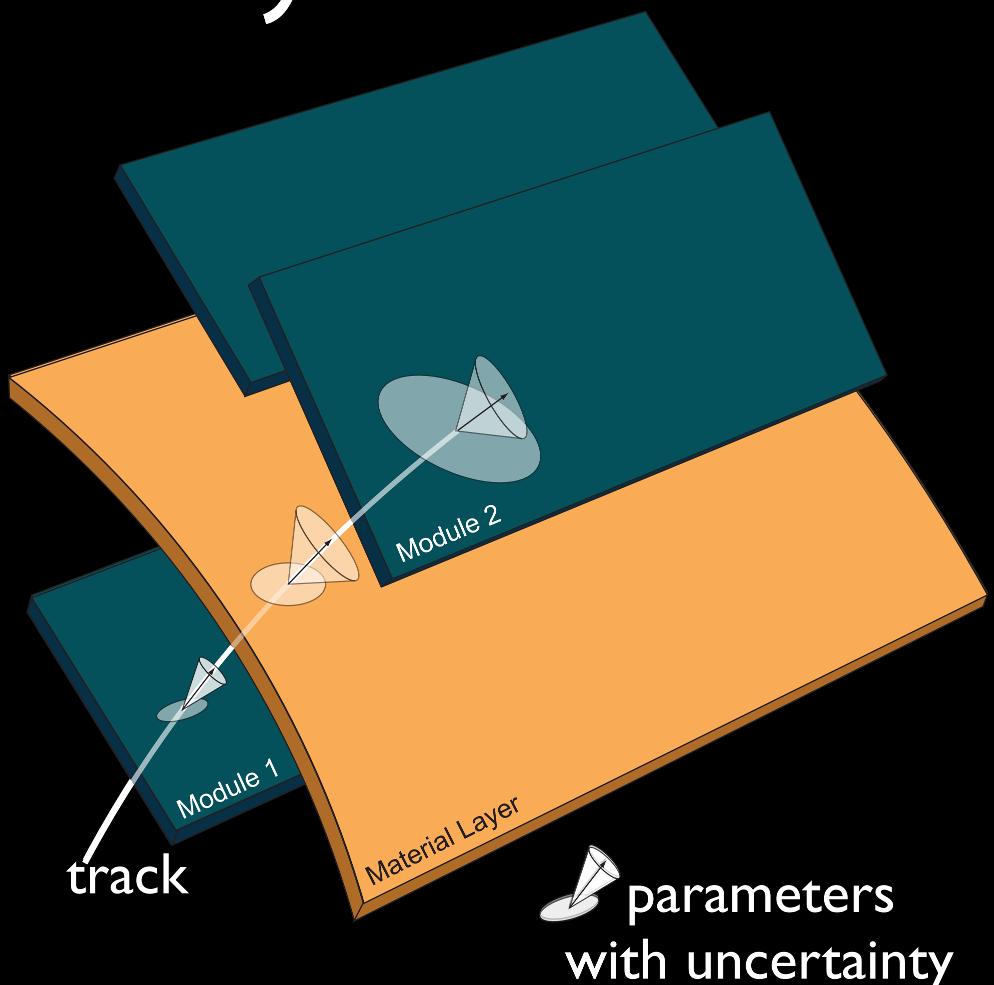
Following the Particle Trajectory

- basic problems to be solved in order to follow a track through a detector:
  - ➡ next detector module that it intersects ?
  - ➡ what are its parameters on this surface ?
    - what is the uncertainty of those parameters ?
  - ➡ for how much material do I have to correct for ?

- requires:
  - ➡ a detector geometry
    - surfaces for active detectors
    - passive material layers
  - ➡ a method to discover which is the next surface (navigation)
  - ➡ a propagator to calculate the new parameters and its errors
    - often referred to as "track model"

Following the Particle Trajectory

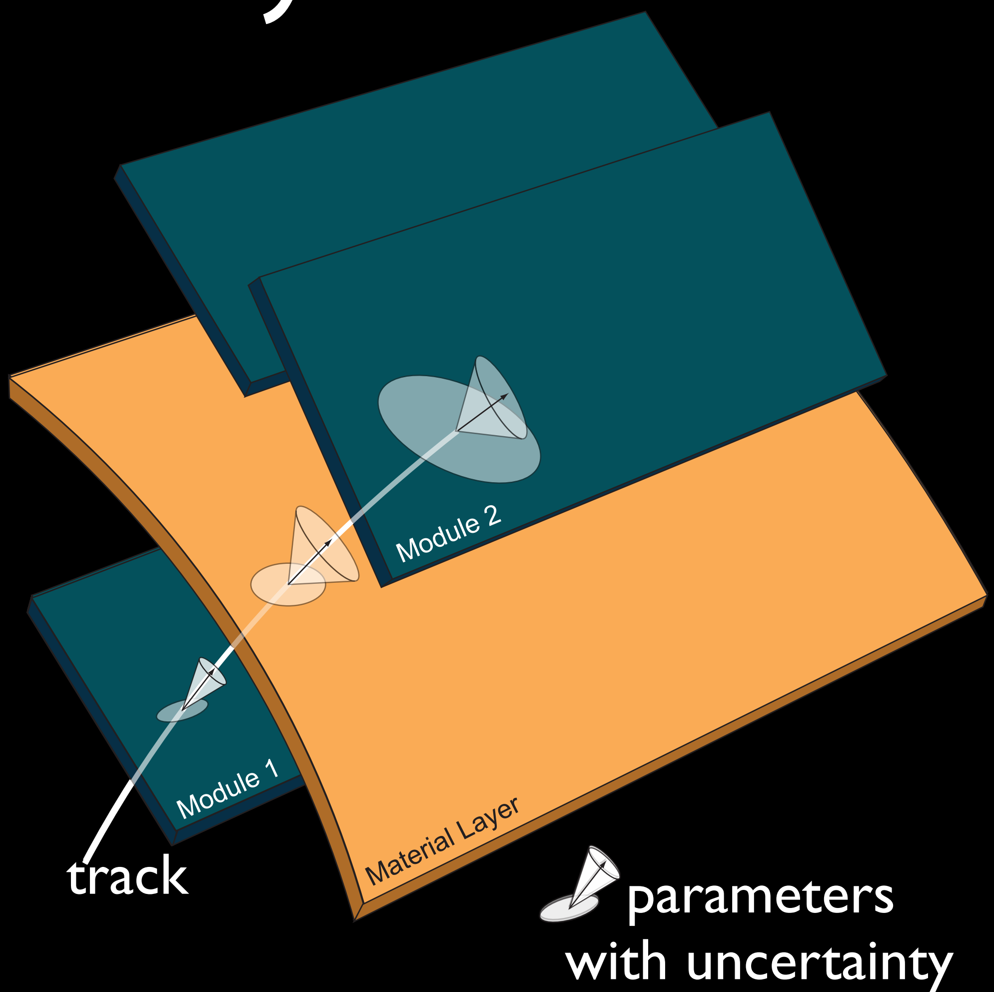
- basic problems to be solved in order to follow a track through a detector:
  - ➡ next detector module that it intersects ?
  - ➡ what are its parameters on this surface ?
    - what is the uncertainty of those parameters ?
  - ➡ for how much material do I have to correct for ?



- requires:
  - ➡ a detector geometry
    - surfaces for active detectors
    - passive material layers
  - ➡ a method to discover which is the next surface (navigation)
  - ➡ a propagator to calculate the new parameters and its errors
    - often referred to as "track model"

Following the Particle Trajectory

- basic problems to be solved in order to follow a track through a detector:
  - ➡ next detector module that it intersects ?
  - ➡ what are its parameters on this surface ?
    - what is the uncertainty of those parameters ?
  - ➡ for how much material do I have to correct for ?



- requires:
  - ➡ a detector geometry
    - surfaces for active detectors
    - passive material layers
  - ➡ a method to discover which is the next surface (navigation)
  - ➡ a propagator to calculate the new parameters and its errors
    - often referred to as "track model"

- for a constant B-field (or no field)
  - ➡ an analytical formula can be calculated for an intersection of a helix (or a straight line) on simple surfaces (plane, cylinder, vertex,...)

Track Propagation in realistic B-Field

- for inhomogeneous B-field there is no analytical solution
  - ➡ start from equation of motion for a particle with charge ***q*** in magnetic field ***B***:

$$\frac{d\vec{p}}{dt} = q\vec{v} \times \vec{B}$$

  - ➡ can be written as set of differential equations for motion along ***z*** with ***x(z)*** and ***y(z)***:

$$\frac{d^2x}{dz^2} = \frac{q}{p}R\left[\frac{dx}{dz}\frac{dy}{dz}B_x - \left(1 + \left(\frac{dx}{dz}\right)^2\right)B_y + \frac{dy}{dz}B_z\right]$$
$$\frac{d^2y}{dz^2} = \frac{q}{p}R\left[\left(1 + \left(\frac{dy}{dz}\right)^2\right)B_x - \frac{dx}{dz}\frac{dy}{dz}B_y - \frac{dx}{dz}B_z\right]$$

with:

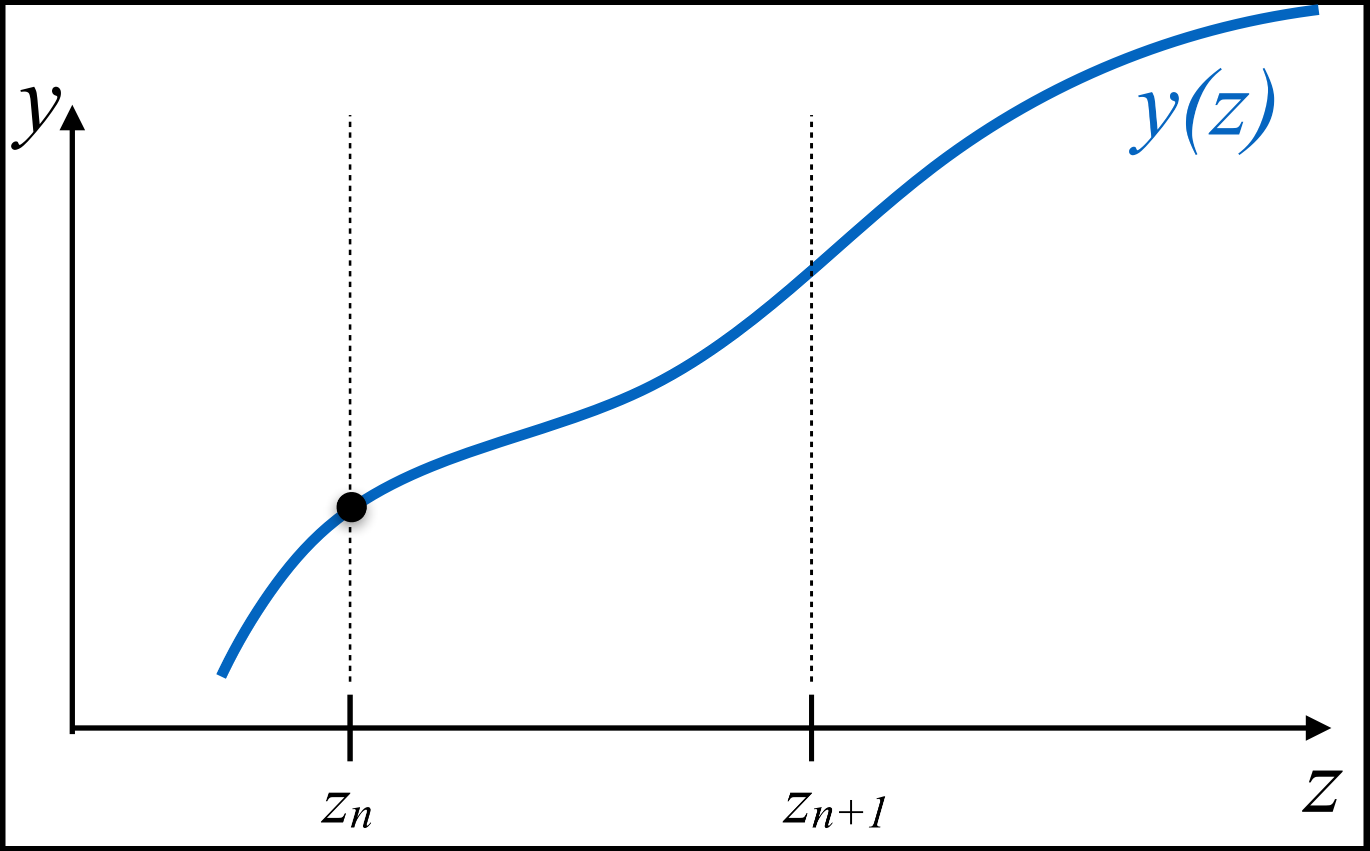
$$R = \frac{ds}{dz} = \sqrt{1 + \left(\frac{dx}{dz}\right)^2 + \left(\frac{dy}{dz}\right)^2}$$

    - no analytical solution for inhomogeneous B-field, requires numerical integration along the path of the trajectory

  - ➡ numerical integration done using Runge-Kutta technique
    - in ATLAS a 4th order **adaptive Runge-Kutta-Nystrom** approach is used, propagates covariance matrix in parallel *(Bugge, Myrheim, 1981, NIM 179, p.365)*

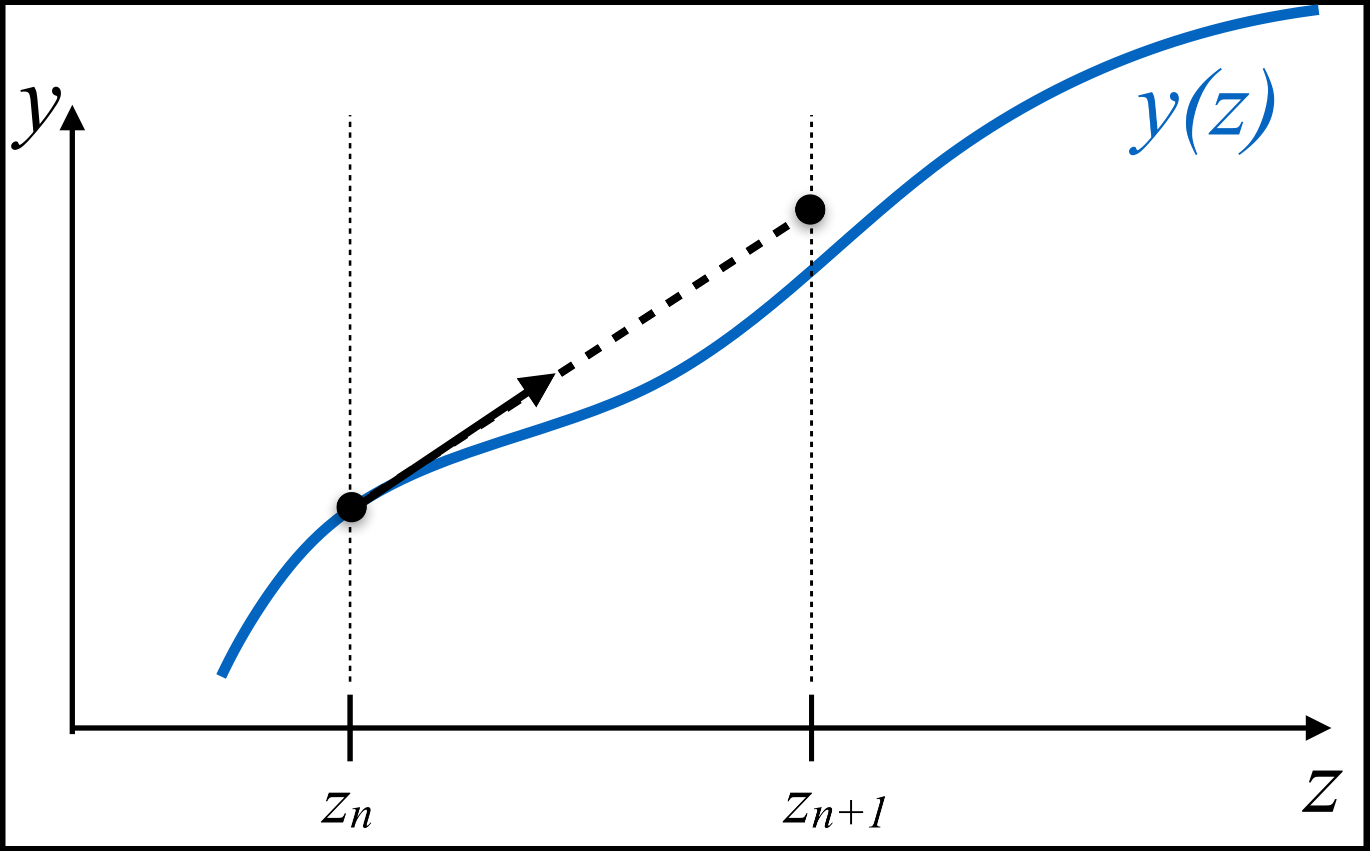
Track Propagation in realistic B-Field

- numerical integration of *y(z)* in a nutshell:
  - ➡ examples for integration methods
    - Euler's method
    - Midpoint method
    - Runge-Kutta integration



Track Propagation in realistic B-Field

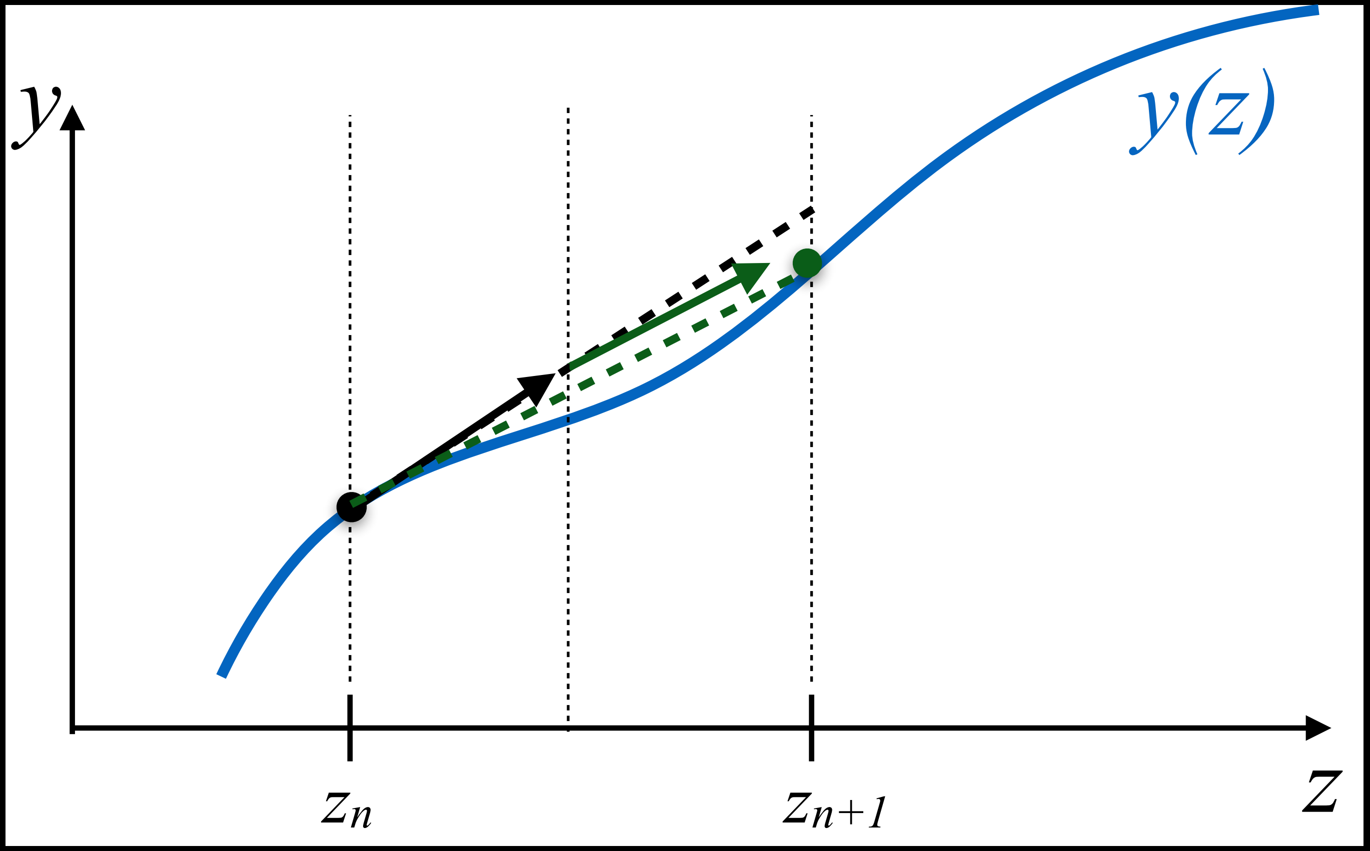
- numerical integration of $y(z)$ in a nutshell:
  - ➡ examples for integration methods
    - Euler's method
    - Midpoint method
    - Runge-Kutta integration
  - ➡ Euler's method:
    - what is the value y at $z_{n+1}=z_n+h$?
    - starting point is y_n at z_n
    - use derivative $f=\partial y/\partial z$ at z_n to approximate y_{n+1}



$$y_{n+1}=y_n+h\cdot f(z_n,y_n)$$
with
$$f(z_n,y_n)=\partial y/\partial z|_{z=z_n}$$

Track Propagation in realistic B-Field

- numerical integration of $y(z)$ in a nutshell:
  - ➡ examples for integration methods
    - Euler's method
    - Midpoint method
    - Runge-Kutta integration
  - ➡ Euler's method:
    - what is the value y at $z_{n+1}=z_n + h$?
    - starting point is y_n at z_n
    - use derivative $f=\partial y/\partial z$ at z_n to approximate y_{n+1}
  - ➡ Midpoint method:
    - evaluate f at z_n this time to stop at midpoint $z_n + h/2$ and evaluate f again



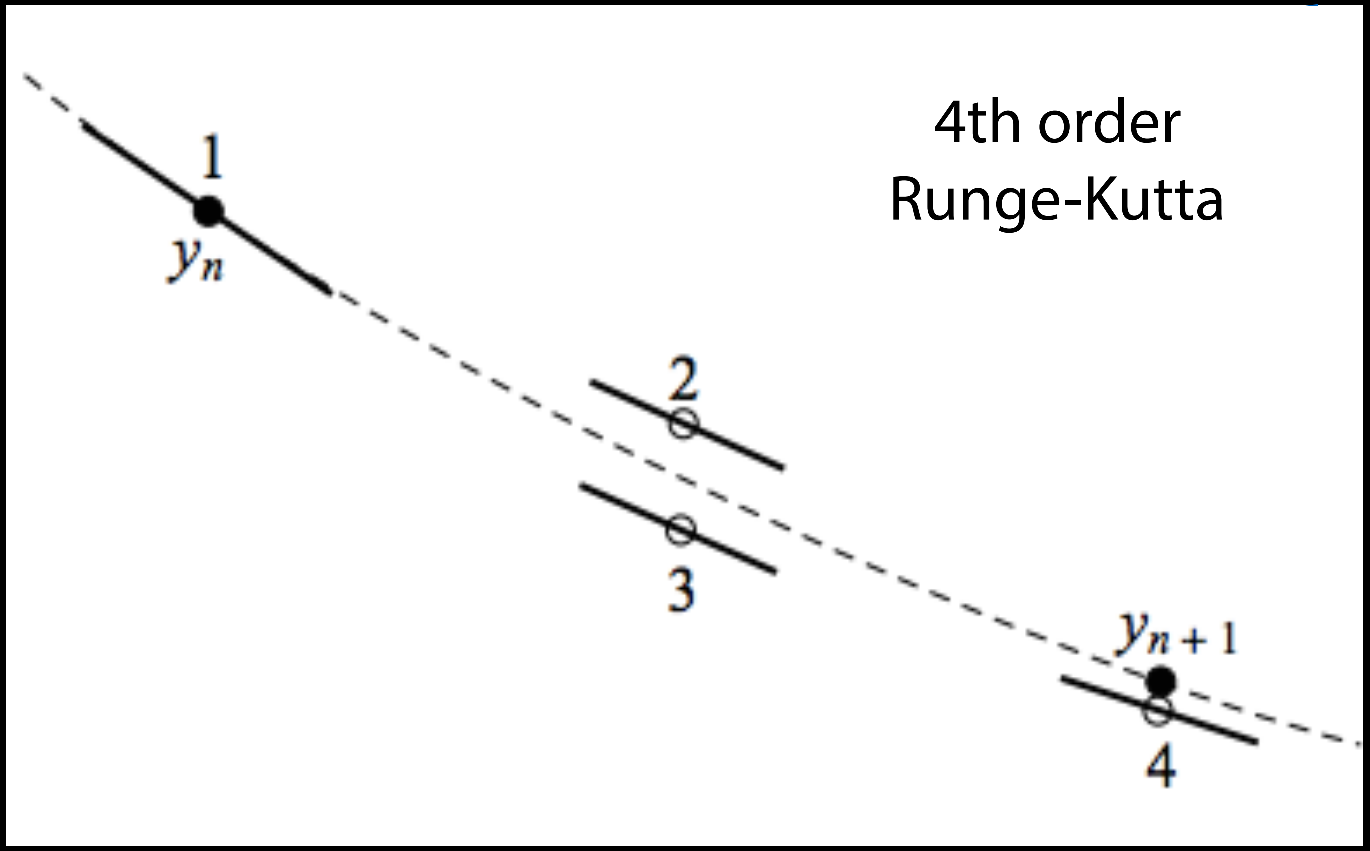
$$k_1=h\cdot f(z_n\,,\,y_n)$$

$$k_2=h\cdot f(z_n+h/2\,,\,y_n+k_1/2)$$

$$y_{n+1}=y_n + k_2 + O(h^3)$$

Track Propagation in realistic B-Field

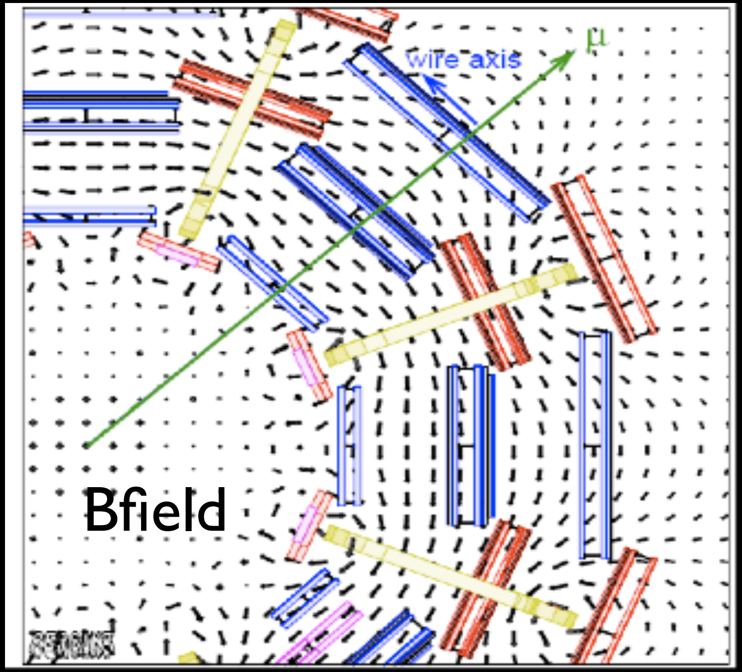
- numerical integration of $y(z)$ in a nutshell:
  - ➡ examples for integration methods
    - Euler's method
    - Midpoint method
    - Runge-Kutta integration
  - ➡ Euler's method:
    - what is the value y at $z_{n+1}=z_n+h$?
    - starting point is y_n at z_n
    - use derivative $f=\partial y/\partial z$ at z_n to approximate y_{n+1}
  - ➡ Midpoint method:
    - evaluate f at z_n this time to stop at midpoint $z_n+h/2$ and evaluate f again
  - ➡ 4th order Runge-Kutta integration:
    - evaluate f at 4 different points: at starting point, twice at midpoint and at endpoint to compute y_{n+1}



$$k_1 = hf(z_n, y_n)$$
$$k_2 = hf(z_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k_3 = hf(z_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$k_4 = hf(z_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$

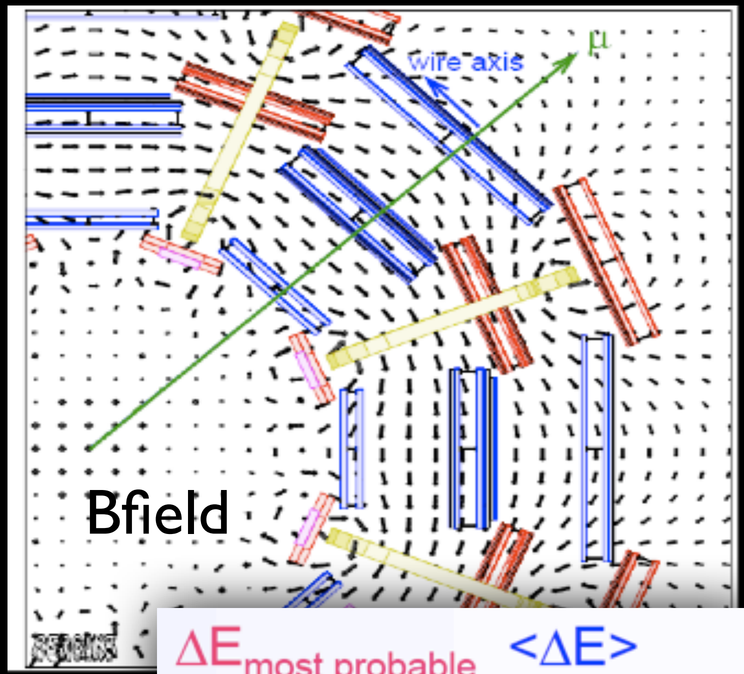
Track Propagation in realistic B-Field

- ATLAS Runge-Kutta propogator:
  - ➡ parameter propagation is 4th order
  - ➡ adaptive: use 3rd order result to monitor step precision and adapt step size (***h***)
  - ➡ monitor the remaining distance to the target surface, if a few µm, use Taylor approximation to reach surface
  - ➡ Nystrom technique: does as well numerical integration of Jacobian for error propagation (fast & precise)

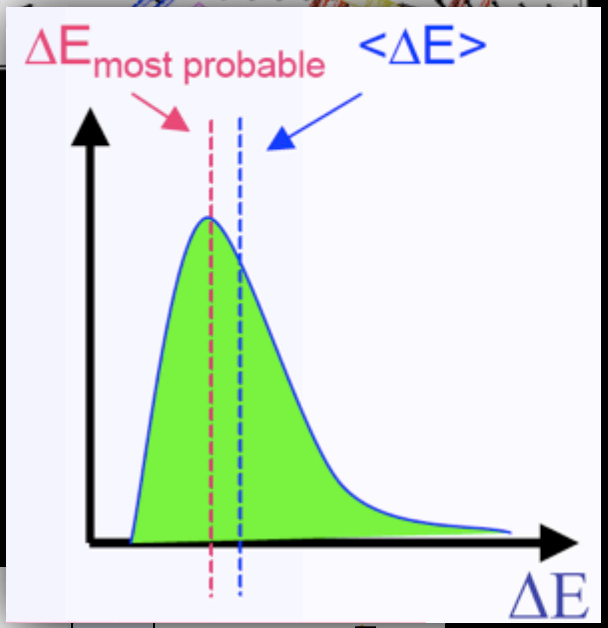


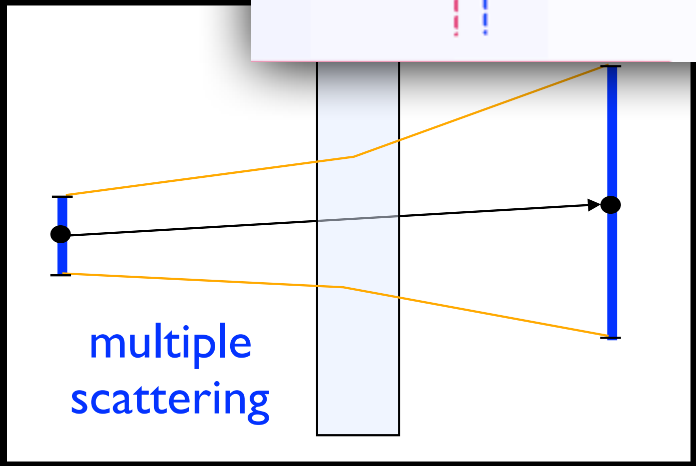
Track Propagation in realistic B-Field

- ATLAS Runge-Kutta propogator:
  - ➡ parameter propagation is 4th order
  - ➡ adaptive: use 3rd order result to monitor step precision and adapt step size (***h***)
  - ➡ monitor the remaining distance to the target surface, if a few µm, use Taylor approximation to reach surface
  - ➡ Nystrom technique: does as well numerical integration of Jacobian for error propagation (fast & precise)



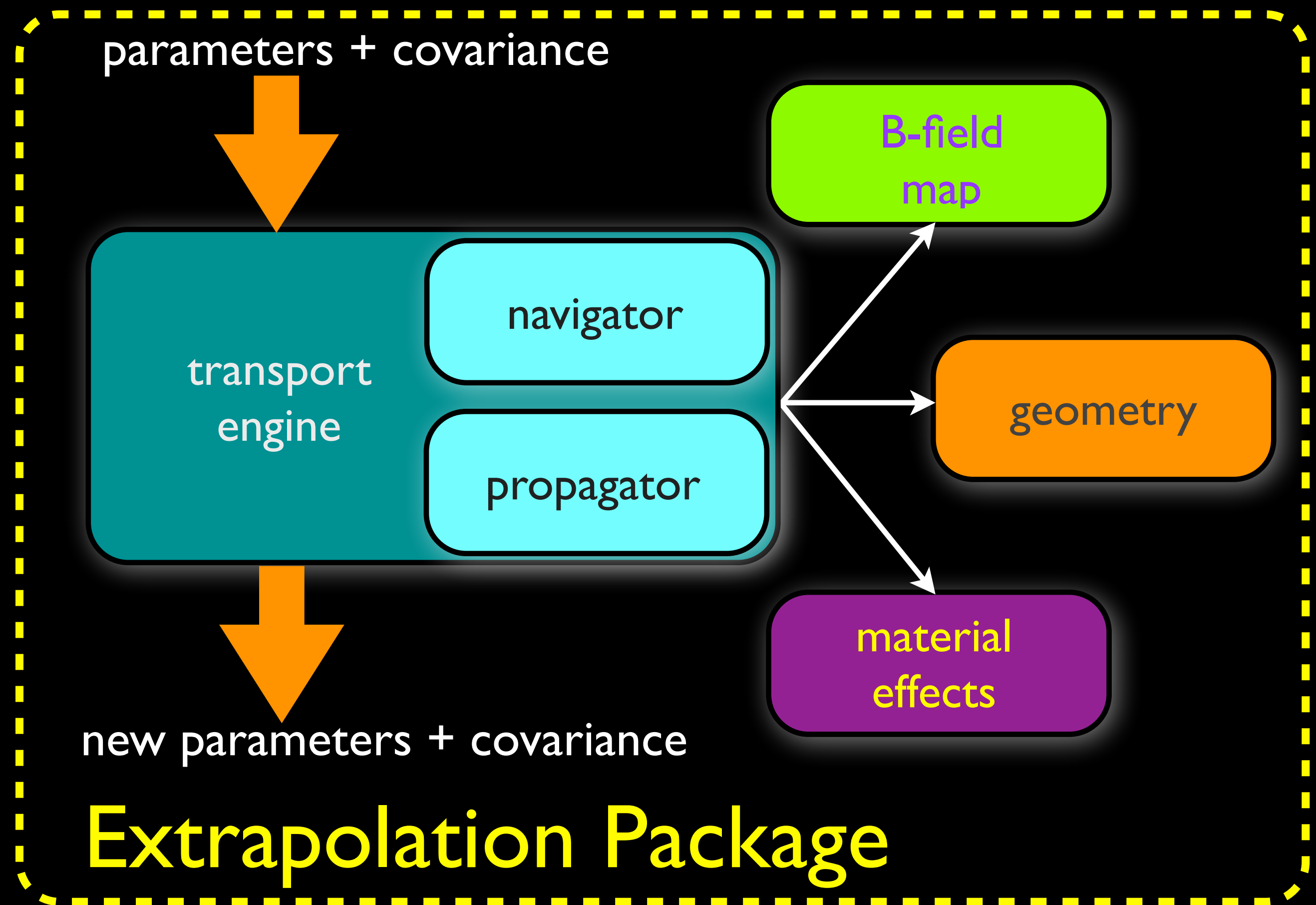
- need to allow for material effects
  - ➡ energy loss
    - use most **probably energy loss** for x/X_0
    - correct momentum (curvature) and its covariance
  - ➡ multiple scattering
    - increases **uncertainty on direction** of track
    - for given x/X_0 traversed add term to covariances of θ and ϕ on a material "layer"





The Track Extrapolation Package

- a **transport engine** used in tracking software
  - ➡ central tool for pattern recognition, track fitting, etc.
  - ➡ parameter transport from **surface to surface**, including covariance
  - ➡ encapsulates the track model, geometry and material corrections

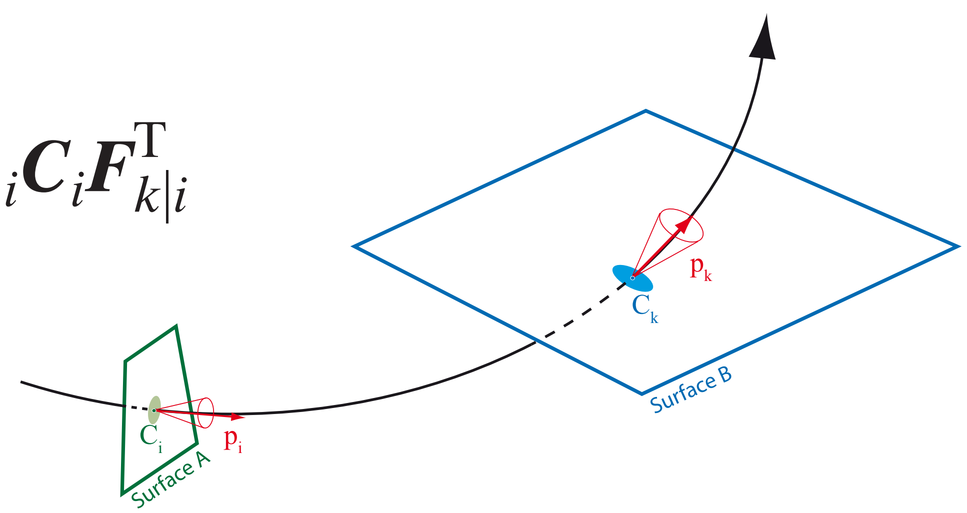


track following in mathematical terms:

$$q_k = f_{k|i}(q_i)$$ convariance: $$C_k = F_{k|i} C_i F_{k|i}^{\mathrm{T}}$$

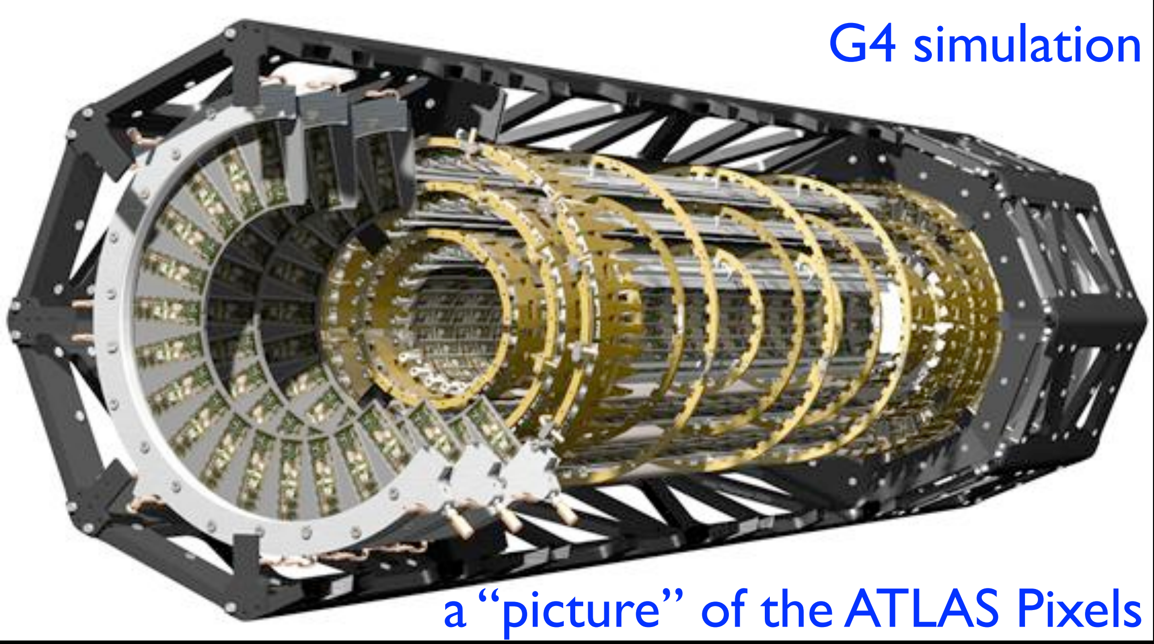
with: $f_{k|i}$ ~ track model

$F_{k|i} = \dfrac{\partial q_k}{\partial q_i}$ ~ Jacobi matrix



Detector Geometry

- interactions in detector material limiting tracking performance
  - ➡ LHC detectors are complex
    - require a very detailed description of their geometry
  - ➡ experiments developed geometry models (translation into G4 simulation)
    - huge number of volumes
- physics requirement to reach LHC goals (e.g. W mass)
  - ➡ control material close to beam pipe at % level



	model	placed volumes
ALICE	Root	4.3 M
ATLAS	GeoModel	4.8 M
CMS	DDD	2.7 M
LHCb	LHCb Det.Des.	18.5 M

Weighing Detectors during Construction

- huge effort in experiments
  - ➡ important to reach good description in simulation and reconstruction
  - ➡ each individual detector part was put on balance and compare with model
    - CMS and ATLAS measured weight of their tracker and all of its components
  - ➡ correct the geometry implementation in simulation and reconstruction

example: ATLAS TRT measured before and after insertion of the SCT

CMS	estimated from measurements	simulation
active Pixels	2598 g	2455 g
full detector	6350 kg	6173 kg

ATLAS	estimated from measurements	simulation
Pixel package	201 kg	197 kg
SCT detector	672 ±15 kg	672 kg
TRT detector	2961 ±14 kg	2962 kg

Preliminary

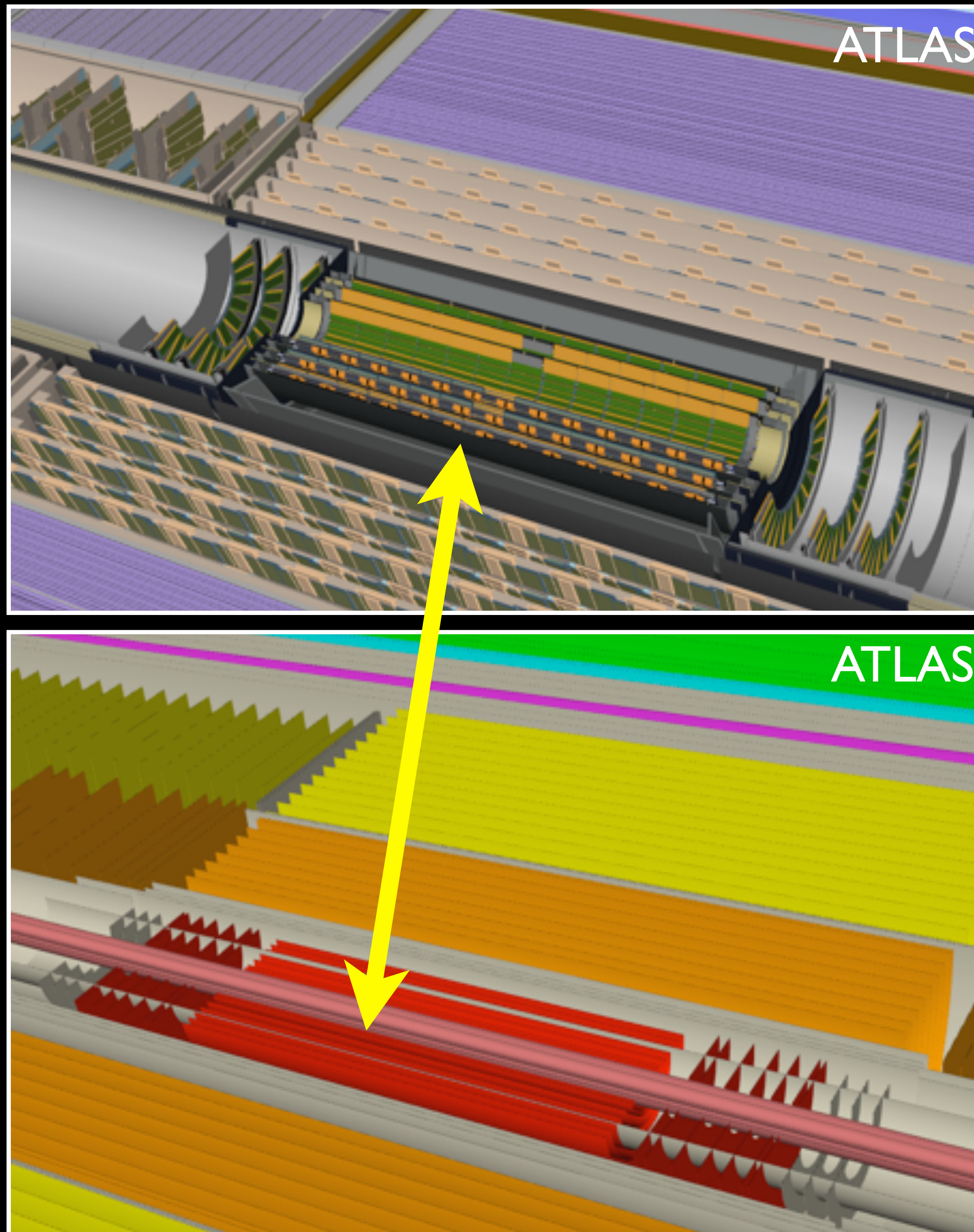
Date	ATLAS $\eta \approx 0$	ATLAS $\eta \approx 1.7$	CMS $\eta \approx 0$	CMS $\eta \approx 1.7$
1994 (Technical Proposals)	0.20	0.70	0.15	0.60
1997 (Technical Design Reports)	0.25	1.50	0.25	0.85
2006 (End of construction)	0.35	1.35	0.35	1.50

Full and Fast (Tracking) Geometries

- complex G4 geometries not optimal for reconstruction
  - ➡ simplified **tracking geometries**
  - ➡ material surfaces, field volumes
- reduced number of volumes
  - ➡ blending details of material onto simple surfaces/volumes
  - ➡ surfaces with 2D material density maps, templates per Si sensor...

