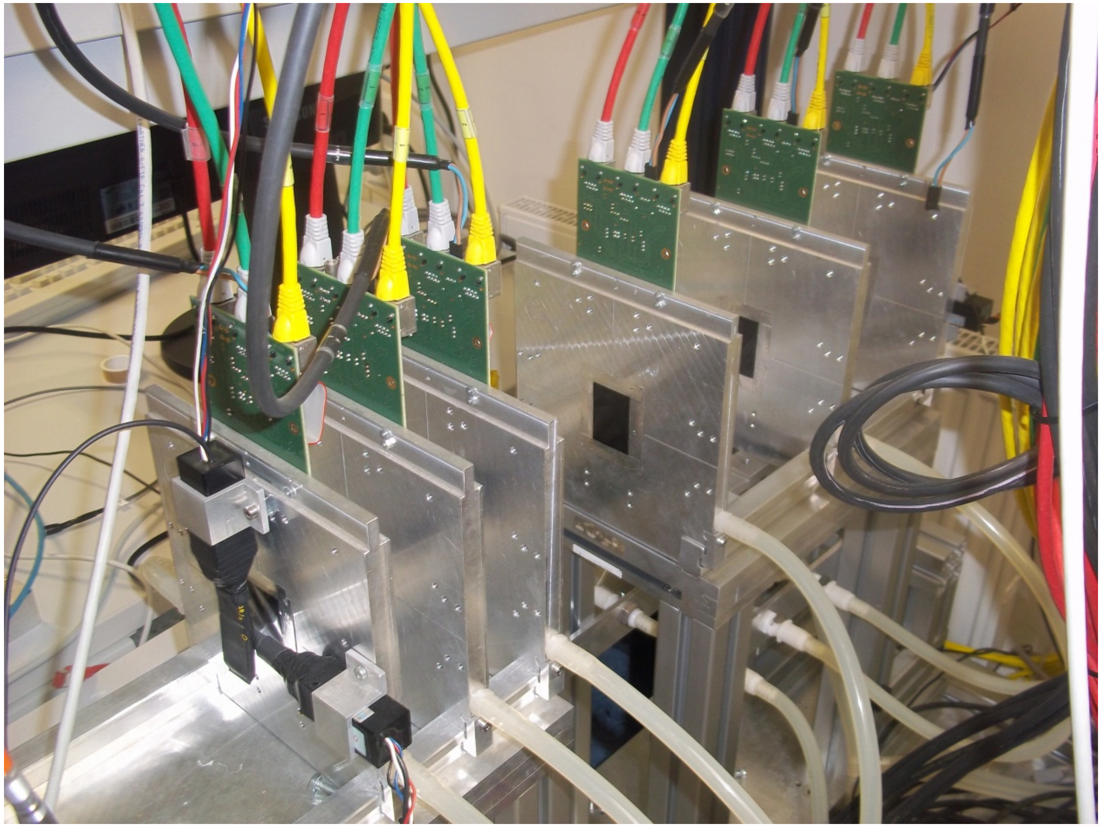
Development of General Broken Lines Track Fitting with EUTelescope

Workshop 1/07/2014



Alexander Morton, DESY
ATLAS

The People to Blame

- General Broken Lines (GBL) algorithm written by Claus Kleinwort.
  - Thanks to all those involved in its creation.
- Daniel Pitzl. Advocate for its use with telescope track reconstruction.
  - Created first working example
- Denys Lontkovskyi reimplementation within EUTelescope
- Work has been continued by myself and Igor Rubinskiy.
  - Refactoring much of the code.
  - Fixing previously unseen problems.
- Example shown today will be in jobsub/examples/GBL
  - Not in the current version but soon!
  - Detailed information on how to run this will also be kept there.

- See last year workshop for more details
  - https://indico.desy.de/conferenceDisplay.py?confId=7597

Motivation

> We want a general purpose track fitter and alignment procedure.

> Must consider:

- Different orientations of sensors,
- Correct radiation lengths of all volumes and how scattering is modelled,
- Equations of motion with magnetic fields.