	G4	tracking
ALICE	4.3 M	same *1
ATLAS	4.8 M	10.2K *2
CMS	2.7 M	3.8K *2
LHCb	18.5 M	30

*1 ALICE uses full geometry (TGeo)
*2 plus a surface per Si sensor



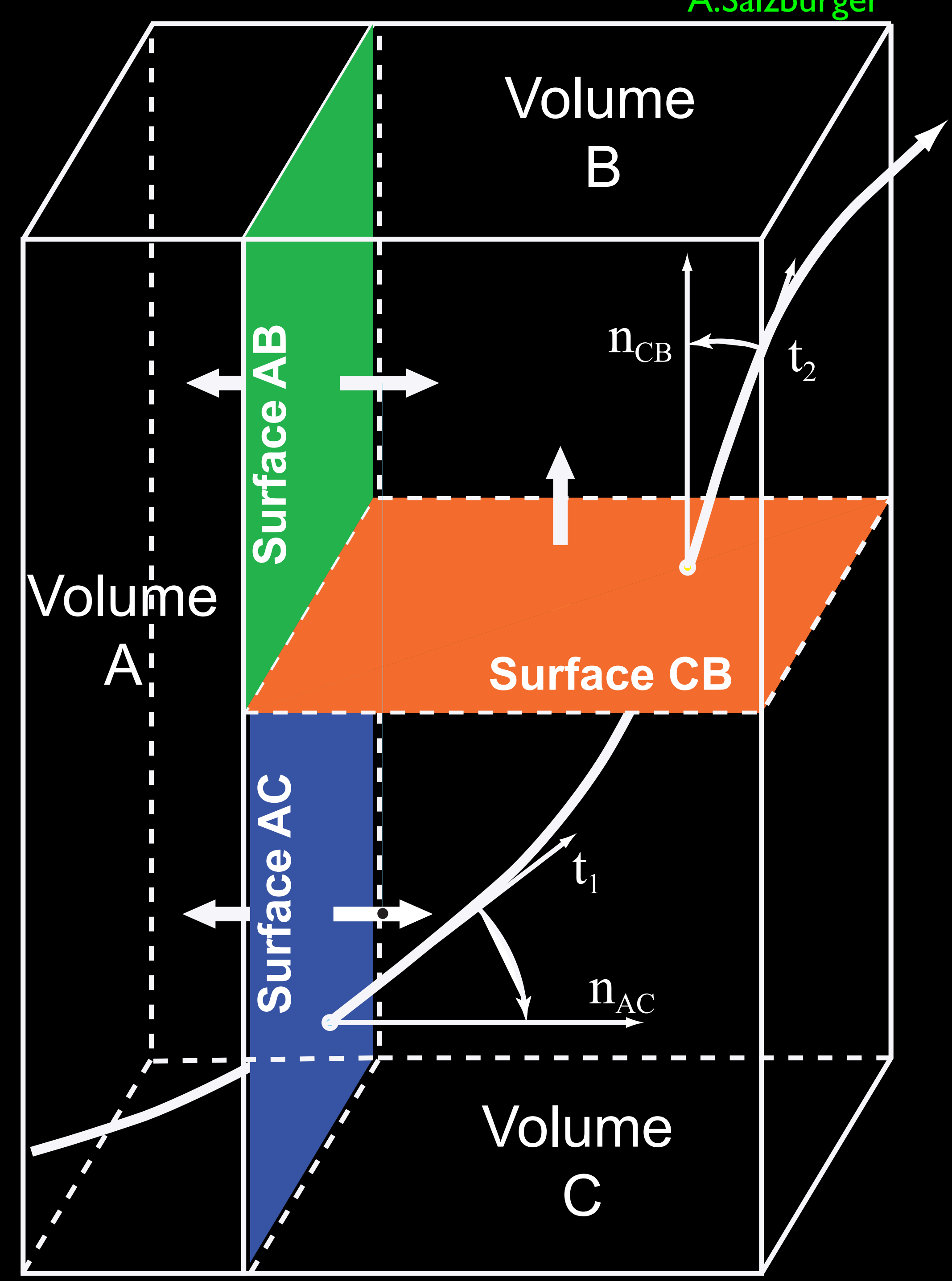
Embedded Navigation Schemes

- **embedded navigation** scheme in tracking geometries
  - ➡ G4 navigation uses voxelisation as generic navigation mechanism
  - ➡ **embedded navigation** for simplified models
    - used in pattern recognition, extrapolation, track fitting and fast simulation

- **example**: ATLAS
  - ➡ developed geometry of connected volumes
  - ➡ boundary surfaces connect neighbouring volumes to predict next step

ATLAS	G4	tracking	ratio
crossed volumes in tracker	474	95	5
time in SI2K sec	19.1	2.3	8.4

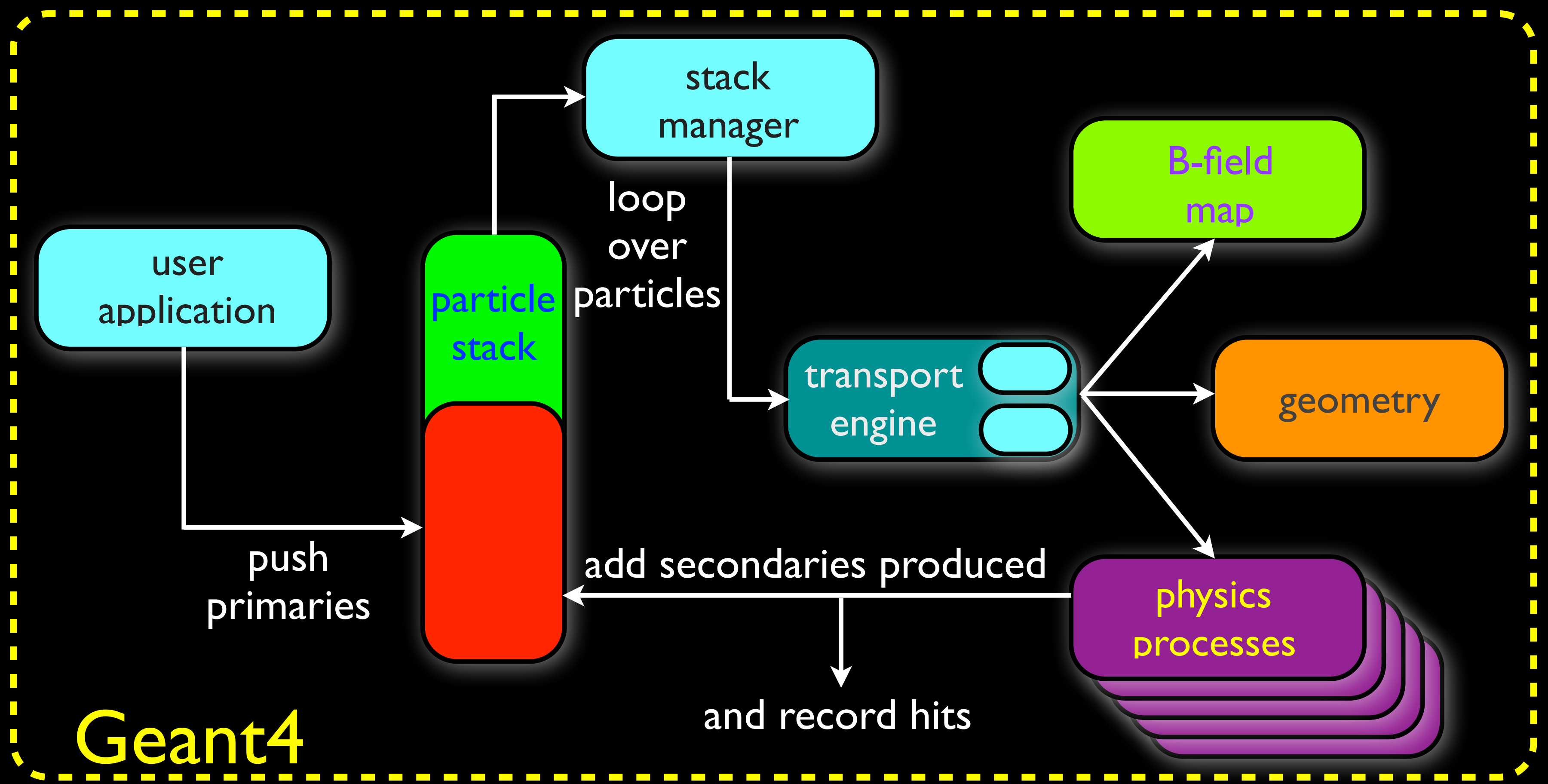
(neutral geantinos, no field lookups)



Detour: Simulation (Geant4)

- Geant4 is based upon
  - ➡ **stack** to keep track of all particles produced and stack manager
  - ➡ **extrapolation system** to propagate each particle:
    - transport engine with navigation
    - geometry model
    - B-field
  - ➡ set of **physics processes** describing interaction of particles with matter
  - ➡ a user application interface, ...

same concept as for track reconstruction



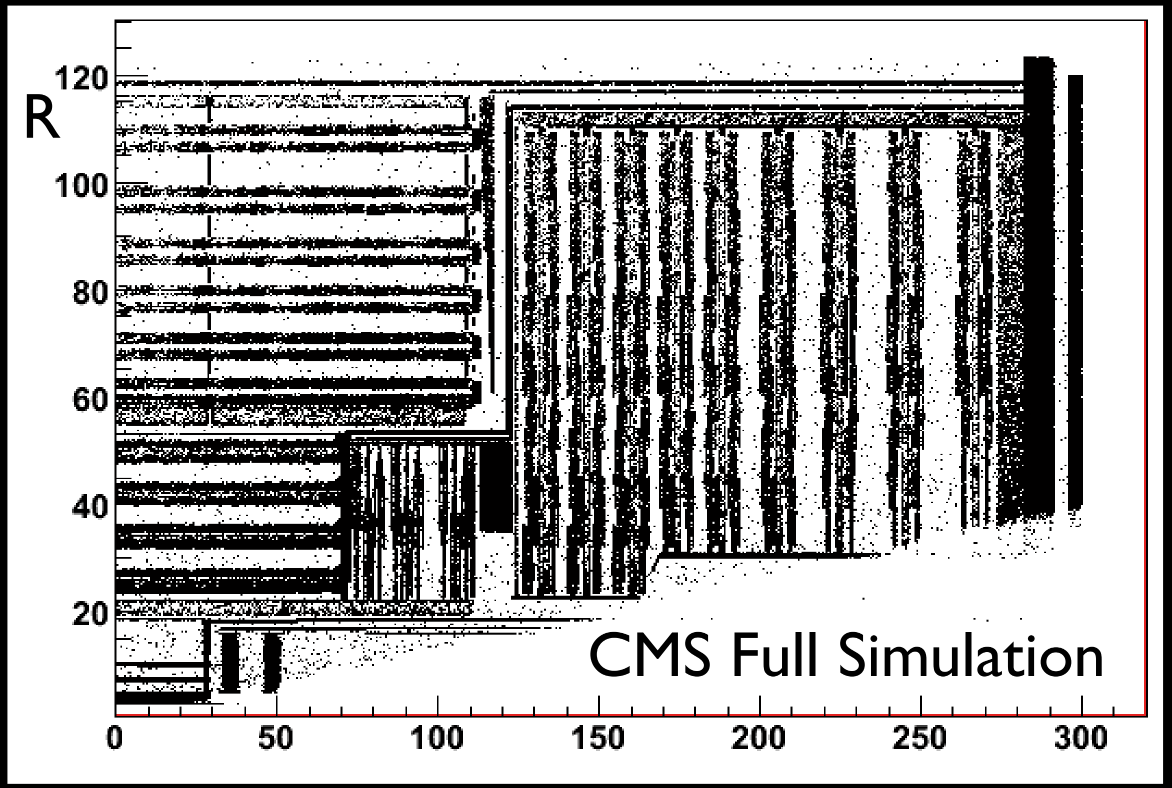
Fast Simulation

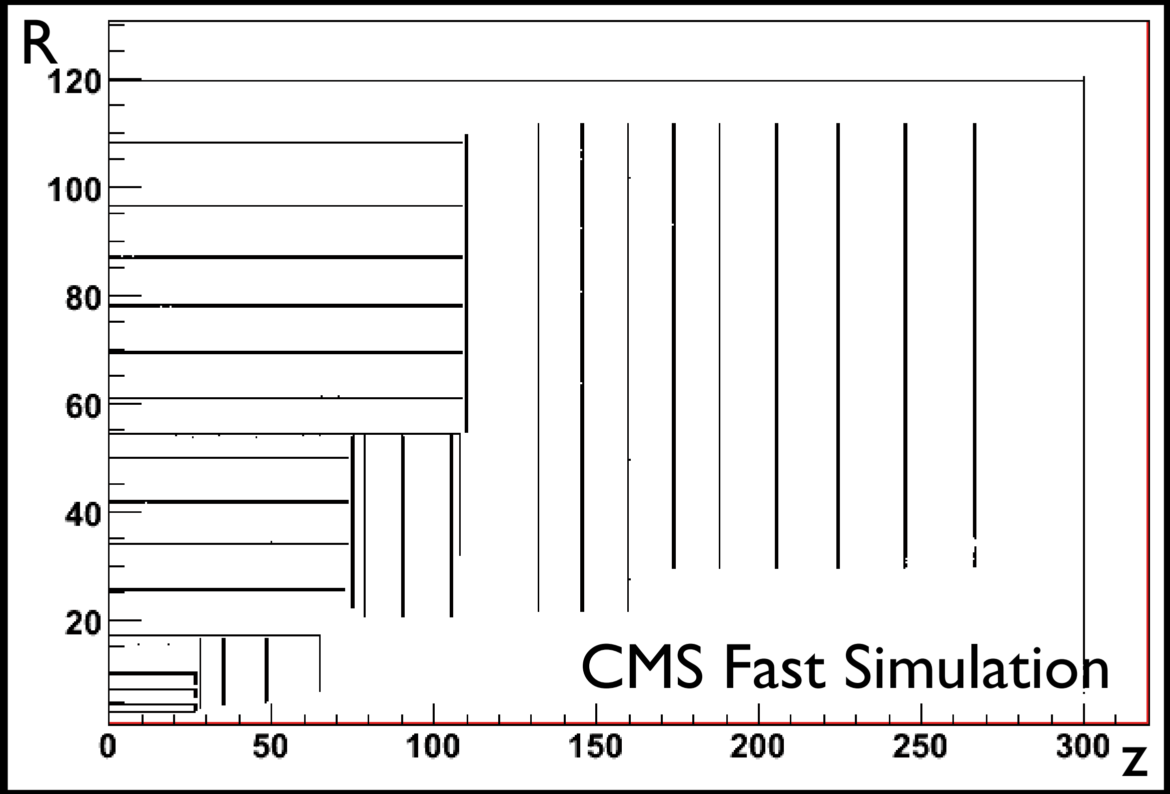
- CPU needs for full G4 exceeds computing models
  - ➡ simulation strategies of experiments mix full G4 and fast simulation

	G4	fast sim.
CMS	360	0.8
ATLAS	1990	7.4

ttbar events, in kSI2K sec
G4 differences: calo.modeling , phys.list, η cuts, b-field

- fast simulation engines
  - ➡ fast calo. simulation (parameterisation, showers libraries, ...)
  - ➡ simplified tracking geometries
  - ➡ simplify physics processes w.r.t. G4
  - ➡ output in same data model as full sim.
  - ➡ able to run full reconstruction (trigger)





Track Fitting

From Measurement Model to Track Fitting

- measurements m_k of a track
  - ➡ in mathematical terms a model:

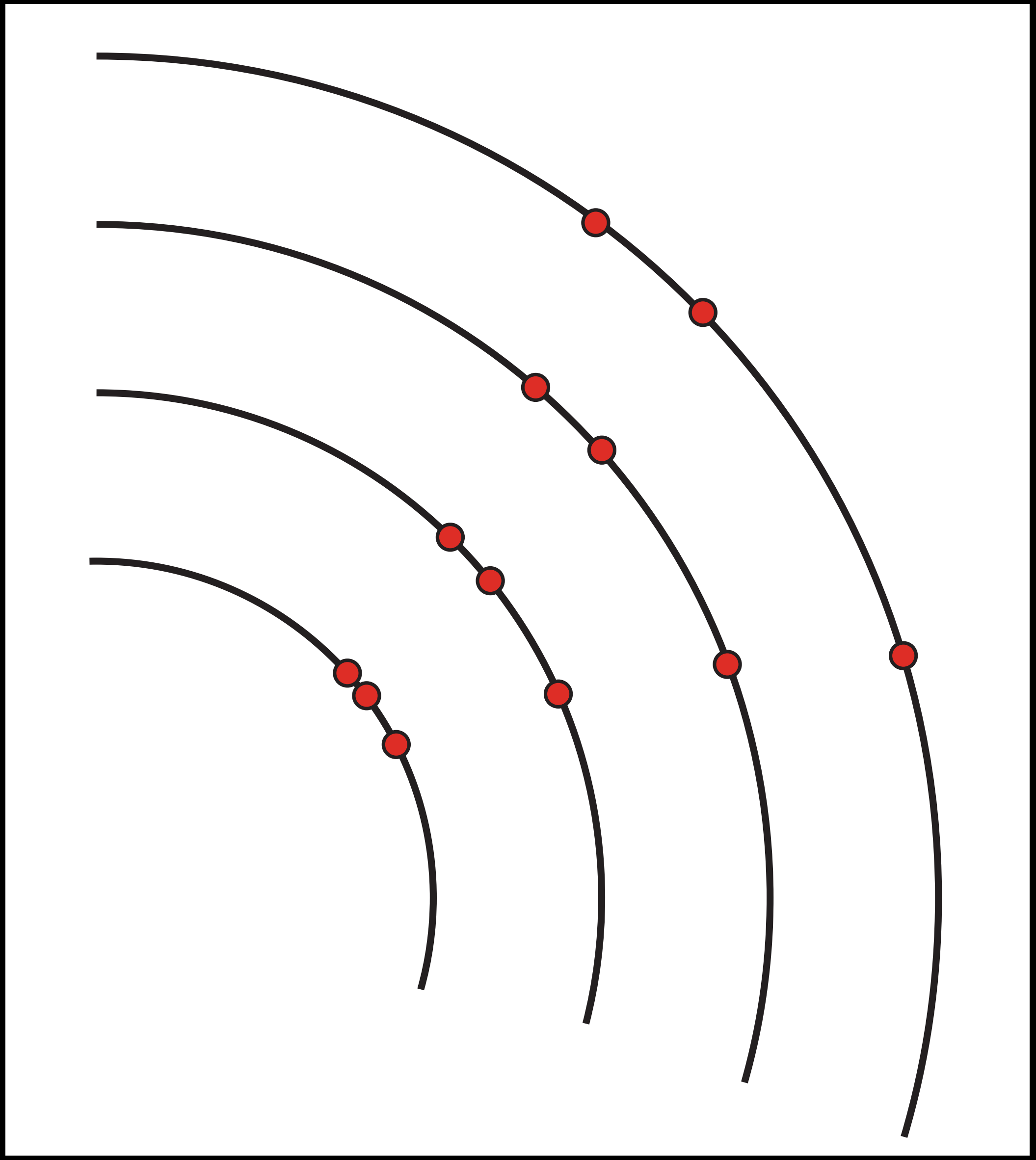
$$\boldsymbol{m}_k = \boldsymbol{h}_k(\boldsymbol{q}_k) + \boldsymbol{\gamma}_k$$

with: $\boldsymbol{h}_k$ ~ functional dependency of measurement on e.g. track angle

$\boldsymbol{\gamma}_k$ ~ error (noise term)

$$\boldsymbol{H}_k = \frac{\partial \boldsymbol{m}_k}{\partial \boldsymbol{q}_k}$$ ~ Jacobian, often contains only rotations and projections

  - ➡ in practice those m_k are clusters, drift circles, ...



From Measurement Model to Track Fitting

- measurements m_k of a track
  - ➡ in mathematical terms a model:

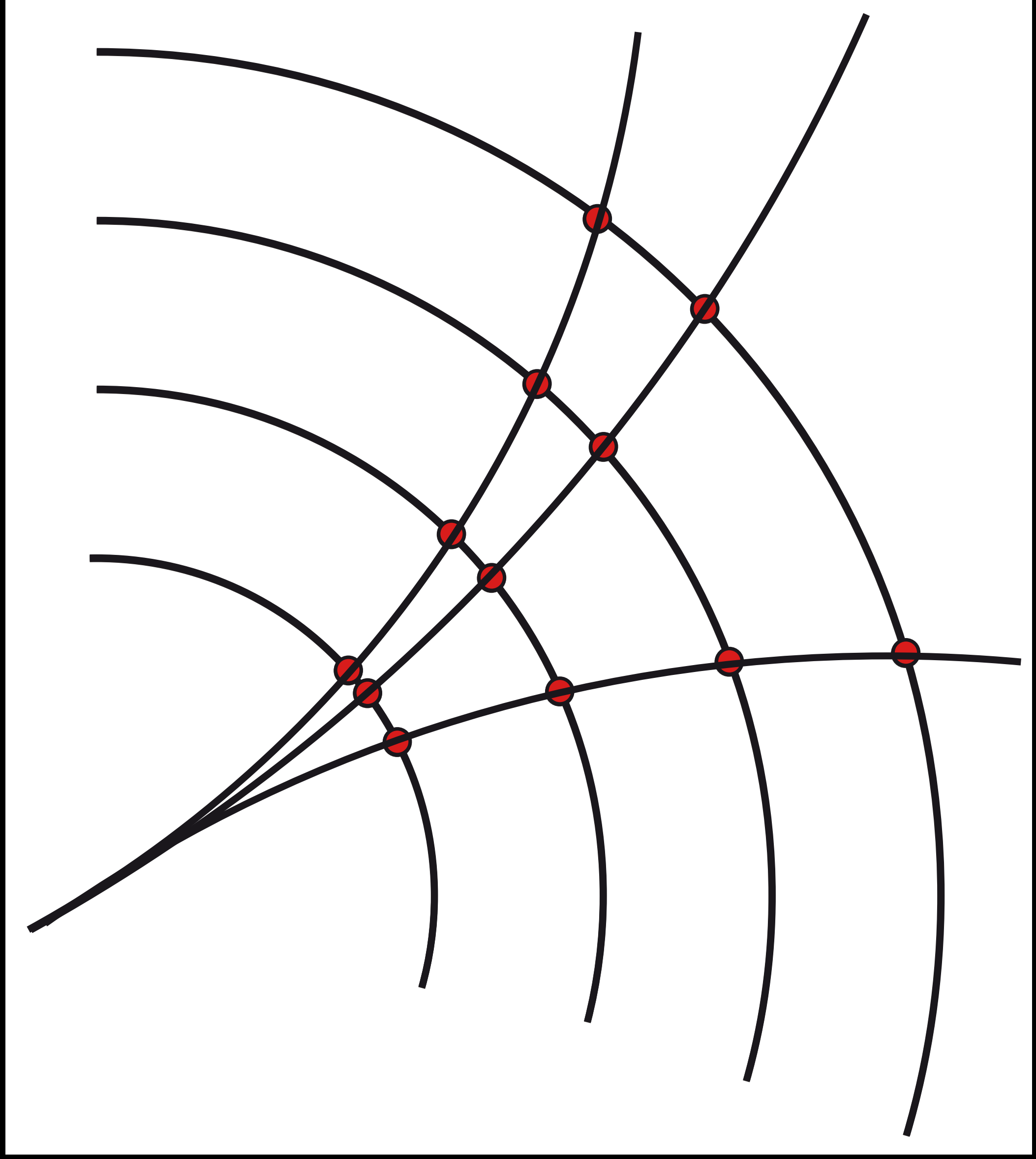
$$m_k = h_k(q_k) + \gamma_k$$

with: h_k ~ functional dependency of measurement on e.g. track angle

γ_k ~ error (noise term)

$$H_k = \frac{\partial m_k}{\partial q_k}$$ ~ Jacobian, often contains only rotations and projections

  - ➡ in practice those m_k are clusters, drift circles, ...



- task of a track fit
  - ➡ estimate the track parameters from a set measurements
- examples for fitting techniques
  - ➡ Least Square track fit or Kalman Filter track fit
  - ➡ more specialised versions: Gaussian Sum Filter or Deterministic Annealing Filters

Classical Least Square Track Fit

- construct and minimise the χ^2 function:

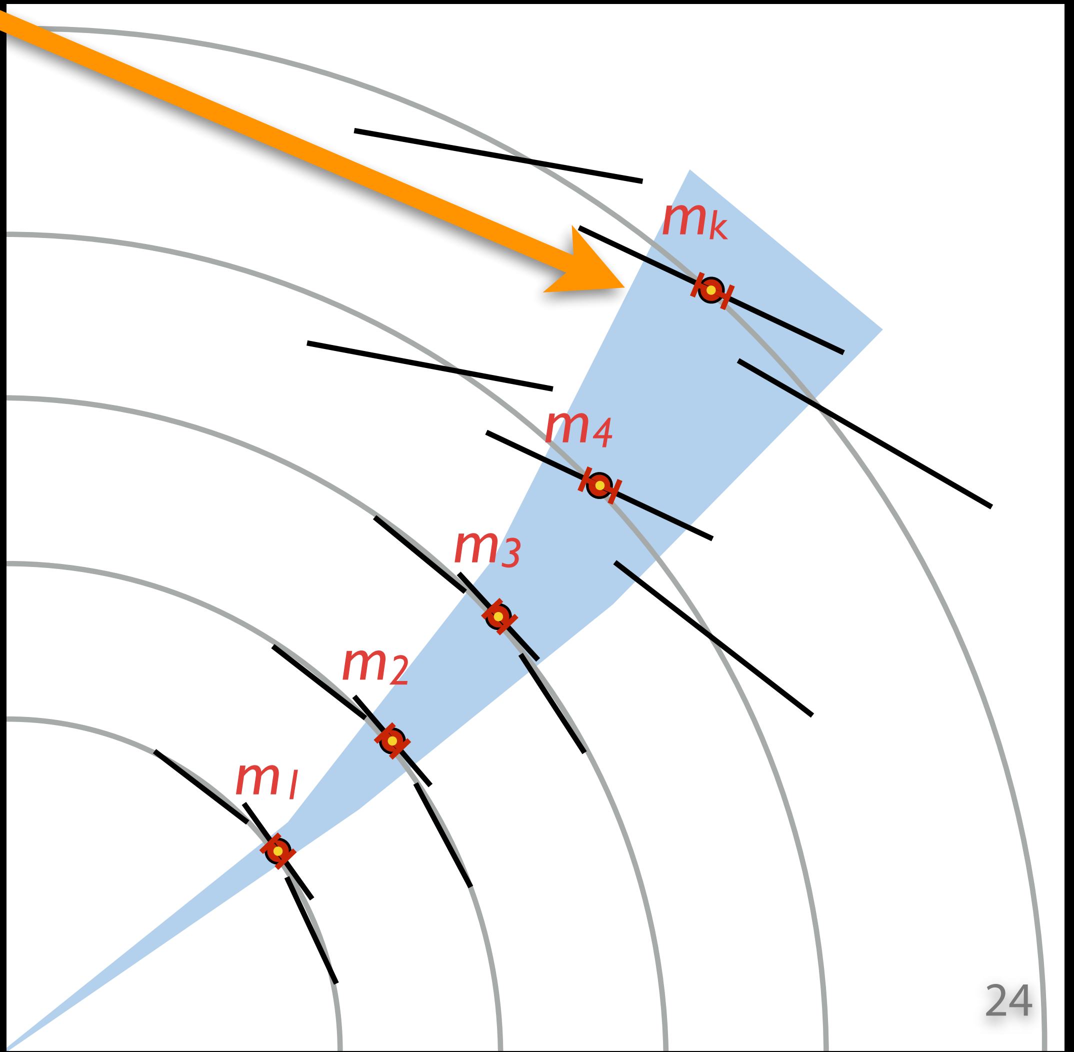
Carl Friedrich Gauss is credited with developing the fundamentals of the basis for least-squares analysis in 1795 at the age of eighteen. **Legendre** was the first to publish the method, however.

➡Write down Least Square function:

$$\chi^2 = \sum_k \Delta m_k^T G_K^{-1} \Delta m_k \quad \text{with:} \quad \Delta m_k = m_k - d_k(p)$$

d_k contains measurement model and propagation of the parameters p : $\boldsymbol{d}_k = \boldsymbol{h}_k \circ \boldsymbol{f}_{k|k-1} \circ \cdots \circ \boldsymbol{f}_{2|1} \circ \boldsymbol{f}_{1|0}$

G_k is the covariance matrix of m_k .



Classical Least Square Track Fit

- construct and minimise the χ^2 function:

Carl Friedrich Gauss is credited with developing the fundamentals of the basis for least-squares analysis in 1795 at the age of eighteen. **Legendre** was the first to publish the method, however.

➡Write down Least Square function:

$$\chi^2 = \sum_k \Delta m_k^T G_K^{-1} \Delta m_k \quad \text{with:} \quad \Delta m_k = m_k - d_k(p)$$

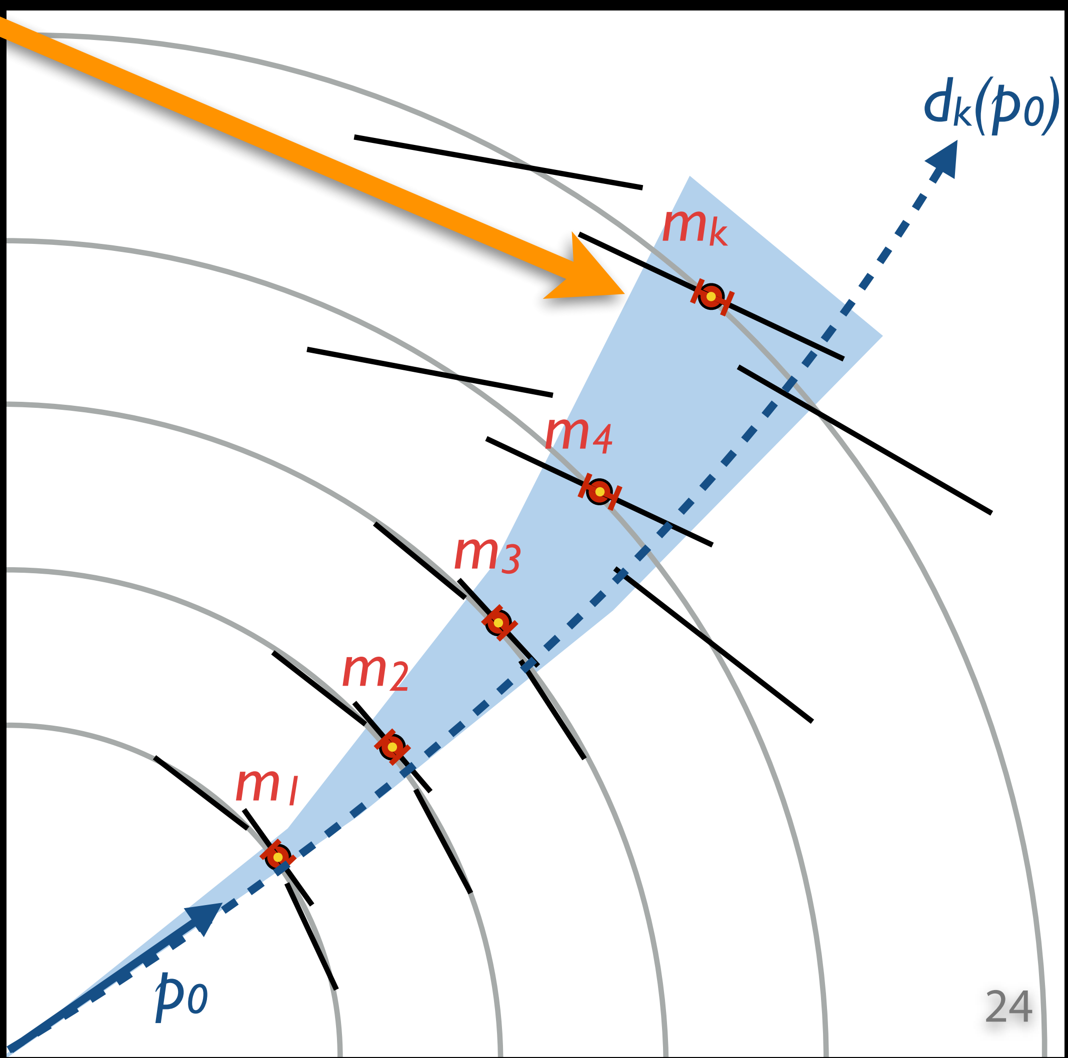
d_k contains measurement model and propagation of the parameters p : $\boldsymbol{d}_k = \boldsymbol{h}_k \circ \boldsymbol{f}_{k|k-1} \circ \cdots \circ \boldsymbol{f}_{2|1} \circ \boldsymbol{f}_{1|0}$

G_k is the covariance matrix of m_k .

➡Linearise the χ^2 with a Taylor expansion:

$$d_k(p_0 + \delta p) \cong d_k(p_0) + D_k \cdot \delta p \text{ + higher terms}$$

with Jacobian: $\boldsymbol{D}_k = \boldsymbol{H}_k \boldsymbol{F}_{k|k-1} \cdots \boldsymbol{F}_{2|1} \boldsymbol{F}_{1|0}$



Classical Least Square Track Fit

- construct and minimise the χ^2 function:

Carl Friedrich Gauss is credited with developing the fundamentals of the basis for least-squares analysis in 1795 at the age of eighteen. **Legendre** was the first to publish the method, however.

➡Write down Least Square function:

$$\chi^2 = \sum_k \Delta m_k^T G_K^{-1} \Delta m_k \quad \text{with:} \quad \Delta m_k = m_k - d_k(p)$$

d_k contains measurement model and propagation of the parameters p : $\quad \boldsymbol{d}_k = \boldsymbol{h}_k \circ \boldsymbol{f}_{k|k-1} \circ \cdots \circ \boldsymbol{f}_{2|1} \circ \boldsymbol{f}_{1|0}$

G_k is the covariance matrix of m_k .

➡Linearise the χ^2 with a Taylor expansion:

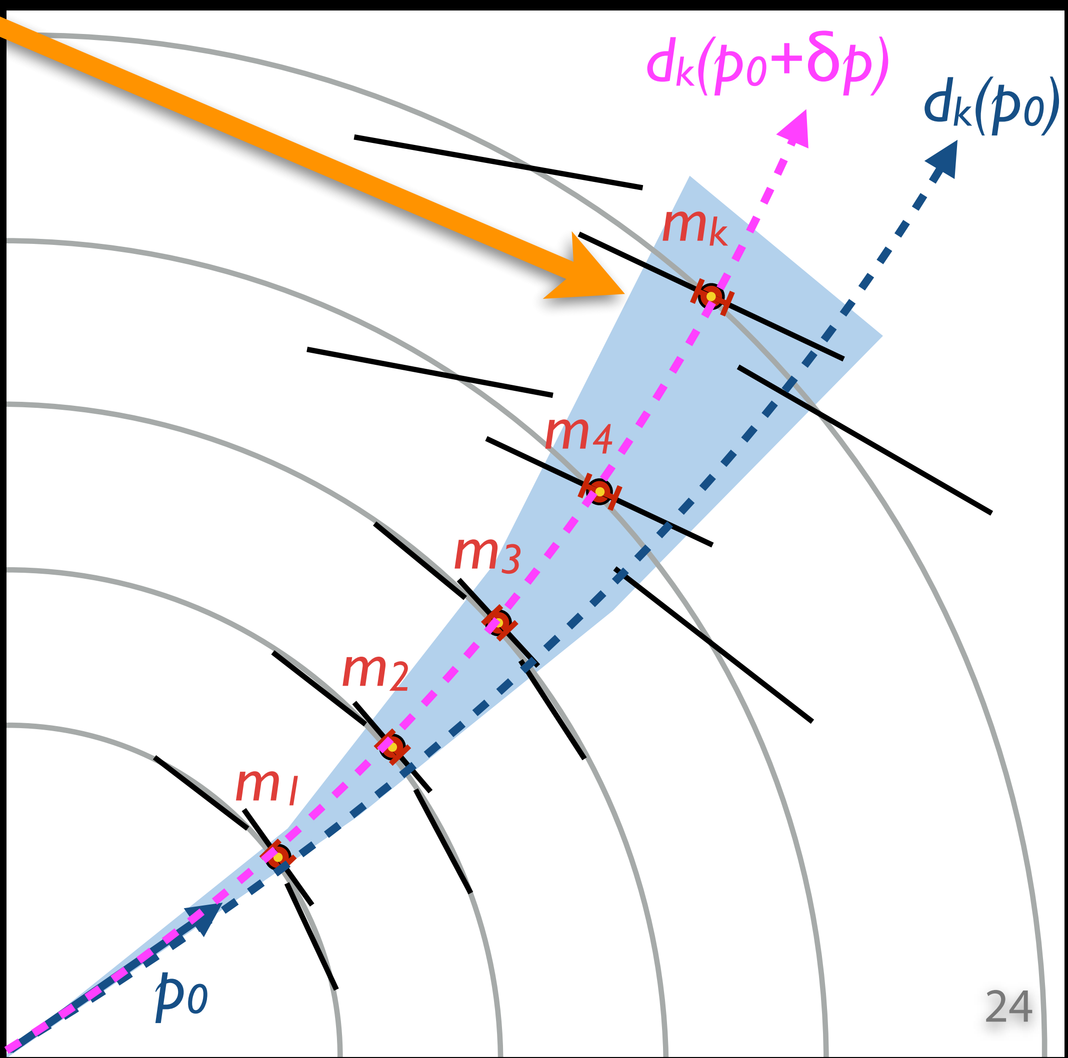
$$d_k(p_0 + \delta p) \cong d_k(p_0) + D_k \cdot \delta p \; + \text{higher terms}$$

with Jacobian: $\quad \boldsymbol{D}_k = \boldsymbol{H}_k \boldsymbol{F}_{k|k-1} \cdots \boldsymbol{F}_{2|1} \boldsymbol{F}_{1|0}$

➡Minimising linearised χ^2 yields system of linear equations:

$$\frac{\partial \chi^2}{\partial p} = 0 \Rightarrow \delta p = \left(\sum_k D_k^T G_k^{-1} D_k \right)^{-1} \sum_k D_k^T G_k^{-1} (m_k - d_k(p_0))$$

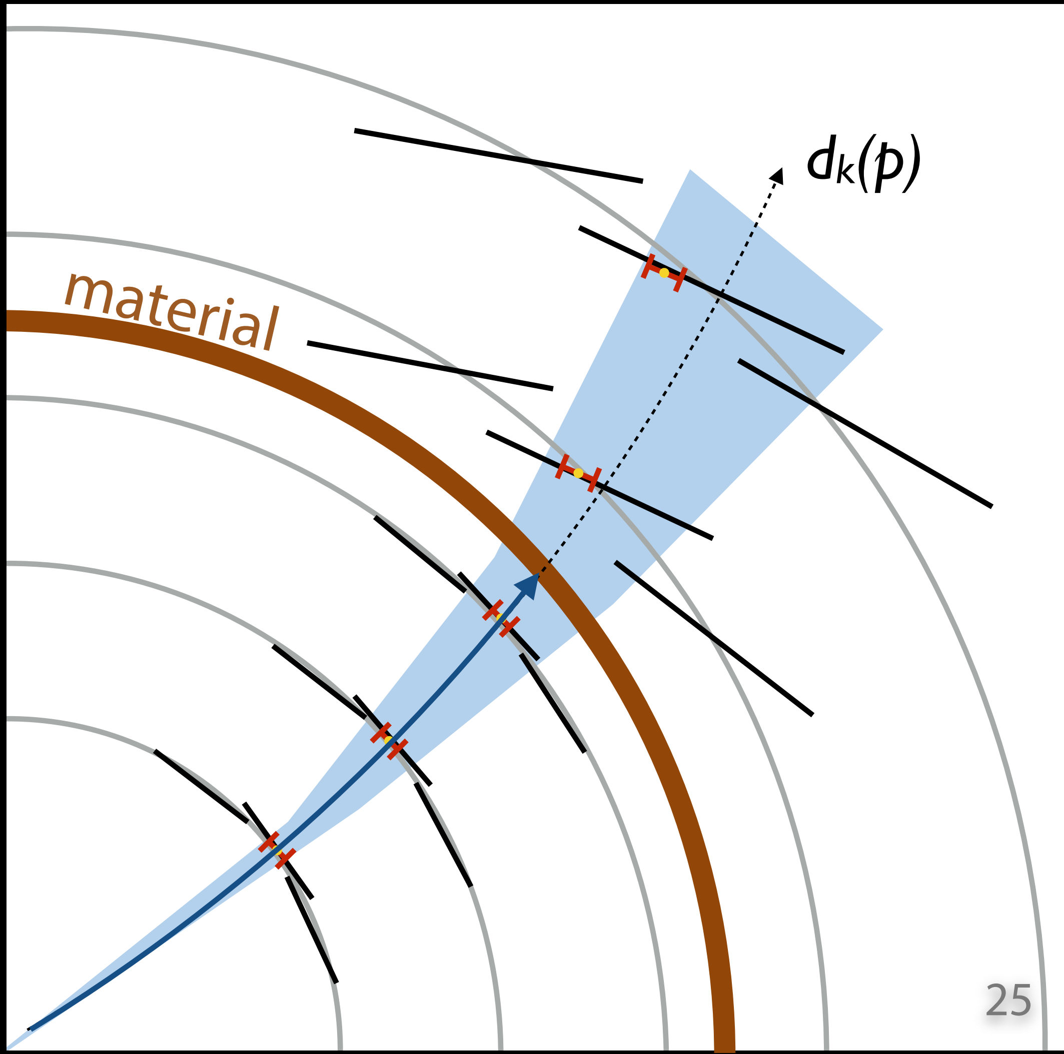
and covariance of δp is: $C = \left(\sum_k D_k^T G_k^{-1} D_k \right)^{-1}$



Classical Least Square Track Fit

similar to Broken Lines fit

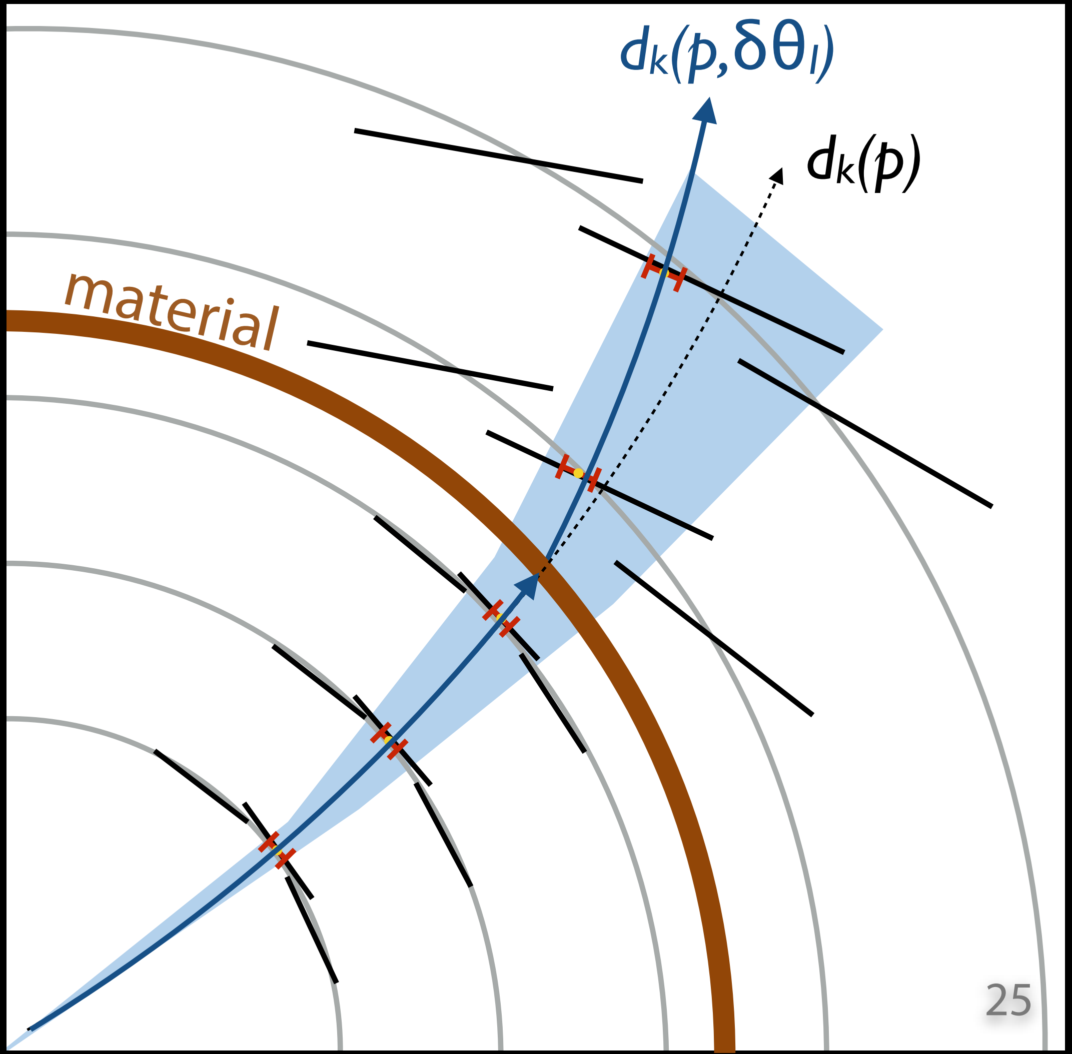
- allowing for material effects in fit:
  - ➡ can be absorbed in track model $\mathbf{f}_{k|i}$, provided effects are small
  - ➡ for substantial multiple scatting, allows for **scattering angles** in the fit



Classical Least Square Track Fit

similar to Broken Lines fit

- allowing for material effects in fit:
  - ➡ can be absorbed in track model $f_{k|i}$, provided effects are small
  - ➡ for substantial multiple scatting, allows for **scattering angles** in the fit
- introduce scattering angles on material surfaces
  - ➡ on each material surface, add 2 angles $\delta\theta_i$ as fee parameters to the fit
  - ➡ expected mean of those angles is 0 (!), their covariance Q_i is given by multiple scattering in x/X_0