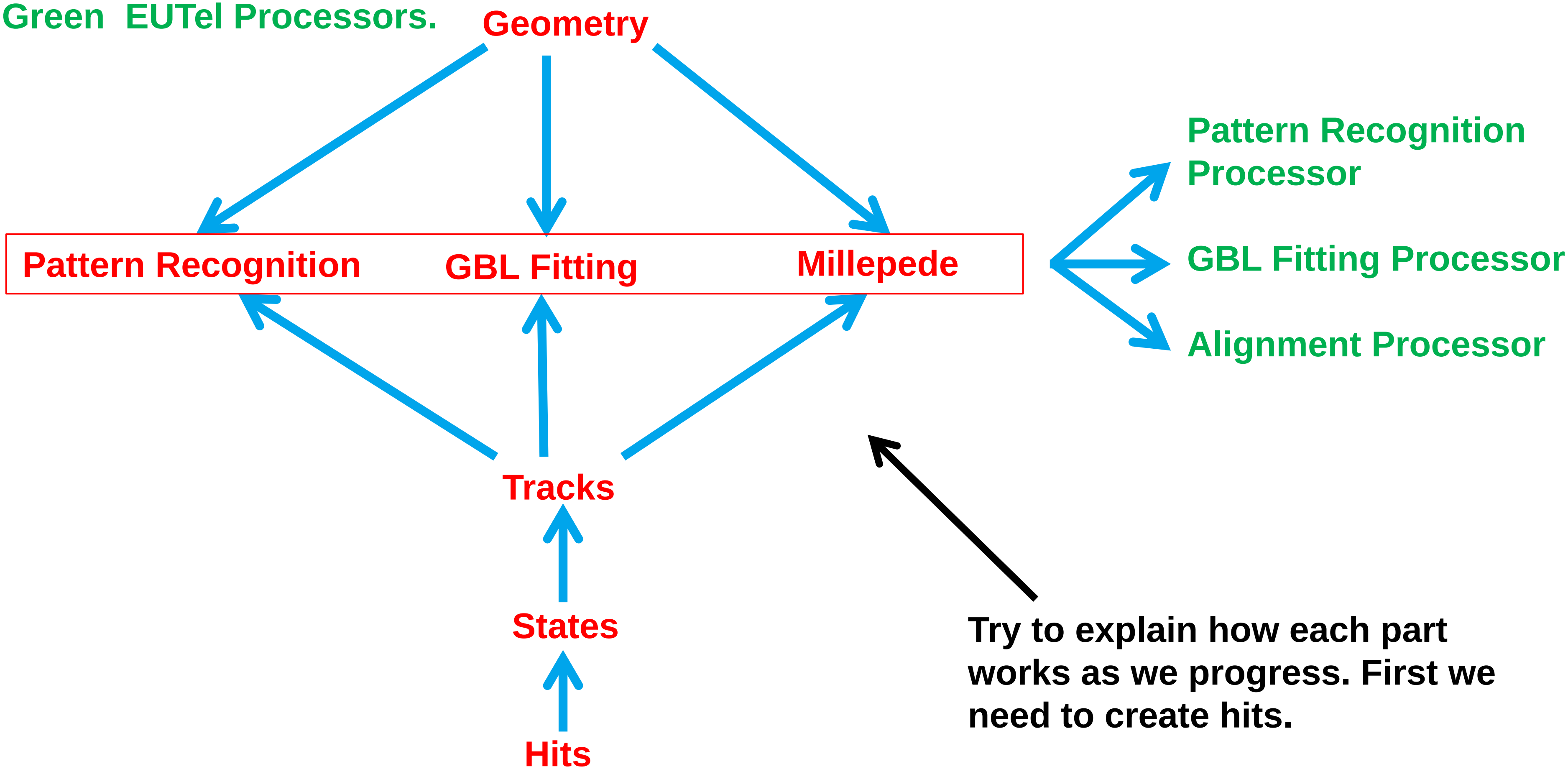
> This is done by factorizing existing code and creating new features.

> Code itself can be split into these areas:

- Geometry,
- Pattern Recognition,
- GBL track fitting,
- Alignment using MILLEPEDE .

EUTelescope after Hit Making and with GBL.

Parts are related as so:
Red different classes,
Blue indicates used by,
Green EUTel Processors.



Try to explain how each part works as we progress. First we need to create hits.

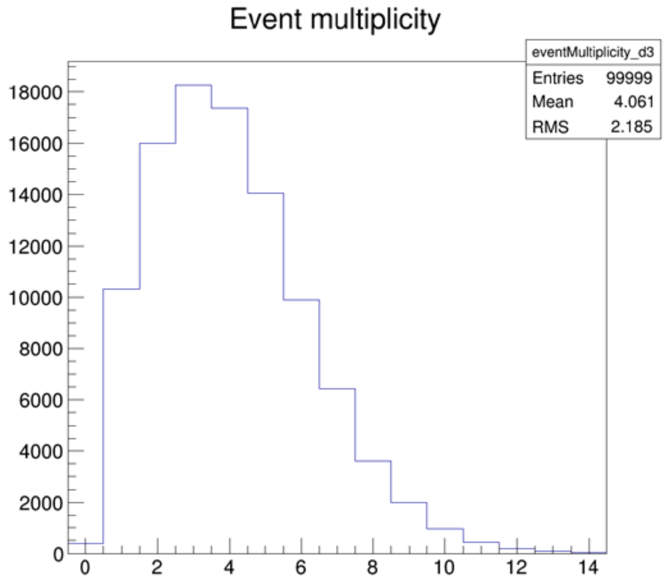
DESY

Conversion, Clustering and Hit Making.

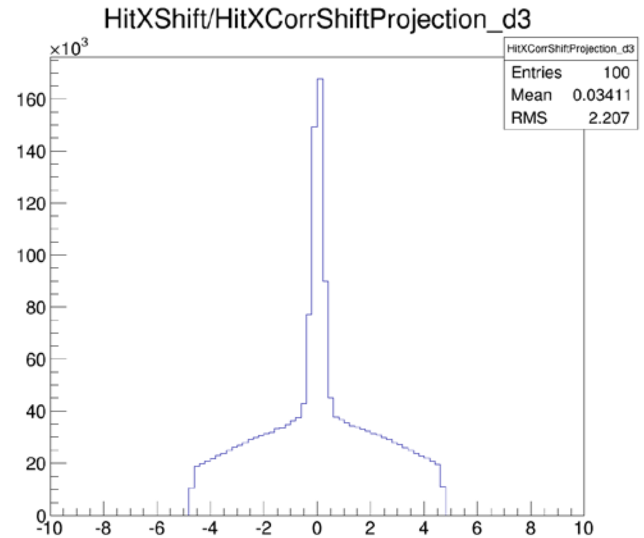
> Conversion from raw to LCIO output with no DUT and 6 Mimosa Planes.
  - Data shown here is for a beam energy of 5 GeV and a threshold setting of 6.
  - No magnetic field.