Classical Least Square Track Fit

similar to Broken Lines fit

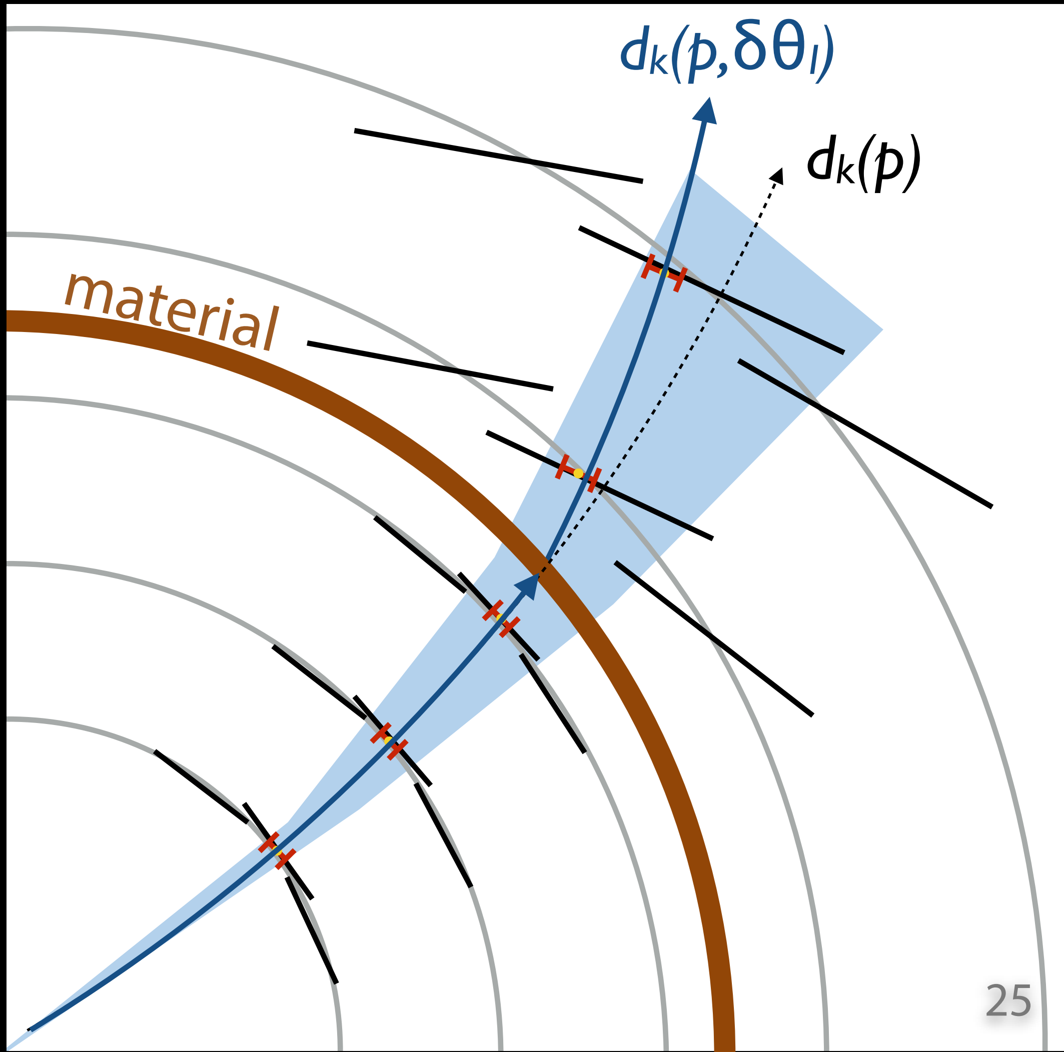
- allowing for material effects in fit:
  - ➡ can be absorbed in track model $f_{k|i}$, provided effects are small
  - ➡ for substantial multiple scatting, allows for **scattering angles** in the fit
- introduce scattering angles on material surfaces
  - ➡ on each material surface, add 2 angles $\delta\theta_i$ as fee parameters to the fit
  - ➡ expected mean of those angles is 0 (!), their covariance Q_i is given by multiple scattering in x/X_0
- results in additional term in χ^2 equations:

$$\chi^2 = \sum_k \Delta m_k^T G_K^{-1} \Delta m_k + \sum_i \delta\theta_i^T Q_i^{-1} \delta\theta_i$$

with: $\Delta m_k = m_k - d_k\left(p, \delta\theta_i\right)$

- ➡ computationally expensive

  *(invert a dimension 5+2*n matrix)*



similar to Broken Lines fit

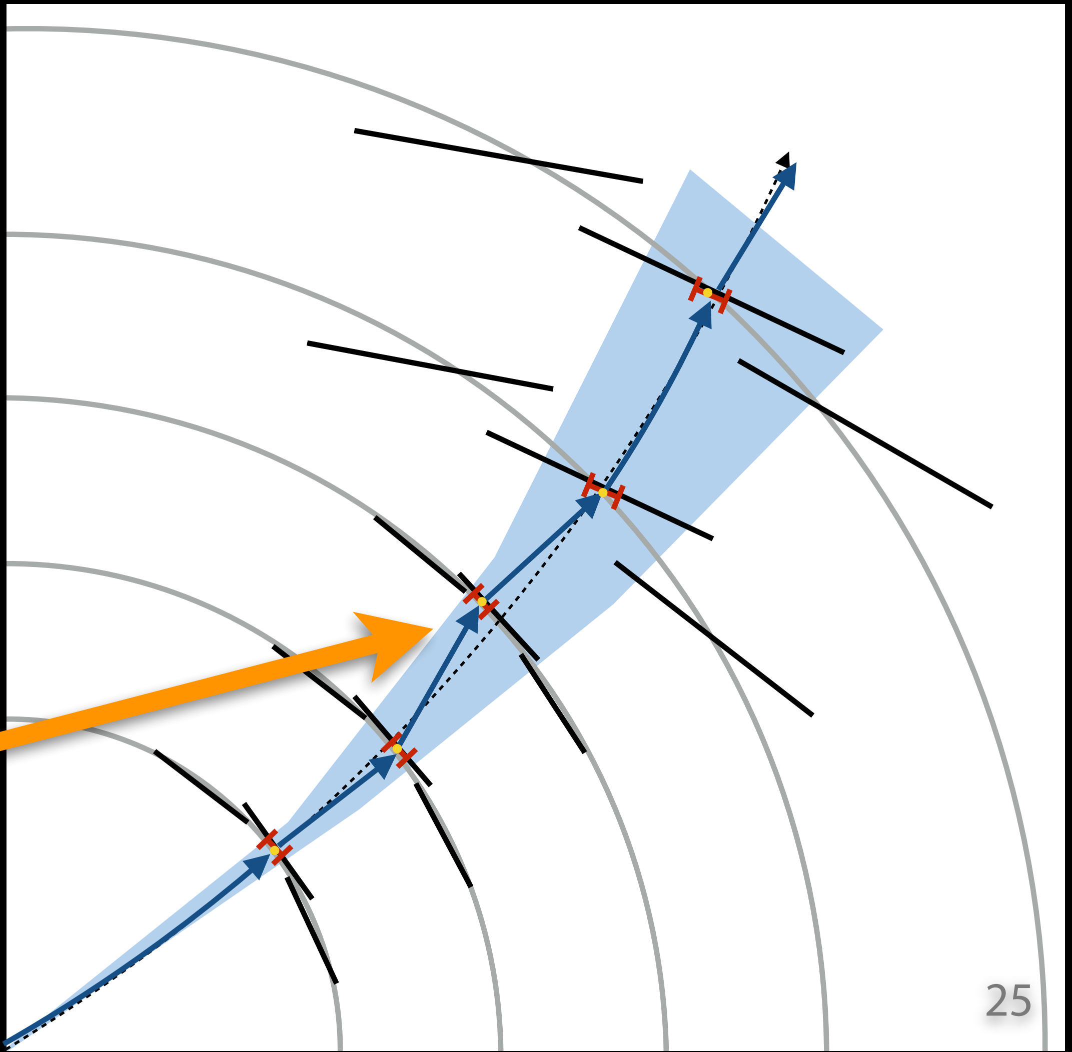
Classical Least Square Track Fit

- allowing for material effects in fit:
  - ➡ can be absorbed in track model $f_{k|i}$, provided effects are small
  - ➡ for substantial multiple scatting, allows for **scattering angles** in the fit
- introduce scattering angles on material surfaces
  - ➡ on each material surface, add 2 angles $\delta\theta_i$ as fee parameters to the fit
  - ➡ expected mean of those angles is 0 (!), their covariance Q_i is given by multiple scattering in x/X_0
- results in additional term in χ^2 equations:

$$\chi^2 = \sum_k \Delta m_k^T G_K^{-1} \Delta m_k + \sum_i \delta\theta_i^T Q_i^{-1} \delta\theta_i$$

with: $\Delta m_k = m_k - d_k\left(p, \delta\theta_i\right)$

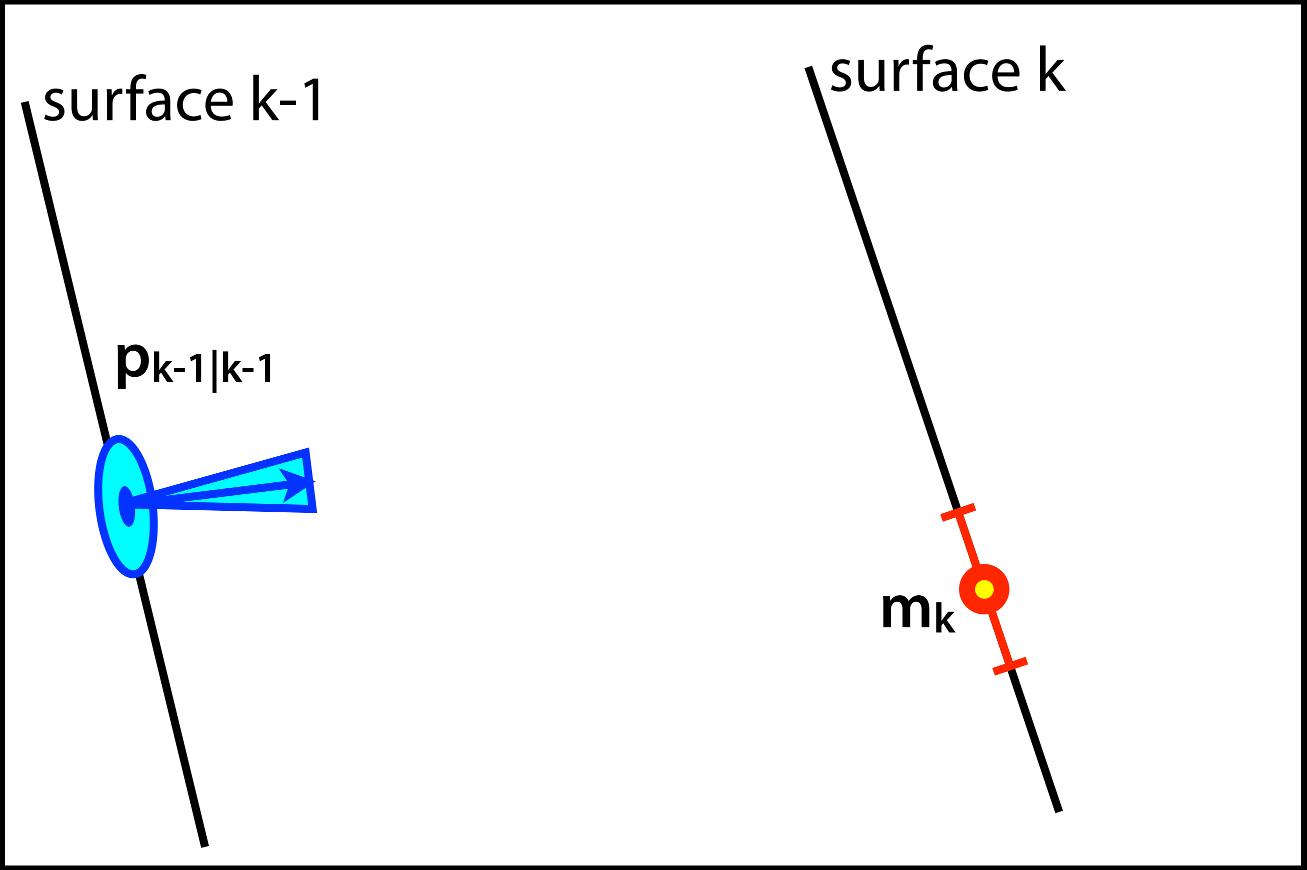
  - ➡ computationally expensive
    *(invert a dimension 5+2*n matrix)*
  - ➡ advantage is that the fitted track follows precisely the particle trajectory
    (e.g. for ATLAS muon reconstruction)



The Kalman Filter Track Fit

- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ can be shown that it is mathematically equivalent

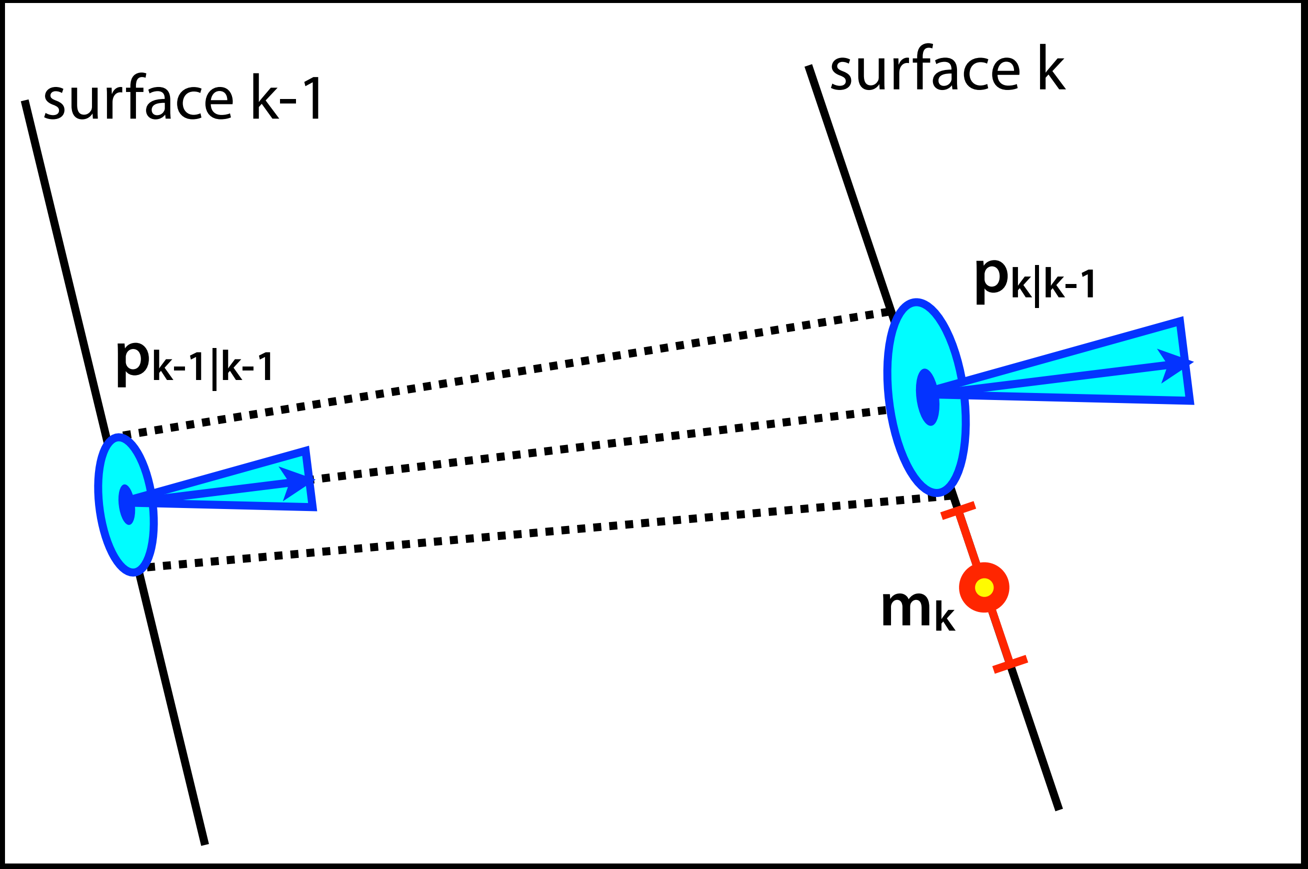
- how does the filter work ?
  - ➡ estimate starting parameters $p_{0|0}$
  - ➡ iterate over all hits **1..K**:
  1. take trajectory parameters $p_{k-1|k-1}$ at point **k-1**



The Kalman Filter Track Fit

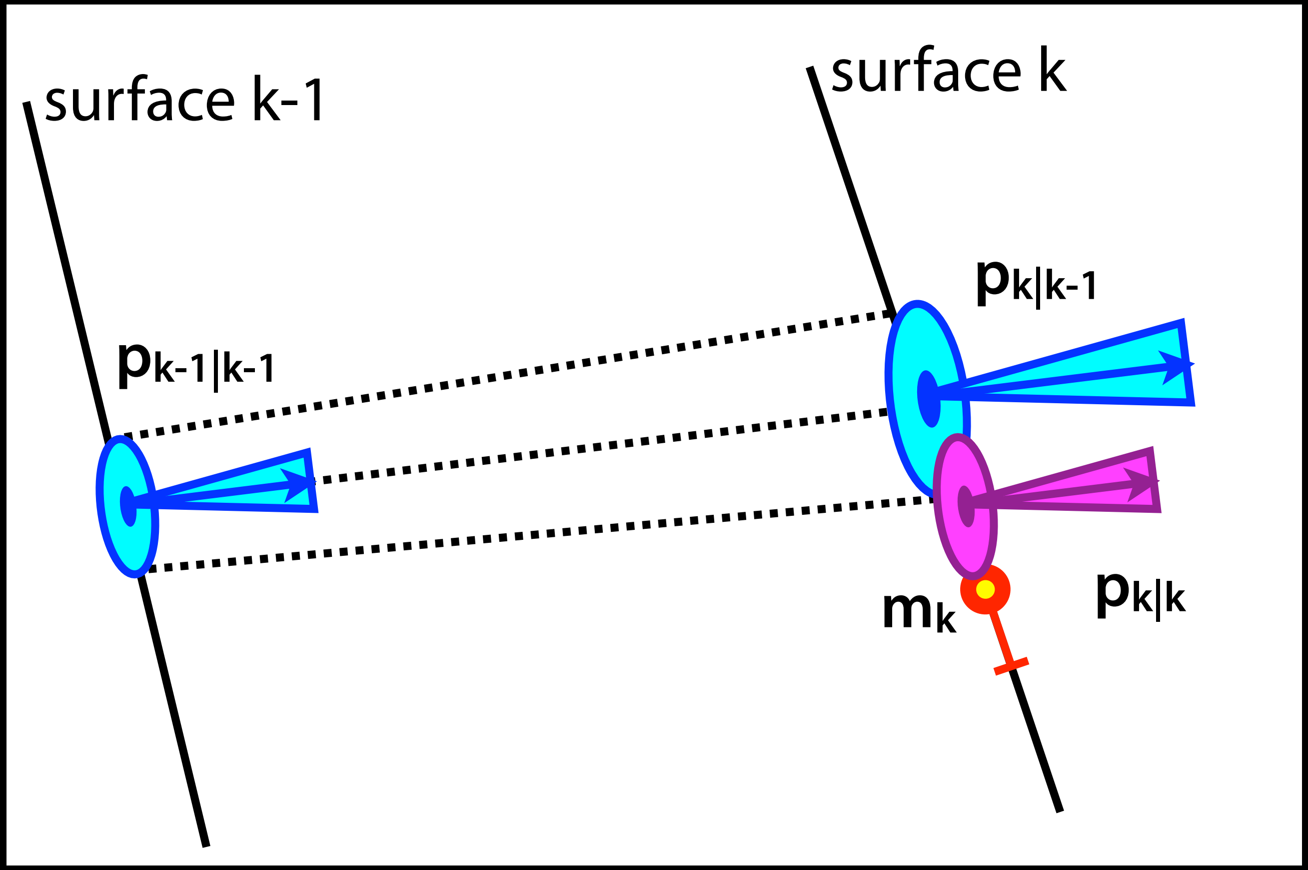
- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ can be shown that it is mathematically equivalent

- how does the filter work ?
  - ➡ estimate starting parameters $p_{0|0}$
  - ➡ iterate over all hits **1..K**:
  1. take trajectory parameters $p_{k-1|k-1}$ at point **k-1**
  2. propagate to point **k** to get predicted parameters $p_{k|k-1}$



The Kalman Filter Track Fit

- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ can be shown that it is mathematically equivalent
- how does the filter work ?
  - ➡ estimate starting parameters $p_{0|0}$
  - ➡ iterate over all hits **1..K**:
  1. take trajectory parameters $p_{k-1|k-1}$ at point **k-1**
  2. propagate to point **k** to get predicted parameters $p_{k|k-1}$
  3. update predicted parameters with measurement m_k to obtain $p_{k|k}$ (simple weighted mean or gain matrix update)

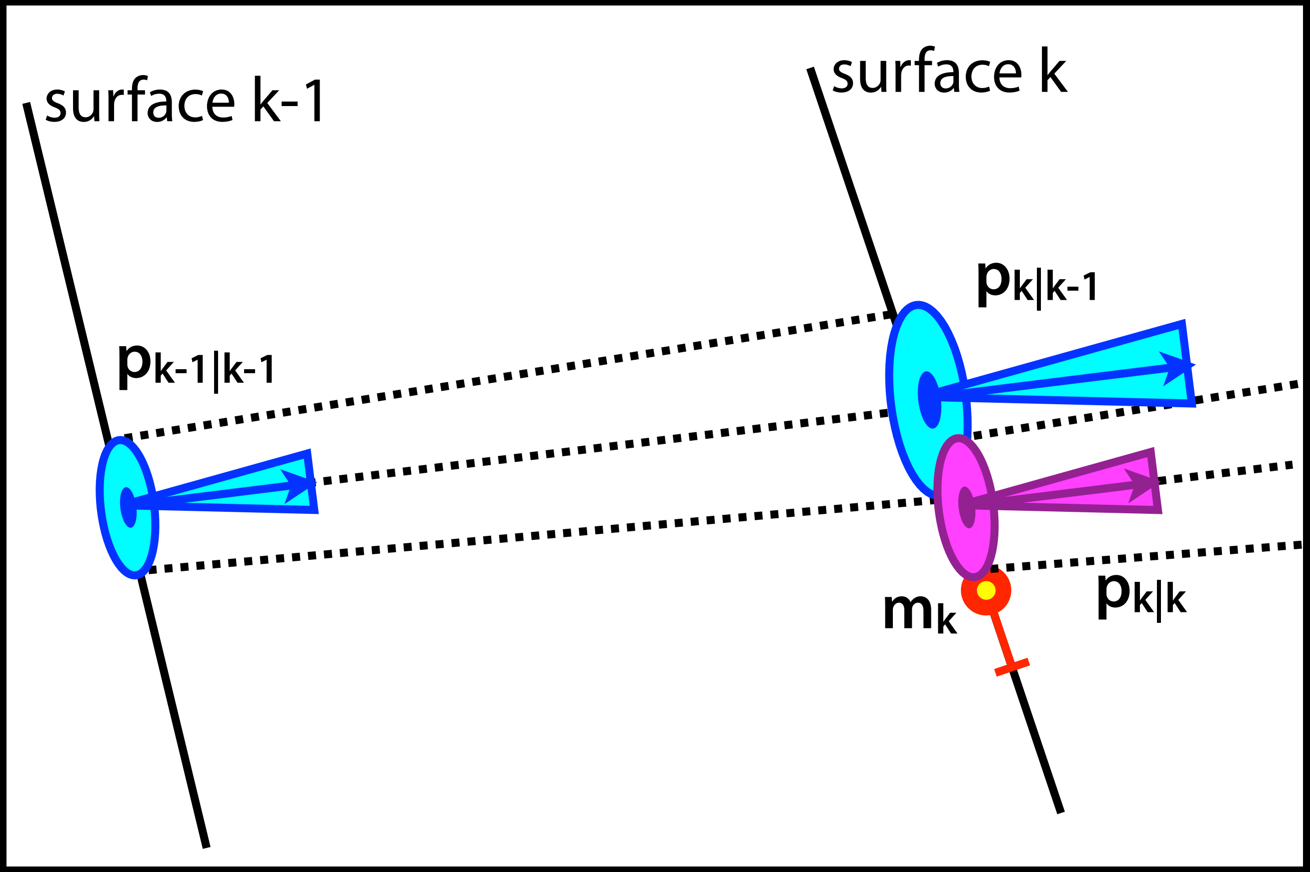


The Kalman Filter Track Fit

- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ can be shown that it is mathematically equivalent

- how does the filter work ?
  - ➡ estimate starting parameters $p_{0|0}$
  - ➡ iterate over all hits **1..K**:

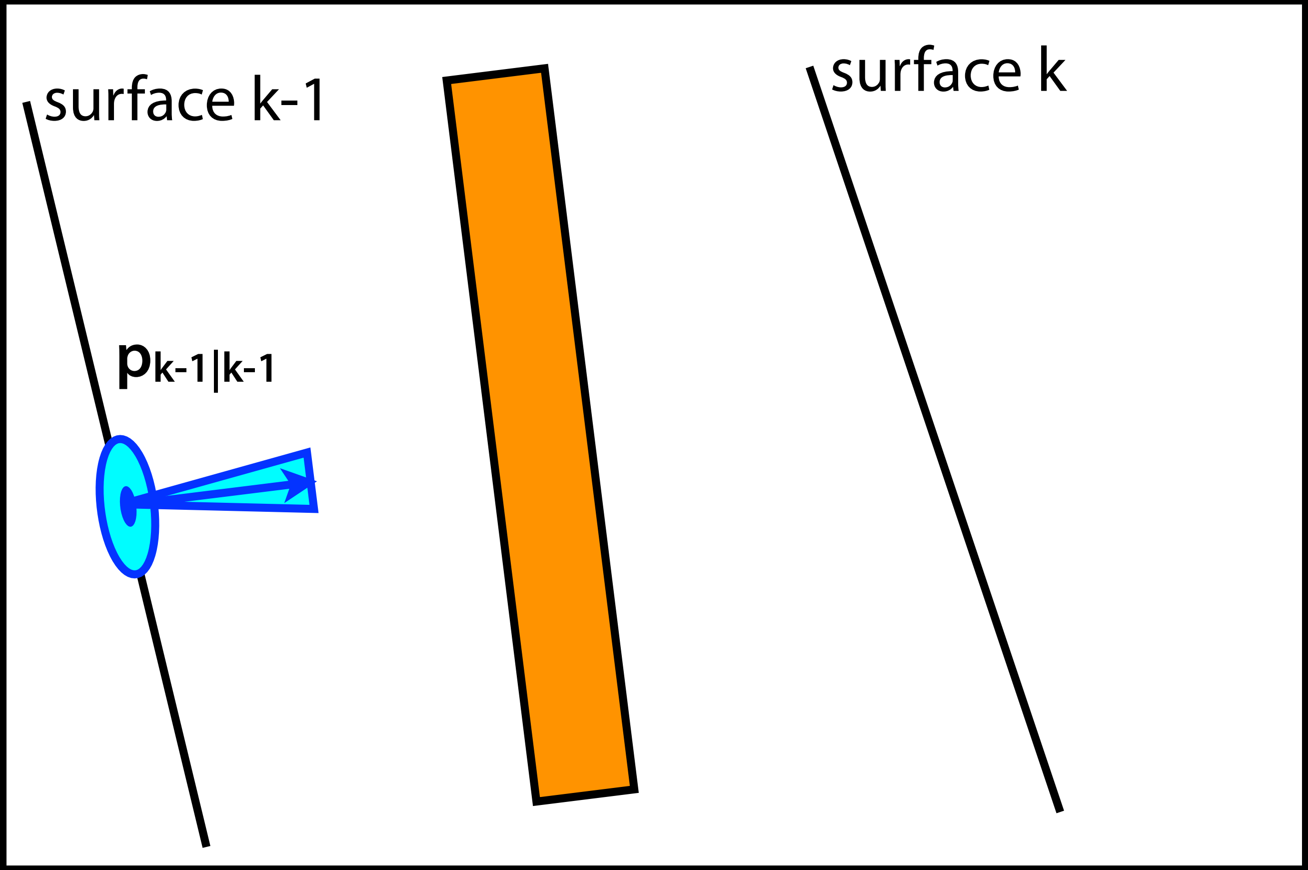
1. take trajectory parameters $p_{k-1|k-1}$ at point **k-1**
2. propagate to point **k** to get predicted parameters $p_{k|k-1}$
3. update predicted parameters with measurement m_k to obtain $p_{k|k}$ (simple weighted mean or gain matrix update)
4. and start over with 1.



The Kalman Filter Track Fit

- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ can be shown that it is mathematically equivalent

- how does the filter work ?
  - ➡ estimate starting parameters $p_{0|0}$
  - ➡ iterate over all hits **1..K**:
    1. take trajectory parameters $p_{k-1|k-1}$ at point **k-1**
    2. propagate to point **k** to get predicted parameters $p_{k|k-1}$
    3. update predicted parameters with measurement m_k to obtain $p_{k|k}$ (simple weighted mean or gain matrix update)
    4. and start over with 1.

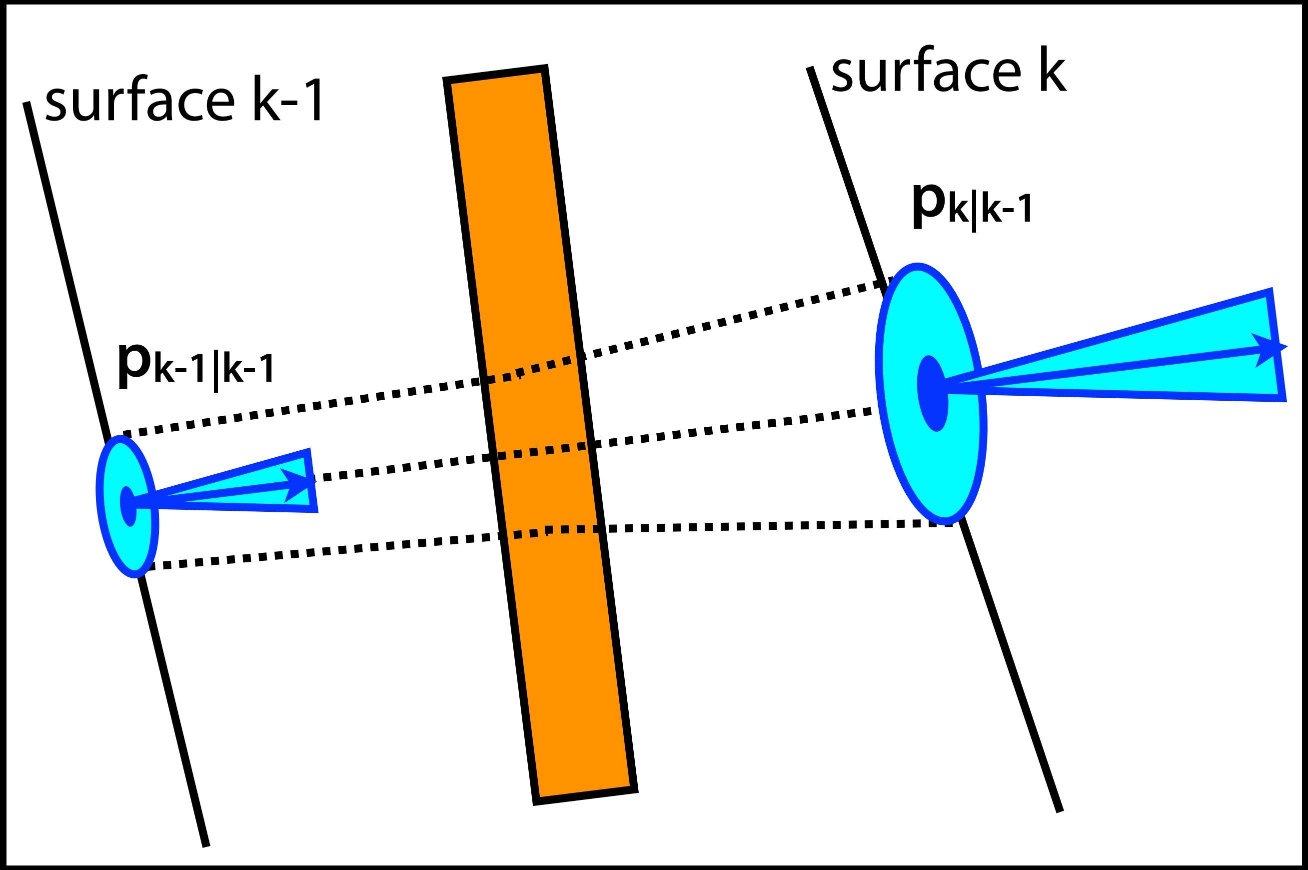


- material effects (multiple scattering and energy loss)

The Kalman Filter Track Fit

- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ can be shown that it is mathematically equivalent

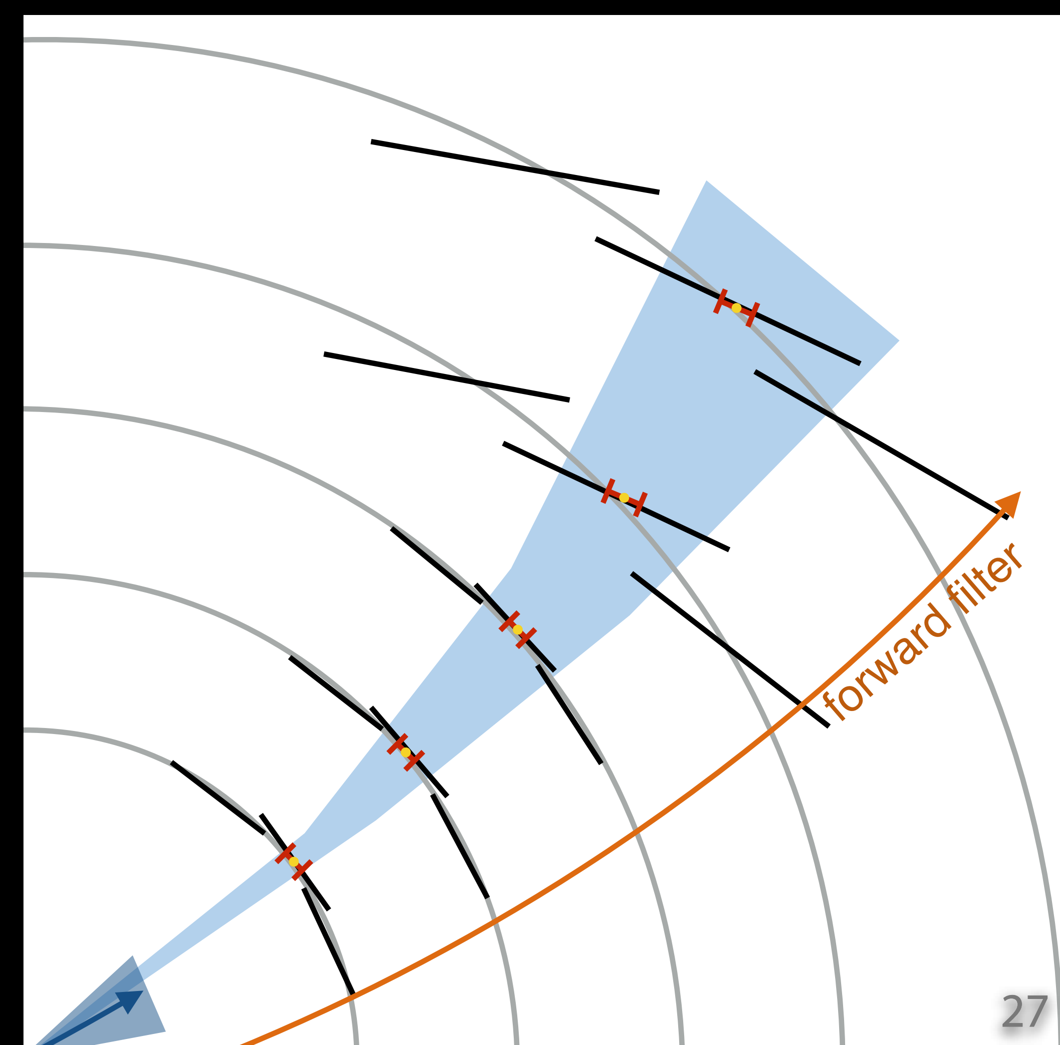
- how does the filter work ?
  - ➡ estimate starting parameters $p_{0|0}$
  - ➡ iterate over all hits **1..K**:
    1. take trajectory parameters $p_{k-1|k-1}$ at point **k-1**
    2. propagate to point **k** to get predicted parameters $p_{k|k-1}$
    3. update predicted parameters with measurement m_k to obtain $p_{k|k}$ (simple weighted mean or gain matrix update)
    4. and start over with 1.



- material effects (multiple scattering and energy loss)
  - ➡ incorporated in the propagated parameters $p_{k|k-1}$ (extrapolated prediction)
  - ➡ and therefore enters automatically in the updated parameters $p_{k|k}$ at point **k**

The Kalman Filter Track Fit

- forward filter
  - ➡ in mathematical terms:



The Kalman Filter Track Fit

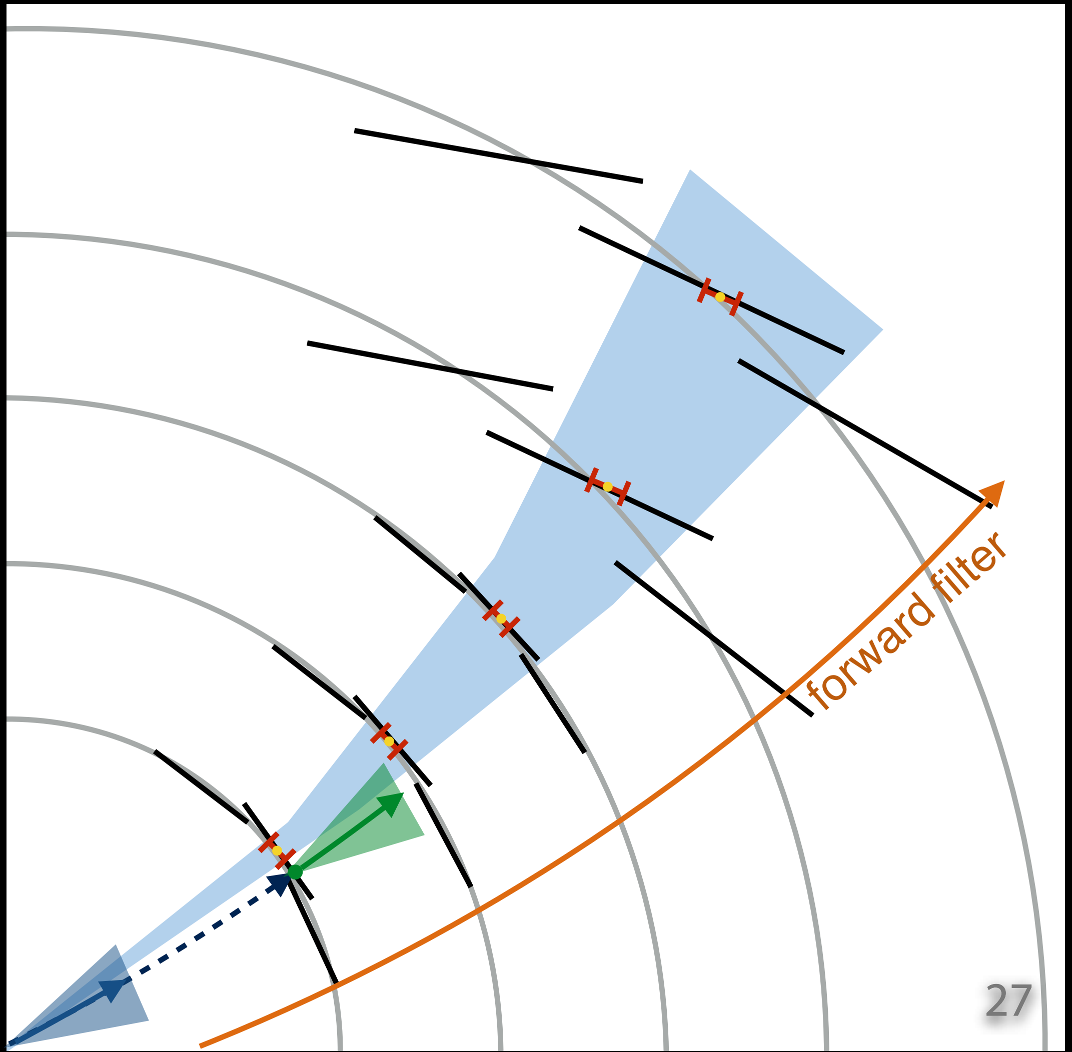
- forward filter
  - ➡ in mathematical terms:

1. propagate p_{k-1} and its covariance C_{k-1} :

$$q_{k|k-1} = f_{k|k-1}(q_{k-1|k-1})$$

$$C_{k|k-1} = F_{k|k-1} C_{k-1|k-1} F^{T}_{k|k-1} + Q_k$$

with Q_k ~ noise term (M.S.)



The Kalman Filter Track Fit

- forward filter
  - ➡ in mathematical terms:

1. propagate p_{k-1} and its covariance C_{k-1} :

$$q_{k|k-1} = f_{k|k-1}(q_{k-1|k-1})$$
$$C_{k|k-1} = F_{k|k-1} C_{k-1|k-1} F^{\mathrm{T}}_{k|k-1} + Q_k$$

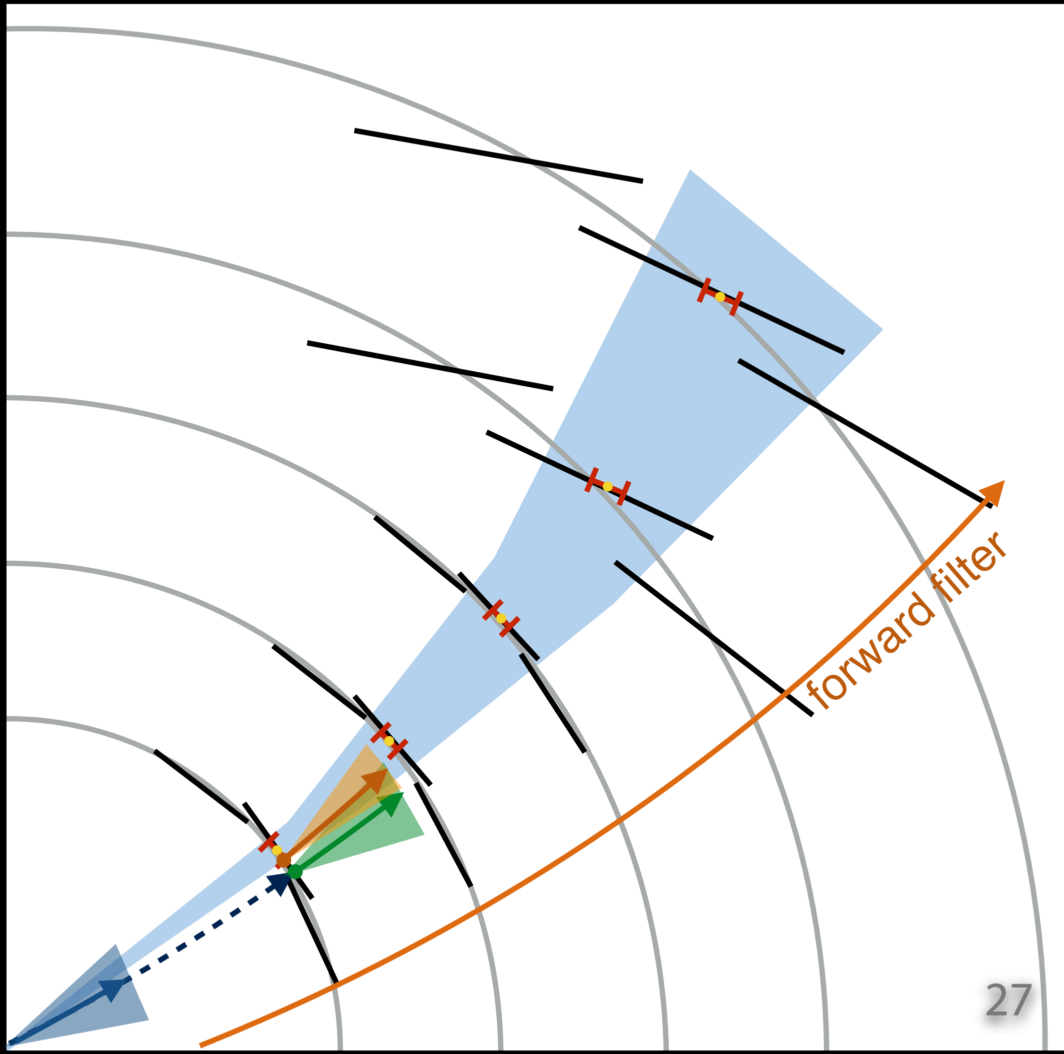
with Q_k ~ noise term (M.S.)

2. update prediction to get $q_{k|k}$ and $C_{k|k}$:

$$q_{k|k} = q_{k|k-1} + K_k [m_k - h_k(q_{k|k-1})]$$
$$C_{k|k} = (I - K_k H_k) C_{k|k-1}$$

with K_k ~ gain matrix :

$$K_k = C_{k|k-1} H^{\mathrm{T}}_k (G_k + H_k C_{k|k-1} H^{\mathrm{T}}_k)^{-1}$$



The Kalman Filter Track Fit

- forward filter
  - ➡ in mathematical terms:

1. propagate p_{k-1} and its covariance C_{k-1} :

$$q_{k|k-1} = f_{k|k-1}(q_{k-1|k-1})$$
$$C_{k|k-1} = F_{k|k-1} C_{k-1|k-1} F^{\mathrm{T}}_{k|k-1} + Q_k$$

with Q_k ~ noise term (M.S.)

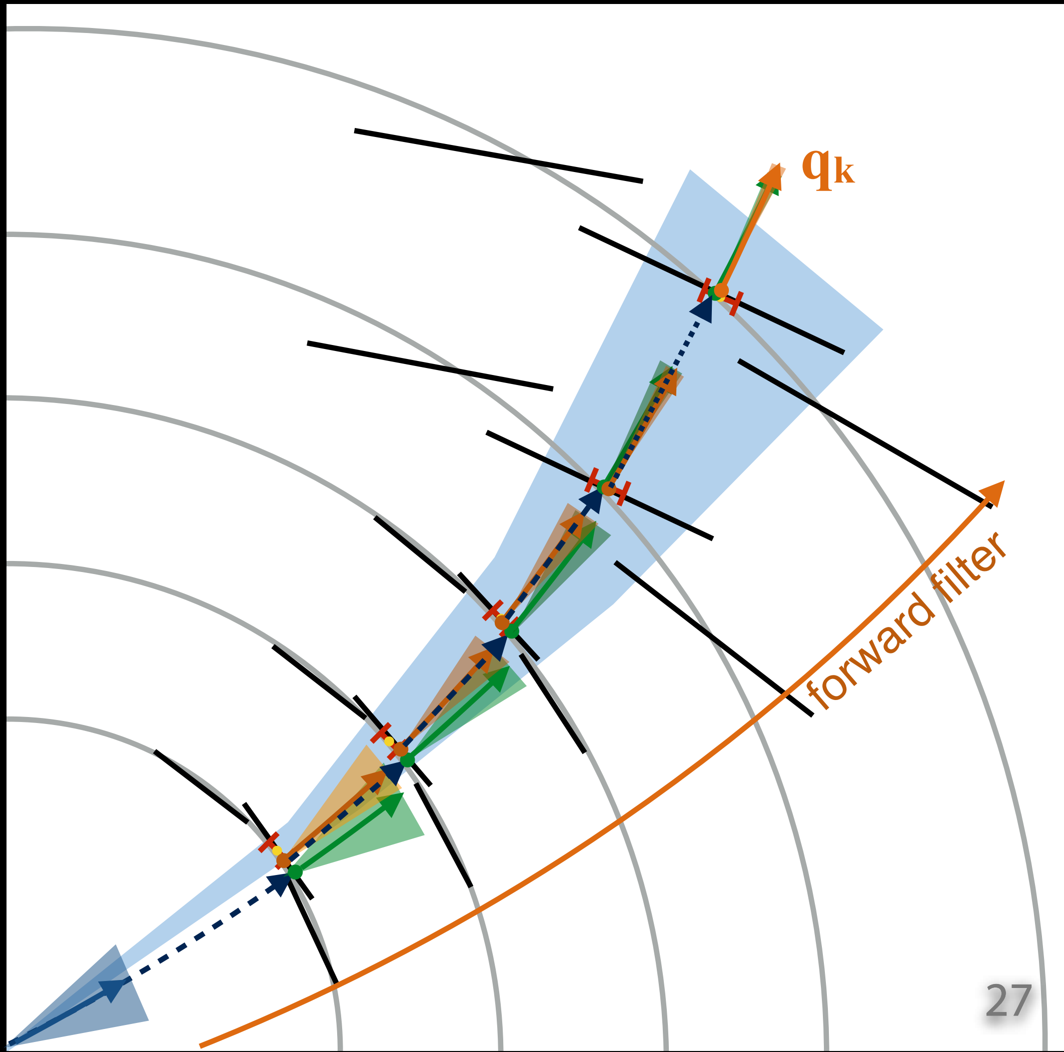
2. update prediction to get $q_{k|k}$ and $C_{k|k}$:

$$q_{k|k} = q_{k|k-1} + K_k [m_k - h_k(q_{k|k-1})]$$
$$C_{k|k} = (I - K_k H_k) C_{k|k-1}$$

with K_k ~ gain matrix :

$$K_k = C_{k|k-1} H^{\mathrm{T}}_k (G_k + H_k C_{k|k-1} H^{\mathrm{T}}_k)^{-1}$$

- ➡ precise fit result q_k at end of fit



The Kalman Filter Track Fit

- forward filter
  - ➡ in mathematical terms:

1. propagate p_{k-1} and its covariance C_{k-1} :

$$q_{k|k-1} = f_{k|k-1}(q_{k-1|k-1})$$

$$C_{k|k-1} = F_{k|k-1} C_{k-1|k-1} F^{T}_{k|k-1} + Q_k$$

with Q_k ~ noise term (M.S.)

2. update prediction to get $q_{k|k}$ and $C_{k|k}$:

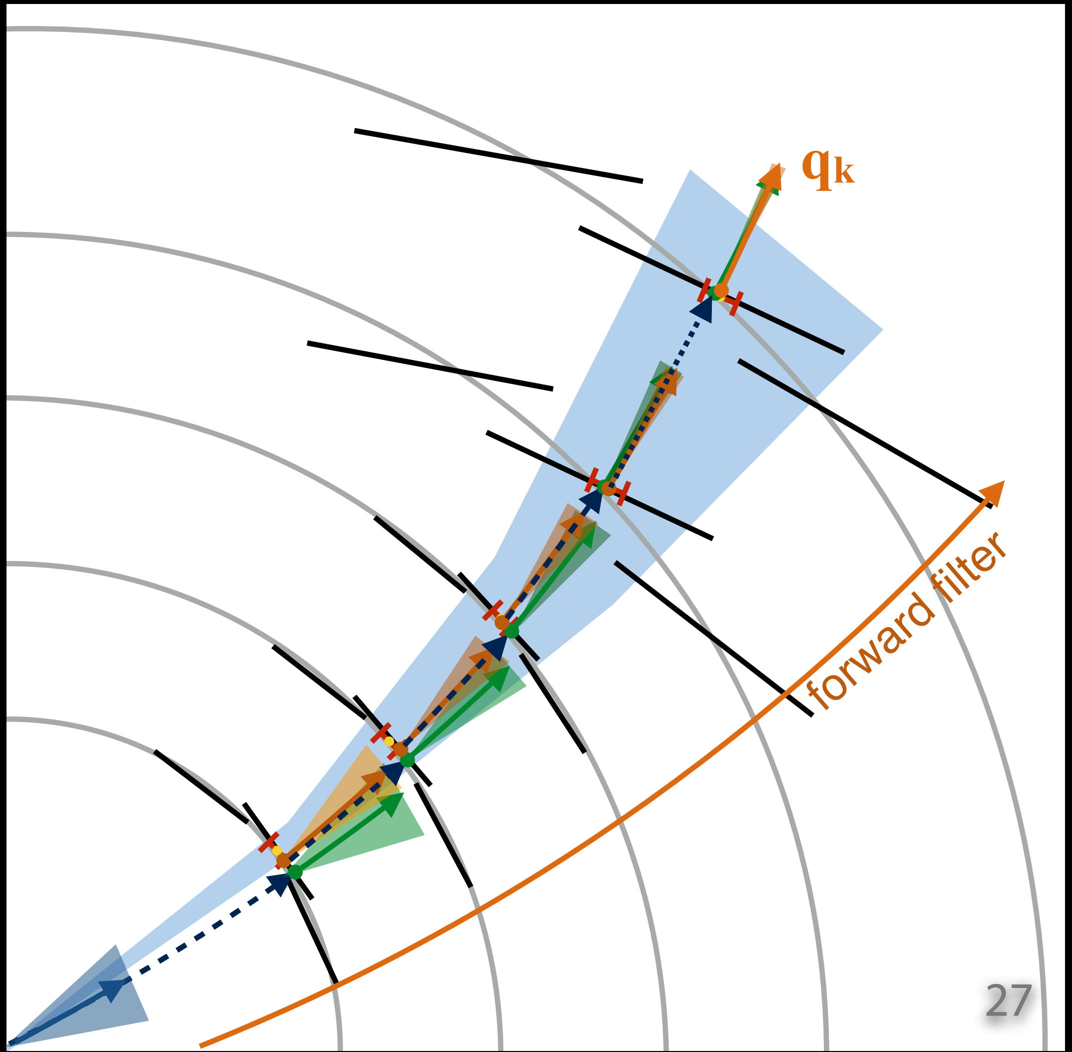
$$q_{k|k} = q_{k|k-1} + K_k [m_k - h_k(q_{k|k-1})]$$

$$C_{k|k} = (I - K_k H_k) C_{k|k-1}$$

with K_k ~ gain matrix :

$$K_k = C_{k|k-1} H^{T}_{k} (G_k + H_k C_{k|k-1} H^{T}_{k})^{-1}$$

- ➡ precise fit result q_k at end of fit
- ➡ alternative to gain matrix approach is a weighted mean to obtian $p_{k|k}$
  - but requires to invert 5x5 matrix instead of a matrix of $rank(G_k)$



The Kalman Filter Track Fit

- forward filter
  - ➡ in mathematical terms:

1. propagate p_{k-1} and its covariance C_{k-1} :

$$q_{k|k-1} = f_{k|k-1}(q_{k-1|k-1})$$
$$C_{k|k-1} = F_{k|k-1} C_{k-1|k-1} F^{\mathrm{T}}_{k|k-1} + Q_k$$

with Q_k ~ noise term (M.S.)

2. update prediction to get $q_{k|k}$ and $C_{k|k}$:

$$q_{k|k} = q_{k|k-1} + K_k [m_k - h_k(q_{k|k-1})]$$
$$C_{k|k} = (I - K_k H_k) C_{k|k-1}$$

with K_k ~ gain matrix :

$$K_k = C_{k|k-1} H^{\mathrm{T}}_k (G_k + H_k C_{k|k-1} H^{\mathrm{T}}_k)^{-1}$$

  - ➡ precise fit result q_k at end of fit

- Kalman Smoother:
  - ➡ provides full information along track
  - ➡ equivalent: average forw./back. filter

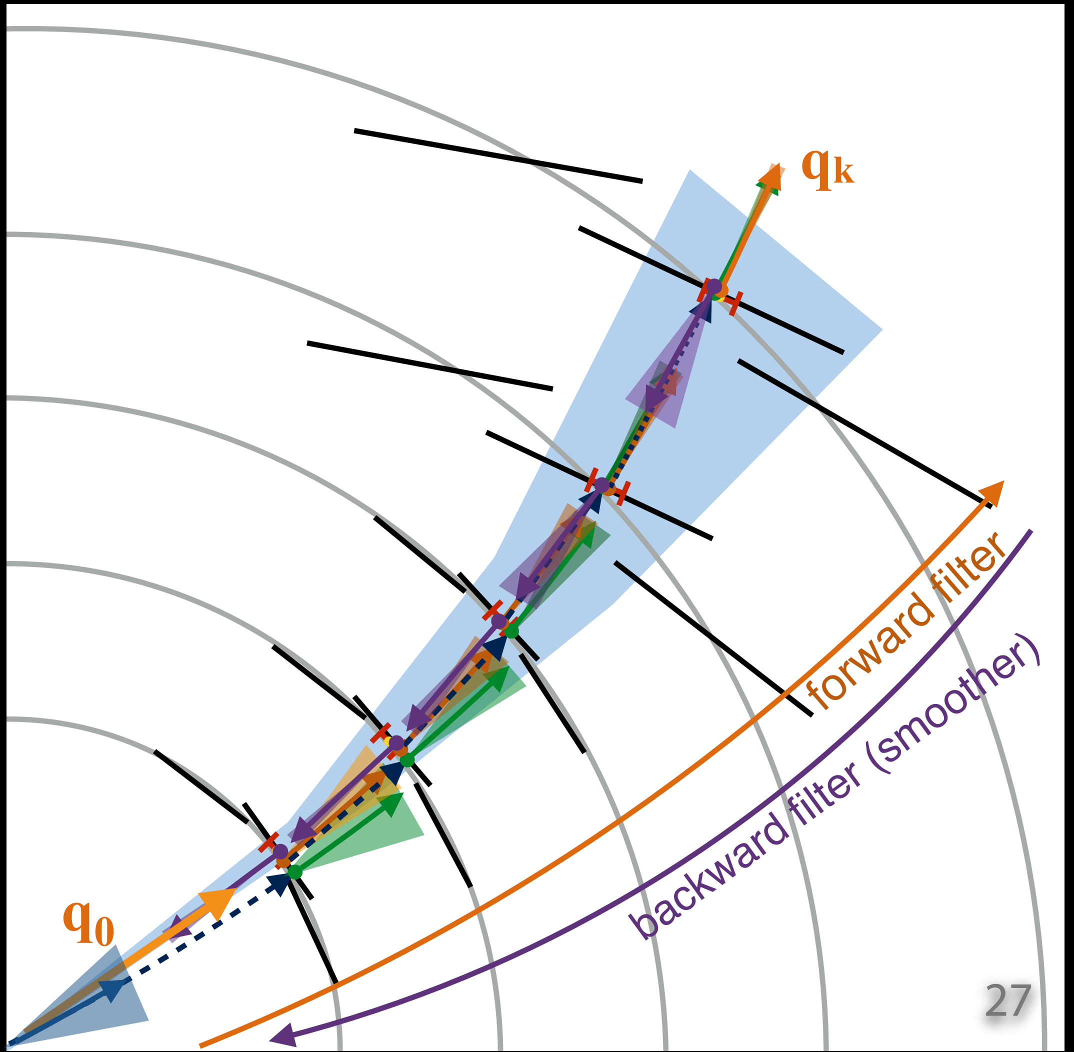
➡ Smoother in mathematical terms:

proceeds from layer $k+1$ to layer k :

$$q_{k|n} = q_{k|k} + A_k (q_{k+1|n} - q_{k+1|k})$$
$$C_{k|n} = C_{k|k} - A_k (C_{k+1|k} - C_{k+1|n}) A^{\mathrm{T}}_k$$

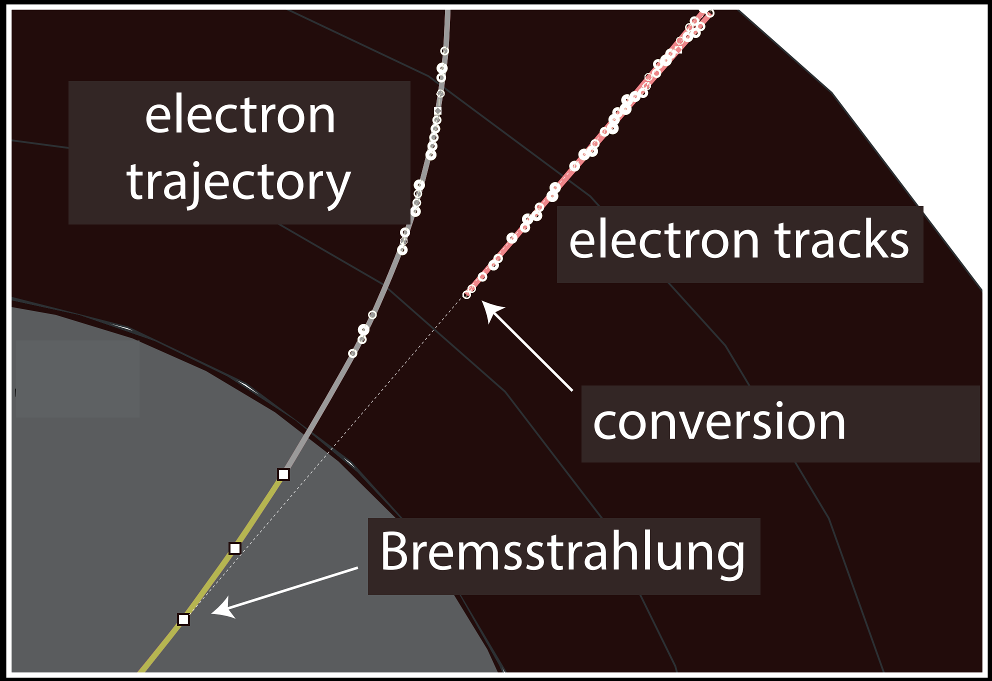
with A_k ~ smoother gain matrix :

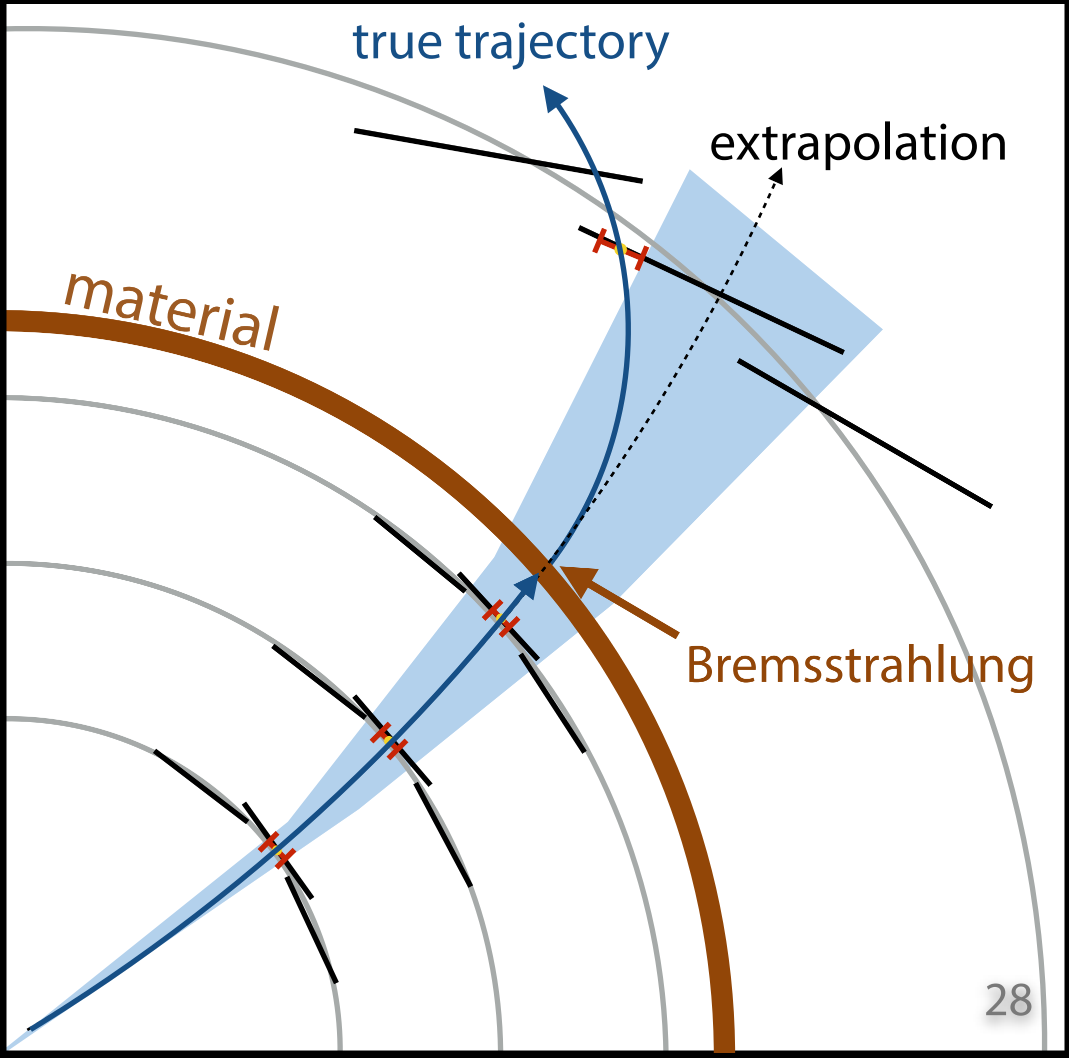
$$A_k = C_{k|k} F^{\mathrm{T}}_{k+1|k} (C_{k+1|k})^{-1}$$



Fitting for Electron Bremsstrahlung

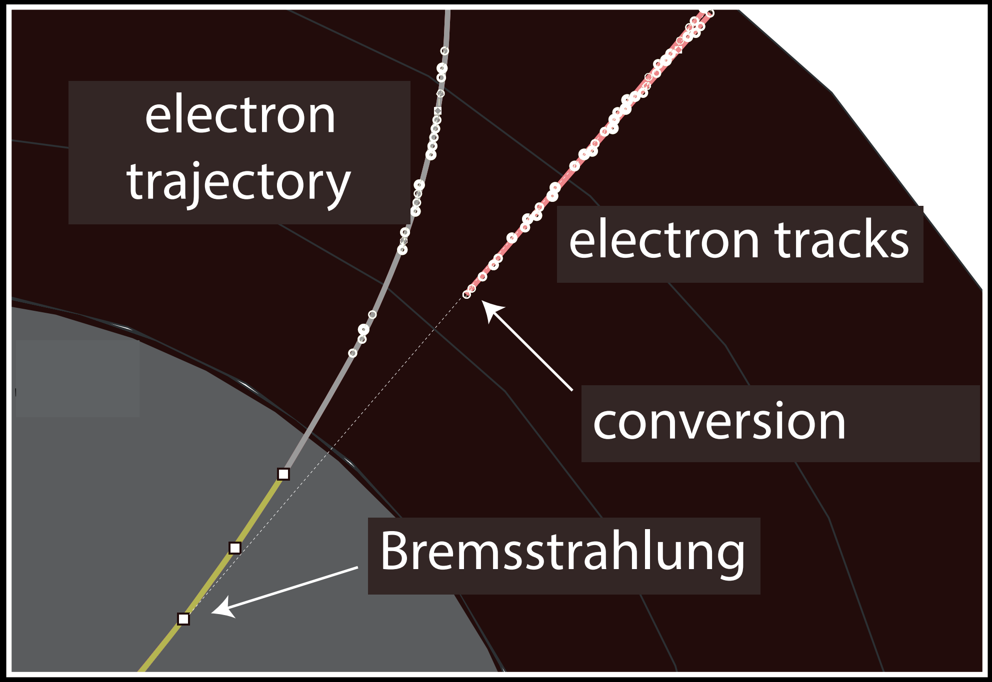
- material in tracker
  - ➡ e-Bremsstrahlung and γ-conversions
- electron efficiency limited
  - ➡ momentum loss due to Bremsstrahlung leads to sudden large changes in track curvature
  - ➡ loosing hits after Brem. leads to inefficiency
  - ➡ fit either biased towards small momenta or fails completely because of bad χ^2

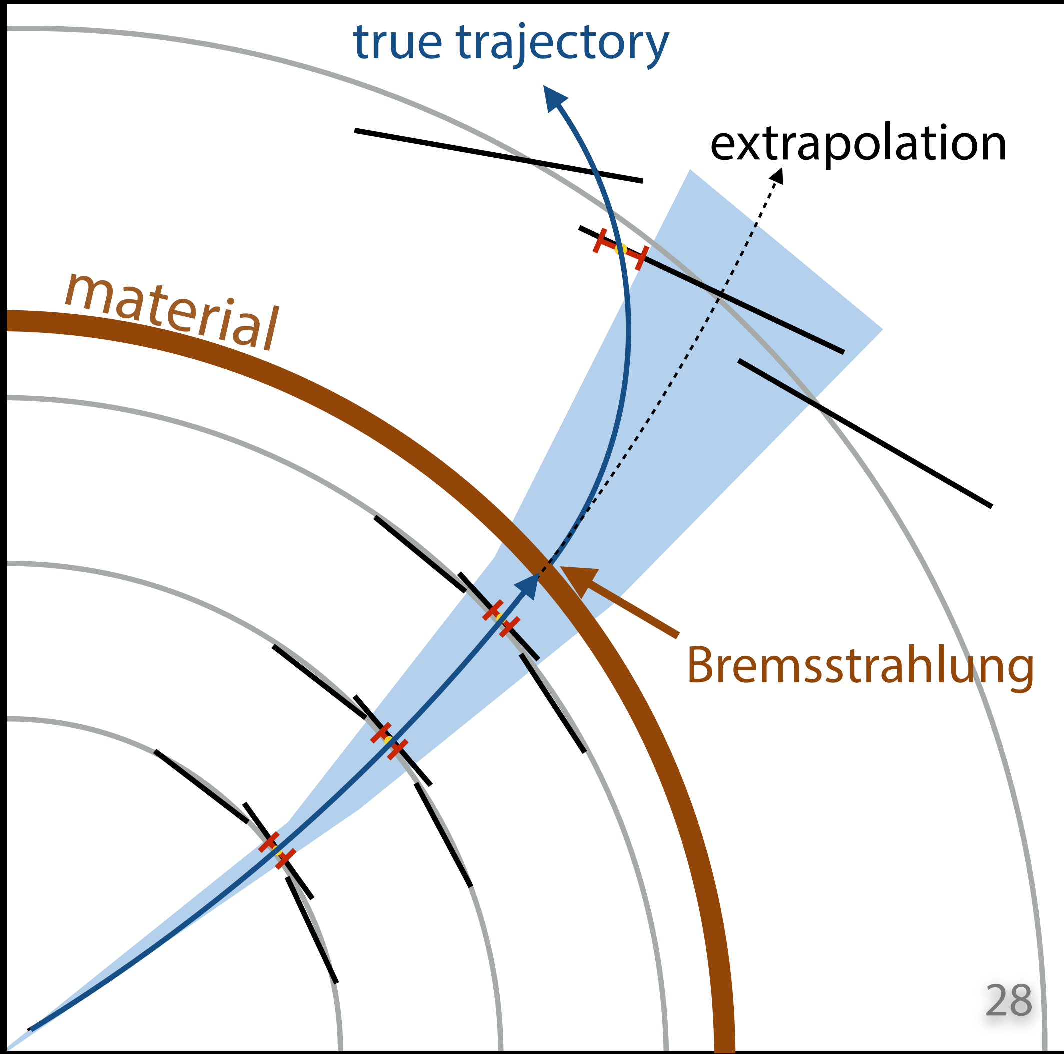




Fitting for Electron Bremsstrahlung

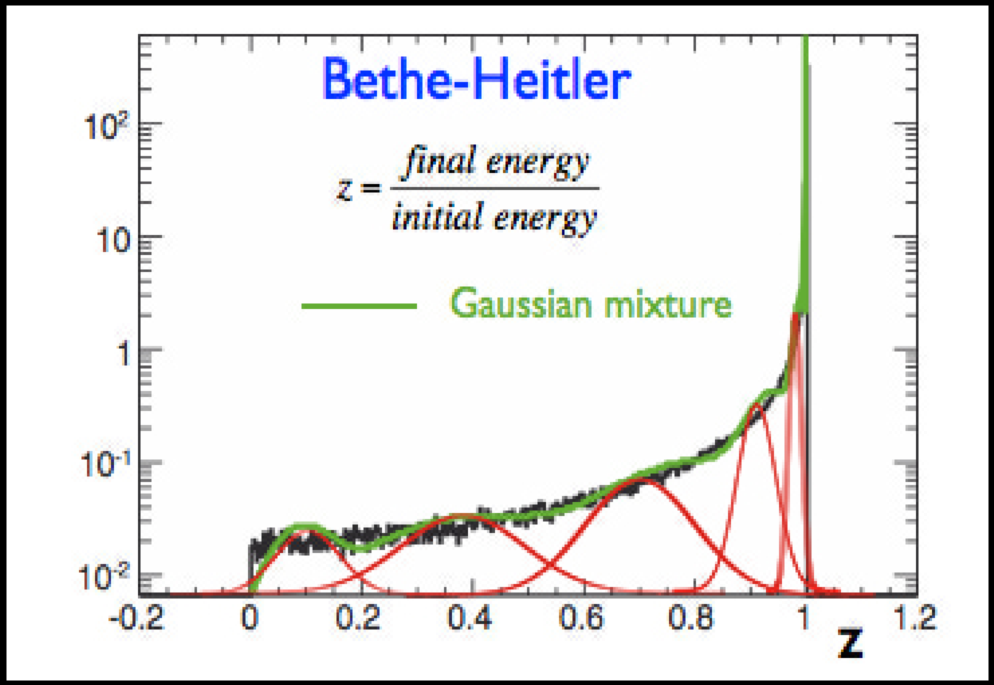
- material in tracker
  - ➡ e-Bremsstrahlung and γ-conversions
- electron efficiency limited
  - ➡ momentum loss due to Bremsstrahlung leads to sudden large changes in track curvature
  - ➡ loosing hits after Brem. leads to inefficiency
  - ➡ fit either biased towards small momenta or fails completely because of bad χ^2
- techniques to allow for Bremsstrahlung in track fitting
  - ➡ for Least Square track fit
    - allow Brem. effect to change curvature, additional term similar is to scattering angle
  - ➡ for Kalman Filter
    - increase correction for material effects in propagation to allow for Brem.
  - ➡ better: Gaussian Sum Filter

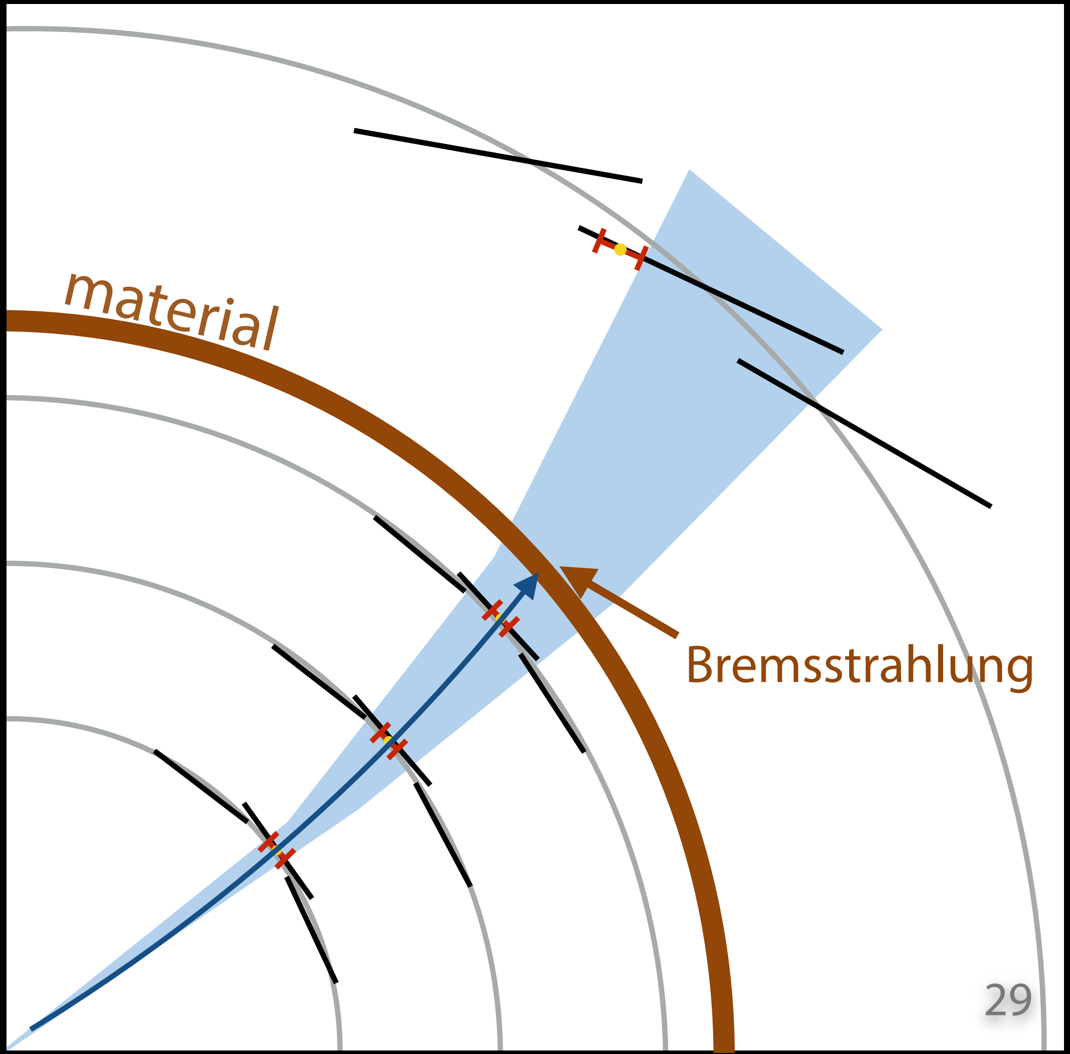




The Gaussian Sum Filter

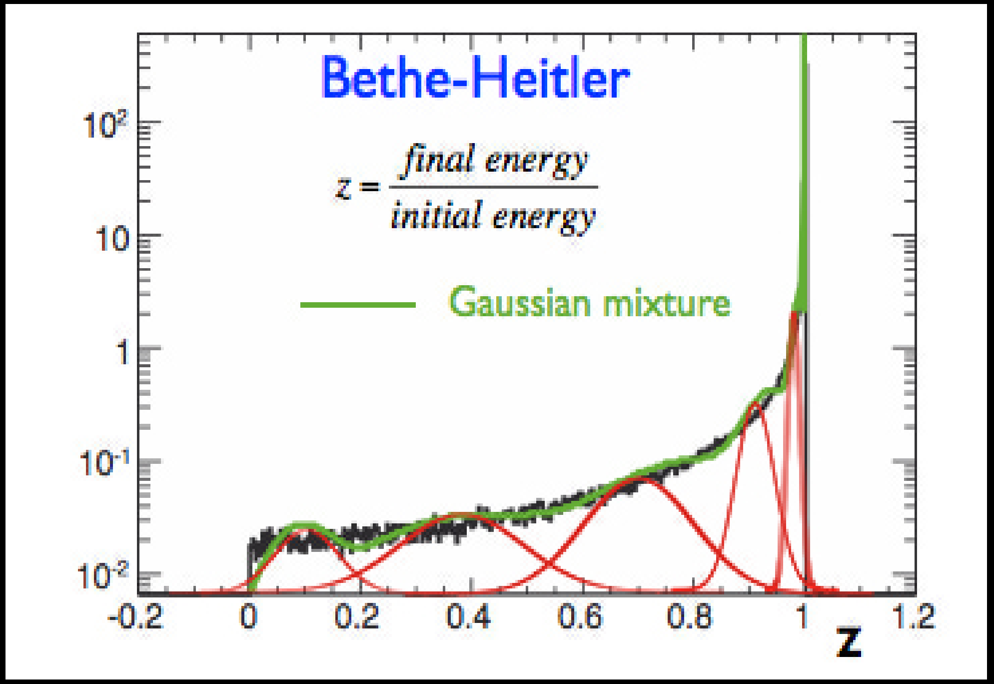
- approximate Bethe-Heitler distribution as Gaussian mixture

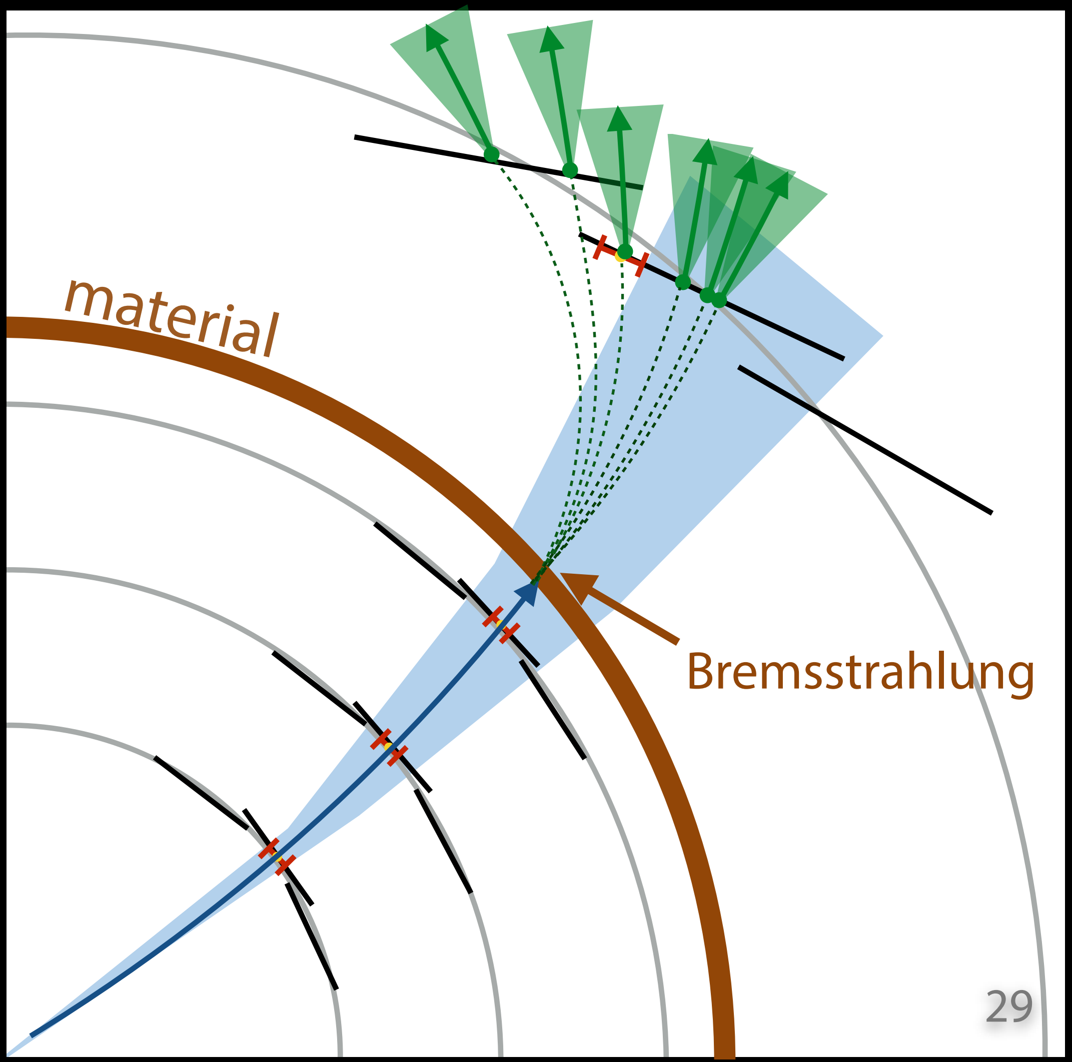




The Gaussian Sum Filter

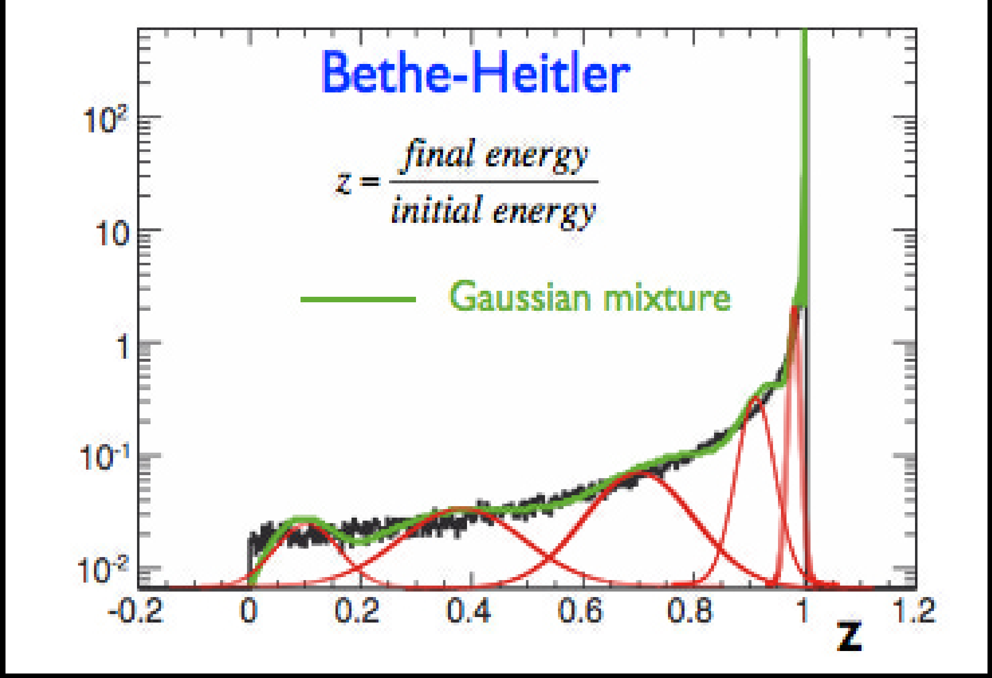
- approximate Bethe-Heitler distribution as Gaussian mixture
  - ➡ state vector after material correction becomes sum of Gaussian components
    - relative weights from Bethe-Heitler distribution
    - GSF step resembles set of parallel Kalman Filters
    - computationally expensive !

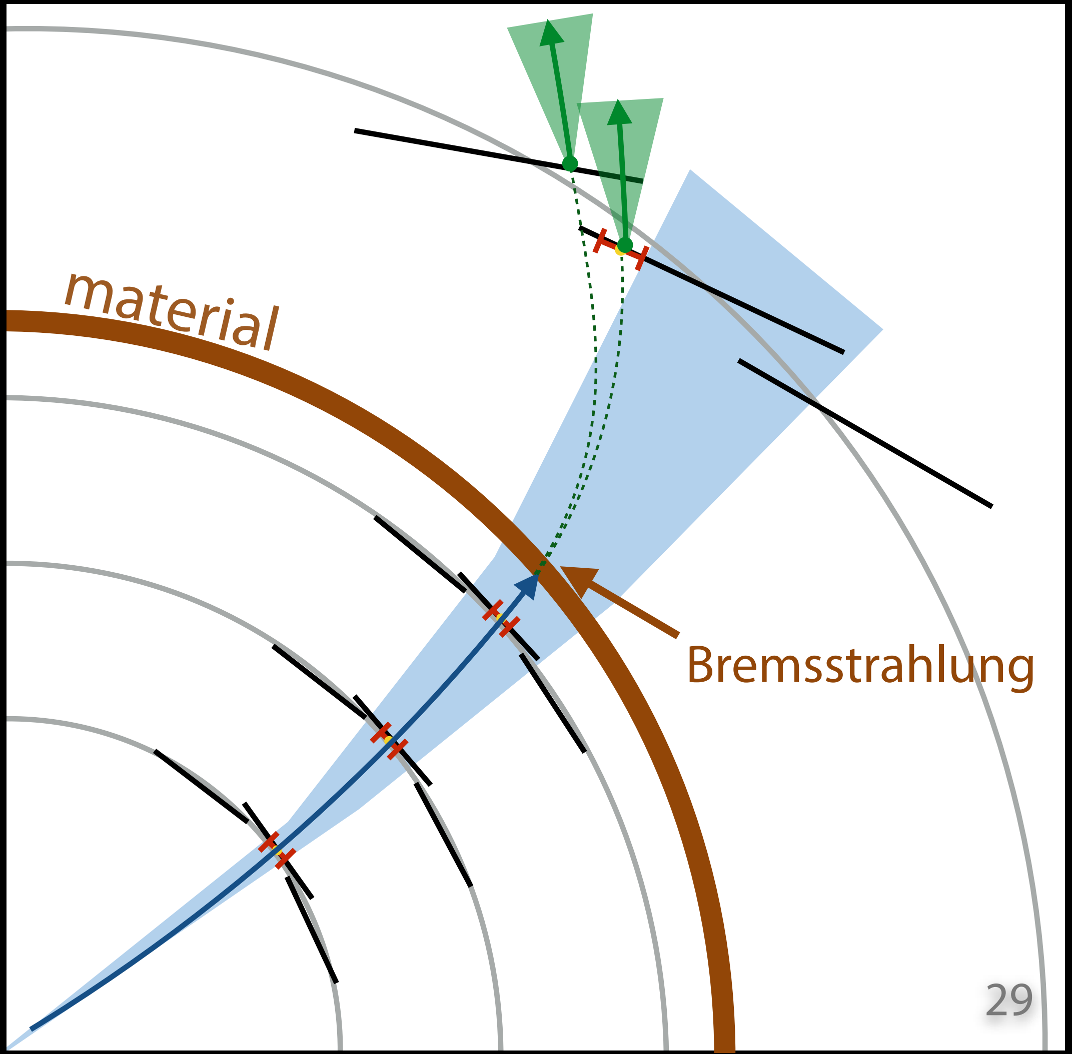




The Gaussian Sum Filter

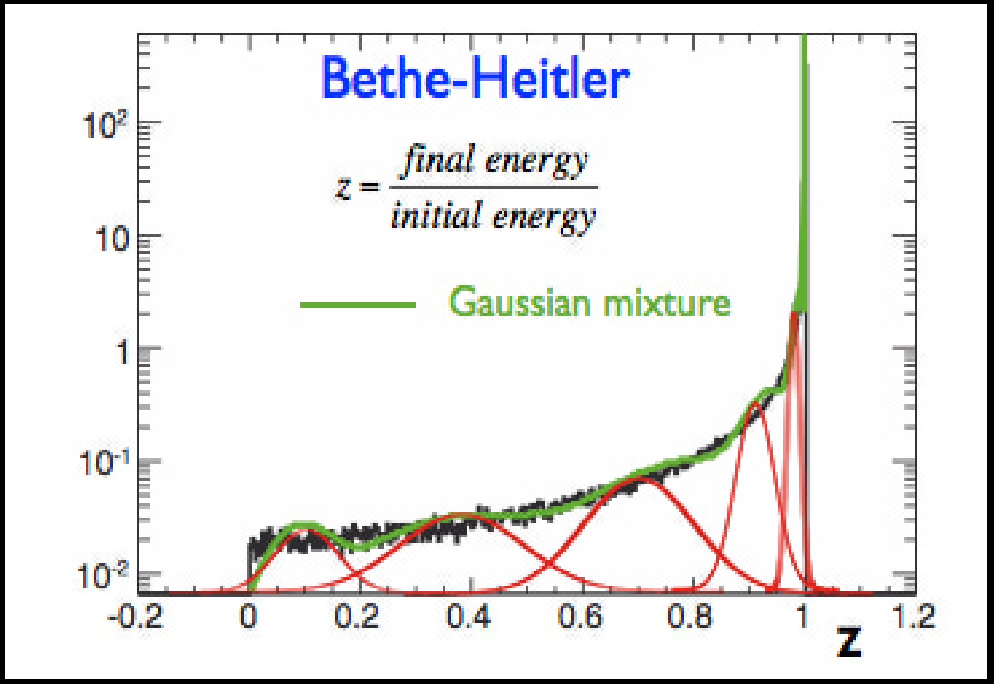
- approximate Bethe-Heitler distribution as Gaussian mixture
  - ➡ state vector after material correction becomes sum of Gaussian components
    - relative weights from Bethe-Heitler distribution
    - GSF step resembles set of parallel Kalman Filters
    - computationally expensive !
  - ➡ component reduction to avoid combinatorial explosion after several material layers
    - re-evaluate weights of components based on compatibility with hits
    - drop components with too low weights

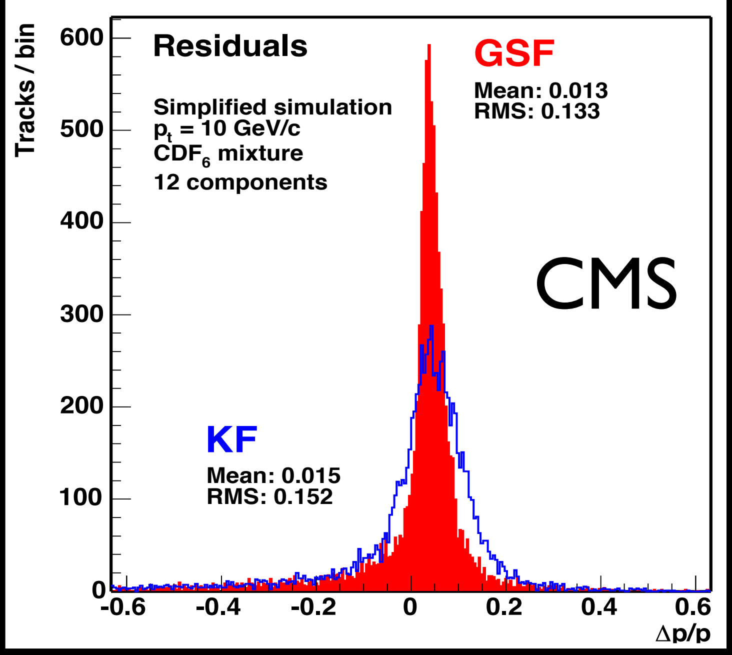


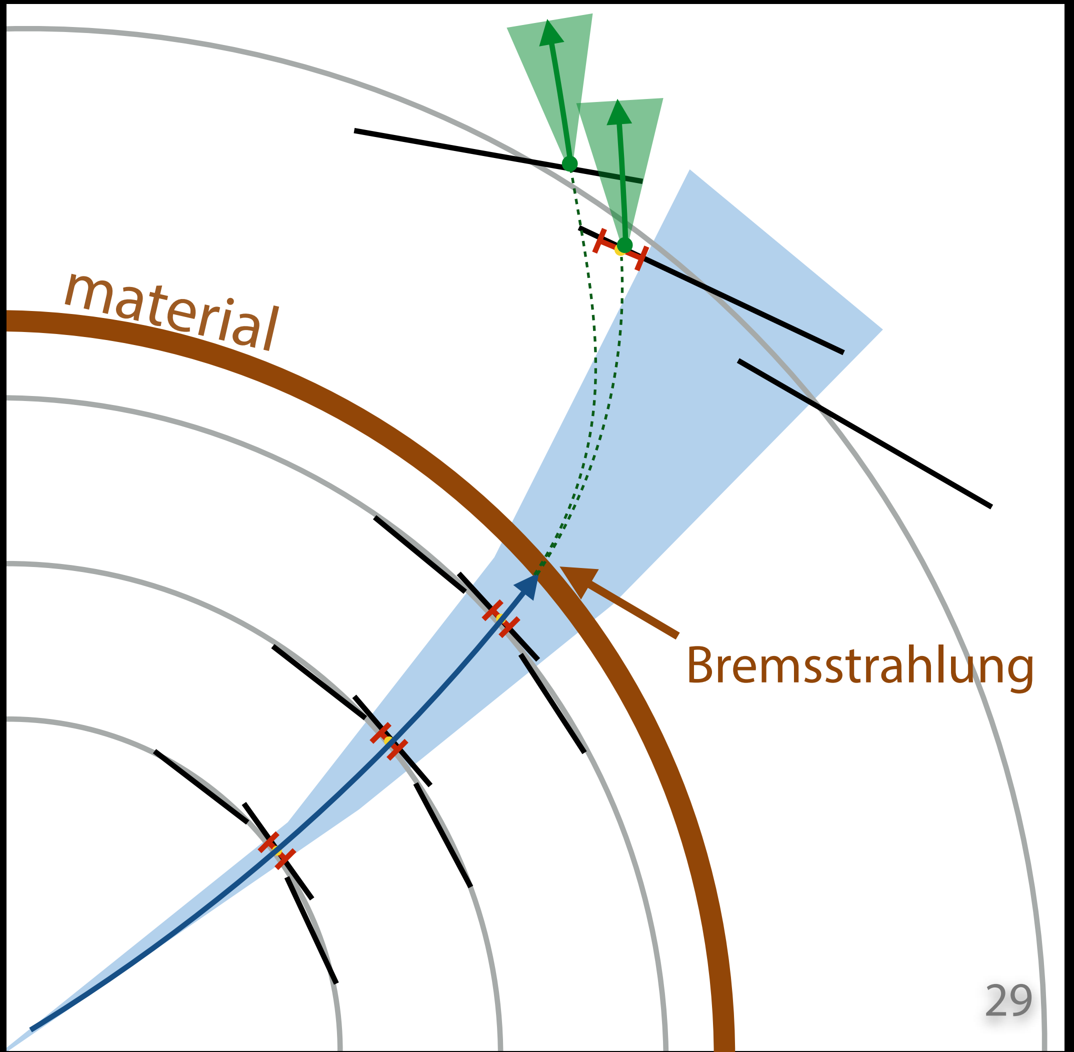


The Gaussian Sum Filter

- approximate Bethe-Heitler distribution as Gaussian mixture
  - ➡ state vector after material correction becomes sum of Gaussian components
    - relative weights from Bethe-Heitler distribution
    - GSF step resembles set of parallel Kalman Filters
    - computationally expensive !
  - ➡ component reduction to avoid combinatorial explosion after several material layers
    - re-evaluate weights of components based on compatibility with hits
    - drop components with too low weights
  - ➡ GSF improves fit performance w.r.t. Kalman Filter

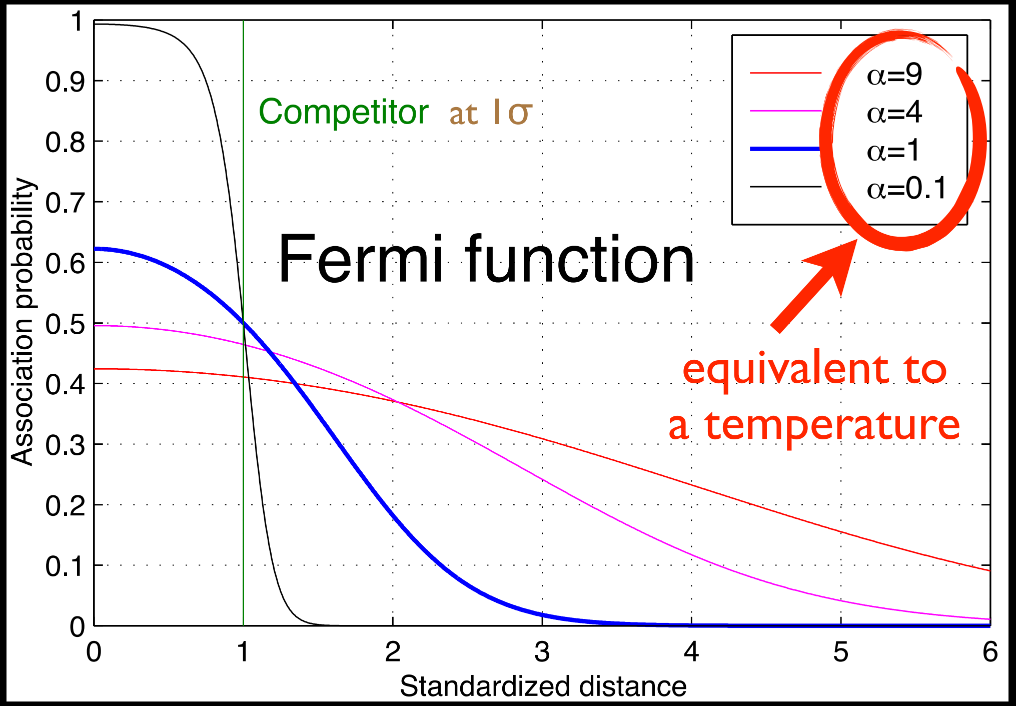






Deterministic Annealing Filters

- robust technique
  - ➡ developed for fitting with high occupancies
    - e.g. ATLAS TRT with high event pileup
    - reconstruction of 3-prong τ decays
  - ➡ can deal with several close by hits on a layer