> Clustering and removal of hotpixels.

> Create hit is local coordinate system (Measurement system).



Number of clusters per event



Residual of hit positions from sensor 0

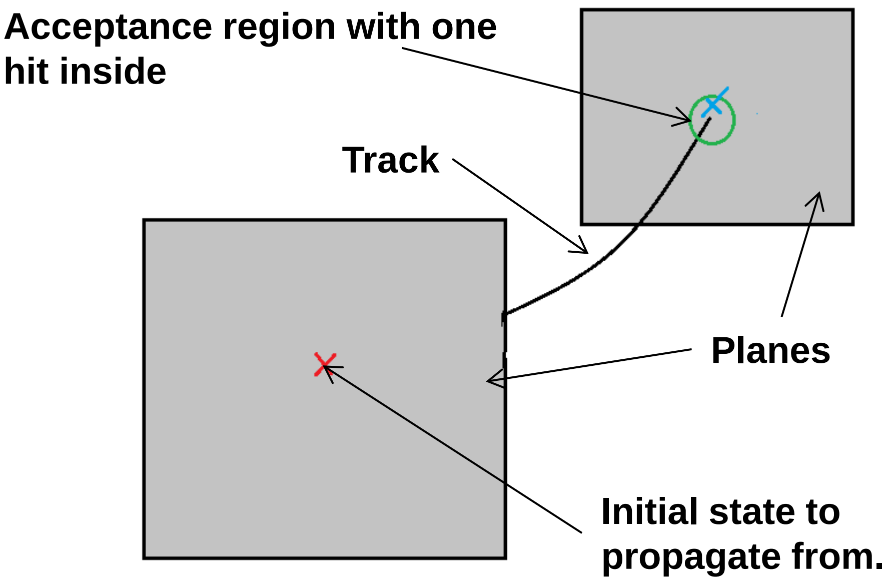
> Everything looks good so now we create our initial trajectory.

Pattern Recognition

> We need an initial trajectory to give to the GBL processor.
> A simple algorithm is used to detemine this initial trajectory.
  - Propagate from a plane to the next, using the state information and equations of motion.
  - Look for hit within some region.
  - Using hit and track covariance matrix update trajectory. (Kalman Filter Part).

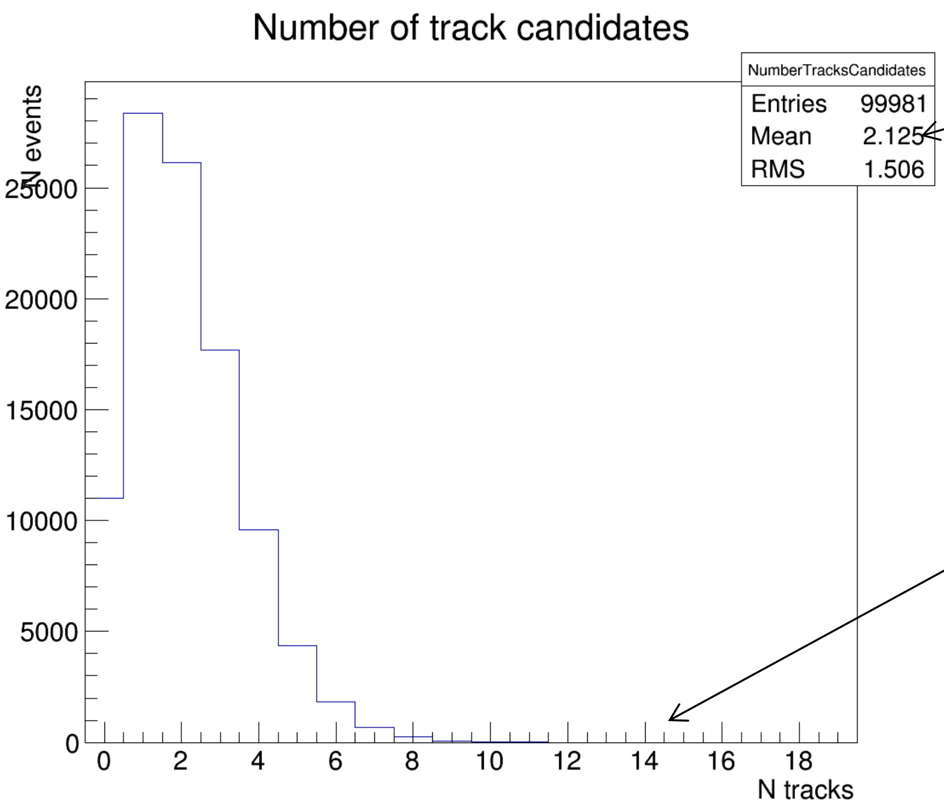
> Track contains states.
> A state contains:
  - A hit,
  - Estimate of momentum and position,
  - Other functionality.

> Cuts on tracks are:
  - Tracks that have too few hits,
  - If tracks share the same hits.



Pattern Recognition without Magnetic field

> **Cuts made on tracks**
  - **All 6 planes must have a hit within 500 microns and no similar hits**

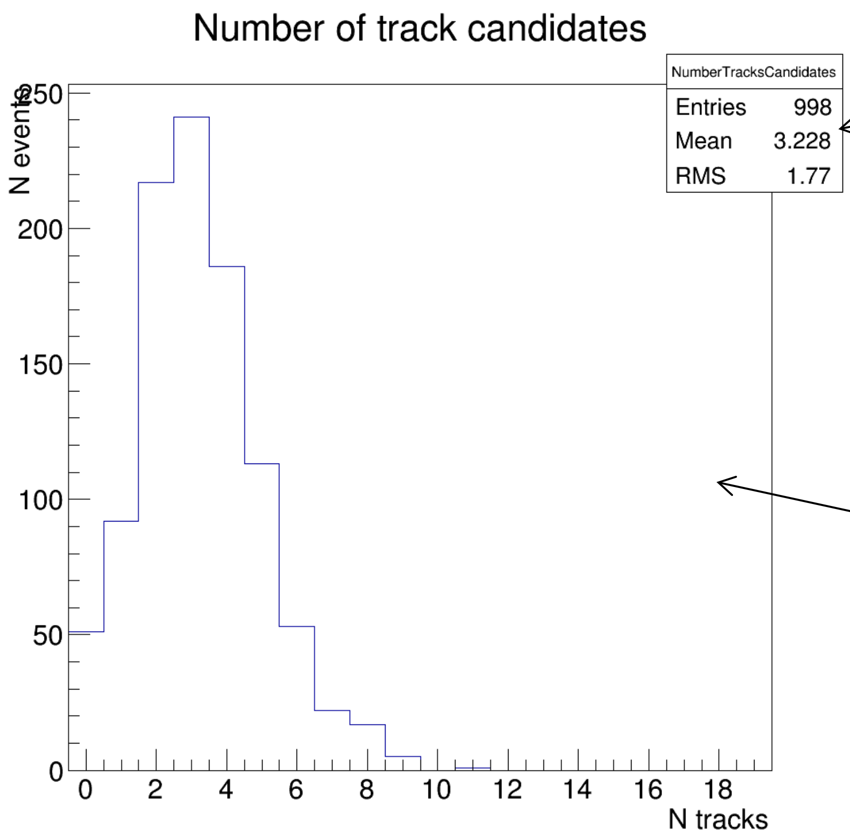


Mean reasonable after cuts taken into account.

With noisy data you can see some high track numbers here

Pattern Recognition with Magnetic field

> **Cuts made on tracks**
- **Reduce the stringency of the cuts**
- **All planes must have a hit within 1000 microns and only two similar hits**