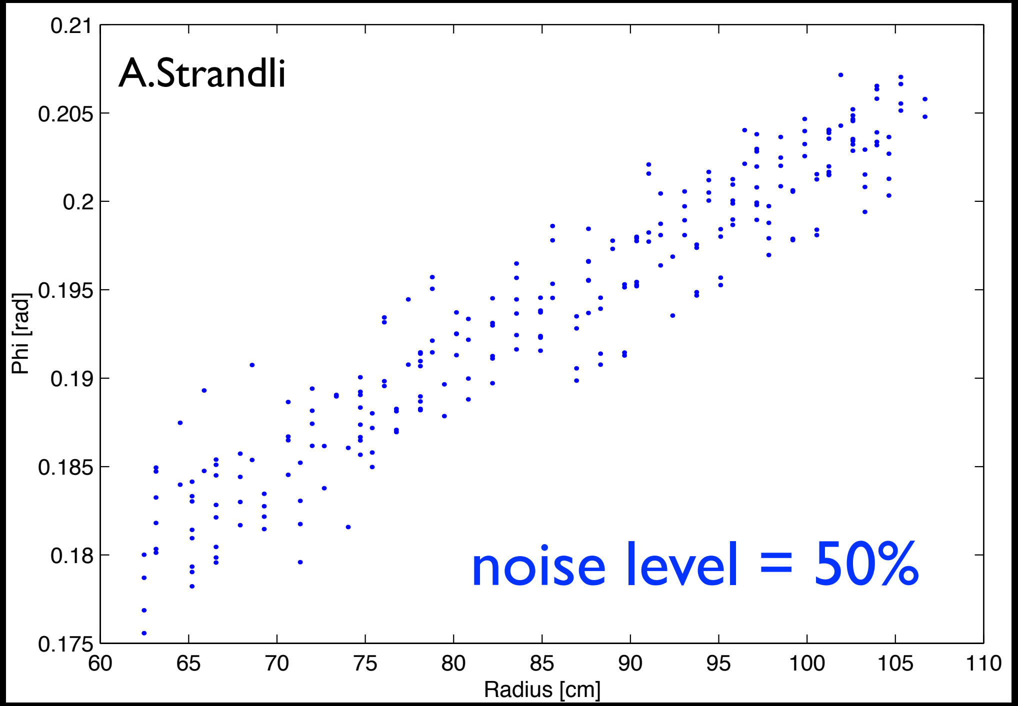
- adaptive fit
  - ➡ multiply weight of each hit in layer with assignment probability:

$$p_{ik} = \frac{\exp\left(-\hat{d}_{ik}^2/T\right)}{\sum_{j=1}^{n_k} \exp\left(-\hat{d}_{jk}^2/T\right)}$$

Boltzman factor

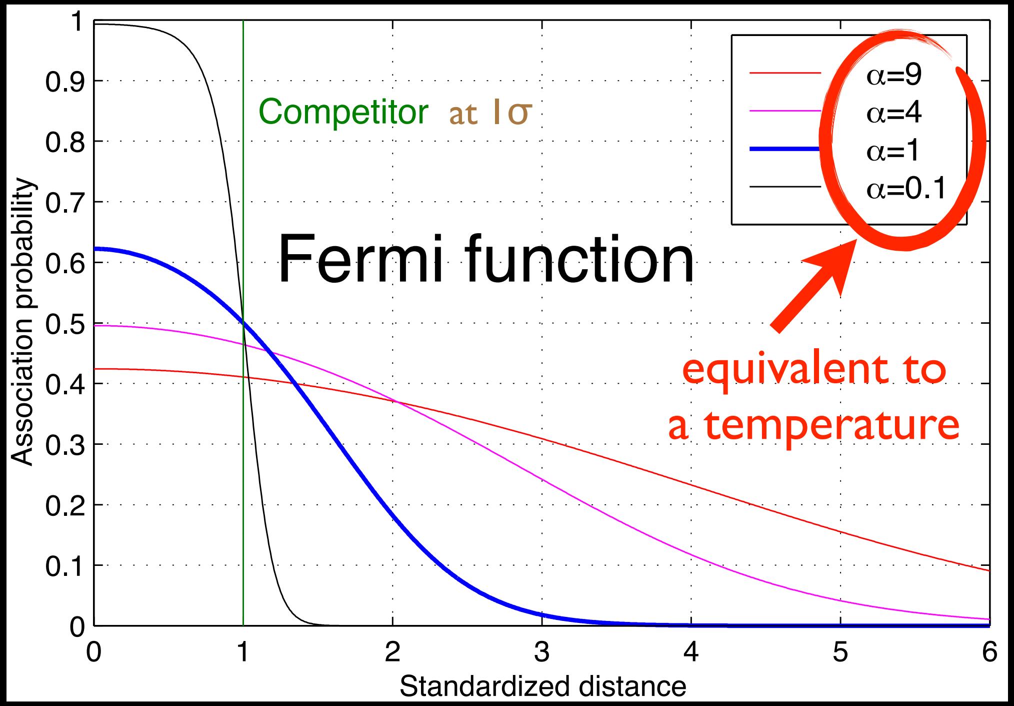
with: $\hat{d}_{ik} = d_{ik}/\sigma_k$

normalised distance



Deterministic Annealing Filters

- robust technique
  - ➡ developed for fitting with high occupancies
    - e.g. ATLAS TRT with high event pileup
    - reconstruction of 3-prong τ decays
  - ➡ can deal with several close by hits on a layer



- adaptive fit
  - ➡ multiply weight of each hit in layer with assignment probability:

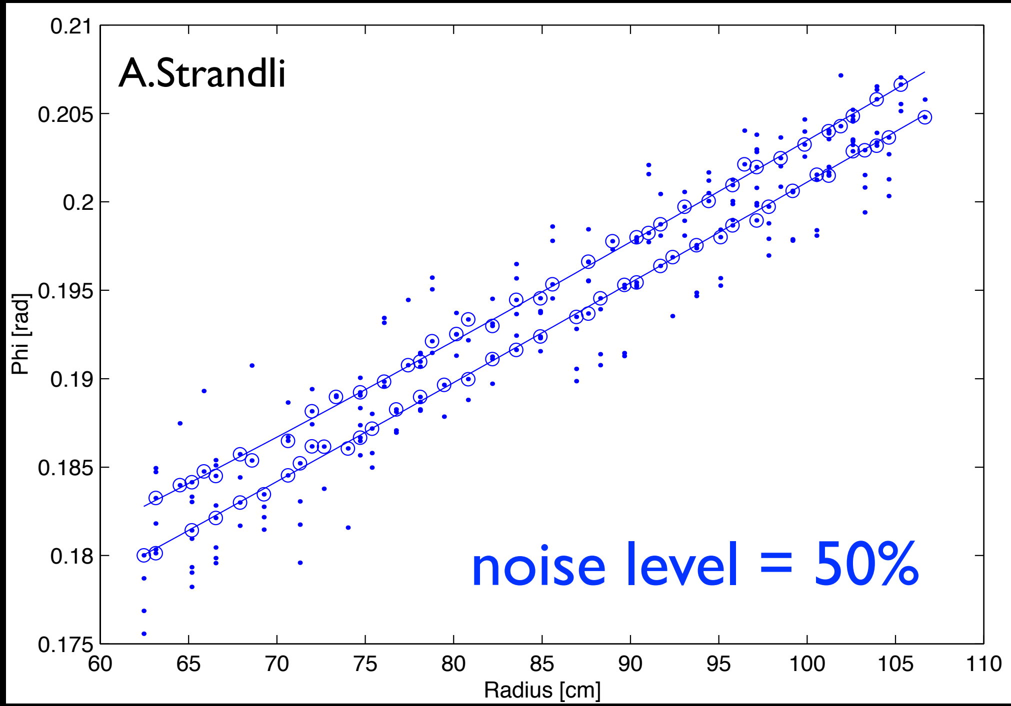
$$p_{ik} = \frac{\exp\left(-\hat{d}_{ik}^2/T\right)}{\sum_{j=1}^{n_k} \exp\left(-\hat{d}_{jk}^2/T\right)}$$

Boltzman factor

with: $\hat{d}_{ik} = d_{ik}/\sigma_k$

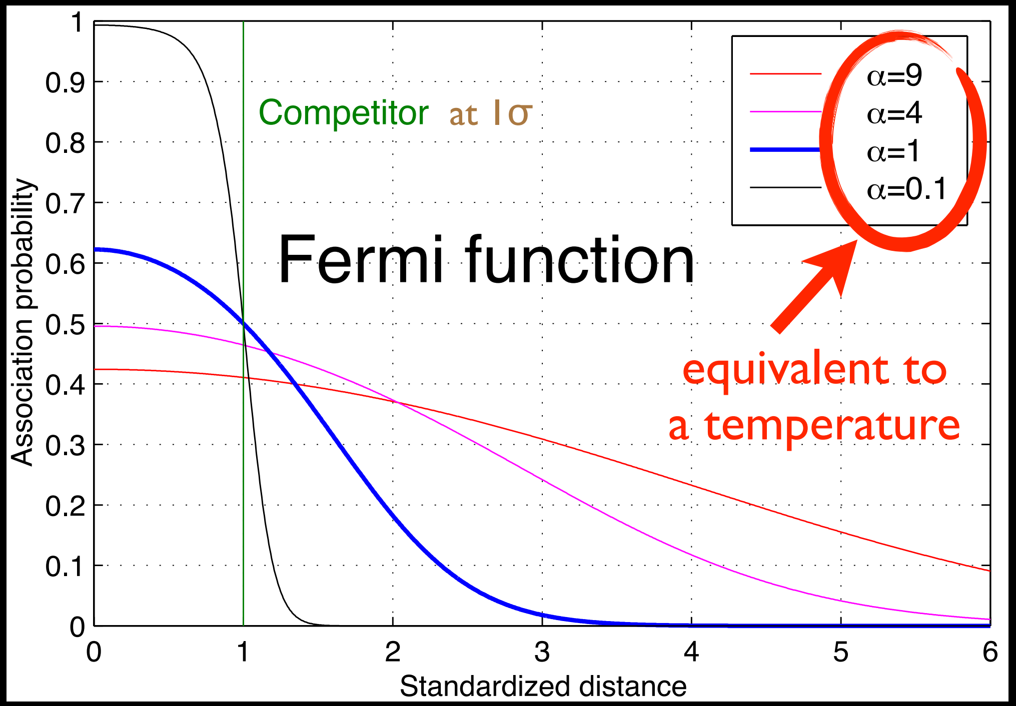
normalised distance

  - ➡ process decreasing temperature T is called annealing (iterative)
    - start at high T ~ all hits contribute same
    - at low T ~ close by hits remain



Deterministic Annealing Filters

- **robust technique**
  - ➡ developed for fitting with high occupancies
    - e.g. ATLAS TRT with high event pileup
    - reconstruction of 3-prong τ decays
  - ➡ can deal with several close by hits on a layer



- **adaptive fit**
  - ➡ multiply weight of each hit in layer with assignment probability:

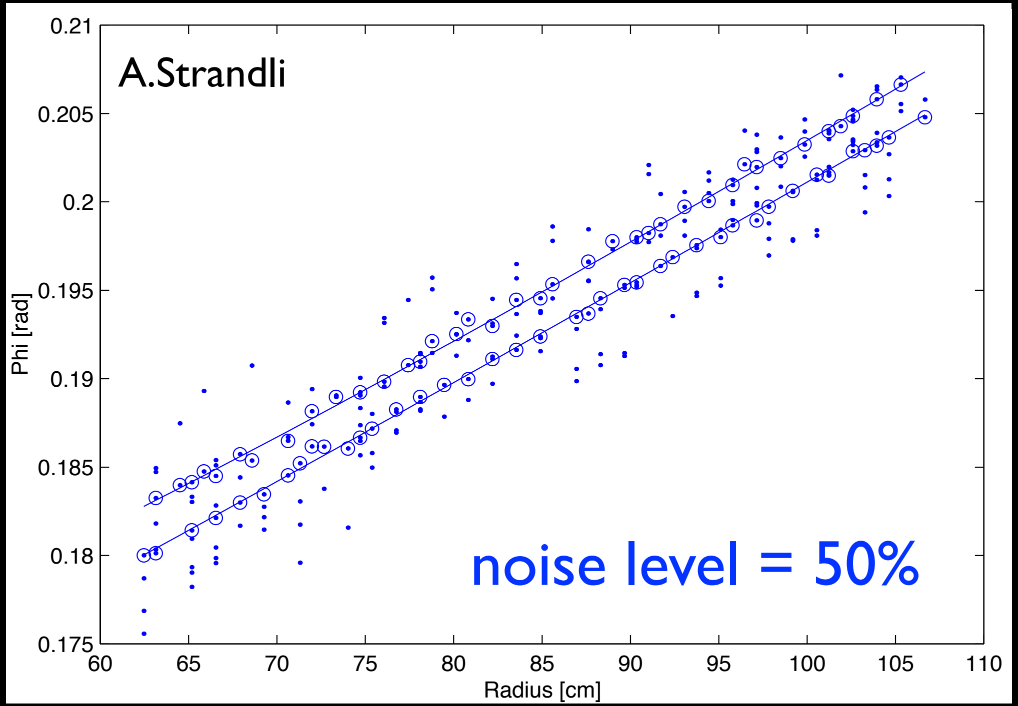
$$p_{ik} = \frac{\exp\left(-\hat{d}_{ik}^2/T\right)}{\sum_{j=1}^{n_k} \exp\left(-\hat{d}_{jk}^2/T\right)}$$

Boltzman factor

with: $\hat{d}_{ik} = d_{ik}/\sigma_k$

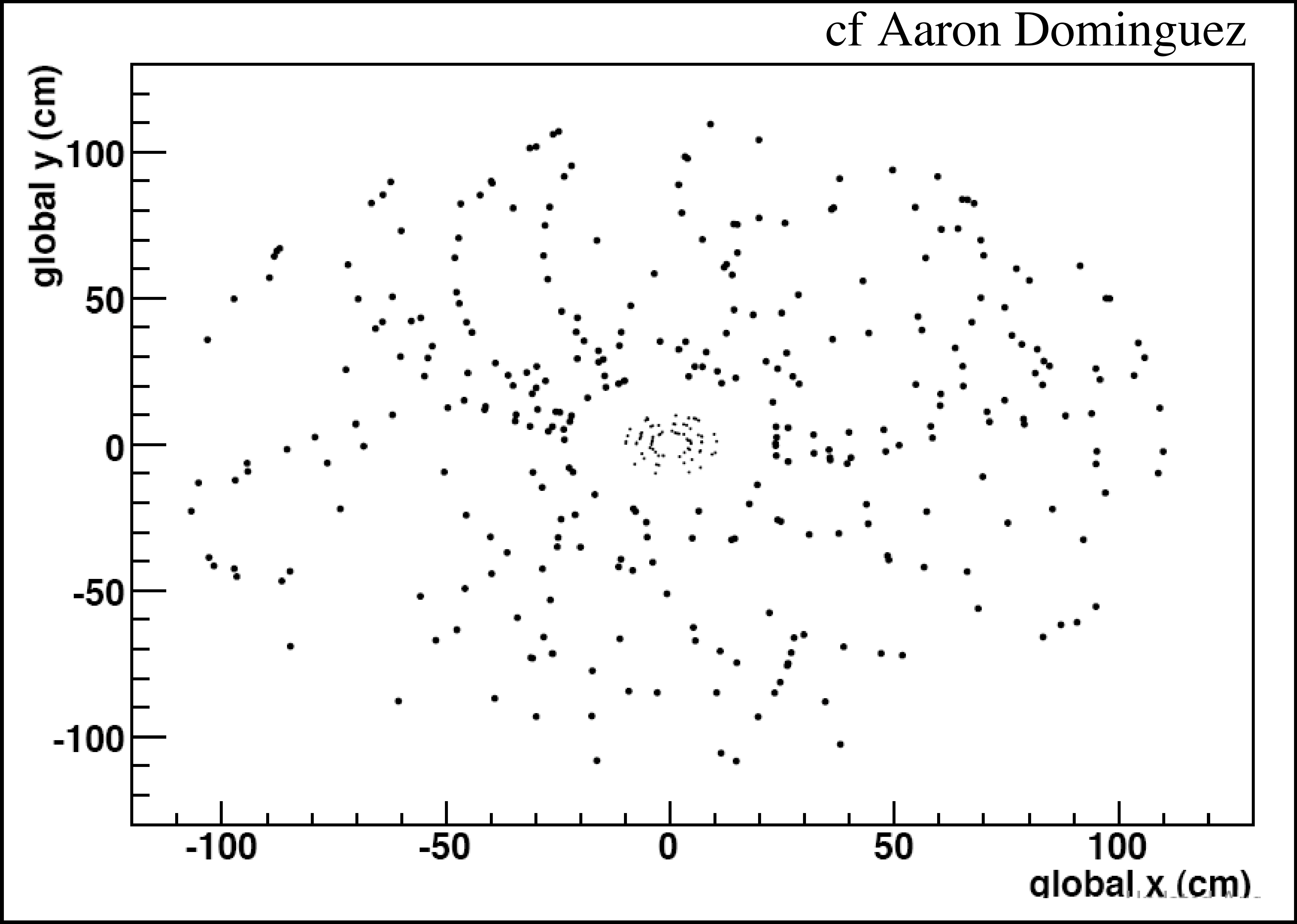
normalised distance

  - ➡ process decreasing temperature T is called annealing (iterative)
    - start at high T ~ all hits contribute same
    - at low T ~ close by hits remain
  - ➡ can be written as a Multi Track Filter

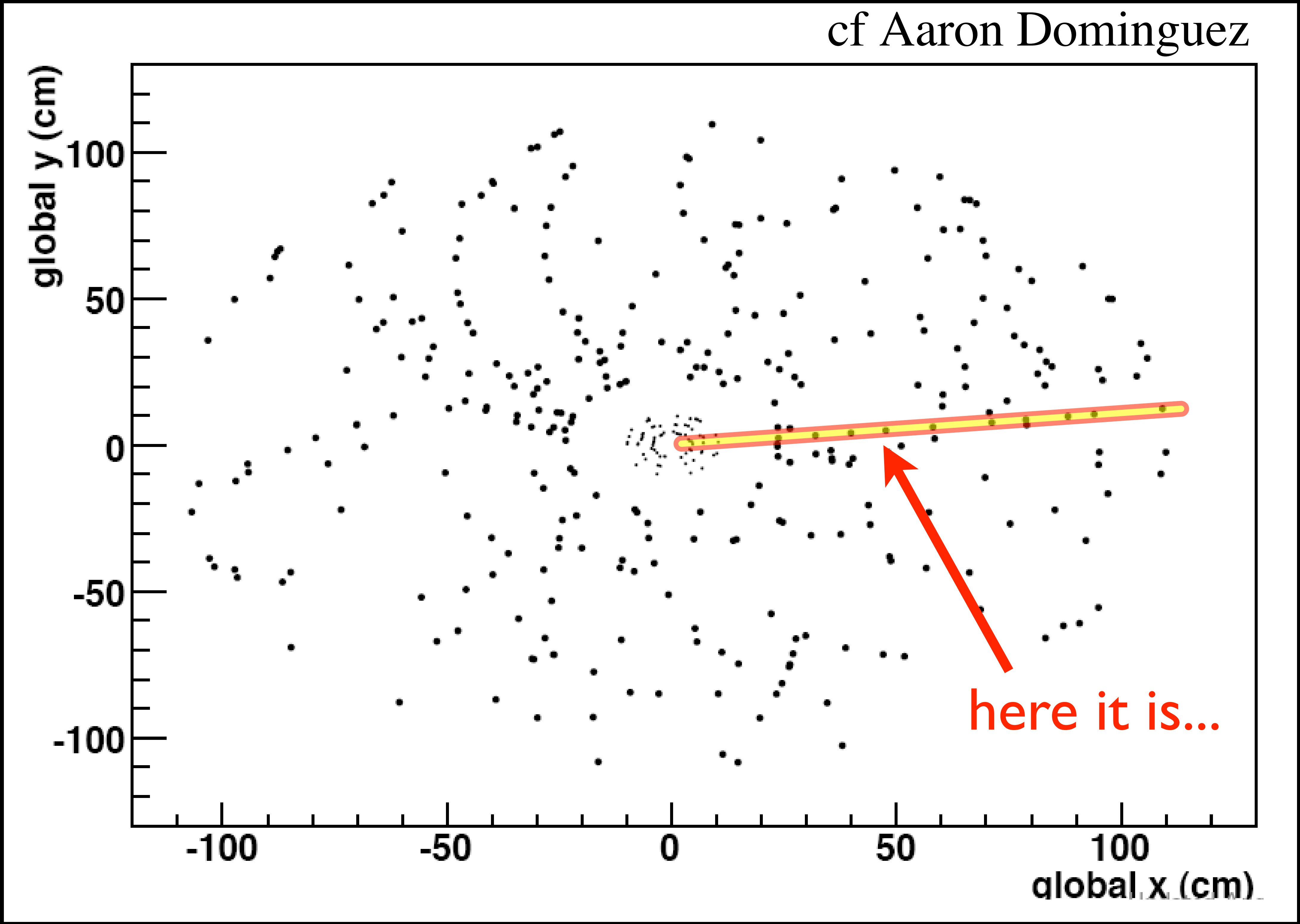


Track Finding

Track Finding: Can you find the 50 GeV track?

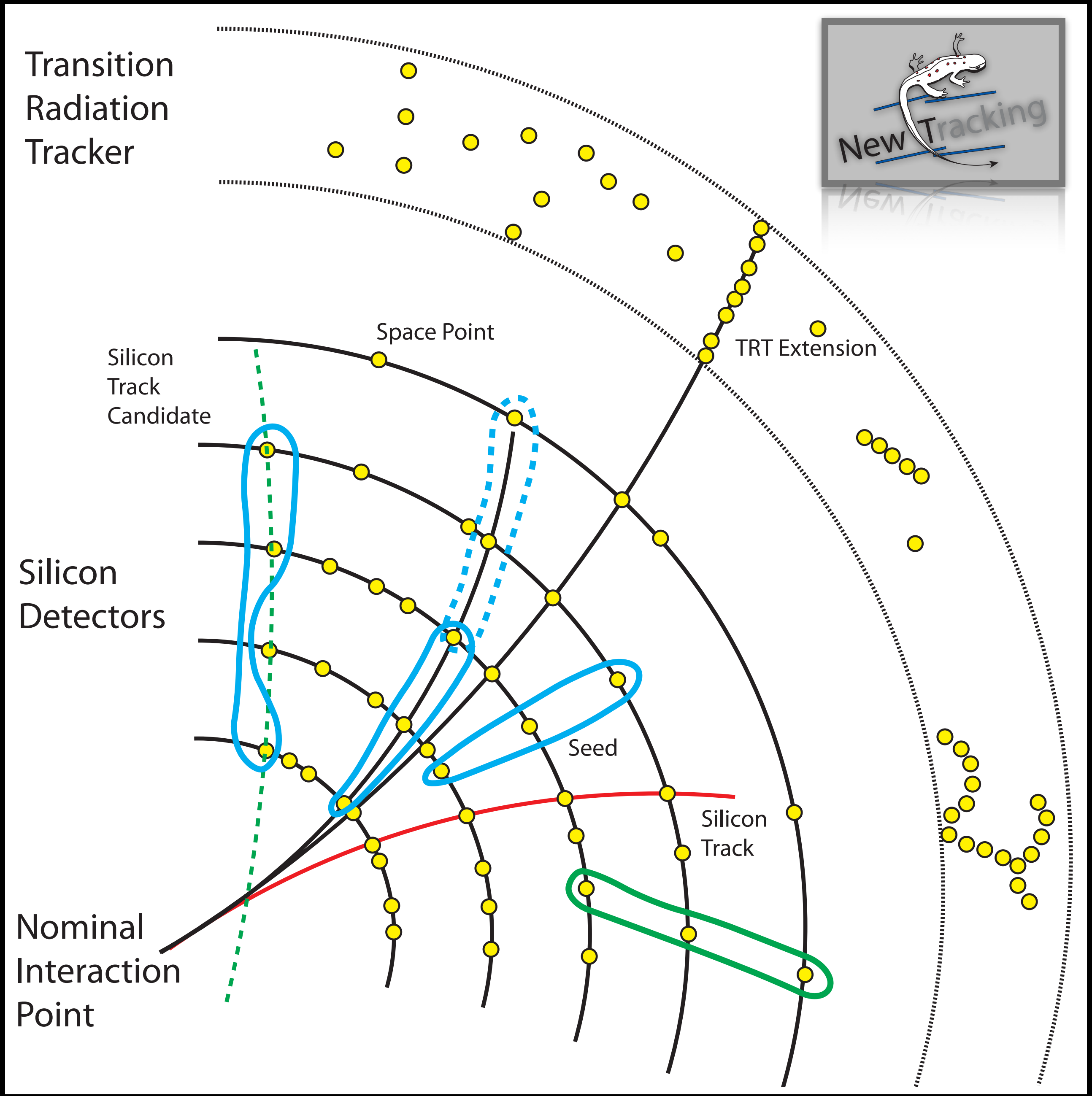


Track Finding: Can you find the 50 GeV track?



Track **Finding**

- the task of the track finding
  - ➡ identify **track candidates** in event
  - ➡ cope with the combinatorial explosion of possible **hit combinations**
- different techniques
  - ➡ rough distinction: **local/sequential** and **global/parallel** methods
  - ➡ local method: generate **seeds and complete** them to track candidates
  - ➡ global method: **simultaneous clustering** of detector hits into track candidates
- some **local** methods
  - ➡ track road
  - ➡ track following
  - ➡ progressive track finding



- some **global** methods
  - ➡ conformal mapping
    - Hough and Legendre transform
  - ➡ adaptive methods
    - Elastic Net, Cellular Automaton ...

      (will not discuss the latter)

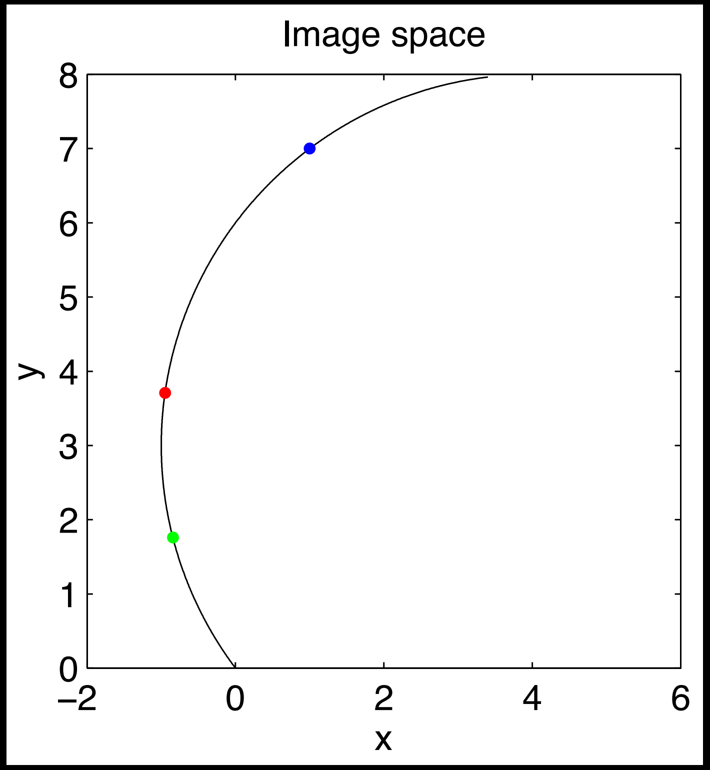
Conformal Mapping

- **Hough transform**
  - ➡ cycles through the origin in x-y transform into point in u-v

$$u = \frac{x}{x^2 + y^2}, \quad v = \frac{y}{x^2 + y^2}$$

$$\Rightarrow v = -\frac{x}{y}u + \frac{x^2 + y^2}{2y}$$

  - each hit becomes a straight line



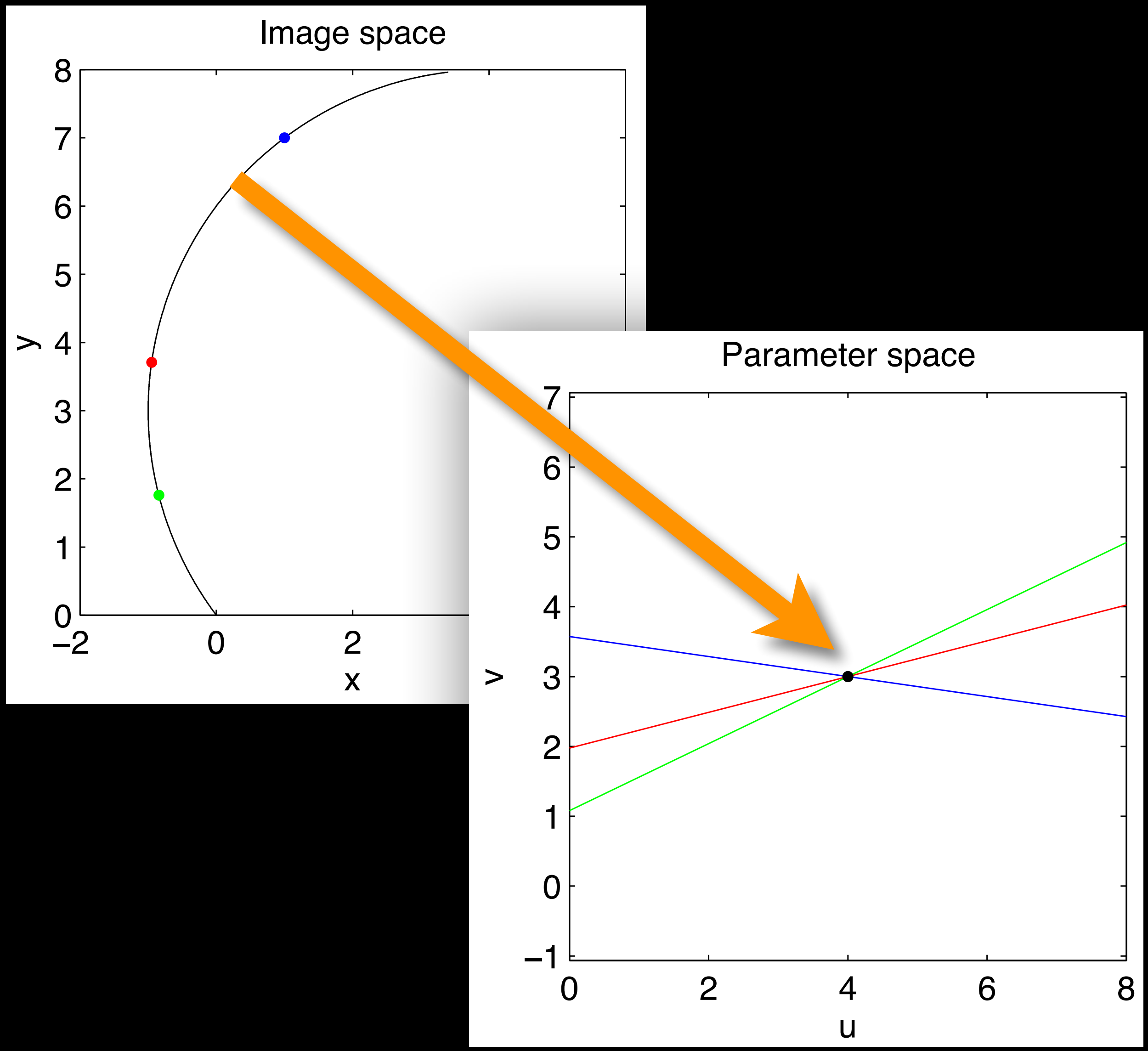
Conformal Mapping

- **Hough transform**
  - ➡ cycles through the origin in x-y transform into point in u-v

$$u = \frac{x}{x^2 + y^2}, \quad v = \frac{y}{x^2 + y^2}$$

$$\Rightarrow v = -\frac{x}{y} u + \frac{x^2 + y^2}{2y}$$

  - • each hit becomes a straight line
  - ➡ search for maxima (histogram) in **parameter space** to find track candidates



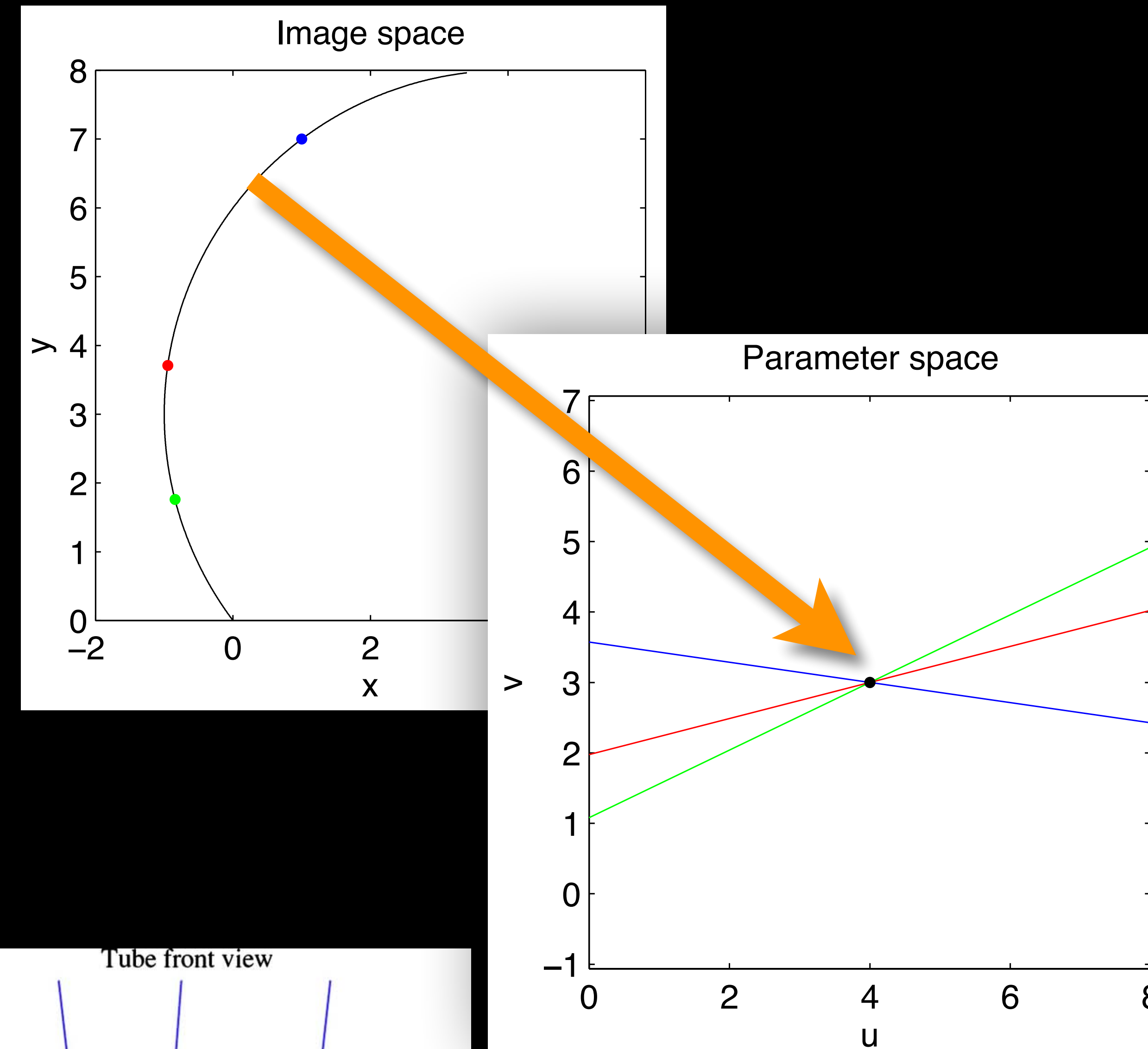
Conformal Mapping

- **Hough transform**
  - ➡ cycles through the origin in x-y transform into point in u-v

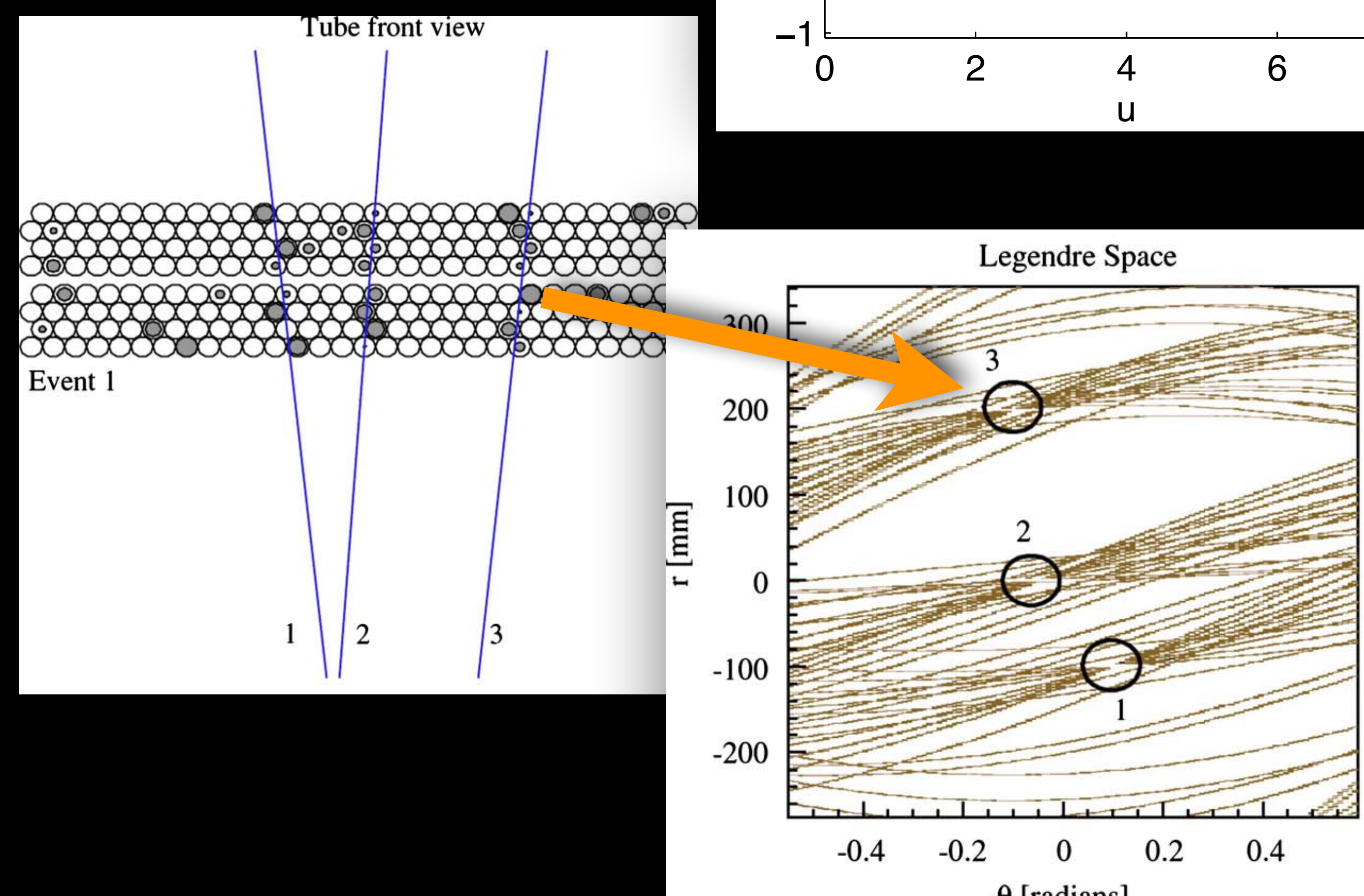
$$u = \frac{x}{x^2 + y^2}, \quad v = \frac{y}{x^2 + y^2}$$

$$\Rightarrow \quad v = -\frac{x}{y}u + \frac{x^2 + y^2}{2y}$$

  - • each hit becomes a straight line
  - ➡ search for maxima (histogram) in **parameter space** to find track candidates

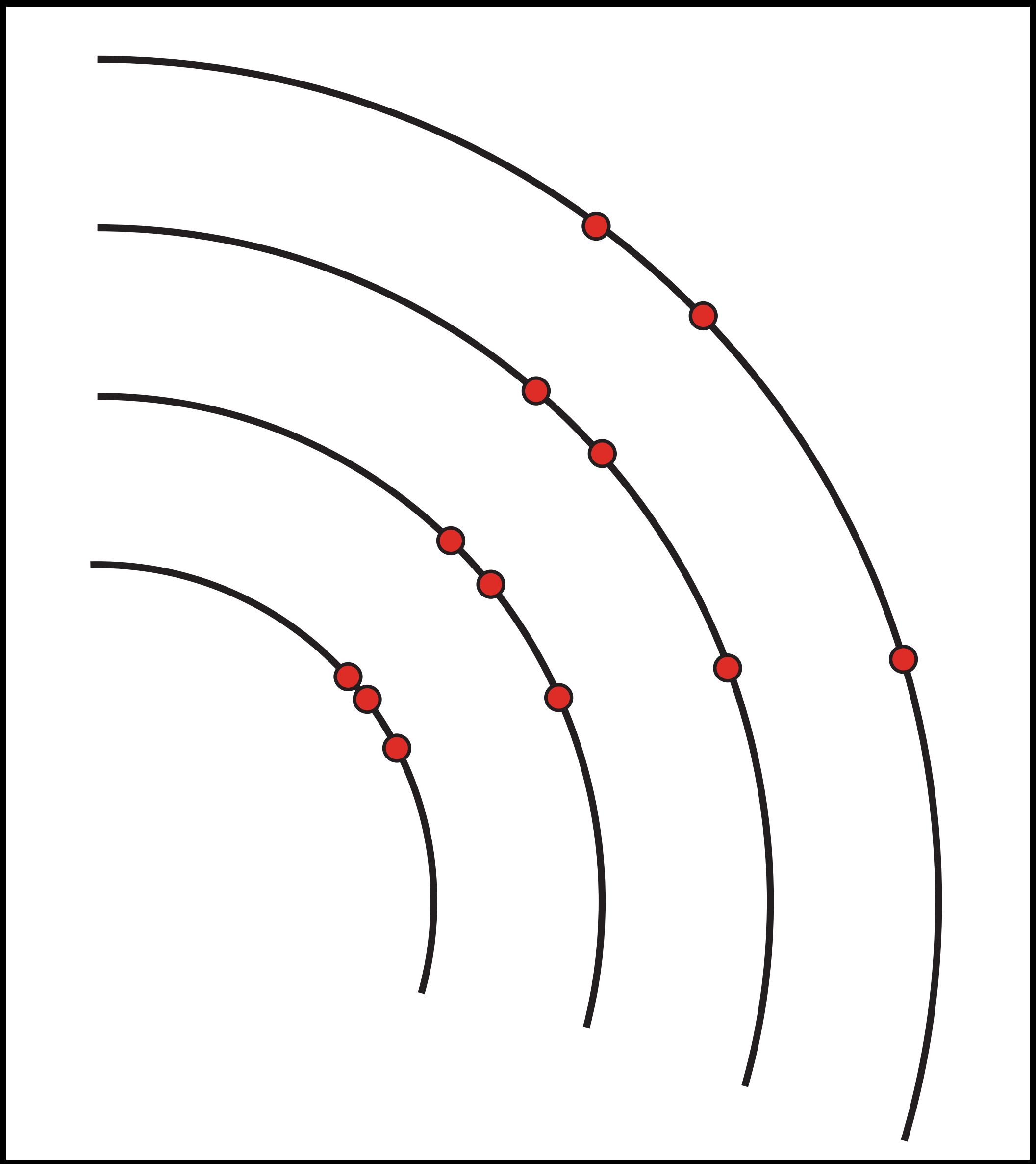


- **Legendre transform**
  - ➡ used for track finding in drift tubes
  - ➡ drift radius is transformed into sine-curves in **Legendre space**
  - ➡ solves as well L-R ambiguity



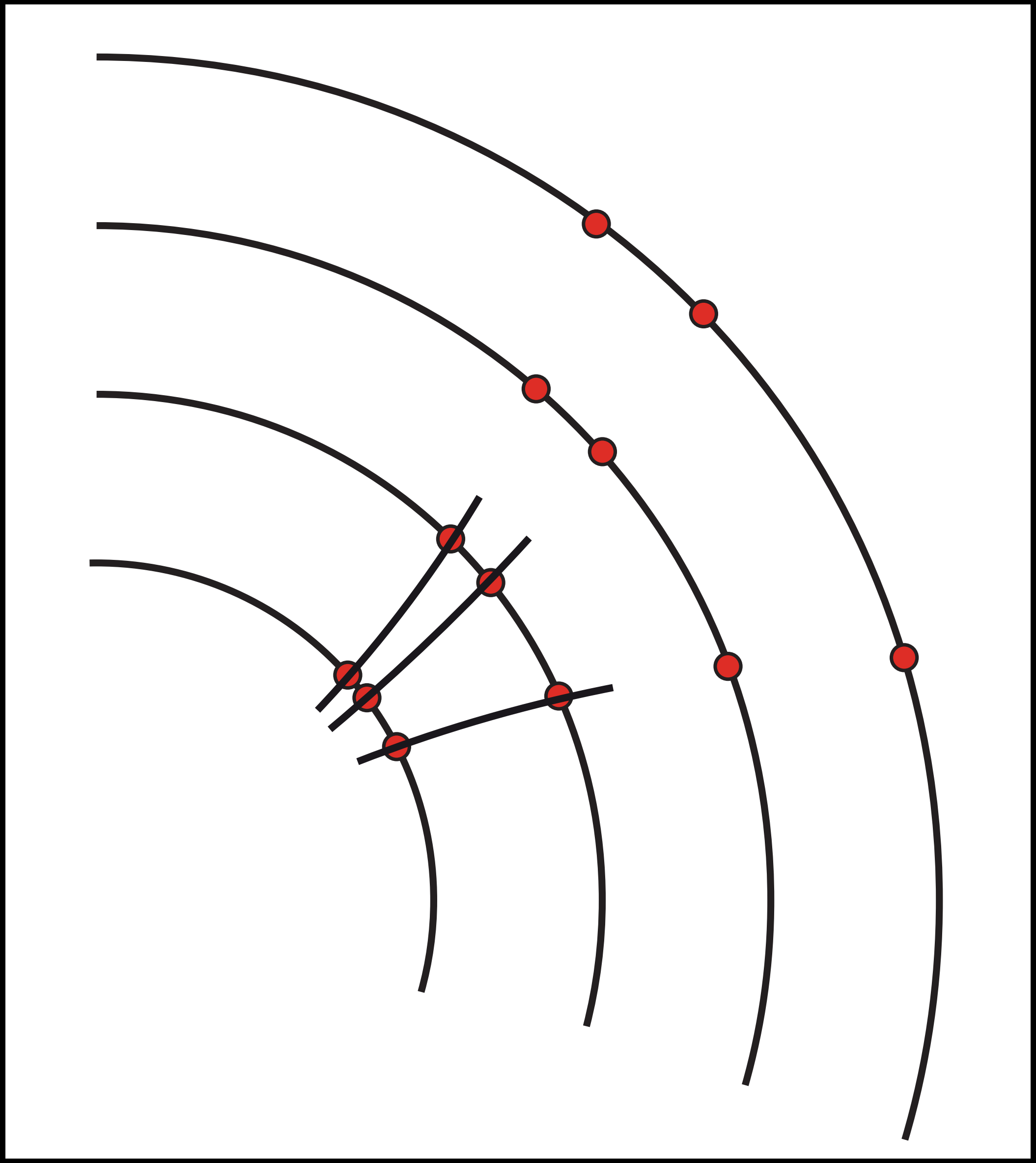
Local Track Finding

- Track Road algorithm



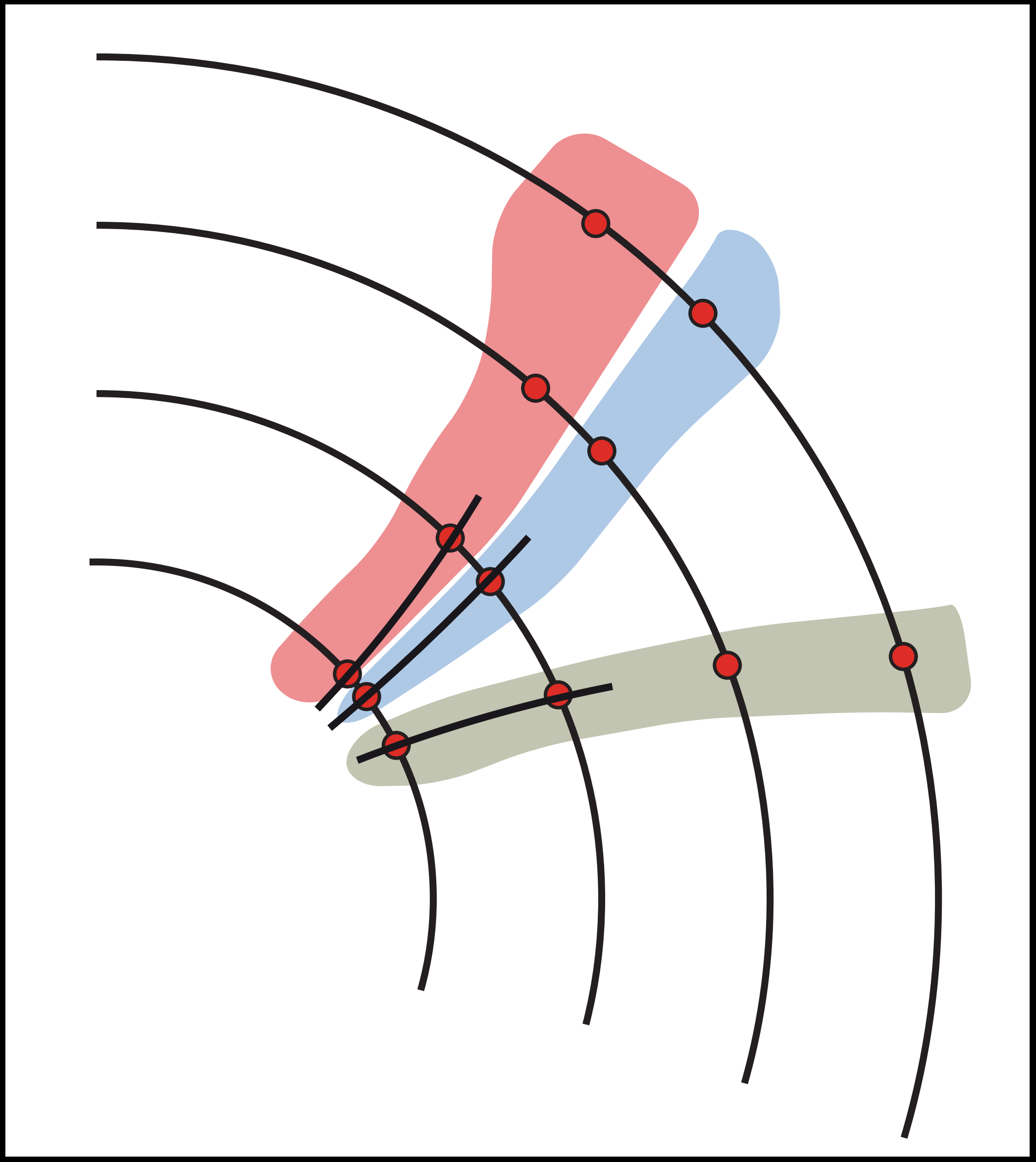
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits



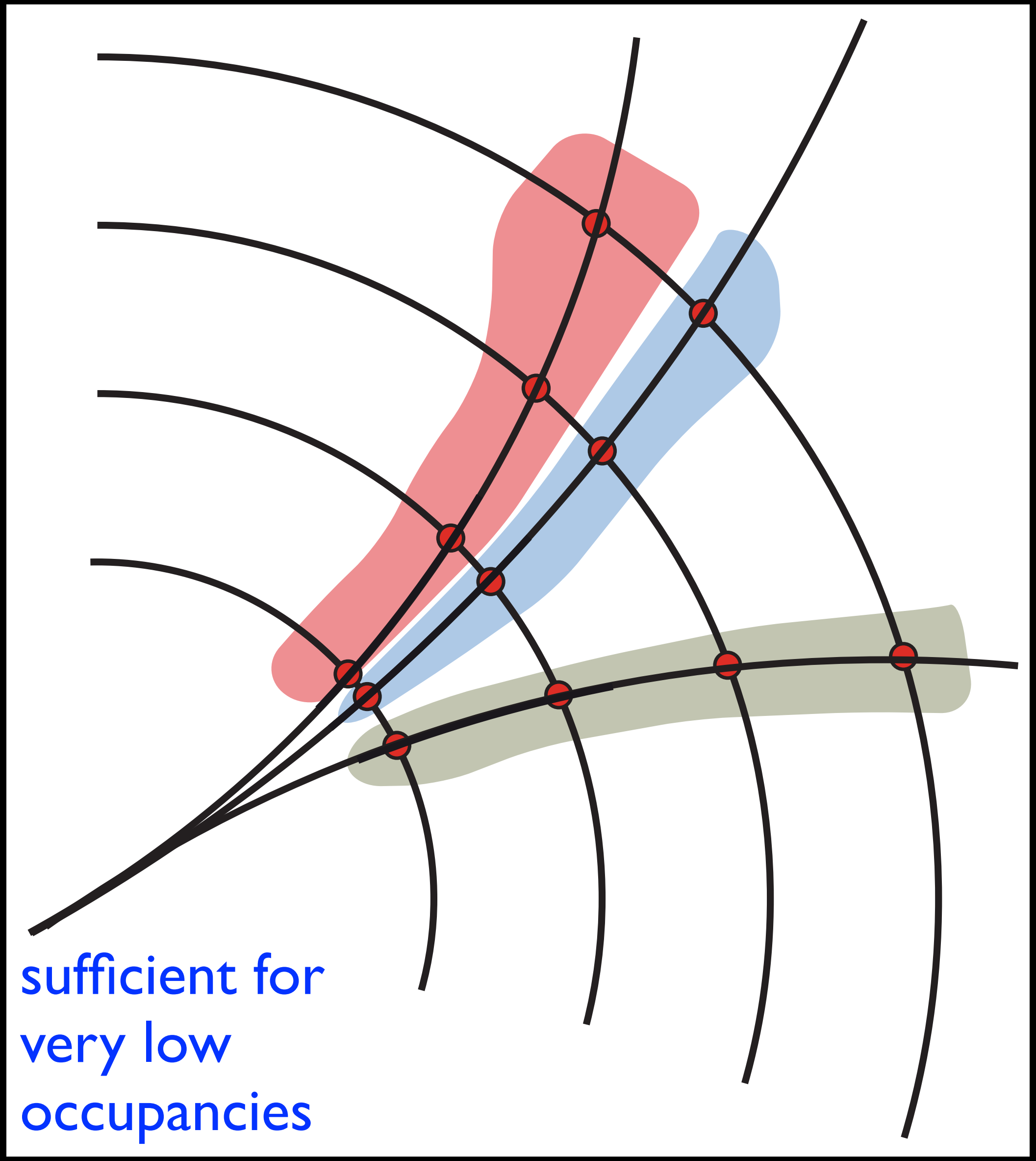
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory



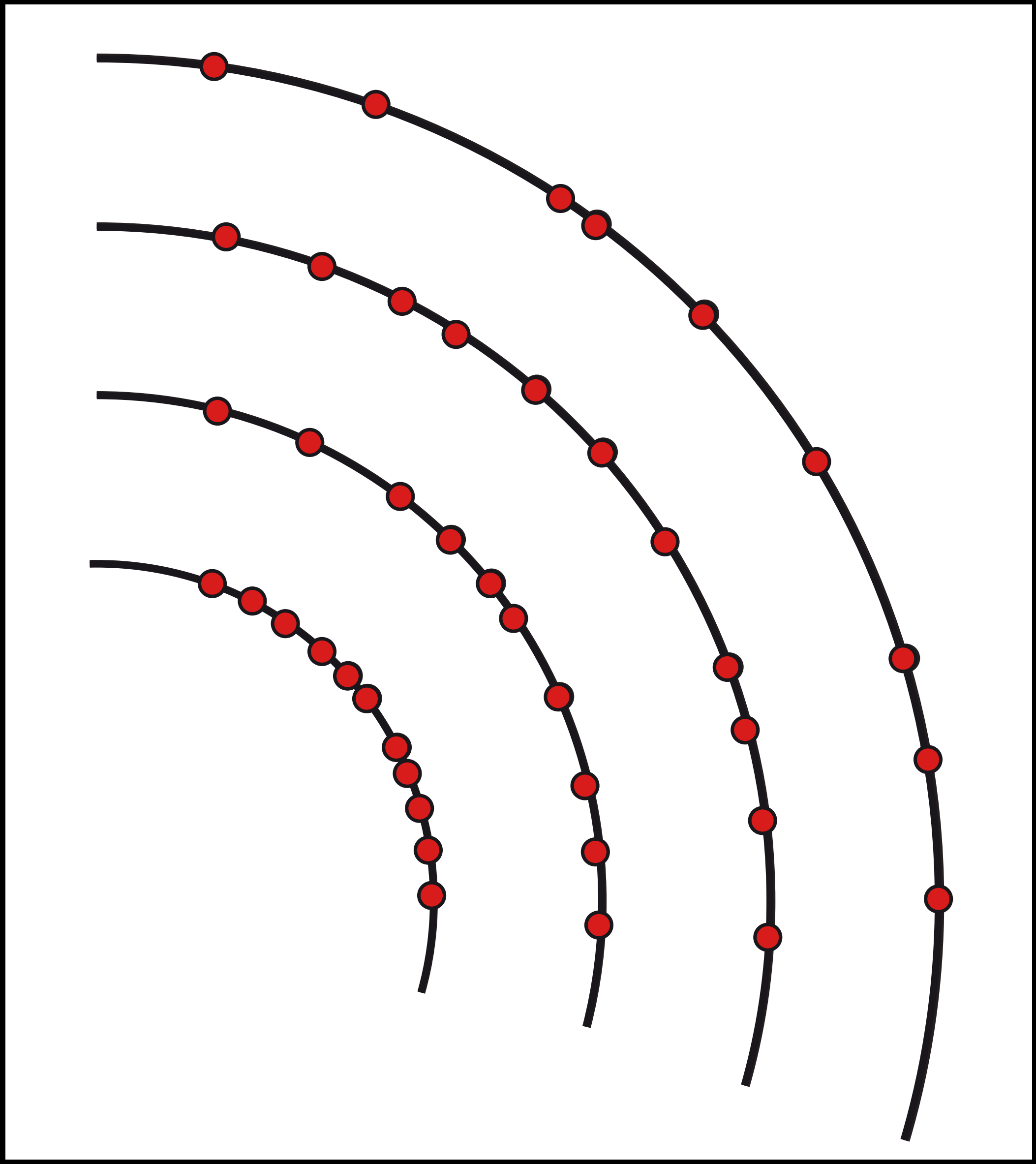
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**



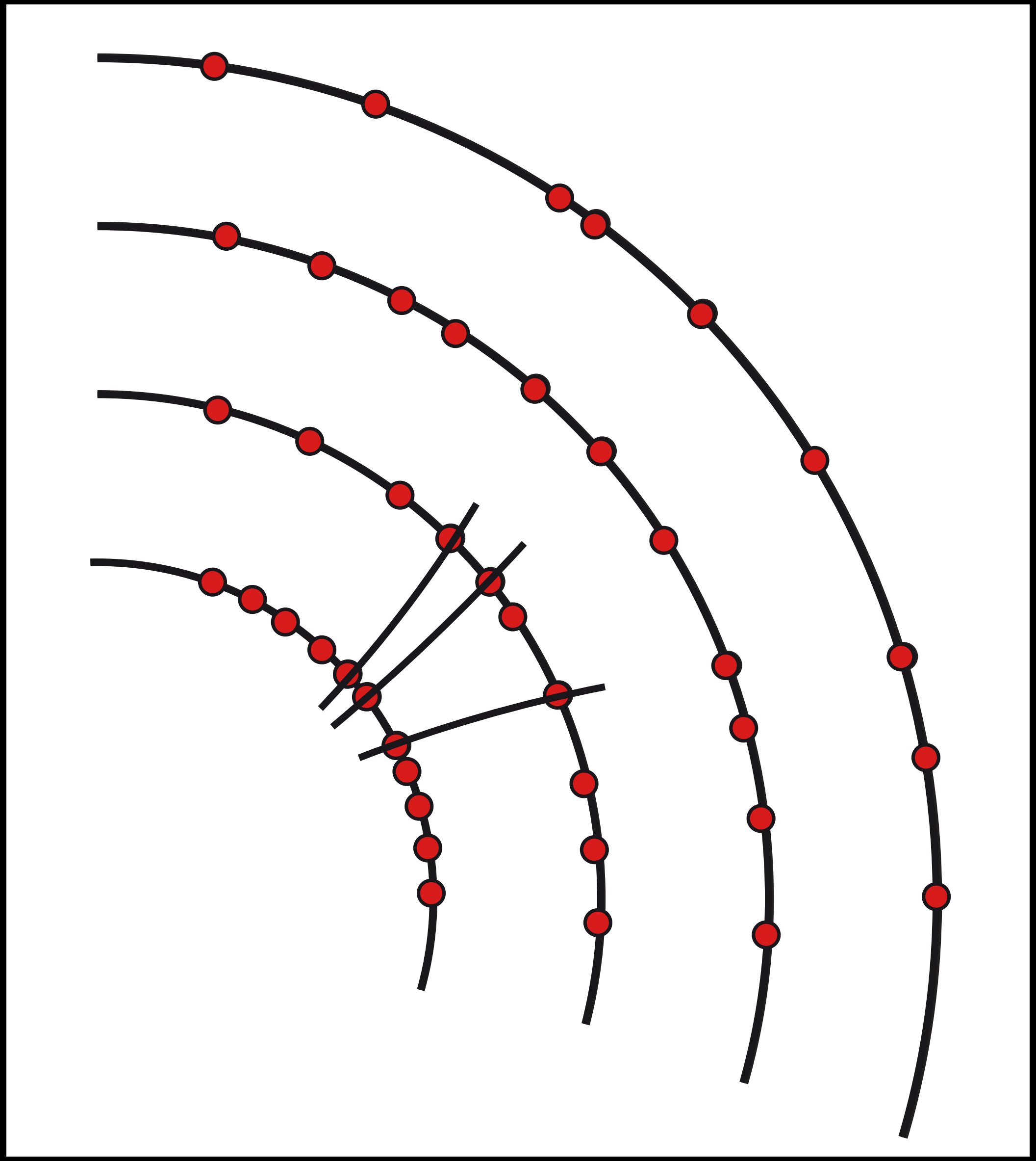
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following



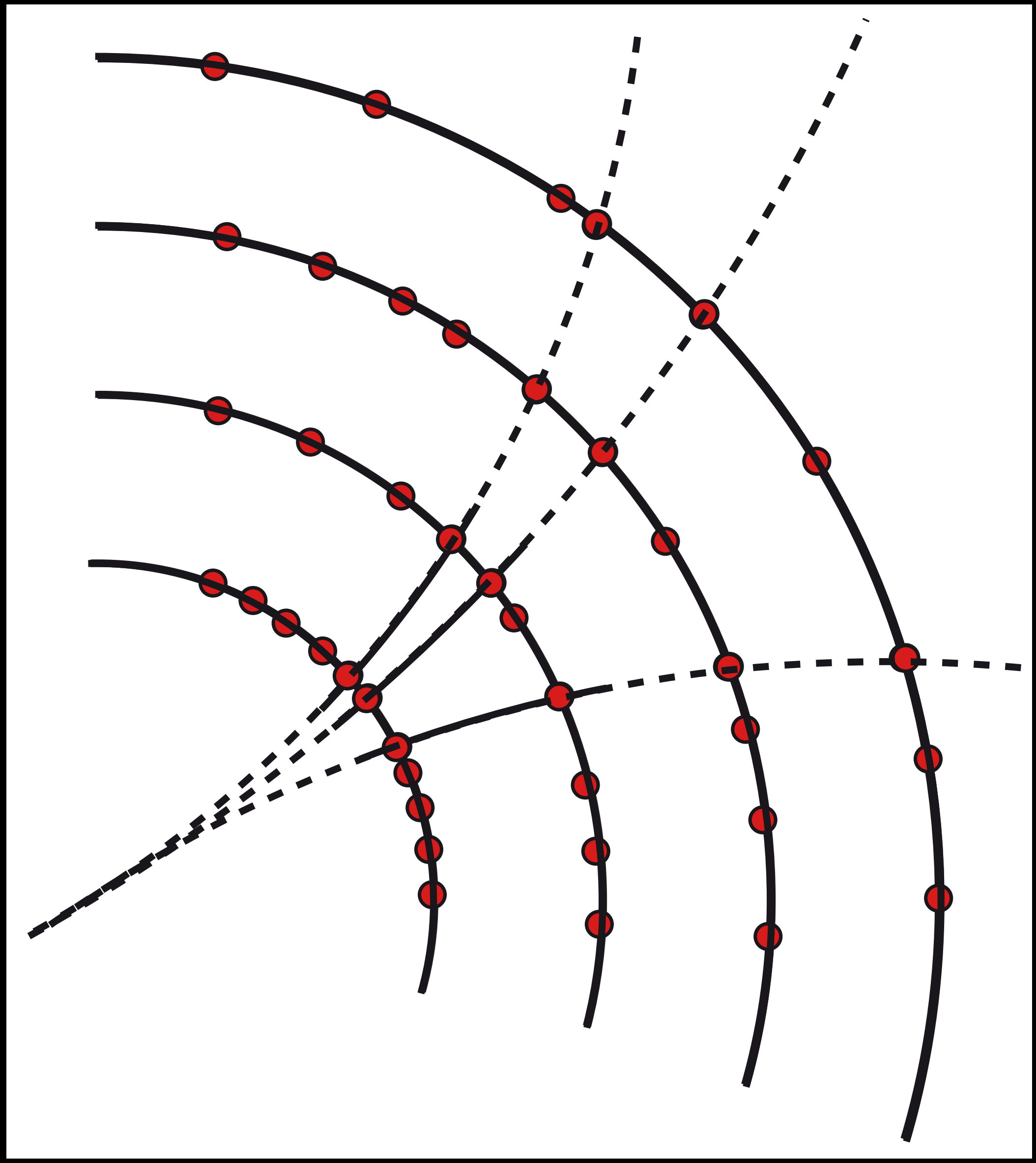
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits



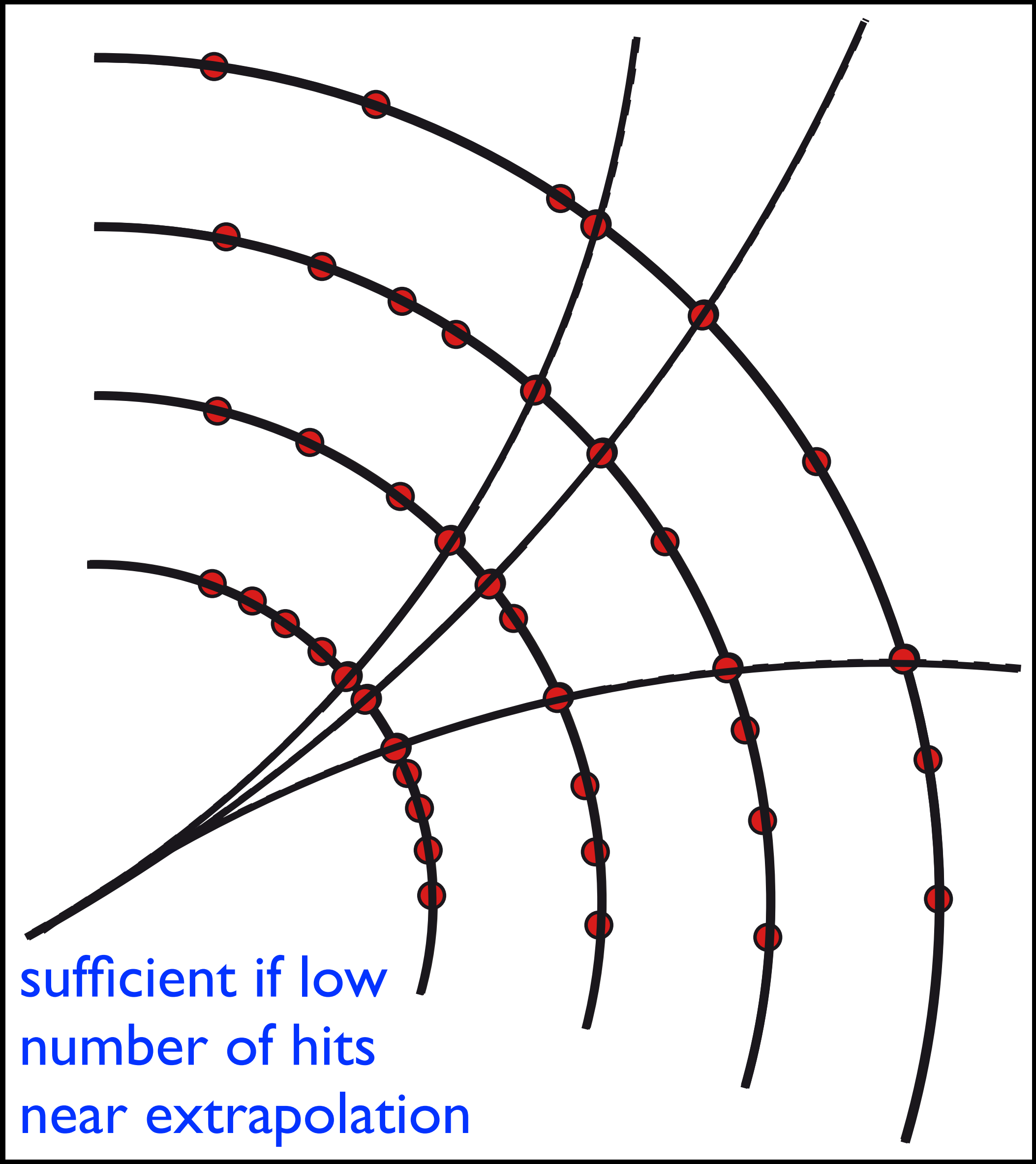
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory



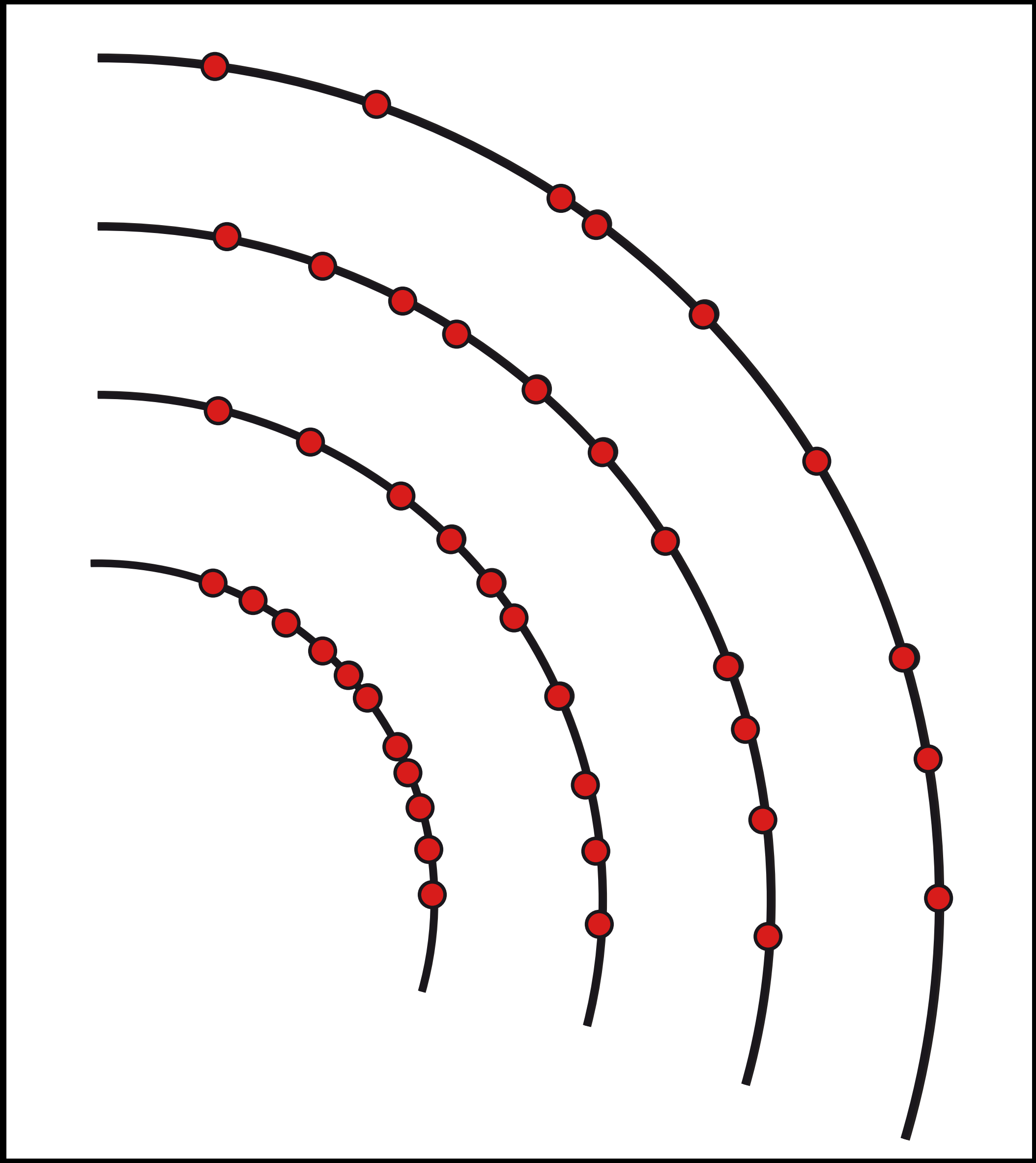
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**



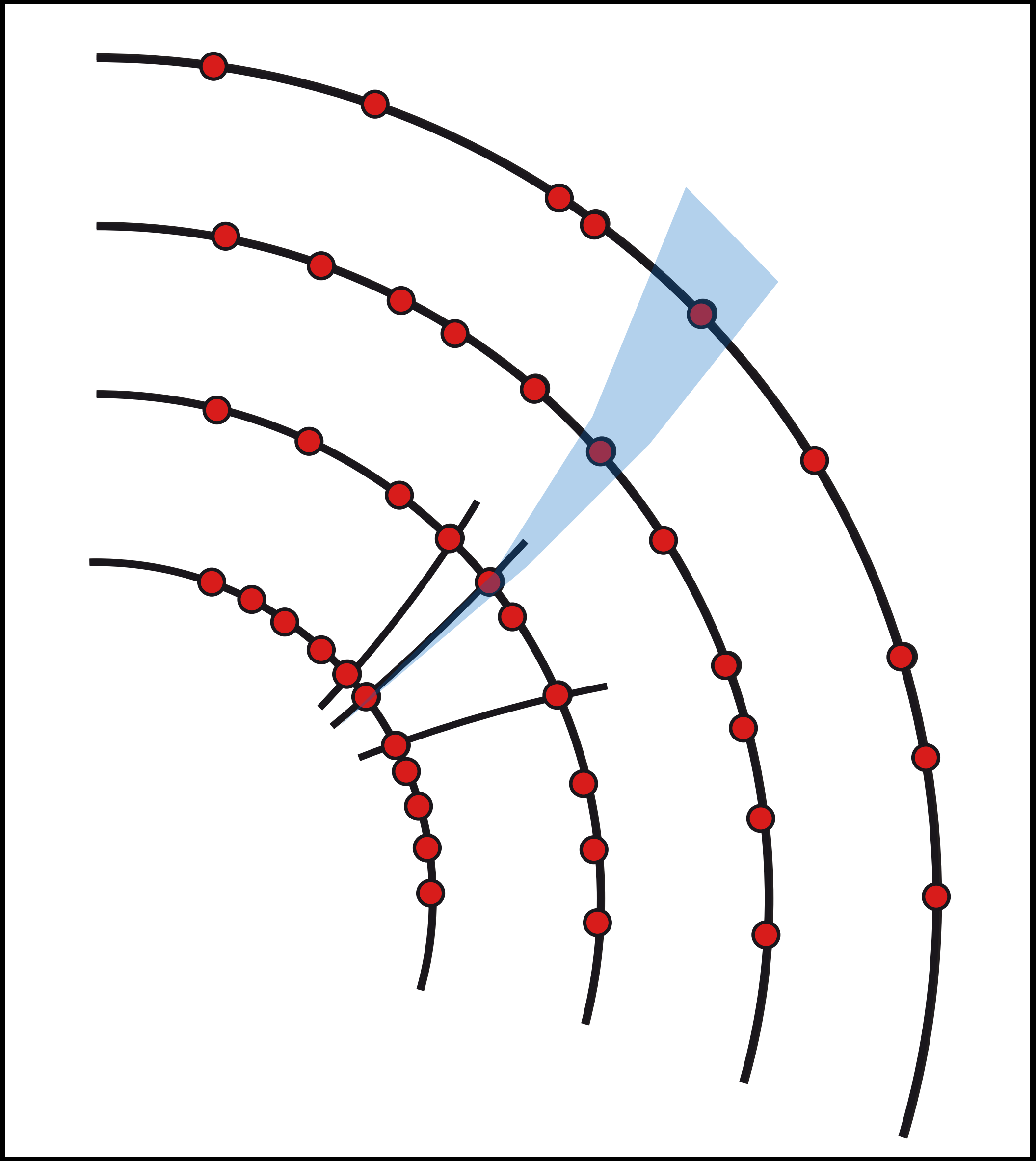
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder



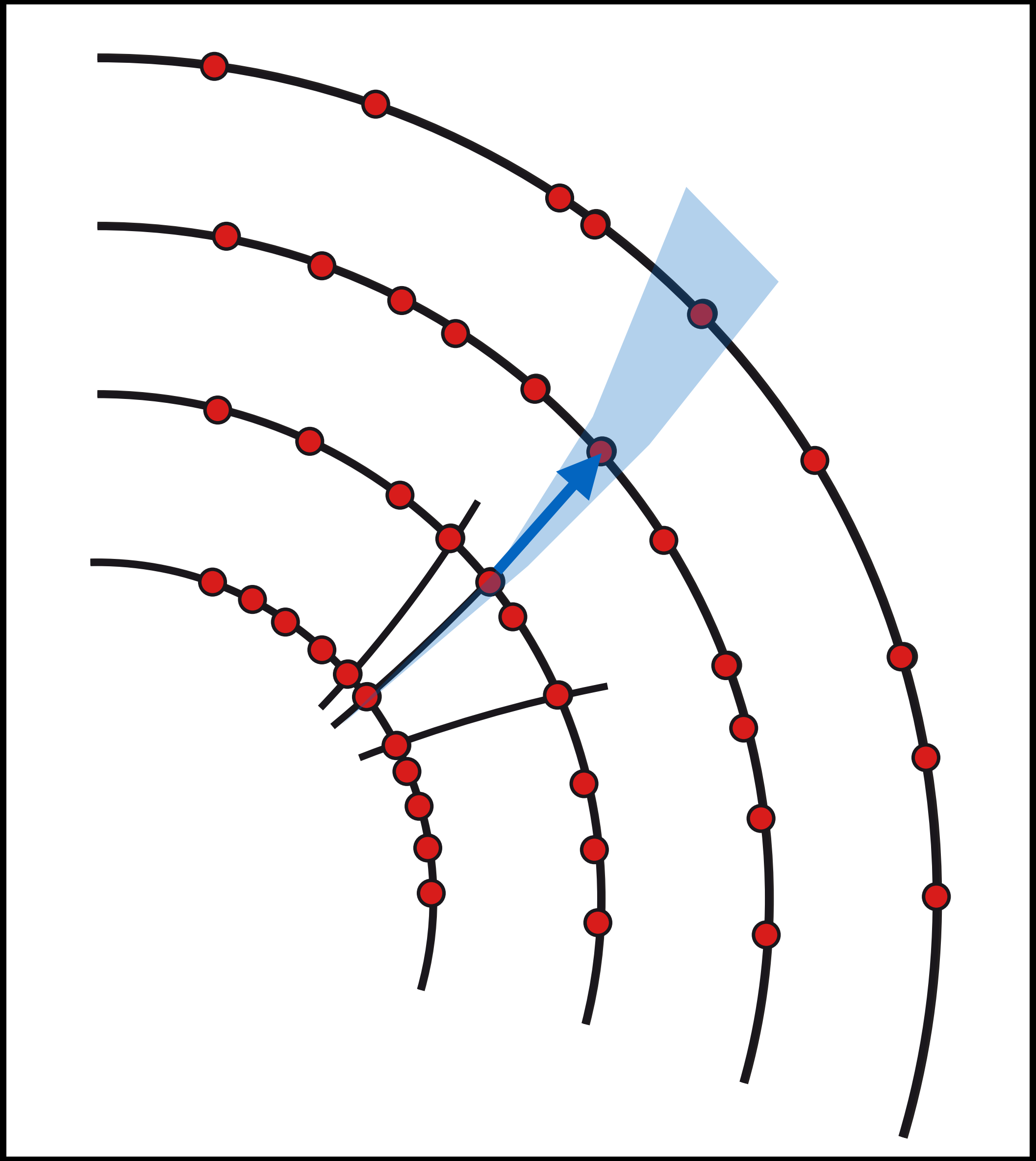
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder
  - ➡ find **seeds** ~ combinations of 2-3 hits



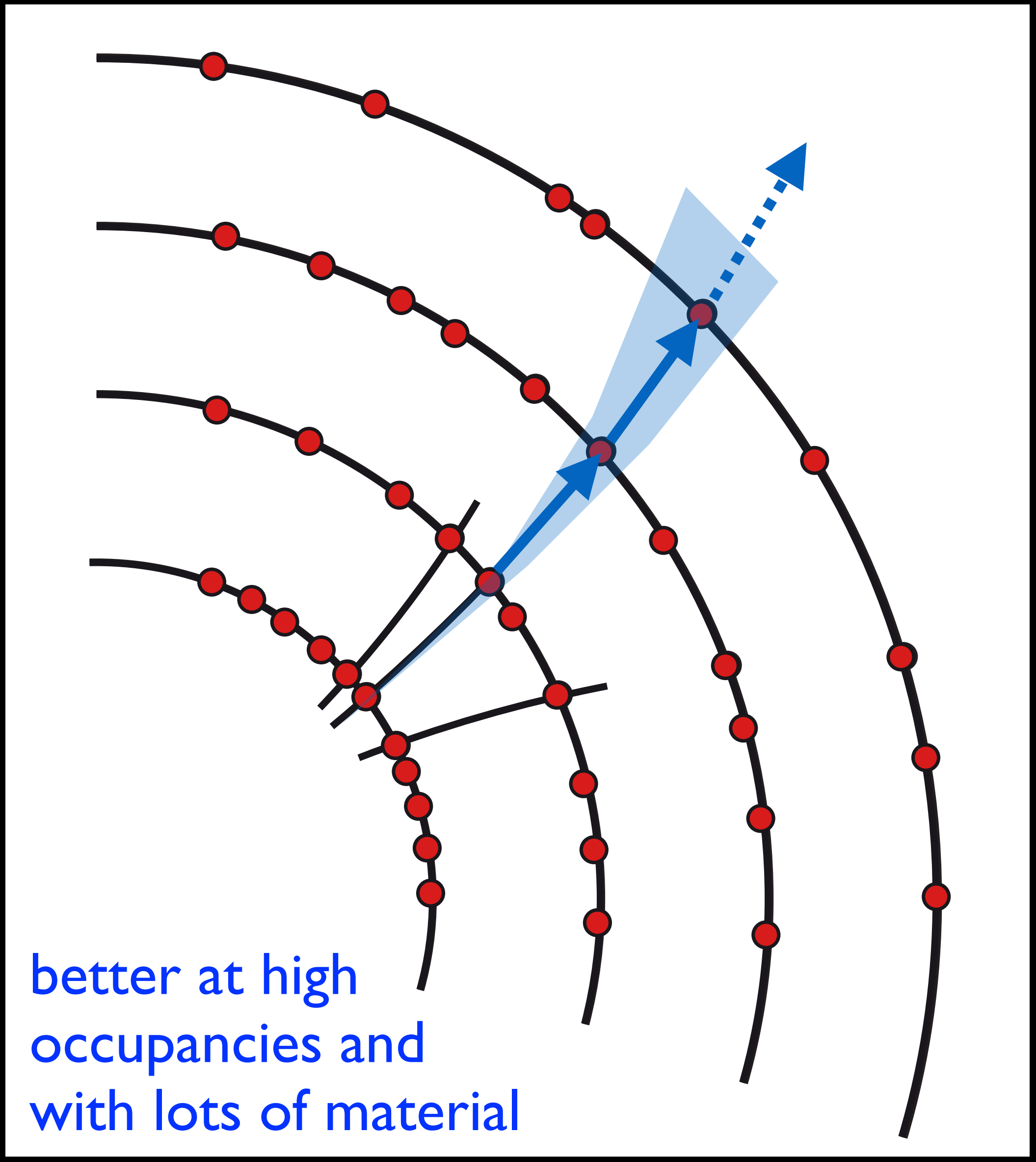
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** to next layer, find **best hit** and **update** trajectory



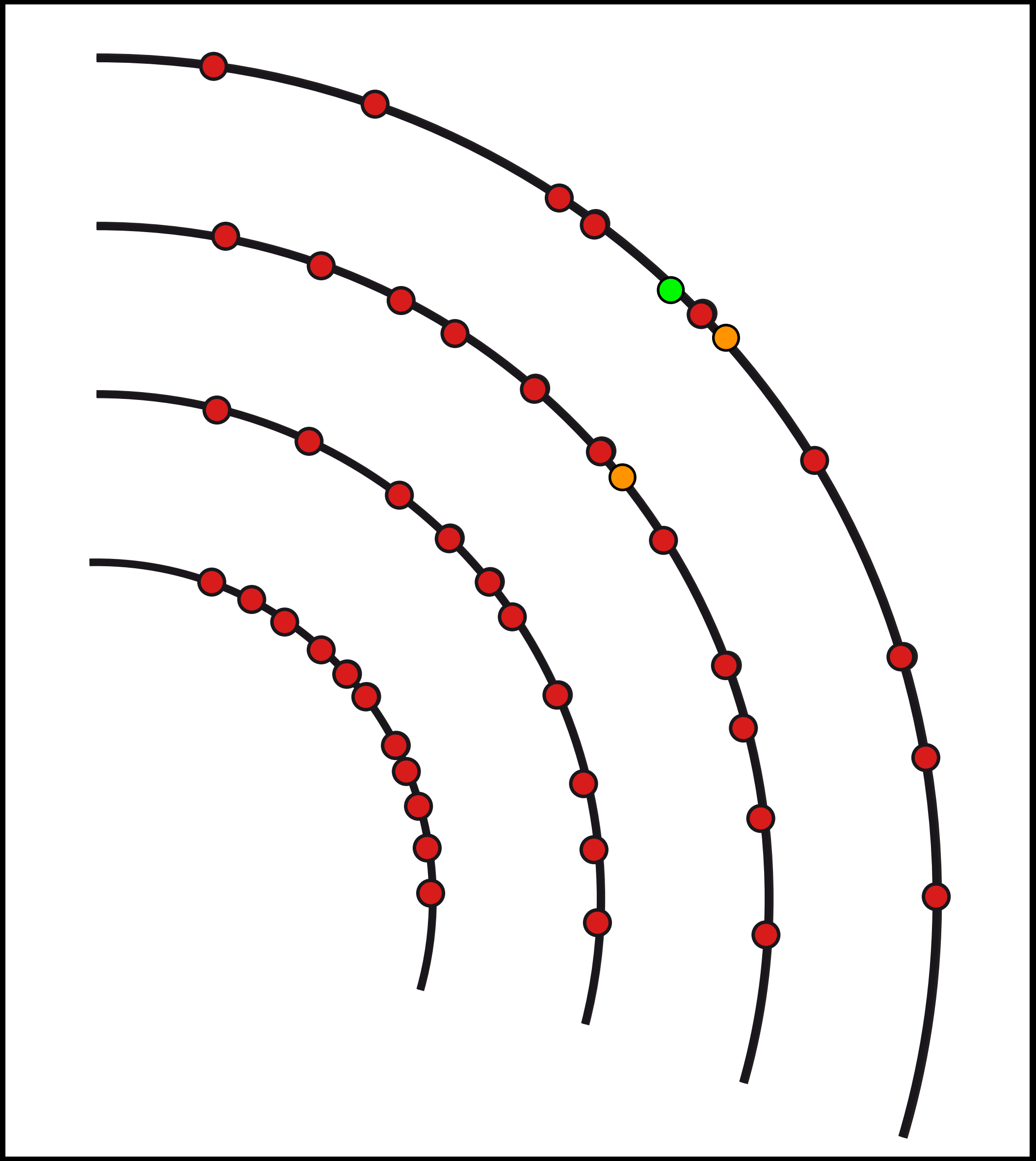
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** to next layer, find **best hit** and **update** trajectory
  - ➡ repeat until last layers to obtain **candidates**



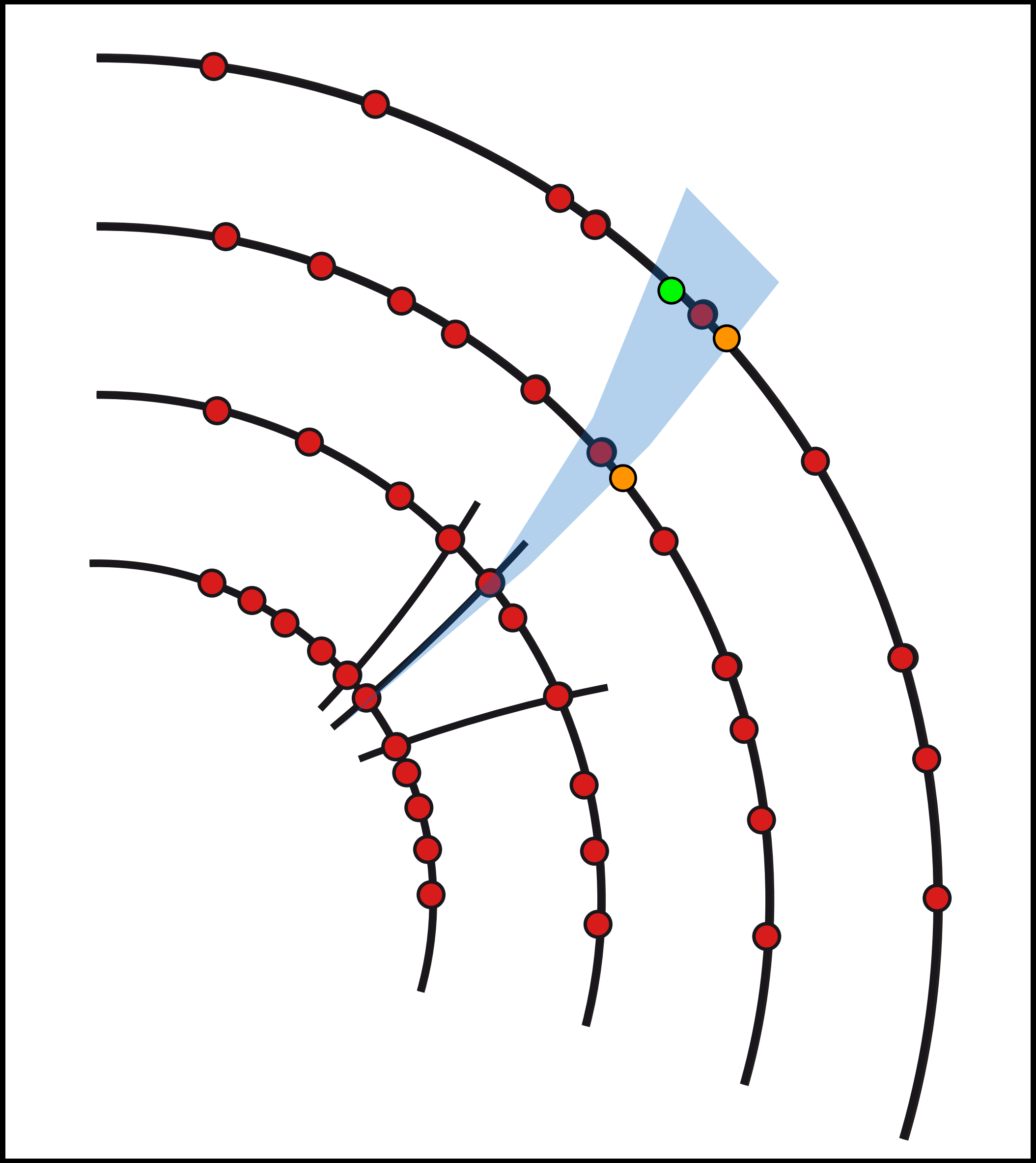
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** to next layer, find **best hit** and **update** trajectory
  - ➡ repeat until last layers to obtain **candidates**
- Combinatorial Kalman Filter



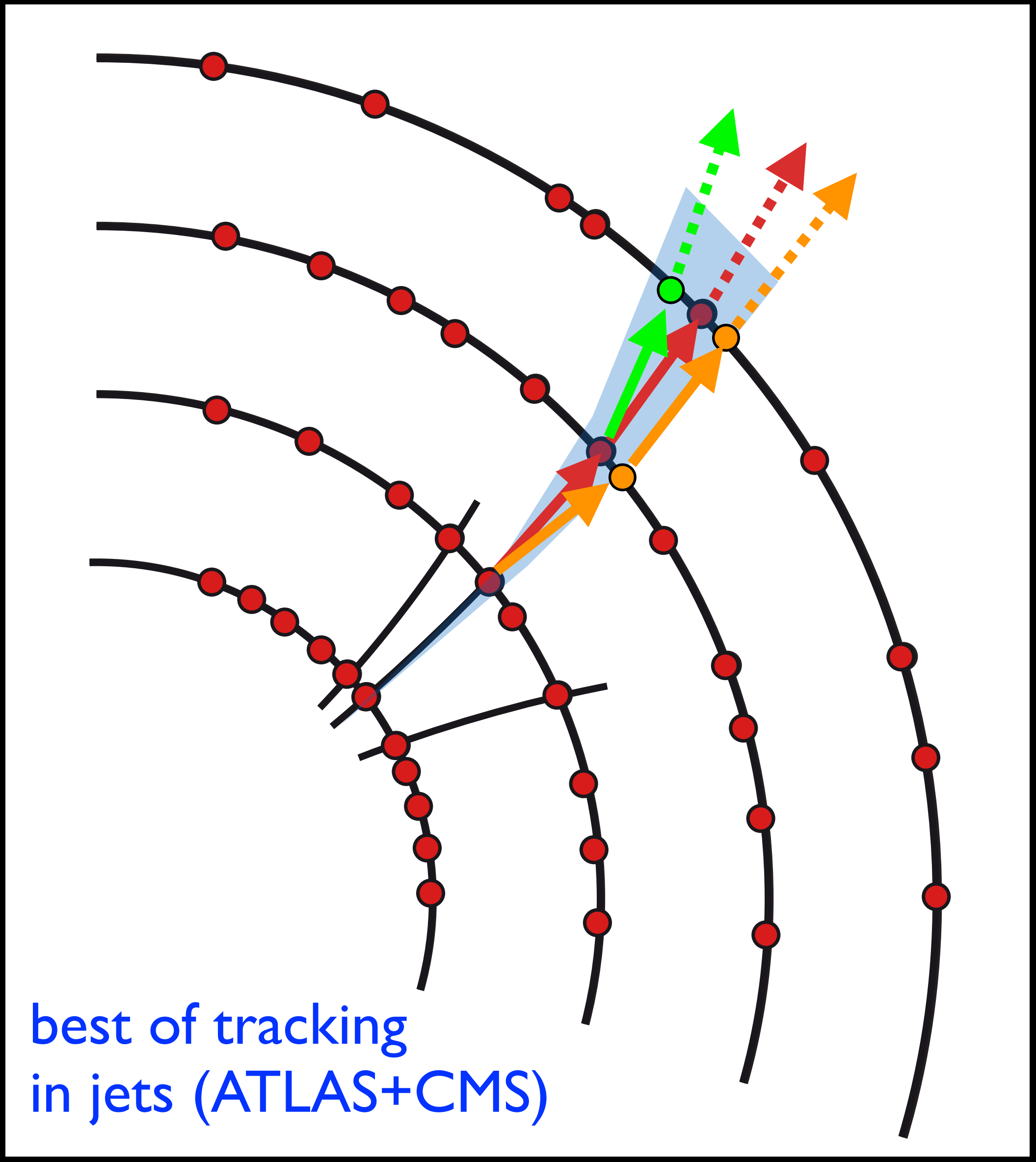
Local Track Finding

- **Track Road algorithm**
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- **Track Following**
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- **Progressive Track Finder**
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** to next layer, find **best hit** and **update** trajectory
  - ➡ repeat until last layers to obtain **candidates**
- **Combinatorial Kalman Filter**
  - ➡ extension of a Progressive Track Finder for dense environments



Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** to next layer, find **best hit** and **update** trajectory
  - ➡ repeat until last layers to obtain **candidates**

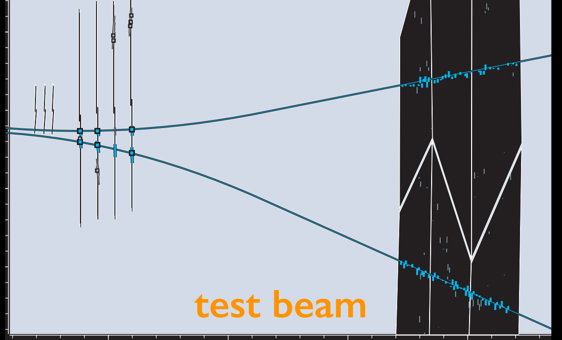


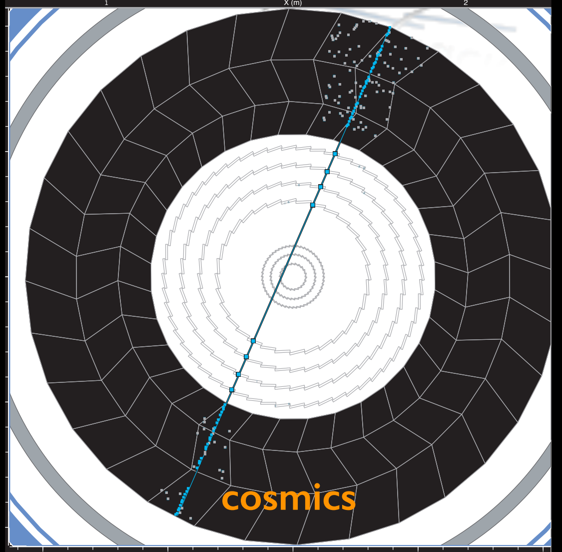
- Combinatorial Kalman Filter
  - ➡ extension of a Progressive Track Finder for dense environments
  - ➡ full **combinatorial exploration**, follow all hits to find all possible **track candidates**

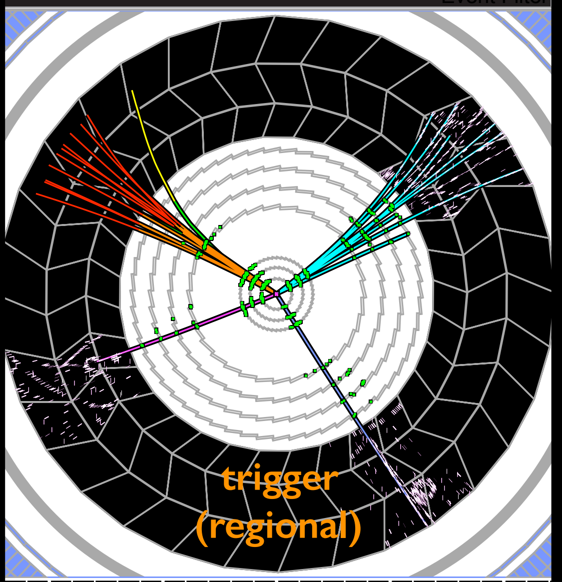
The ATLAS Track Reconstruction

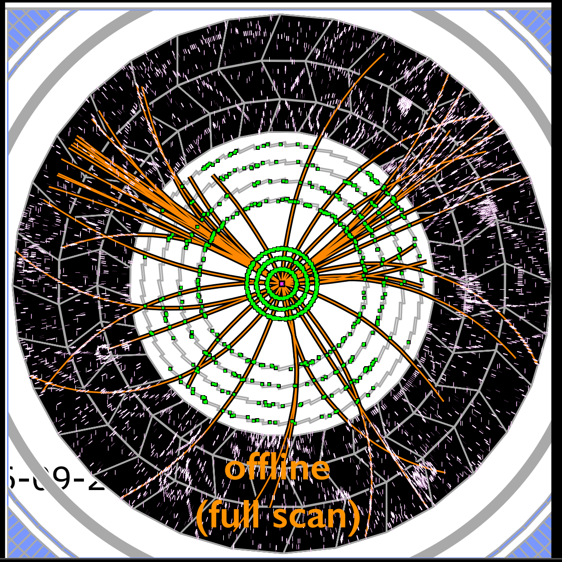
... and in Practice ?