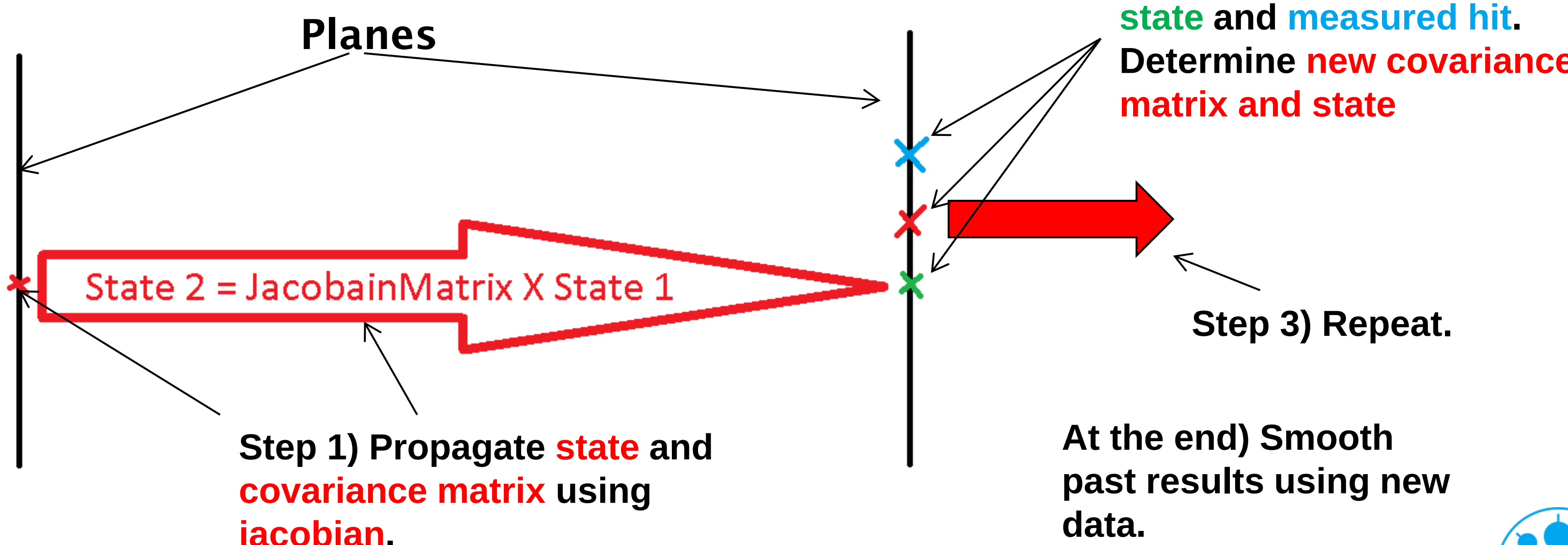
Mean looks a bit too high

Still need to test this with GBL fitter

General Broken Lines and Kalman Filtering

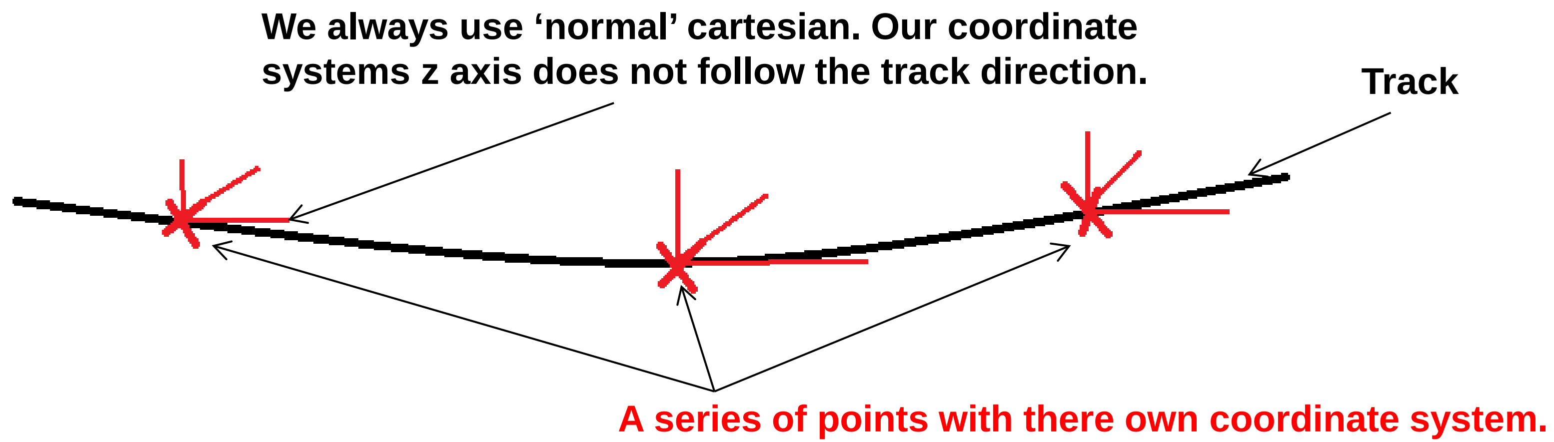
- Track fitting made complicated by multiple scattering and magnetic fields.
- Many different algorithms exist that can take these into account.
- The Kalman filter is the optimal linear filter.
  - Minimizes estimation error better than any other linear methods.
- GBL is mathematically equivalent to a Kalman filter. However computationally different.

Kalman Filter in action



Requirements to Fit Trajectory with GBL

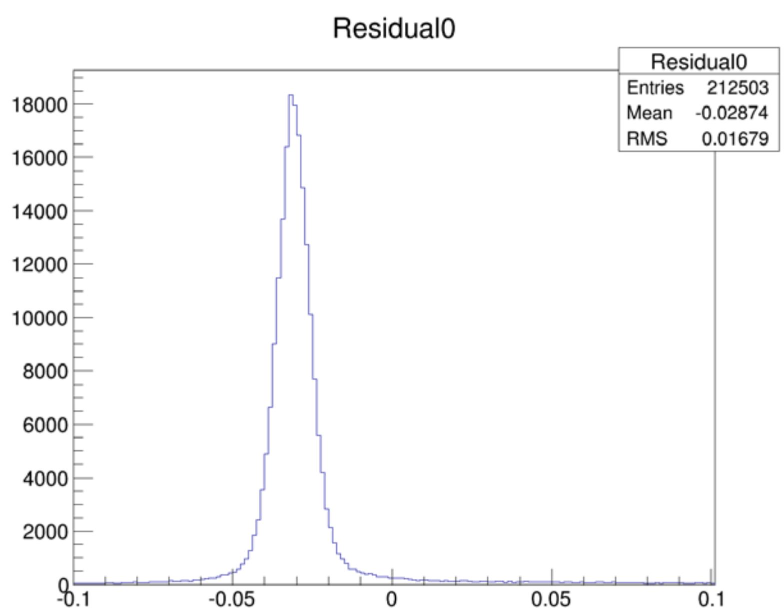
- A Trajectory is composed of a list of GBL points (states) ordered by arc length on the initial seed trajectory.
- A GBL point (state) is some position in the global frame of the telescope that has its own coordinate system.