- choice of reconstruction strategy depends on:
  - ➡ detector technologies
  - ➡ physics/performance requirements
  - ➡ occupancy and backgrounds
  - ➡ technical constraints (CPU, memory)
- even for same detector setup one looks at different types of events:
  - ➡ test beam
  - ➡ cosmics
  - ➡ trigger (regional)
  - ➡ offline (full scan)
- track reconstruction used by experiments
  - ➡ usually apply a **combination of different techniques**
  - ➡ often **iterative** ~ different strategies run one after the other to obtain best possible performance within resource constraints



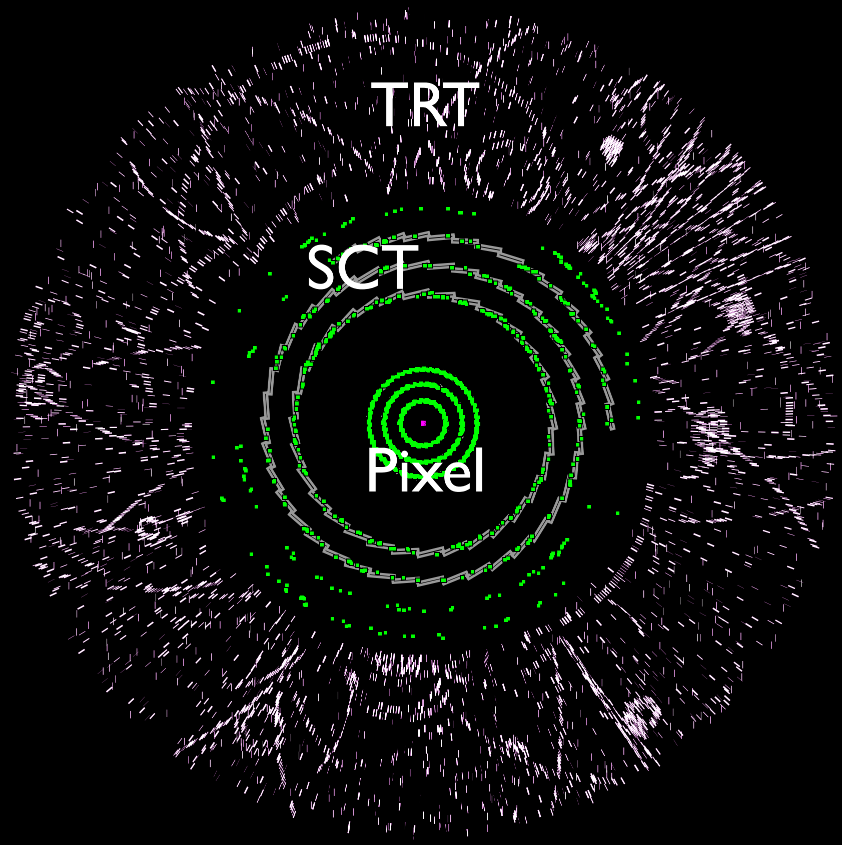






ATLAS NewTracking Software Chain

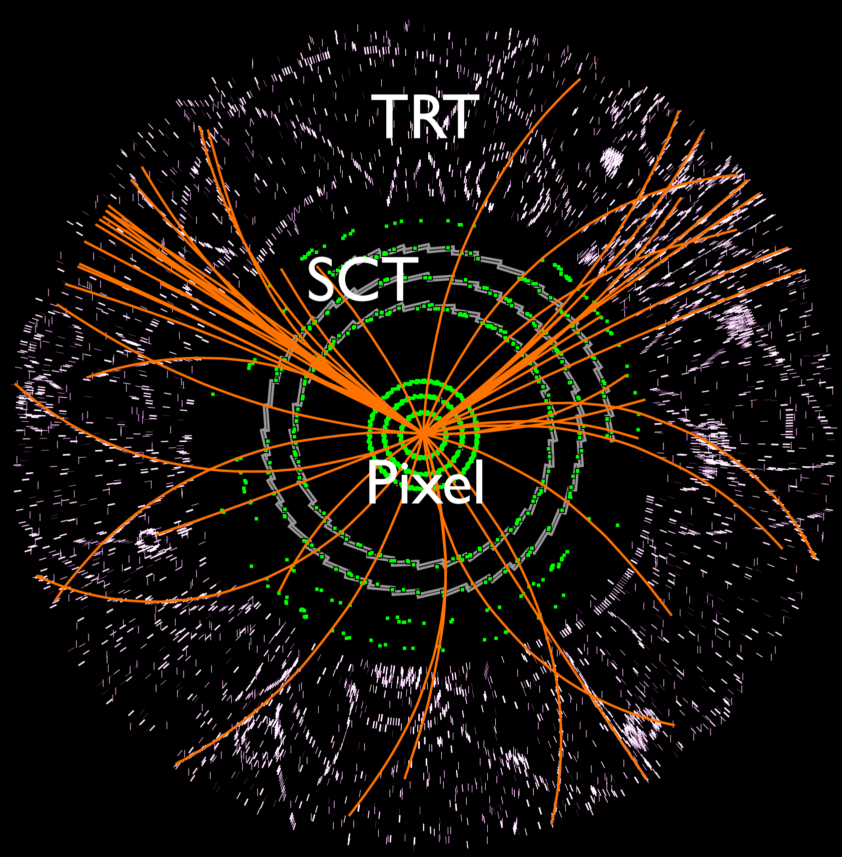
pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation



ATLAS NewTracking Software Chain

pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder
- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates



ambiguity solution
- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

extension into TRT
- ➡ progressive finder
- ➡ refit of track and selection

ATLAS NewTracking Software Chain

pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder
- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates

ambiguity solution
- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

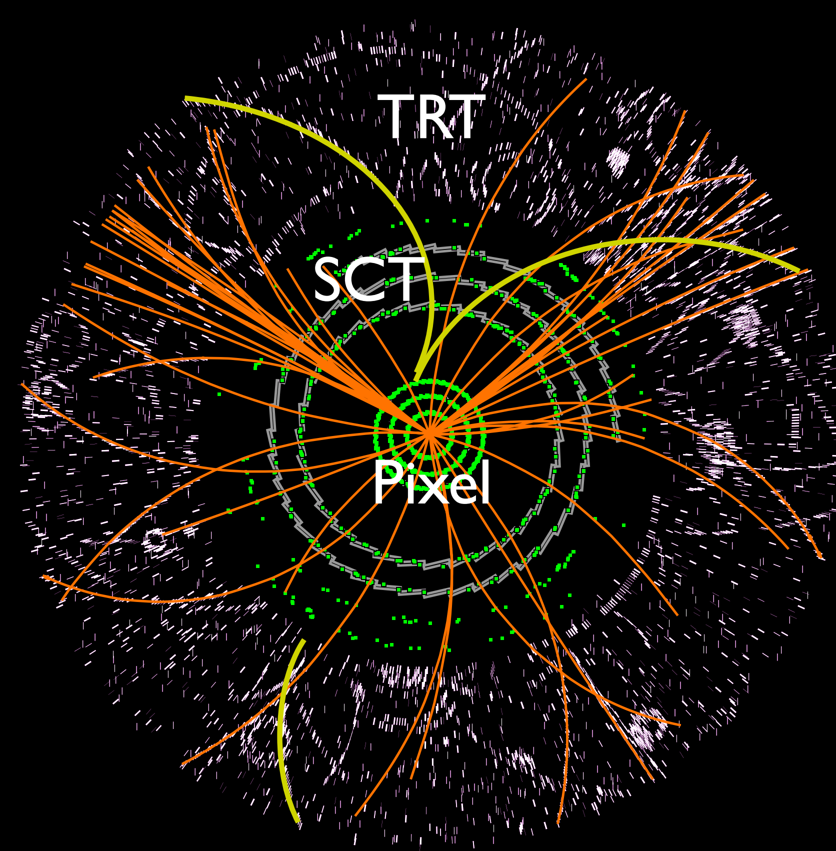
extension into TRT
- ➡ progressive finder
- ➡ refit of track and selection

TRT segment finder
- ➡ on remaining drift circles
- ➡ uses Hough transform

TRT seeded finder
- ➡ from TRT into SCT+Pixels
- ➡ combinatorial finder

ambiguity solution
- ➡ precise fit and selection
- ➡ TRT seeded tracks

standalone TRT
- ➡ unused TRT segments



ATLAS NewTracking Software Chain

pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder
- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates

ambiguity solution
- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

extension into TRT
- ➡ progressive finder
- ➡ refit of track and selection

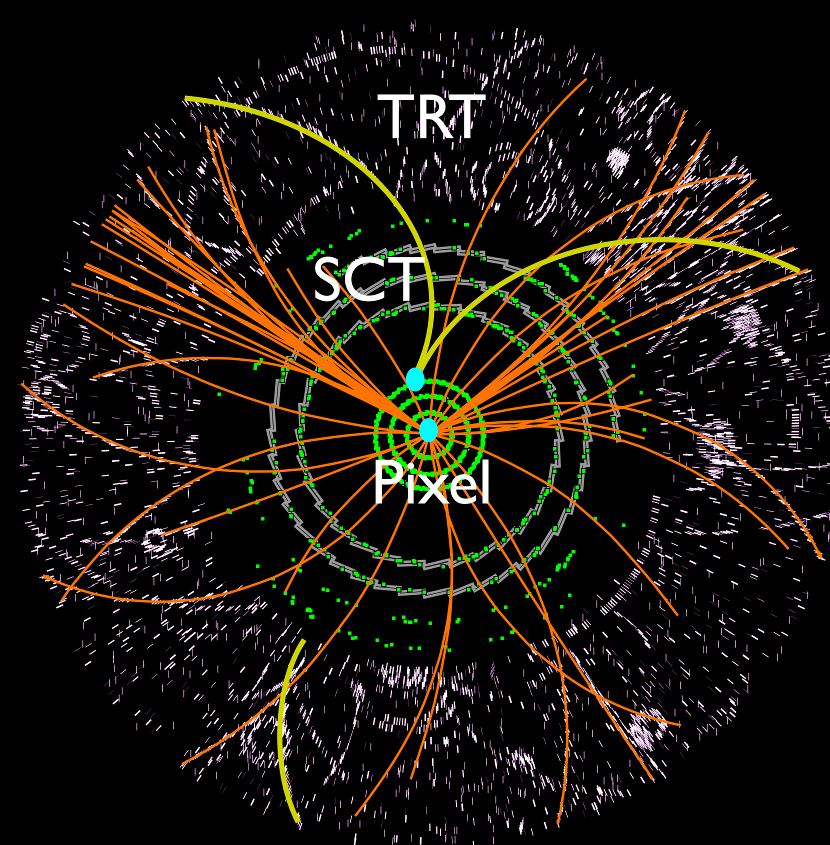
TRT segment finder
- ➡ on remaining drift circles
- ➡ uses Hough transform

TRT seeded finder
- ➡ from TRT into SCT+Pixels
- ➡ combinatorial finder

ambiguity solution
- ➡ precise fit and selection
- ➡ TRT seeded tracks

standalone TRT
- ➡ unused TRT segments

vertexing
- ➡ primary vertexing
- ➡ conversion and V0 search



ATLAS NewTracking Software Chain

pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder
- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates

ambiguity solution
- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

extension into TRT
- ➡ progressive finder
- ➡ refit of track and selection

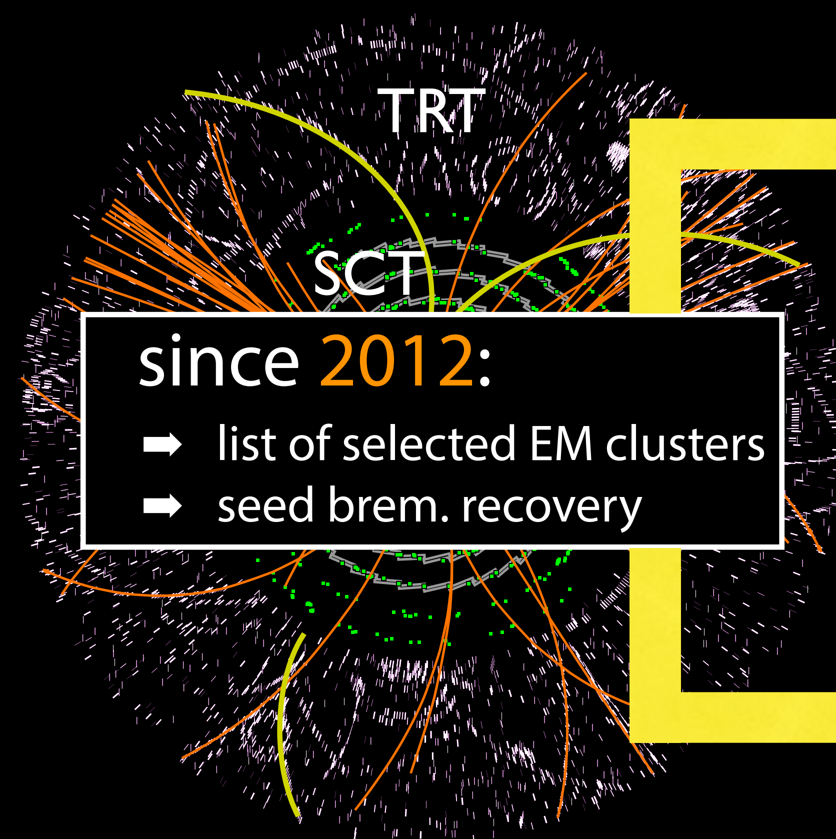
TRT segment finder
- ➡ on remaining drift circles
- ➡ uses Hough transform

TRT seeded finder
- ➡ from TRT into SCT+Pixels
- ➡ combinatorial finder

ambiguity solution
- ➡ precise fit and selection
- ➡ TRT seeded tracks

standalone TRT
- ➡ unused TRT segments

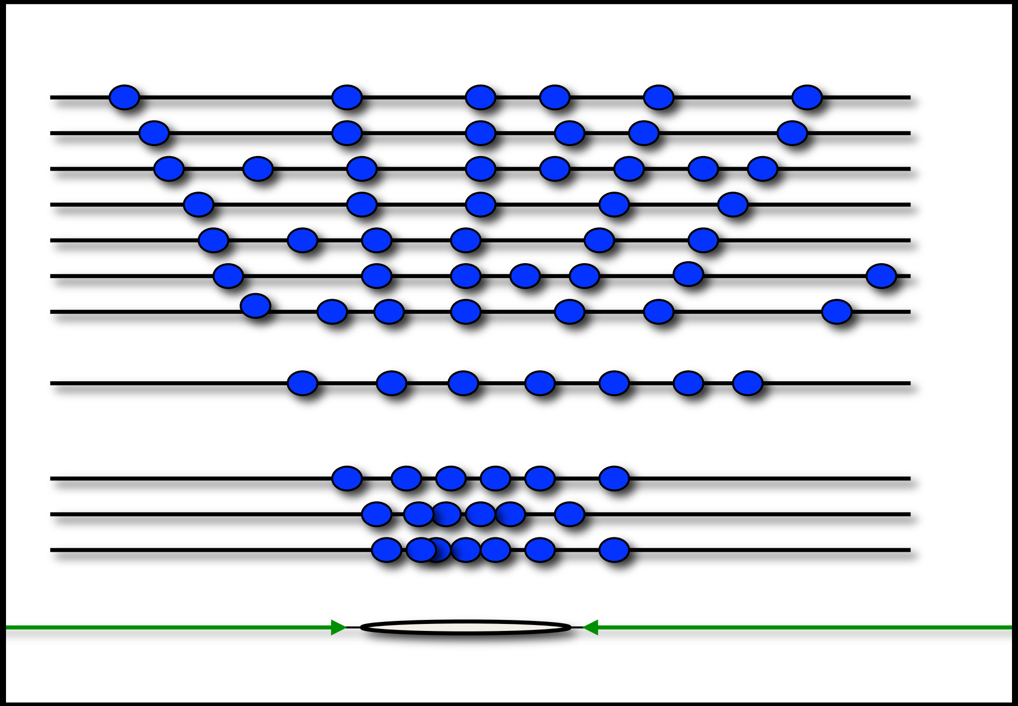
vertexing
- ➡ primary vertexing
- ➡ conversion and V0 search



since 2012:
- ➡ list of selected EM clusters
- ➡ seed brem. recovery

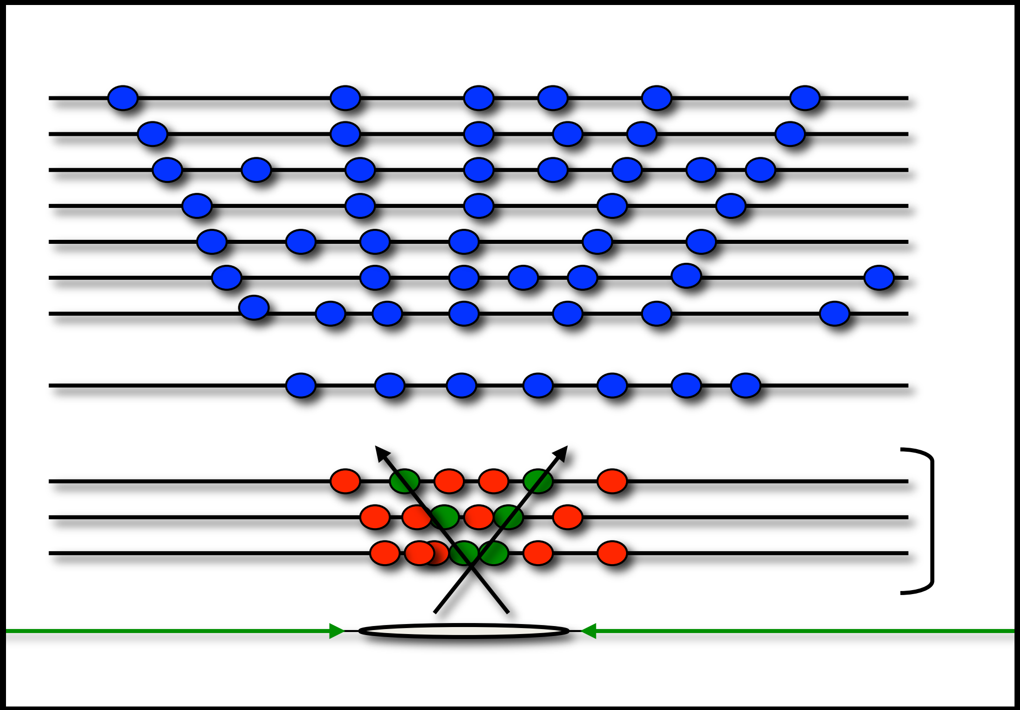
The Iterative Tracking Strategy

- track finding is most time consuming reconstruction step
  - ➡ avoid combinatorial overhead !
  - ➡ iterative seeding approach:



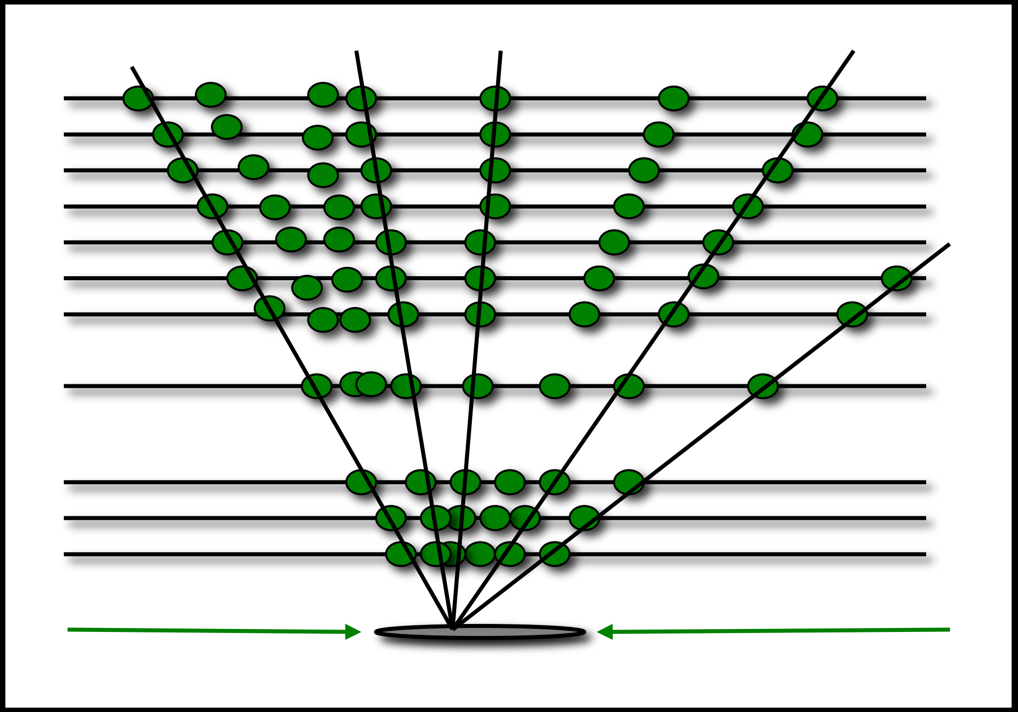
The Iterative Tracking Strategy

- track finding is most time consuming reconstruction step
  - ➡ avoid combinatorial overhead !
  - ➡ iterative seeding approach:
    - restrict seeding for combinatorial Kalman Filter to set of layers



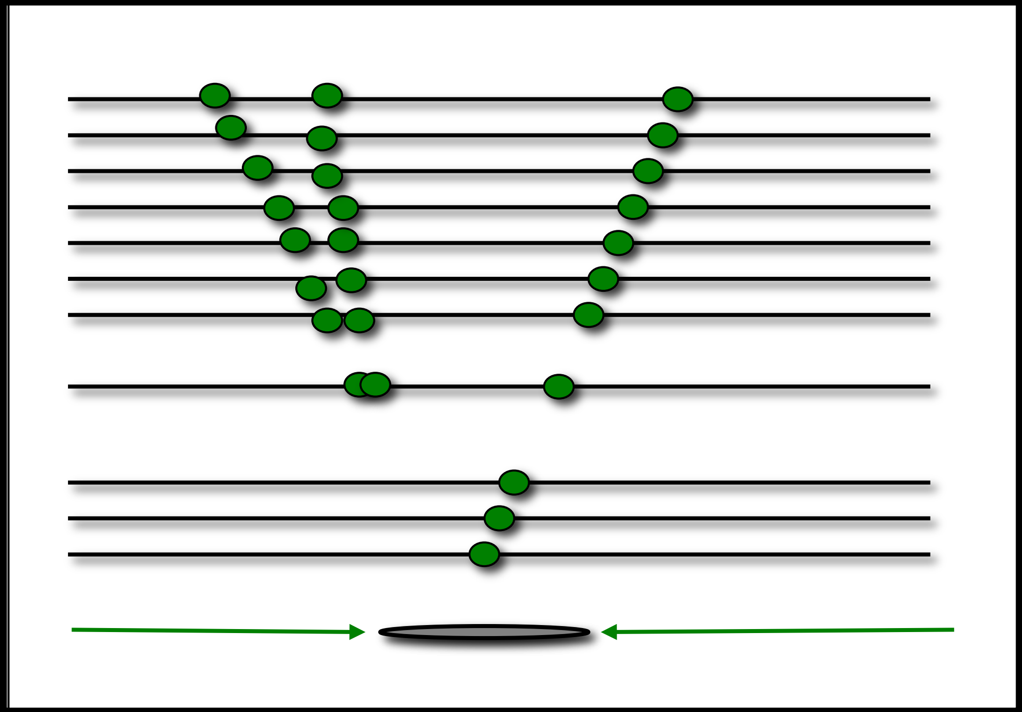
The Iterative Tracking Strategy

- track finding is most time consuming reconstruction step
  - ➡ avoid combinatorial overhead !
  - ➡ iterative seeding approach:
    - restrict seeding for combinatorial Kalman Filter to set of layers
    - find initial set of tracks



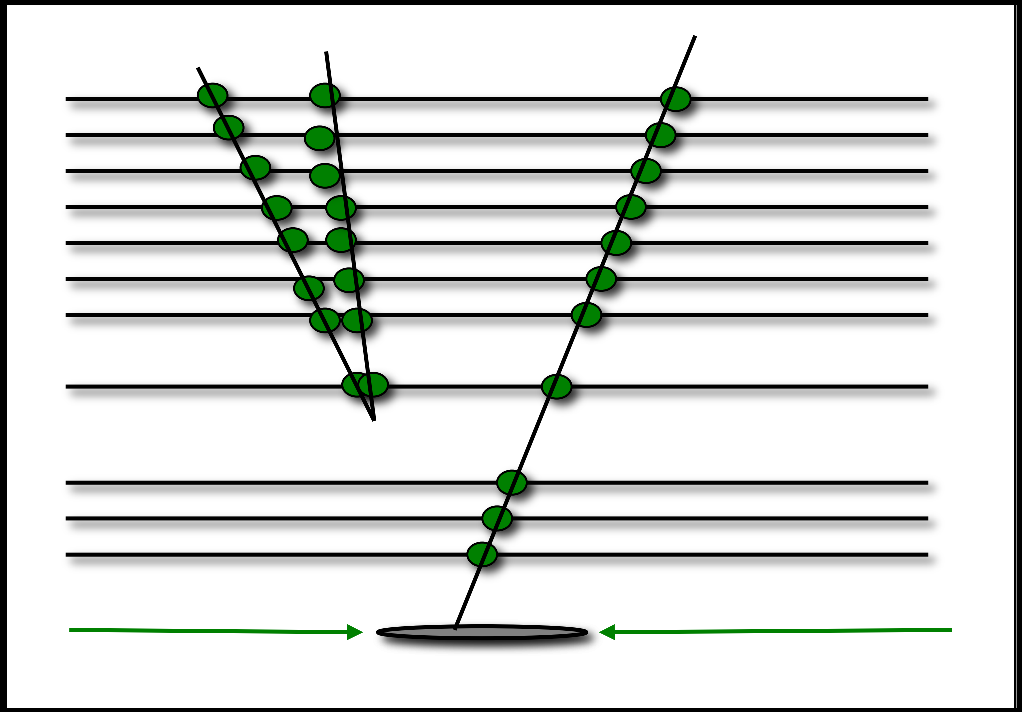
The Iterative Tracking Strategy

- track finding is most time consuming reconstruction step
  - ➡ avoid combinatorial overhead !
  - ➡ iterative seeding approach:
    - restrict seeding for combinatorial Kalman Filter to set of layers
    - find initial set of tracks
    - remove used hits from event



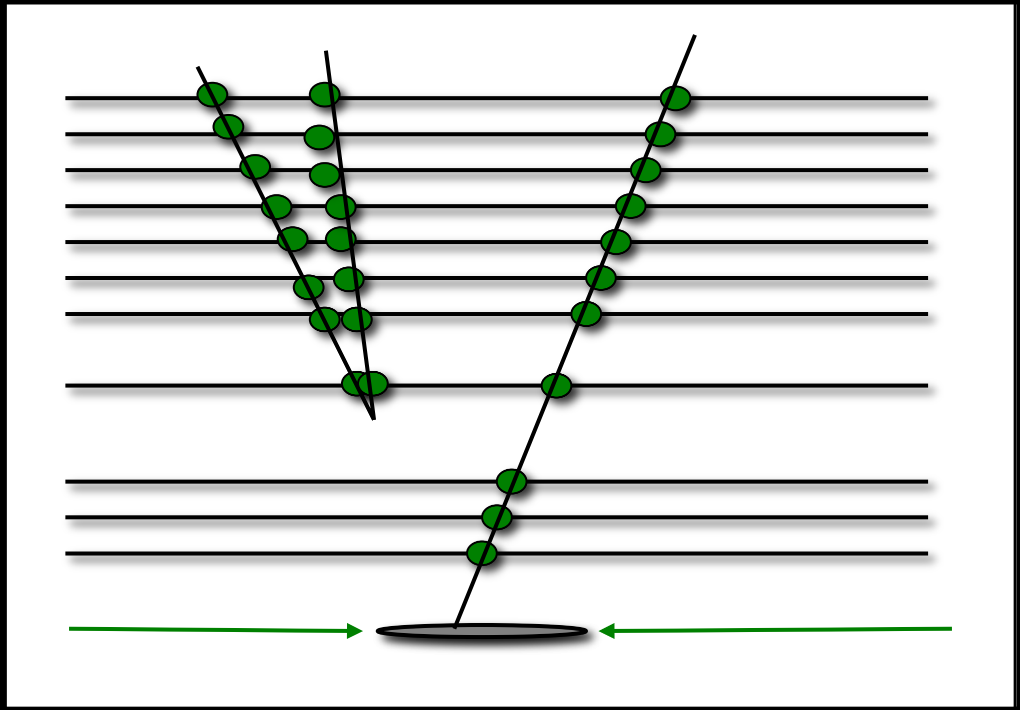
The Iterative Tracking Strategy

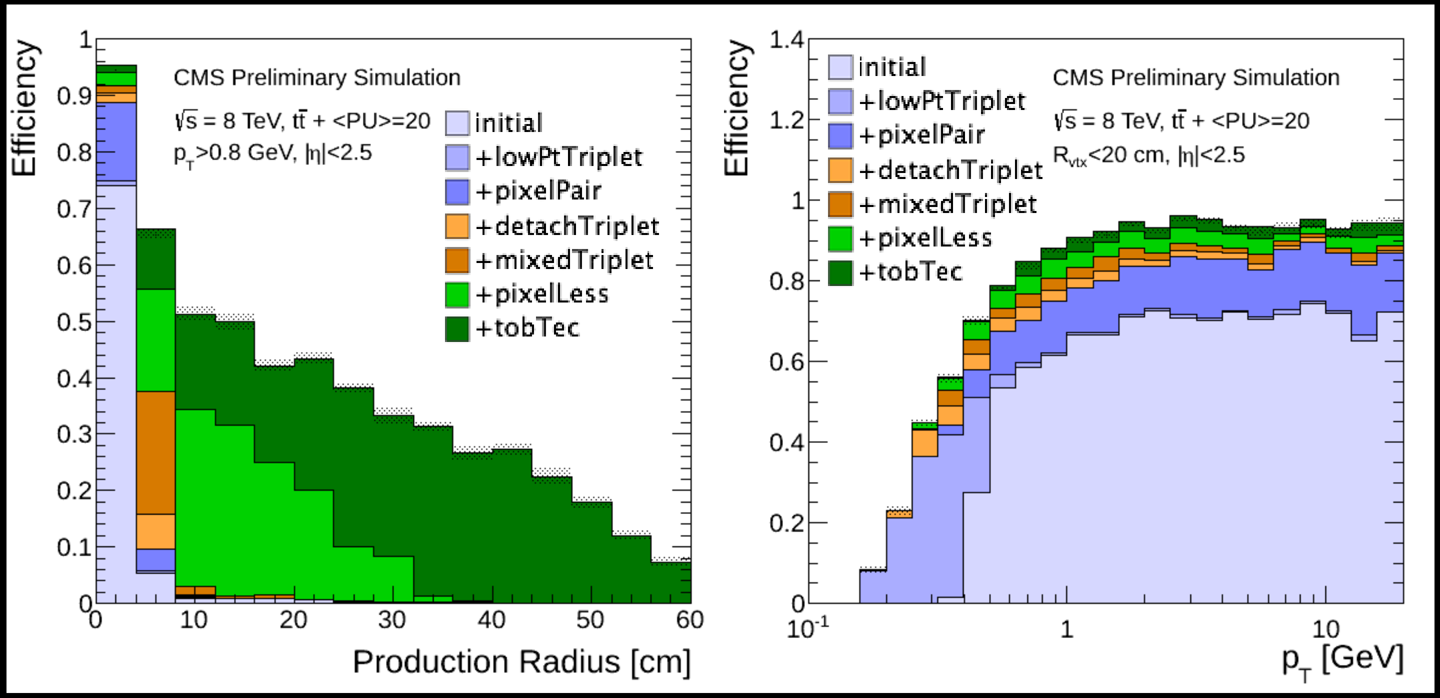
- track finding is most time consuming reconstruction step
  - ➡ avoid combinatorial overhead !
  - ➡ iterative seeding approach:
    - restrict seeding for combinatorial Kalman Filter to set of layers
    - find initial set of tracks
    - remove used hits from event
    - seed tracking from different set of layers to find more tracks
    - ... etc.



The Iterative Tracking Strategy

- **track finding** is most time consuming reconstruction step
  - ➡ avoid combinatorial overhead !
  - ➡ iterative seeding approach:
    - restrict seeding for combinatorial Kalman Filter to set of layers
    - find initial set of tracks
    - remove used hits from event
    - seed tracking from different set of layers to find more tracks
    - ... etc.
  - ➡ optimal choice of iterative seeding strategy is matter of tuning
    - e.g. CMS did 7 iterations in Run-1





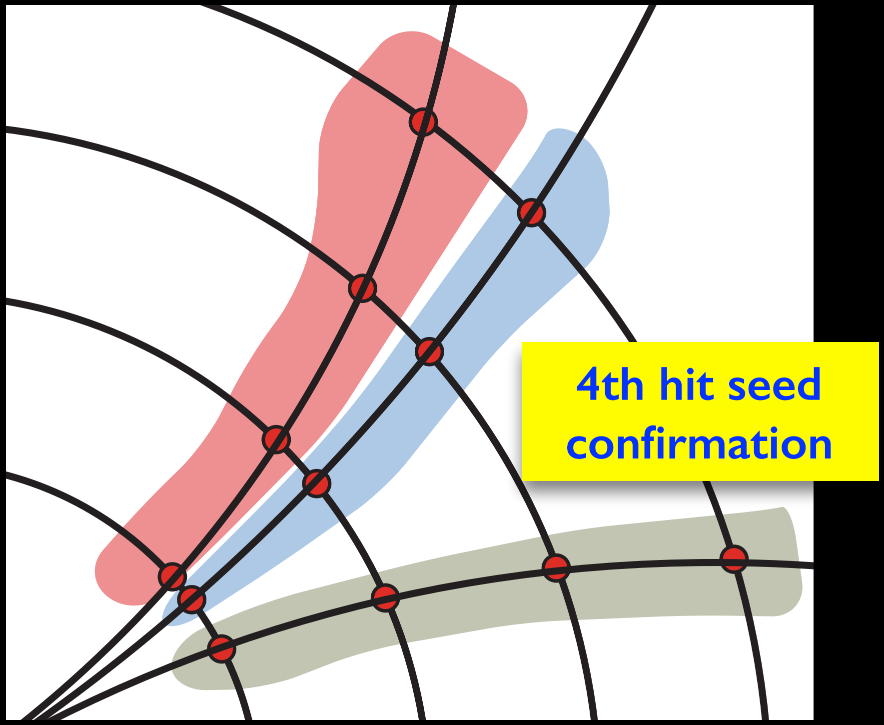
Tuning the Iterative Tracking Strategy

- optimal seeding strategy depends on level of pileup (ATLAS)
  - ➡ fraction of seeds to give a good track candidate:

seed-triplets:
P = Pixel
S = Strips

pileup	"PPP"	"PPS"	"PSS"	"SSS"
0	57%	26%	29%	66%
40	17%	6%	5%	35%

- hence start with SSS at 40 pileup !



Tuning the Iterative Tracking Strategy

- optimal seeding strategy depends on level of pileup (ATLAS)
  - ➡ fraction of seeds to give a good track candidate:

seed-triplets:
P = Pixel
S = Strips

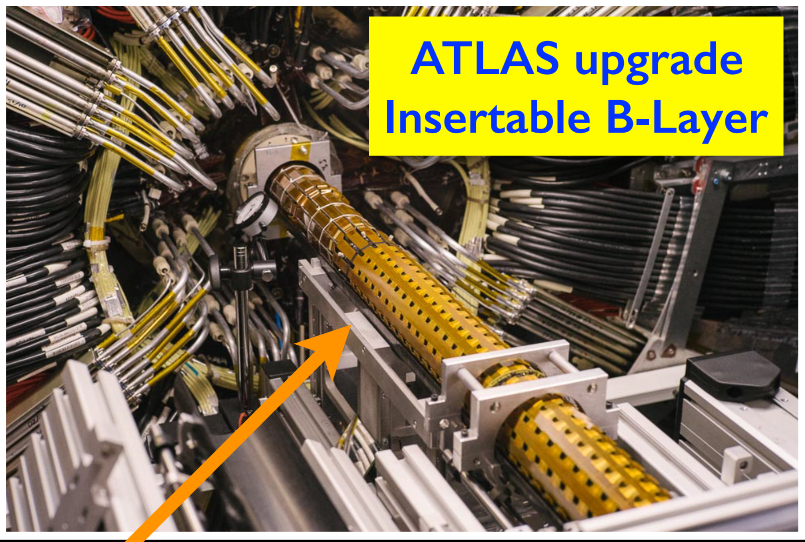
pileup	"PPP"	"PPS"	"PSS"	"SSS"
0	57%	26%	29%	66%
40	17%	6%	5%	35%

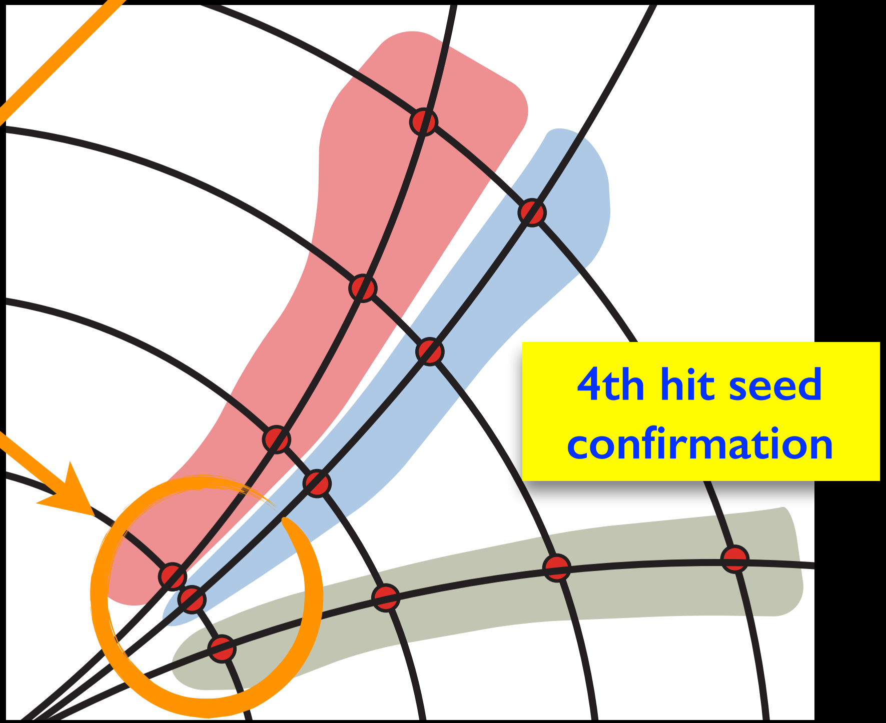
  - hence start with SSS at 40 pileup !