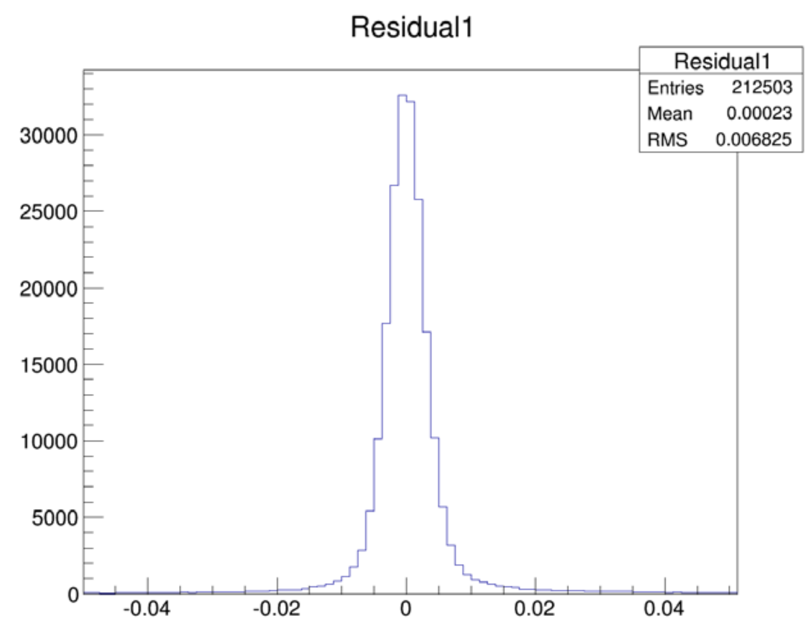
- Each point must contain:
  - Jacobean that relates change in previous point parameters to this one,
  - Projection matrix from curvilinear system to measurement system (local system) .
- Each point can contain:
  - A measurement with respect to the initial trajectory (Residual) and variance,
  - A scatterer which contains information on scattering angle and variance.

The GBL Fitter in Action

- Putting this into use with our initial trajectory.
- Create GBL points from scatterers and measurements.
  - Scattering information calculated from new geometry class and Highland formula.
  - Model mass inbetween measurements as two thin scatterers.
- As output get correction to initial trajectory and new residuals.

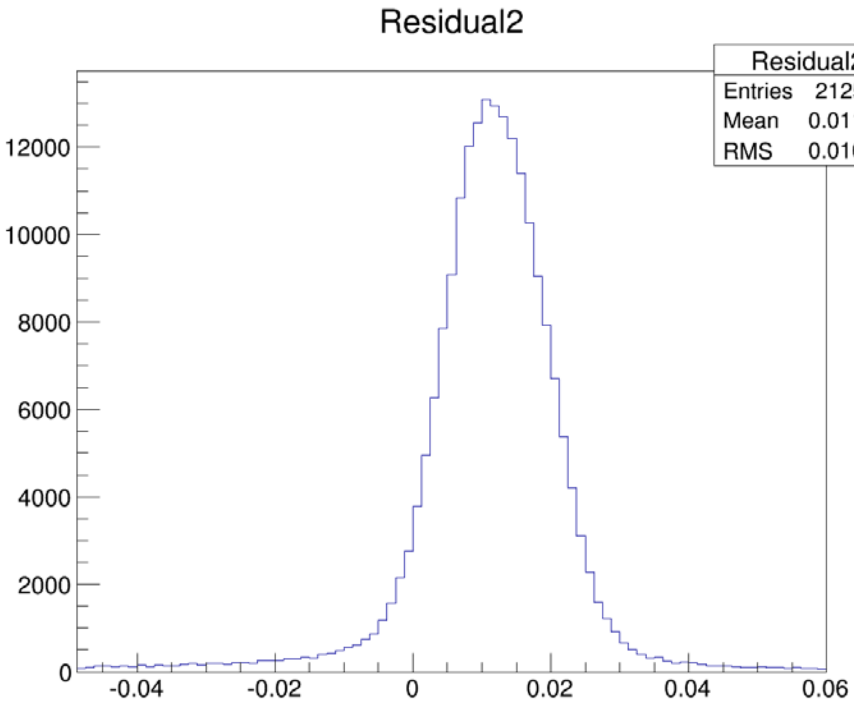


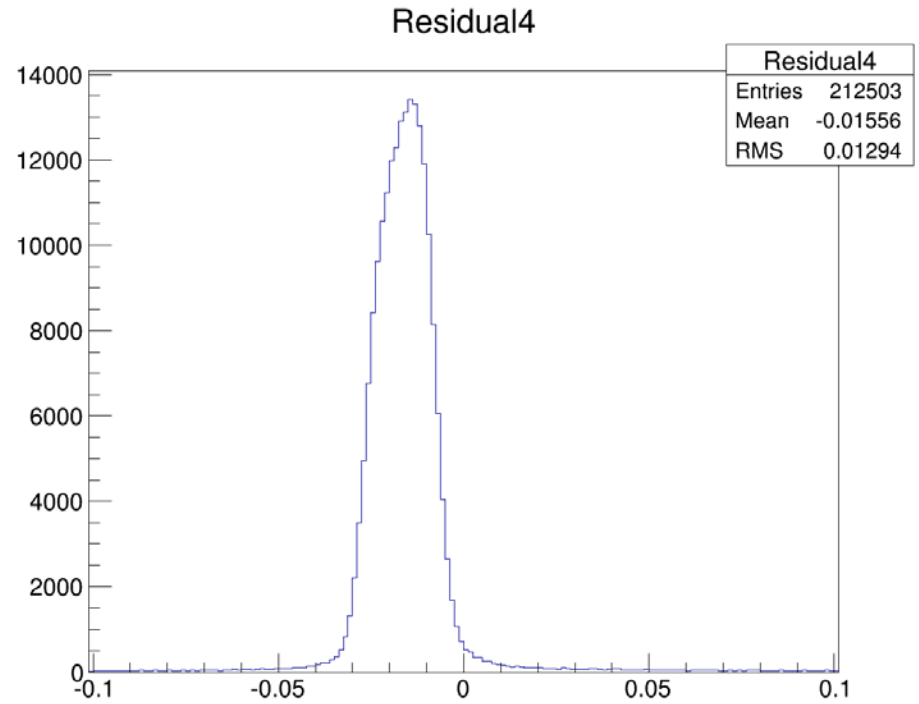
Track – Hit position (mm)



Track – Hit position (mm)

More Residuals





Track – Hit position (mm)

Track – Hit position (mm)

> **Of course the residuals are offset and with large variance due to misalignment.**

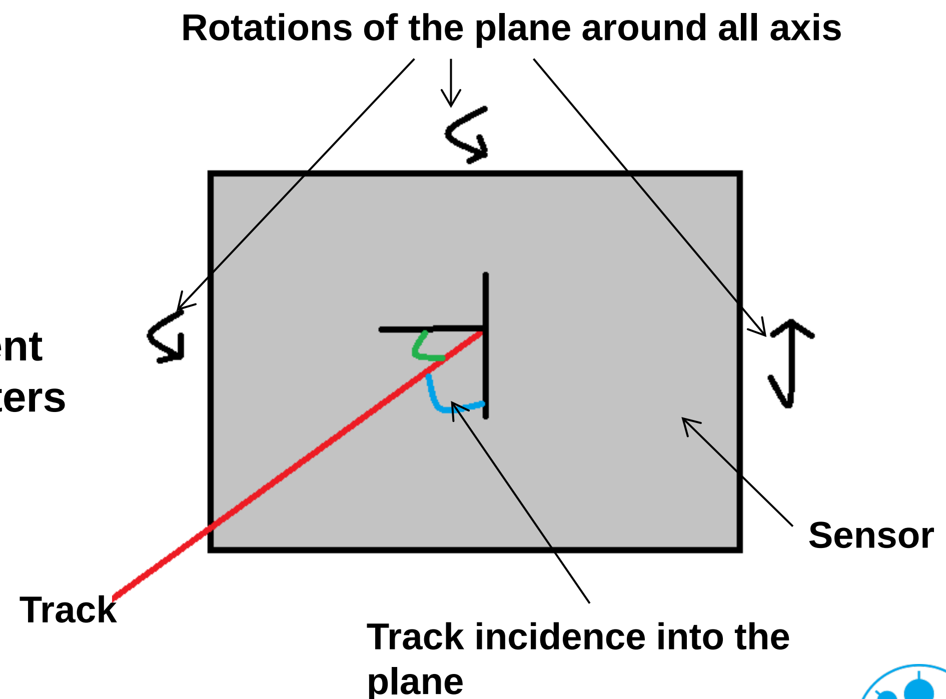
> **Now we have tracks we can move on to alignment.**

Alignment with MILLEPEDE

- > Millepede is used to solve least squares fit problems.
- > GBL software can write a binary file for millepede to run on.
  - Millepede must be given alignment jacobian which depends on sensor and state.
- > GBL software knows nothing about how moving a sensor will affect track hit positions.
  - This information must be calculated separately.

Track Position (local) = Alignment Jacobian ✖ Alignment Parameters

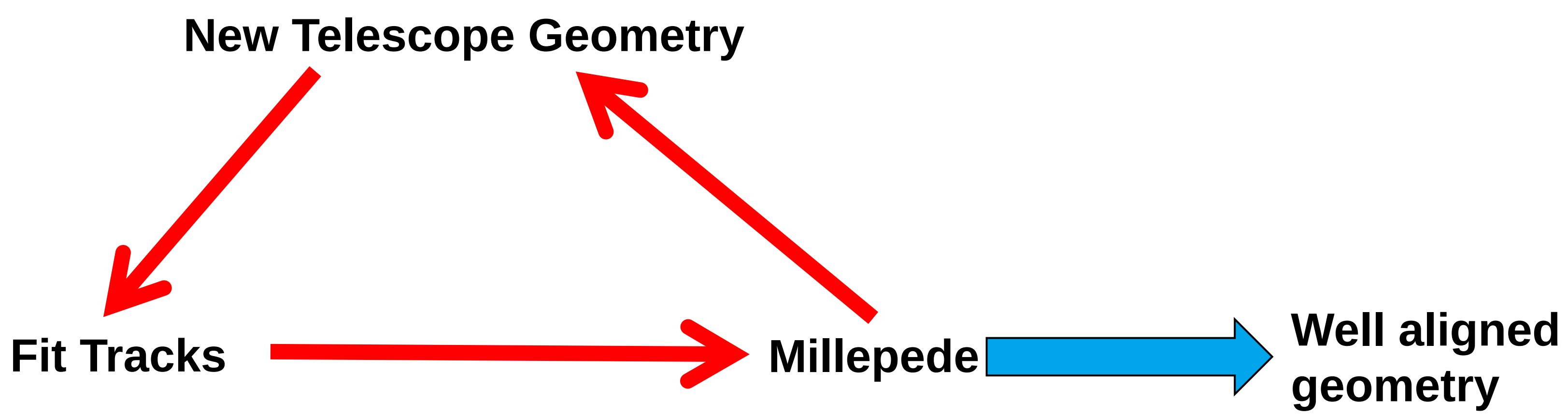
This is what MILLEPEDE needs contained in binary file.