- ➡ further increase good seed fraction using 4th hit

pileup	"PPP+1"	"PPS+1"	"PSS+1"	"SSS+1"
0	79%	53%	52%	86%
40	39%	8%	16%	70%

  - takes benefit from new Insertable B-Layer (IBL)





Tuning the Iterative Tracking Strategy

- optimal seeding strategy depends on level of pileup (ATLAS)
  - ➡ fraction of seeds to give a good track candidate:

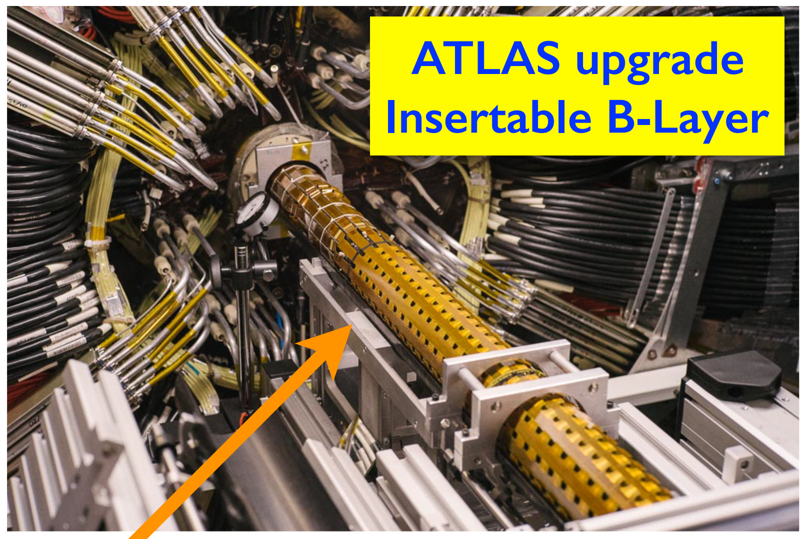
seed-triplets:
P = Pixel
S = Strips

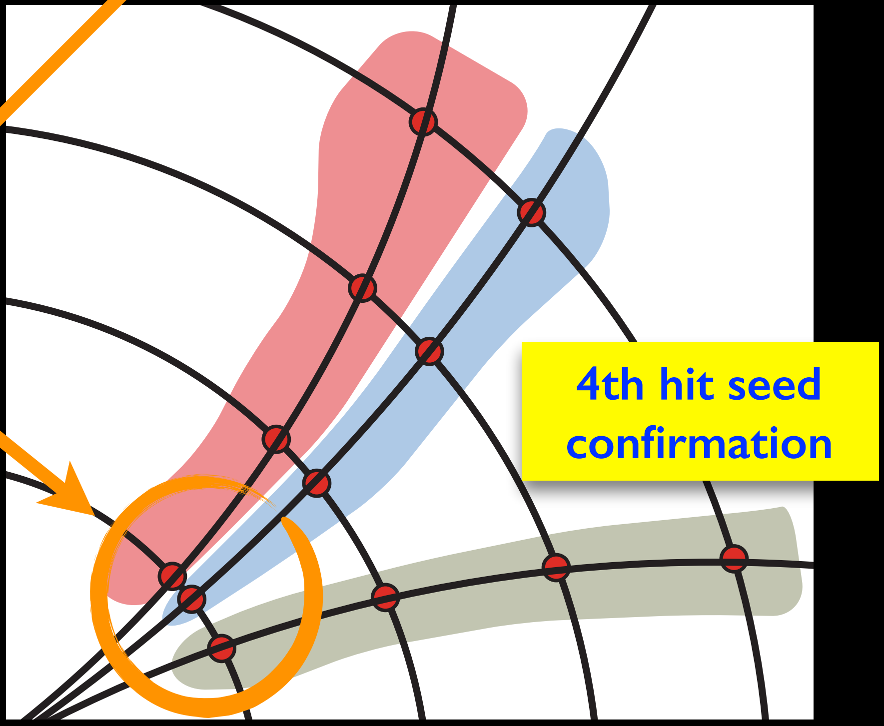
pileup	"PPP"	"PPS"	"PSS"	"SSS"
0	57%	26%	29%	66%
40	17%	6%	5%	35%

  - hence start with SSS at 40 pileup !
  - ➡ further increase good seed fraction using 4th hit

pileup	"PPP+1"	"PPS+1"	"PSS+1"	"SSS+1"
0	79%	53%	52%	86%
40	39%	8%	16%	70%

  - takes benefit from new Insertable B-Layer (IBL)



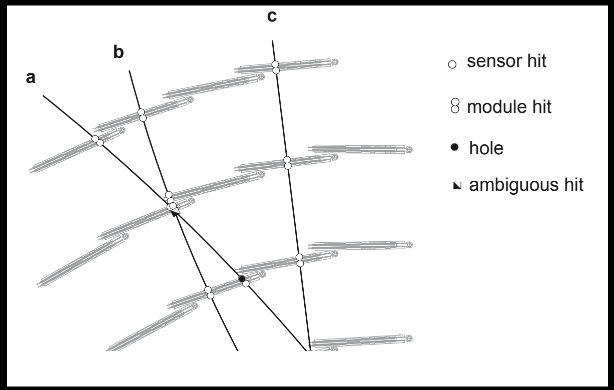


- final ATLAS Run-2 seeding strategy
  - ➡ significant speedup at 40 pileup (and 25 *ns*)

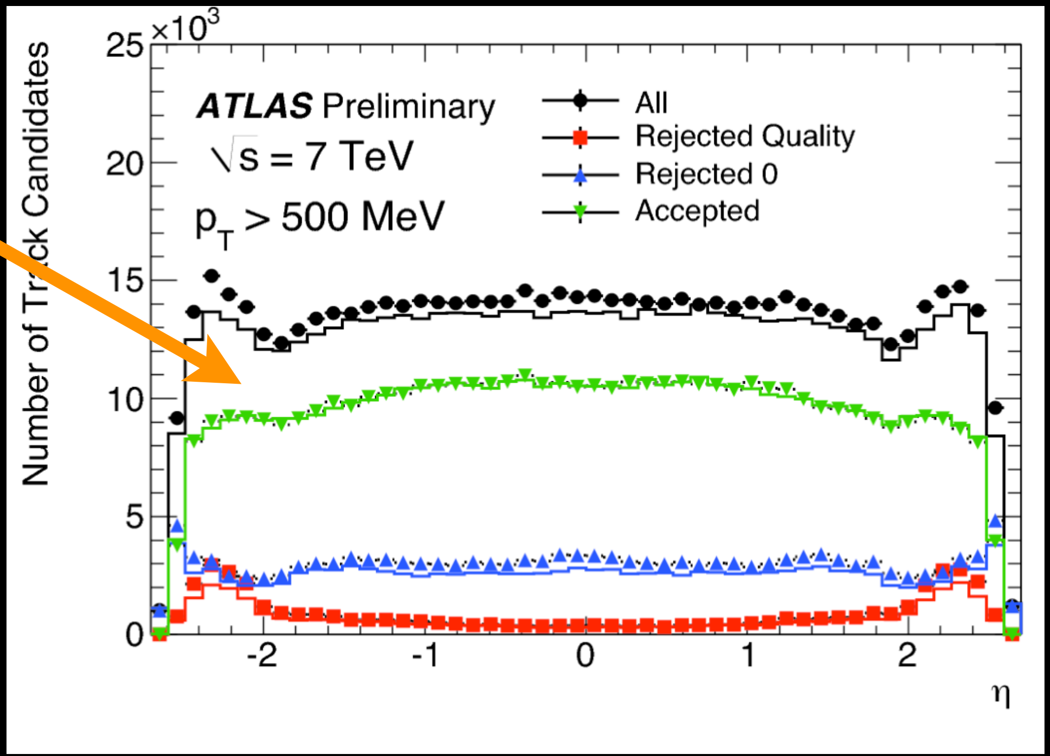
seeding	efficiency	CPU*
"Run-1"	94.0%	9.5 sec
"Run-2"	94.2%	4.7 sec

*on local machine

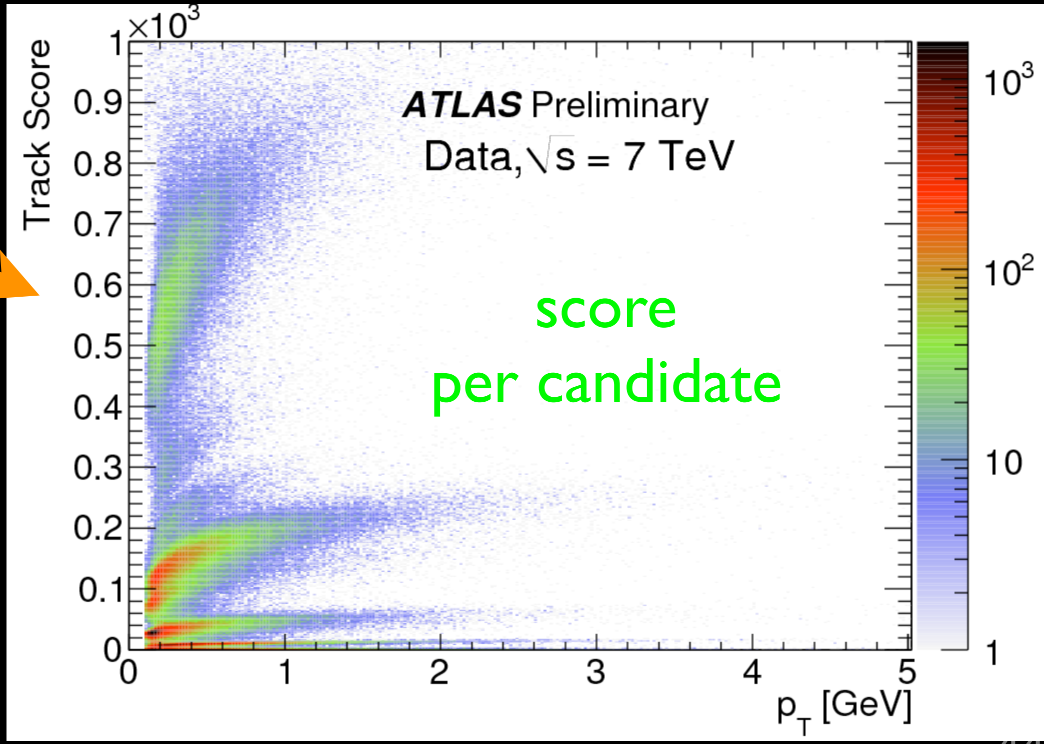
Ambiguity Solution



- track selection cuts
  - ➡ applied at every stage in reconstruction
  - ➡ still more candidates than final tracks and too high rate of fakes

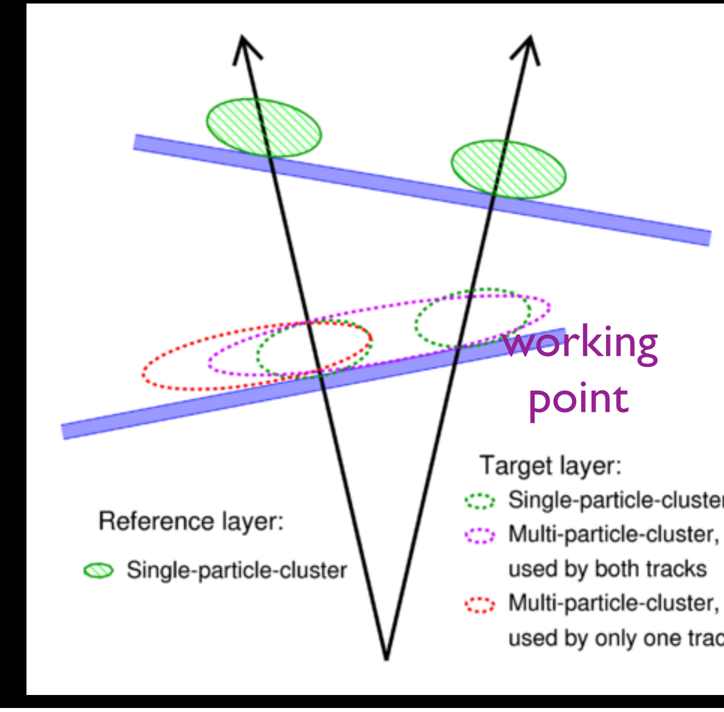


- task of ambiguity solution:
  - ➡ select good tracks and reject fakes
- ordered iterative procedure
  - ➡ in case of ATLAS:
    - precise fit with outlier removal
  - ➡ construct quality function ("score") for each candidate:
    1. hit content, holes
    2. number of shared hits
    3. fit quality...
  - ➡ candidate with best score wins
  - ➡ if too many shared hits, create sub-track if track with remaining hits passes cuts

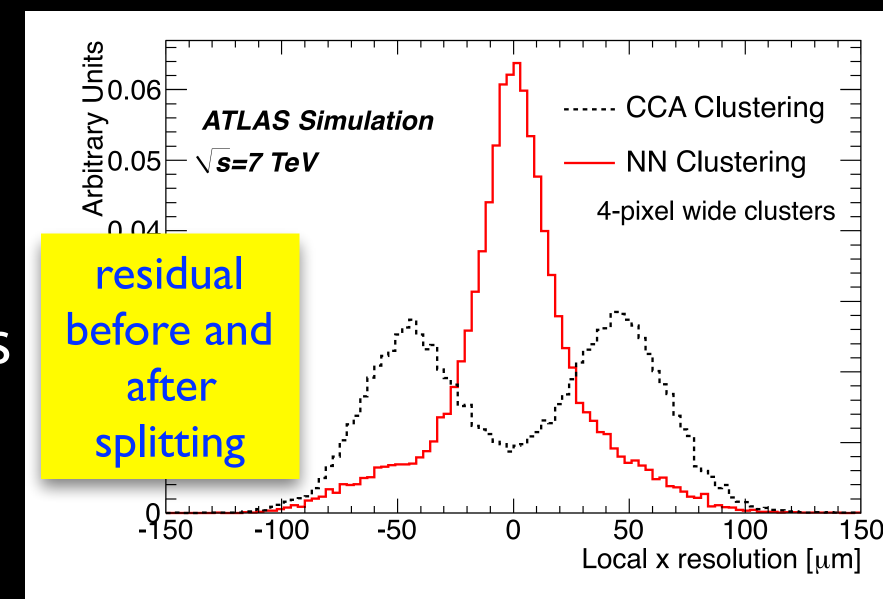


Tracking in dense Jets

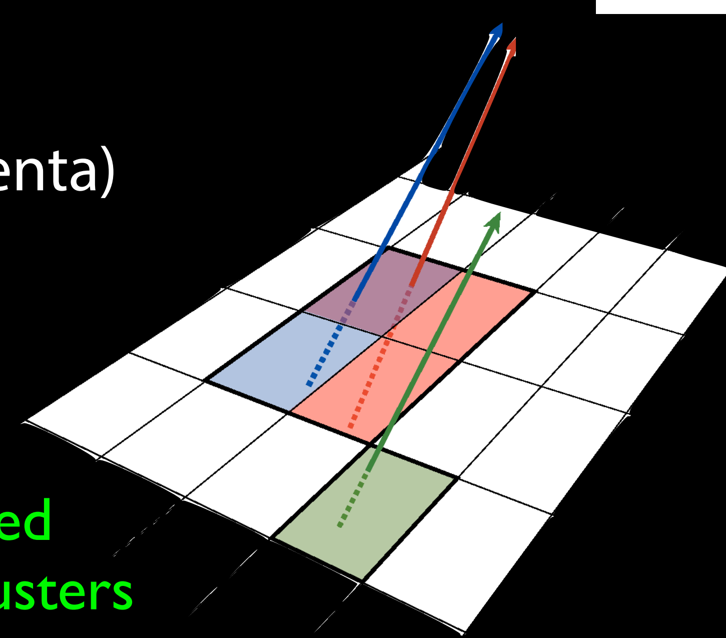
- problem of cluster merging
  - ➡ merging when track separation reaches single Pixel size
  - ➡ during track reconstruction shared clusters are penalised to reduce fakes and duplicate tracks

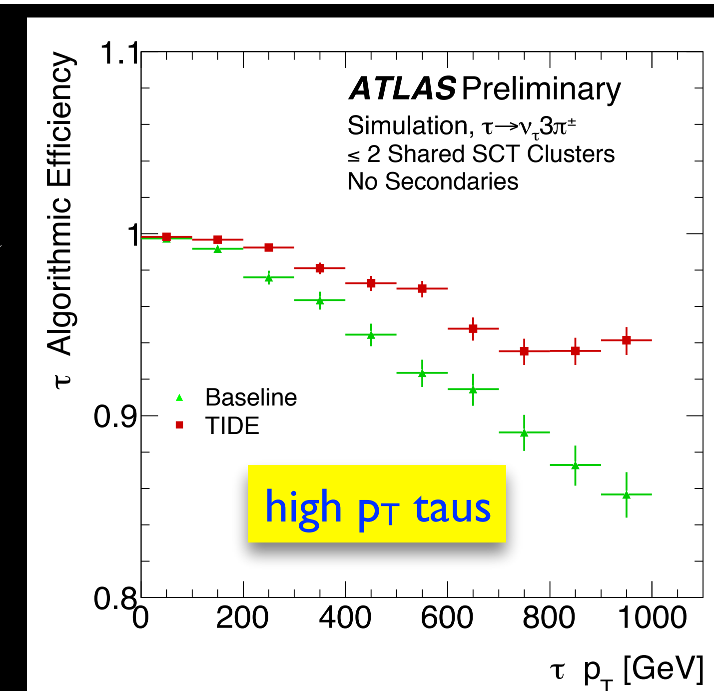


- neural network (NN) Pixel clustering
  - ➡ identify merged clusters and splitting them
    - identify merge clusters, split them and correct positions
  - ➡ splitting/sharing decision done in ambiguity processing
    - full track information for all candidates available



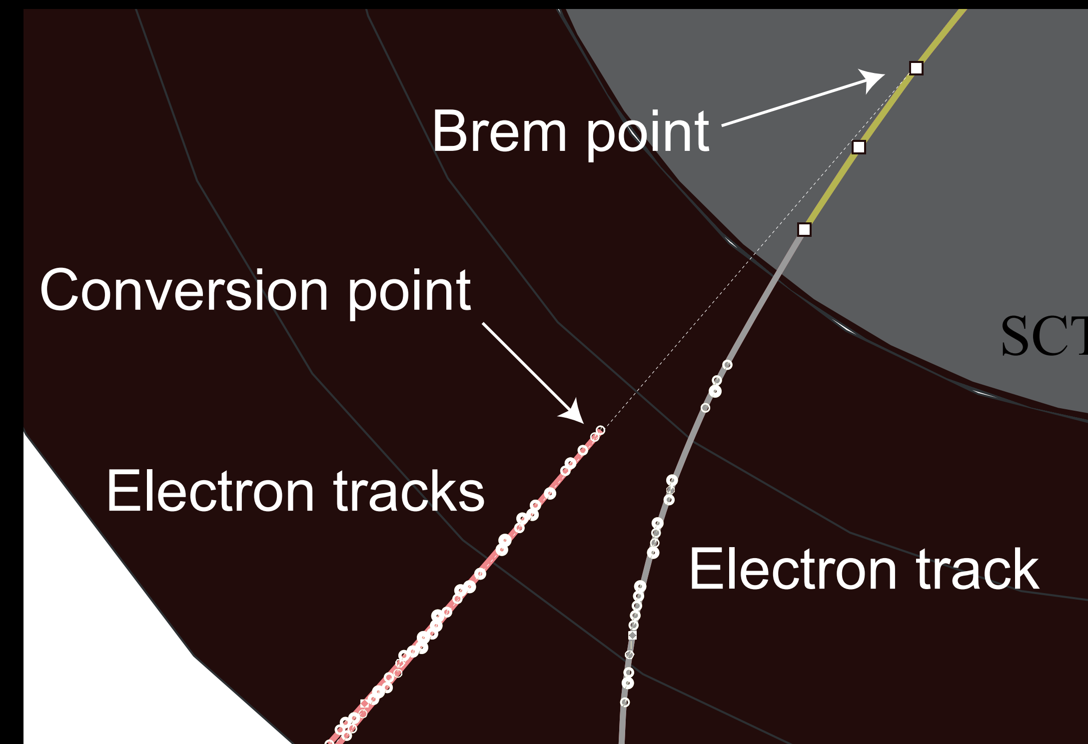
- crucial in many areas:
  - ➡ b-tagging (especially at high momenta)
  - ➡ jet calibration and particle flow
  - ➡ 3-prong τ identification



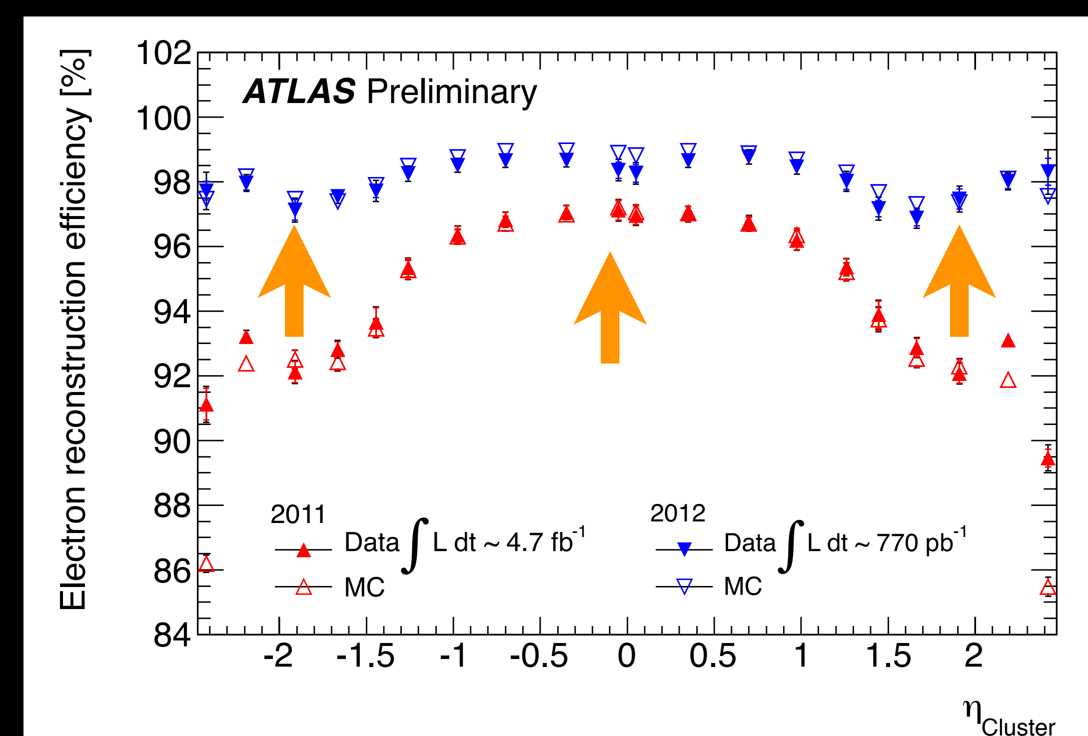


Tracking with Electron Brem. Recovery

- **strategy** for brem. recovery
  - ➡ restrict recovery to regions pointing to electromagnetic clusters (RoI)
  - ➡ pattern: allow for large energy loss in combinatorial Kalman filter
    - adjust noise term for electrons
  - ➡ global-χ^2 fitter allows for brem. point
  - ➡ adapt ambiguity processing (etc.) to ensure e.g. b-tagging is not affected
  - ➡ use full fledged Gaussian-Sum Filter in electron identification code

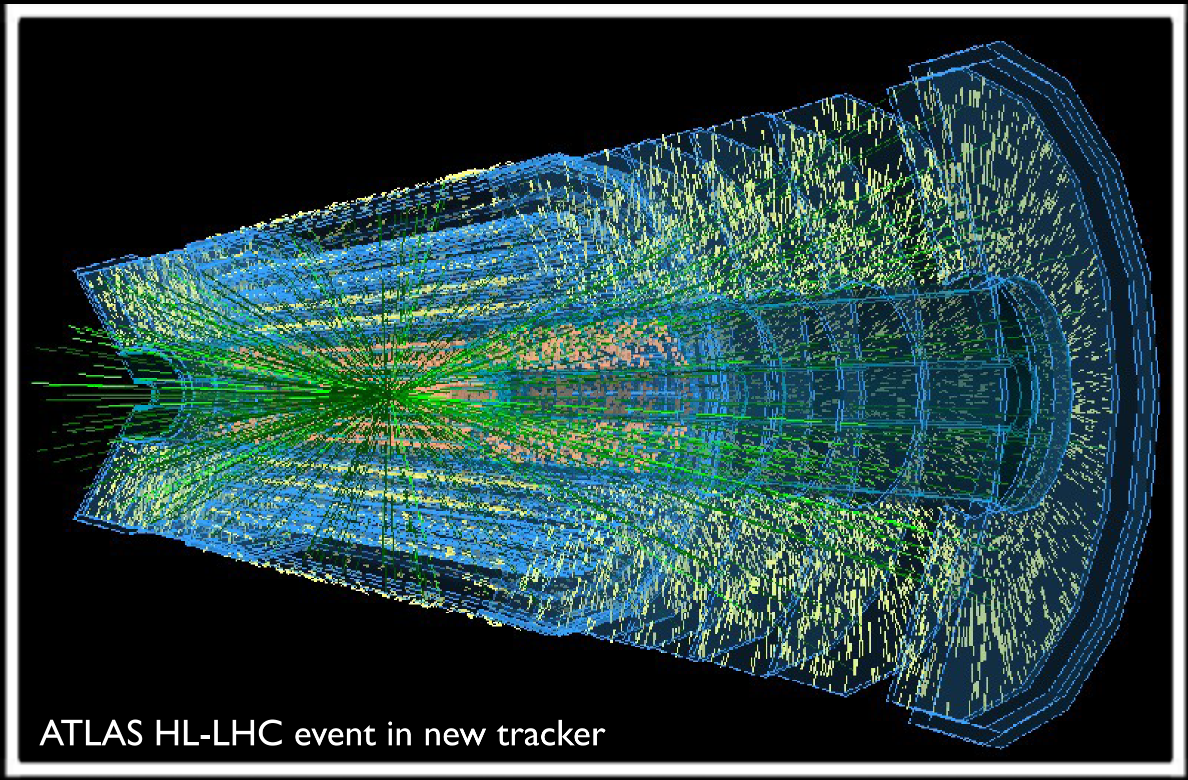


- **tracking update deployed in 2012**
  - ➡ improvements especially at low p_T (< 15 GeV)
    - limiting factor for $H \to ZZ^* \to 4e$
  - ➡ significant efficiency gain for Higgs discovery



Let's Summarise...

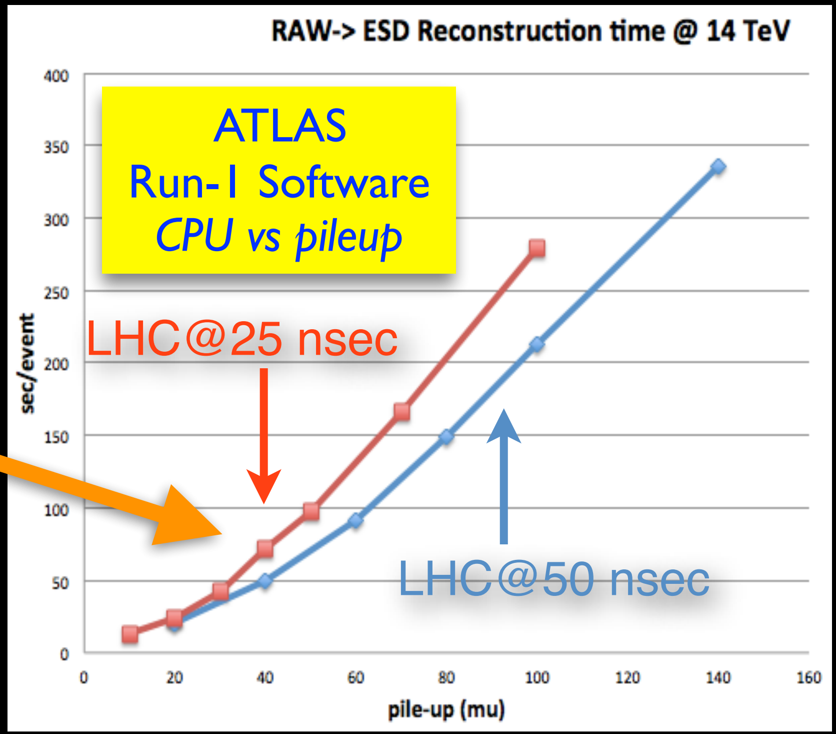
- discussed concepts for track reconstruction
- have overview of strategies and mathematical tools
- discussed an example of a track reconstruction package (ATLAS NewTracking)
- next is to talk about vertexing and its applications



Bonus Slides...

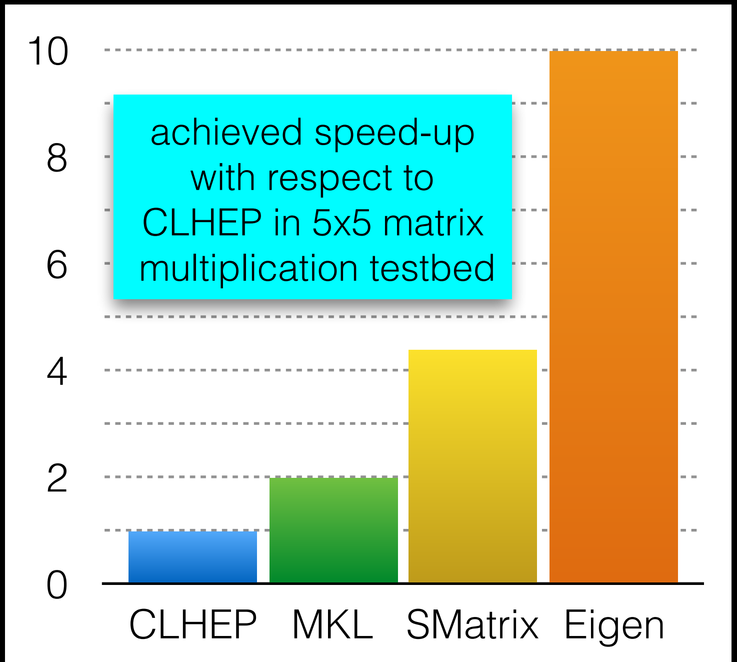
LS-1 Tracking Upgrades

...so what did we do about this so far ?



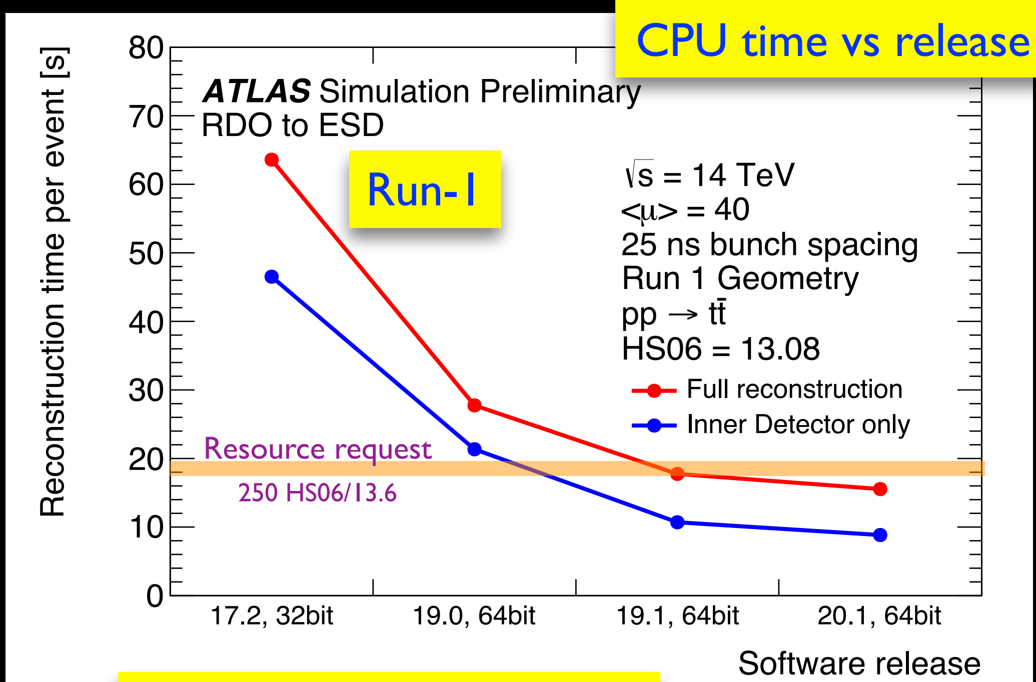
Tracking Developments towards Run-2

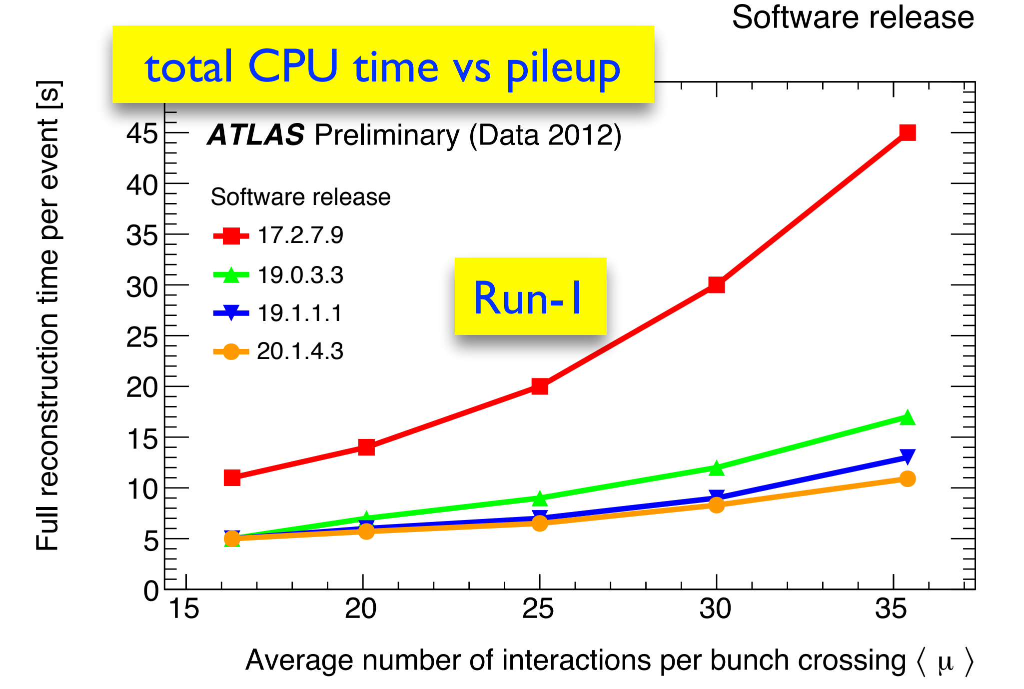
- ATLAS and CMS focus on technology and strategy to improve CURRENT algorithms
  - ➡ improve software technology, including:
    - simplify EDM design to be less OO ("hip" 10 years ago)
    - ATLAS migrated to Eigen - faster vector+matrix algebra (CMS was already using SMatrix)
    - vectorised trigonometric functions (CMS: VDT or ATLAS: intel math lib)
    - work on CPU hot spots (e.g. ATLAS replaced F90 by C++ for B-field service)
  - ➡ tune reconstruction strategy (very similar in ATLAS and CMS):
    - optimise iterative track finding strategy for 40 pileup
    - ATLAS modified track seeding to explore 4th Pixel layer
    - CMS added cluster-shape filter against out-of-time pileup
- hence, mix of SIMD and algorithm tuning
  - ➡ CMS made their tracking as well thread-safe

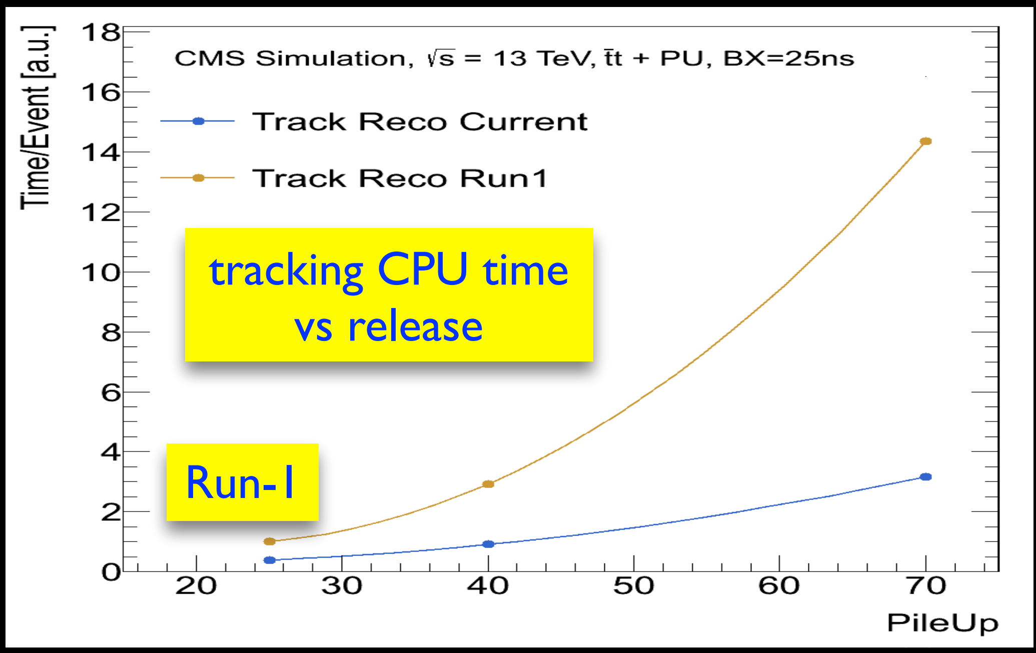


CPU for Reconstruction

- sum of tracking and general software improvements
  - ➡ improved software technology, including:
    - tracking related improvements
    - new 64 bit compilers, new tcmalloc
  - ➡ tune reconstruction strategy (very similar in ATLAS and CMS)
    - optimise track finding strategy for 40 pileup
    - faster versions of things like FastJet, ...
    - addressing other CPU hot spots in reconstruction

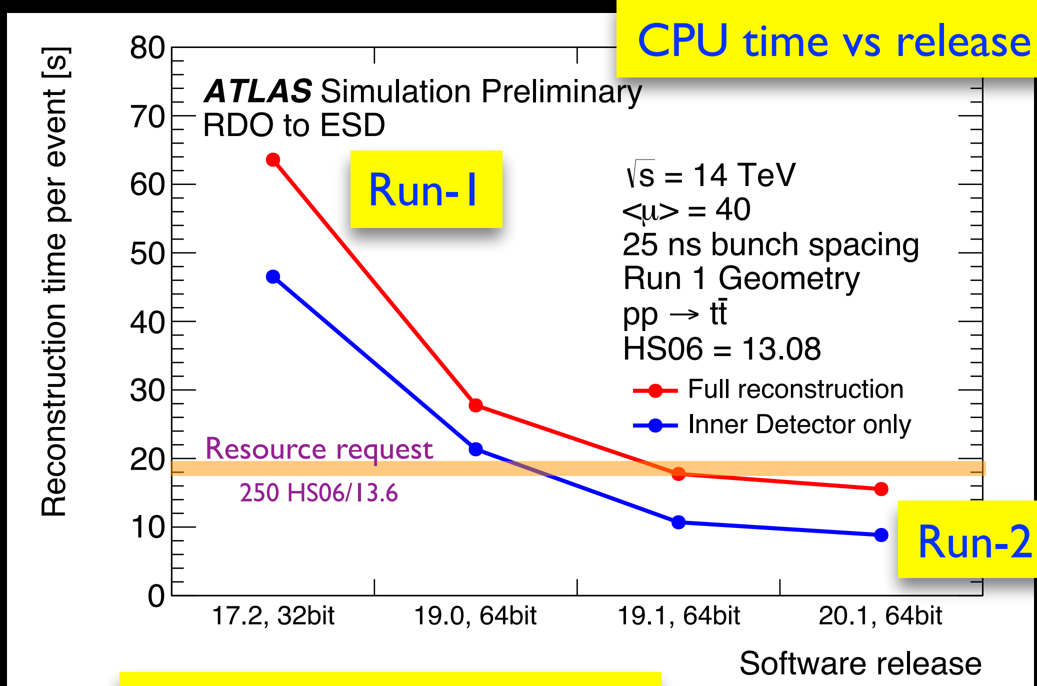


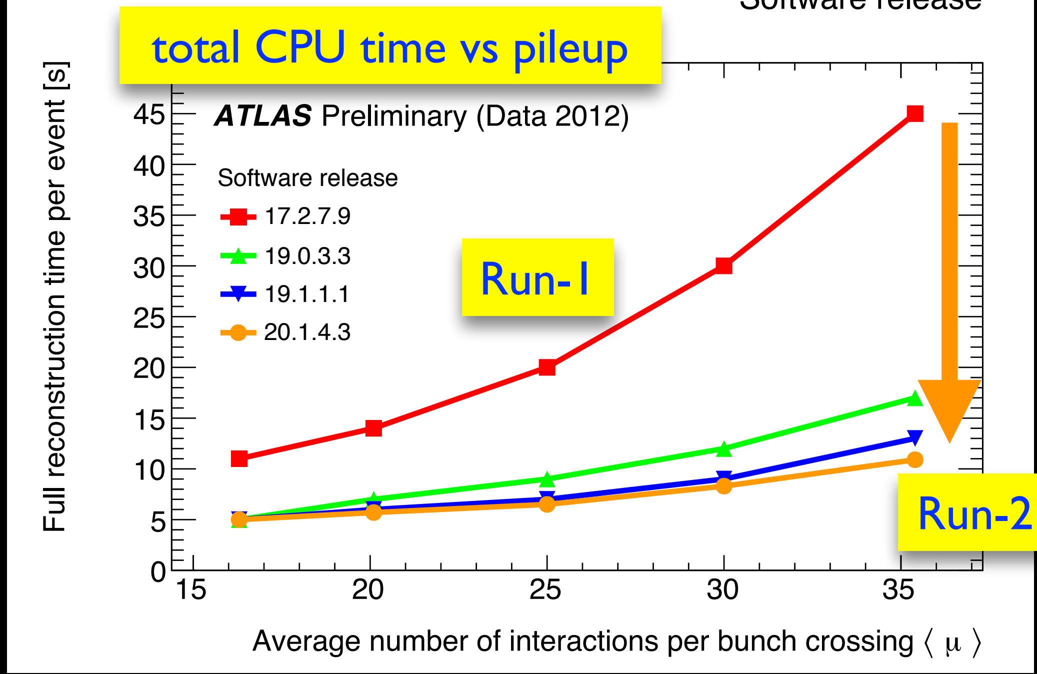


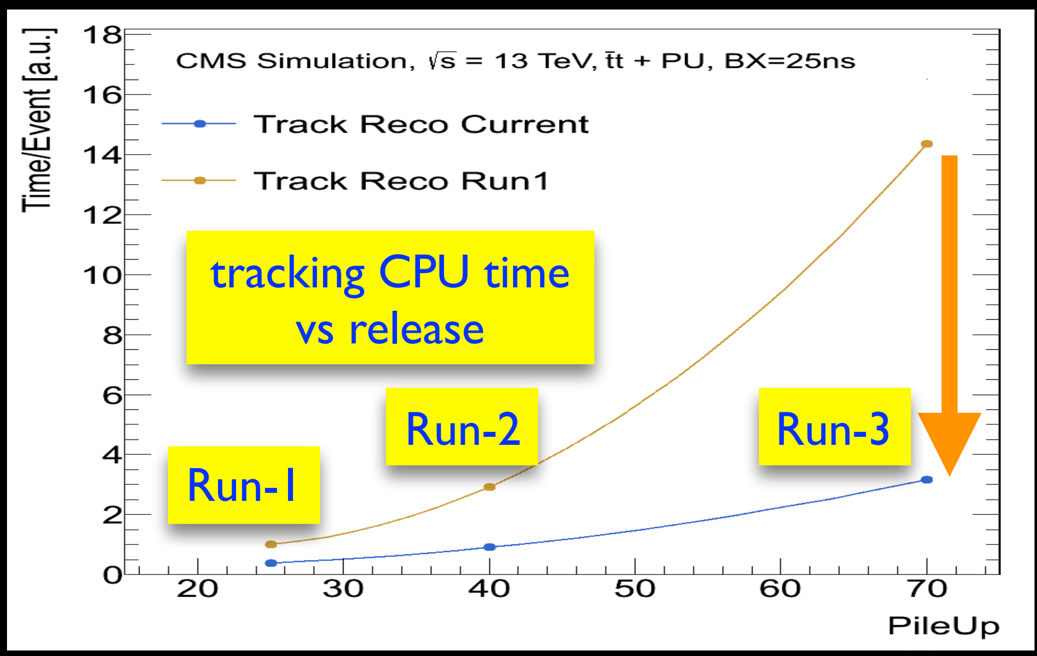


CPU for Reconstruction

- sum of tracking and general software improvements
  - ➡ improved software technology, including:
    - tracking related improvements
    - new 64 bit compilers, new tcmalloc
  - ➡ tune reconstruction strategy (very similar in ATLAS and CMS)
    - optimise track finding strategy for 40 pileup
    - faster versions of things like FastJet, ...
    - addressing other CPU hot spots in reconstruction
- huge gains achieved !
  - ➡ ATLAS reports overall factor > 4 in CPU time
    - touched >1000 packages for factor 5 in tracking
  - ➡ CMS reports overall factor > 2 in CPU time
    - on top of their 2011/12 improvements
    - as well dominated by tracking improvements
  - ➡ both experiments within 1 *kHz* Tier-0 budget
    - required to keep single lepton triggers



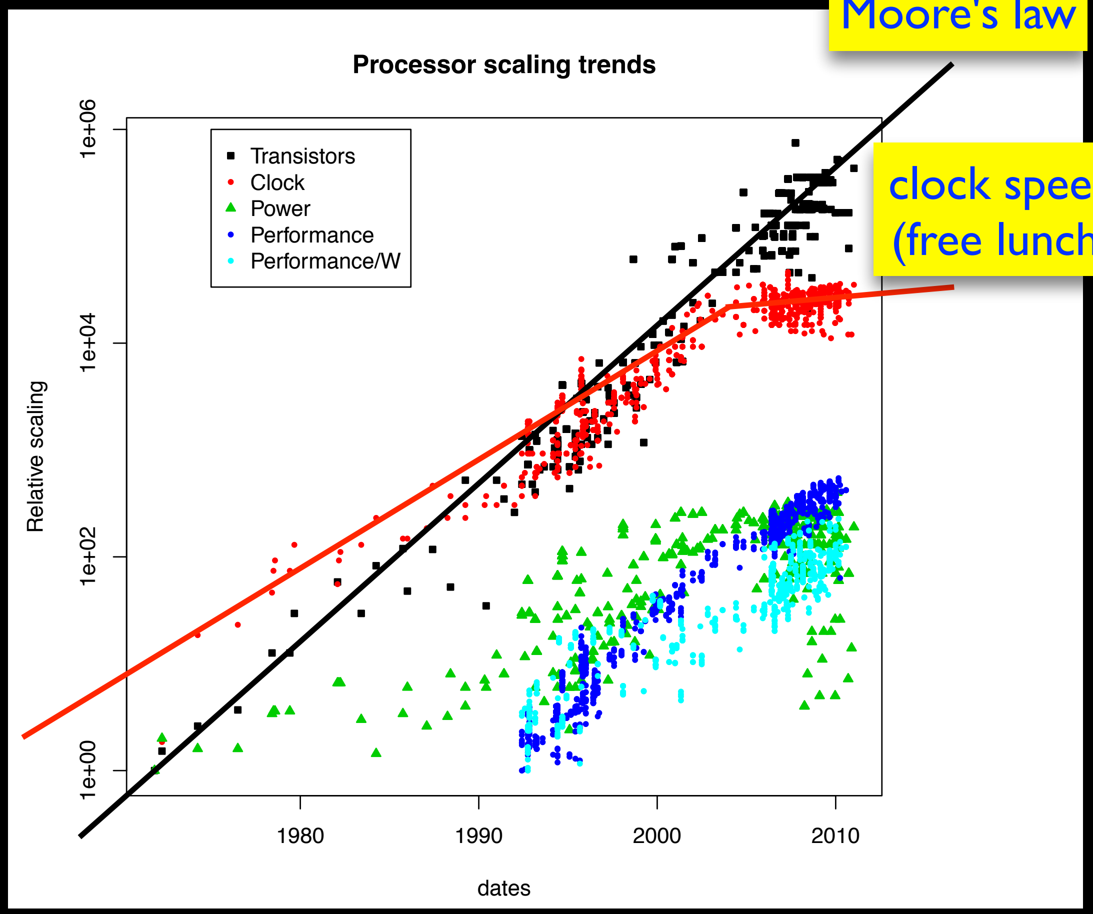




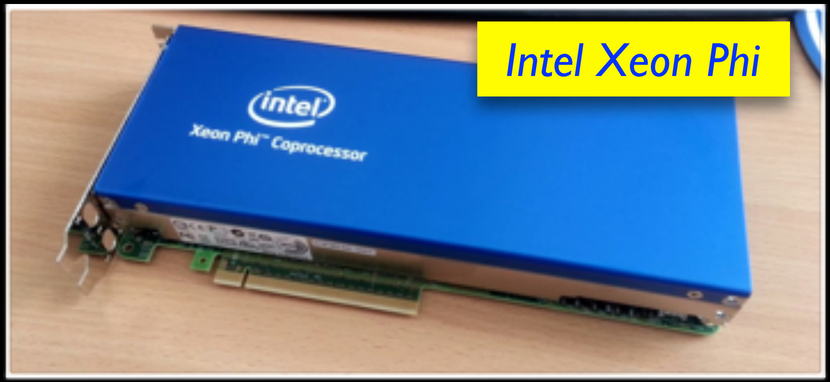
Markus Elsing

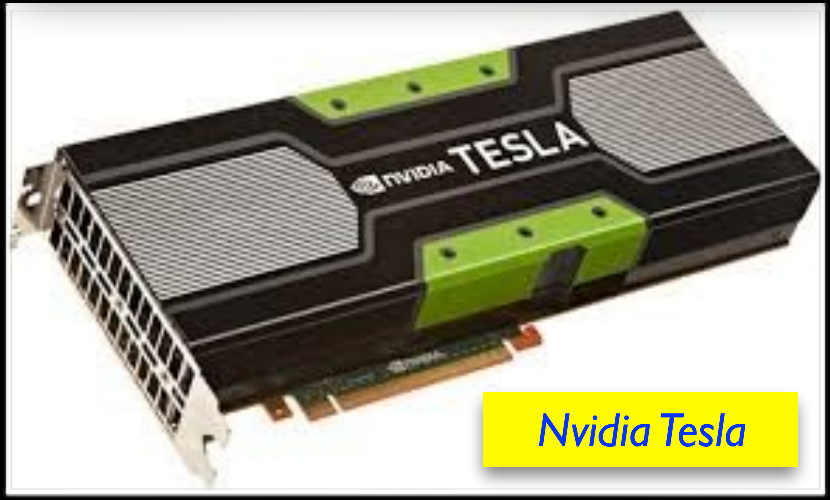
Technology Challenges

- Moore's law is still alive
  - ➡ number of transistors still doubles every 2 years
    - no free lunch, clock speed no longer increasing
  - ➡ lots of transistors looking for something to do:
    - vector registers
    - out of order execution
    - hyper threading
    - multiple cores
  - ➡ many-core processors, including GPGPUs
    - lots of cores with less memory
  - ➡ increase theoretical performance of processors
- challenge will be to adapt HEP software
  - ➡ hard to exploit theoretical processor performance
    - many of our algorithm strategies are sequential
  - ➡ need to parallelise applications (multi-threading)
    (GAUDI-HIVE and CMSSW multi-threading a step in this direction)
    - change memory model for objects, more vectorisation, ...

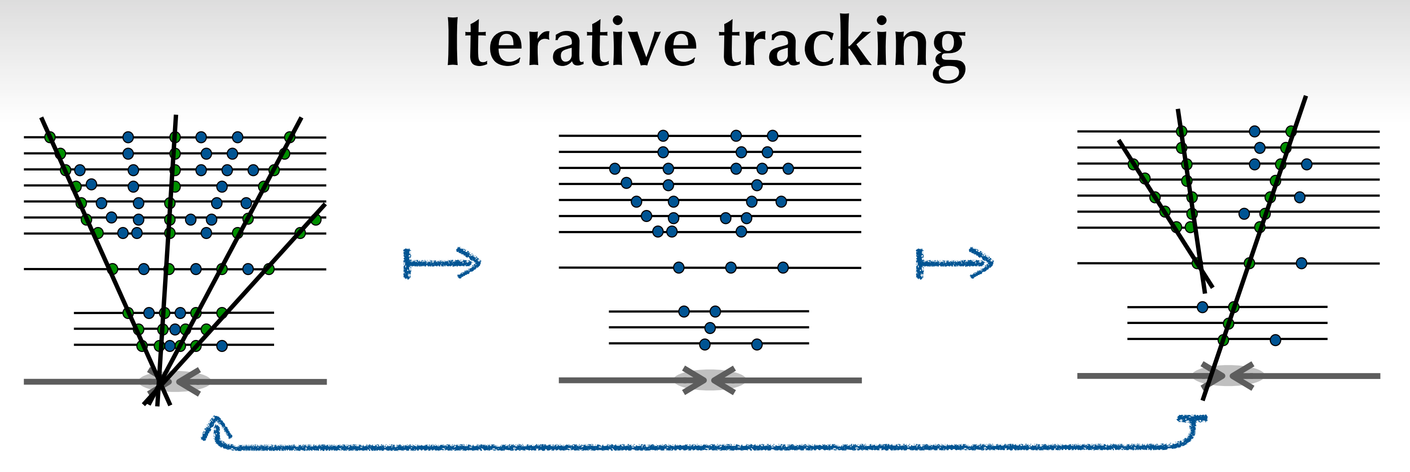


see G.Stewart, CHEP 2015





Massively parallel Tracking ?



- ATLAS/CMS tracking strategy is for early rejection
  - ➡ iterative tracking: avoid combinatorial overhead as much as possible !
    - early rejection requires strategic candidate processing and hit removal
  - ➡ not a heavily parallel approach, it is a SEQUENTIAL approach !
- implications for making it massively parallel ?
  - ➡ Amdahl's law at work:

$$\text{Time}_{||} = \text{Para} / N + \text{Seq}$$

  - ➡ iterative tracking: small parallel part Para, heavy on sequential Seq
    - hence, if we want to gain by a large N threads, we need to reduce Seq
- hence we need to re-think the algorithmic strategy
  - ➡ having concurrency in mind from the very start
  - ➡ as well, look outside the box, e.g. explore using **machine learning** techniques