Iterative Alignment

- Can often have problems with alignment with MILLEPEDE.
  - Too many free parameters.
  - High number of track rejection.

- This can be helped by iterative alignment.
  - Fix some planes or alignment parameters.
  - Vary the covariance of your input hit.



Known Issues and Things to Do

> **Alignment:**
>
> - Still not getting residuals as expect.
> - Worked for past versions of this code so not likely something fundamental.

> **Pattern Recognition:**
>
> - Currently we use a semi form of a Kalman filter. Is this a good idea?
> - Are there other clever forms of pattern recogntion that would serve this purpose?

> **GBL fitter:**
>
> - Still need to test new code with magnetic field and tilted sensors.
> - Clean up some of the code.

> **So still some work to do but close to the final result.**

> **Hopefully we can learn from other peoples experience here!**

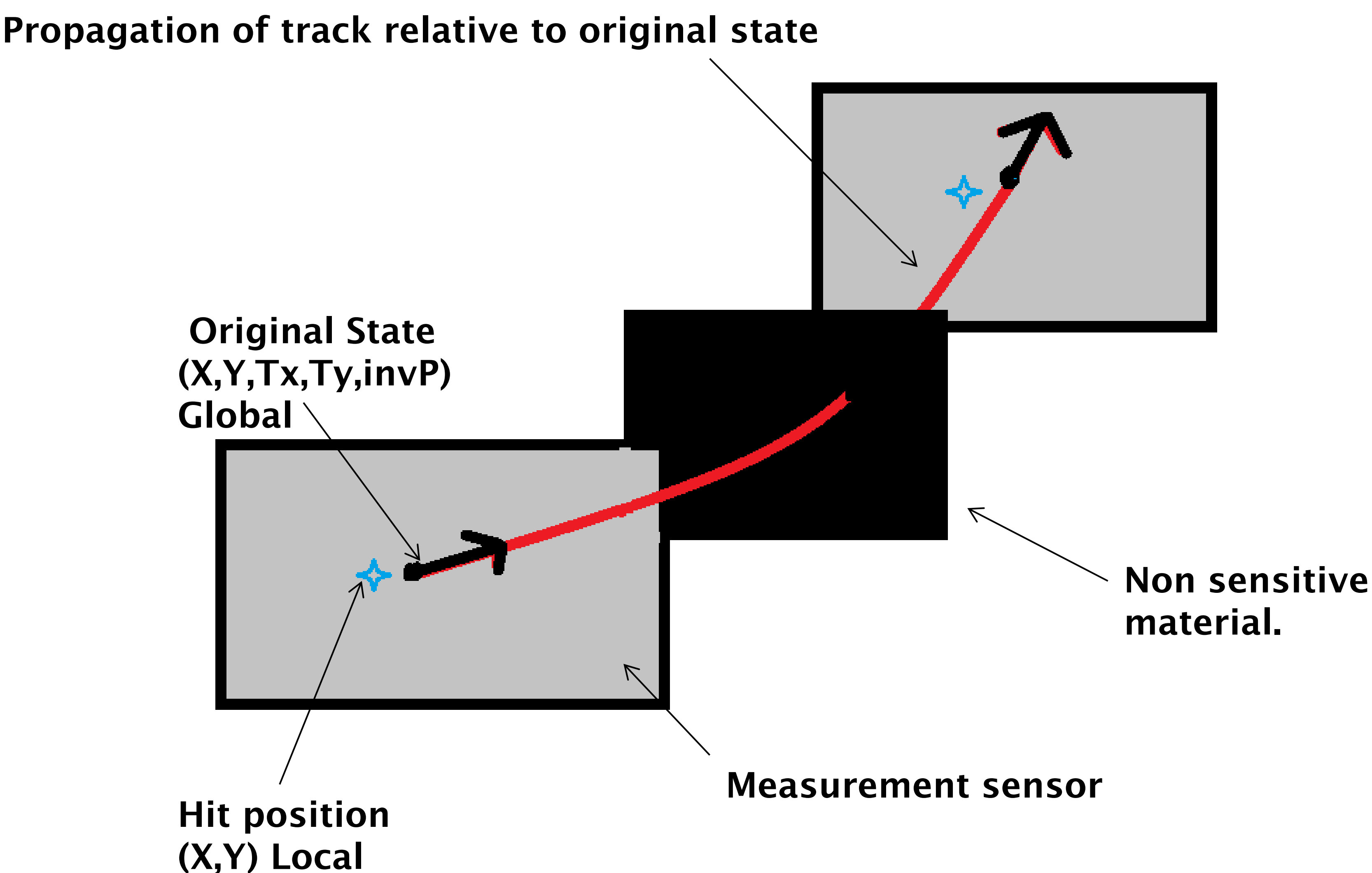
Thank you!

BACKUP SLIDES

Alignment Jacobian

- Here is the alignment jacobian input as global parameters by GBL into millepede
  - Assumes linear track close to the sensor

The setup

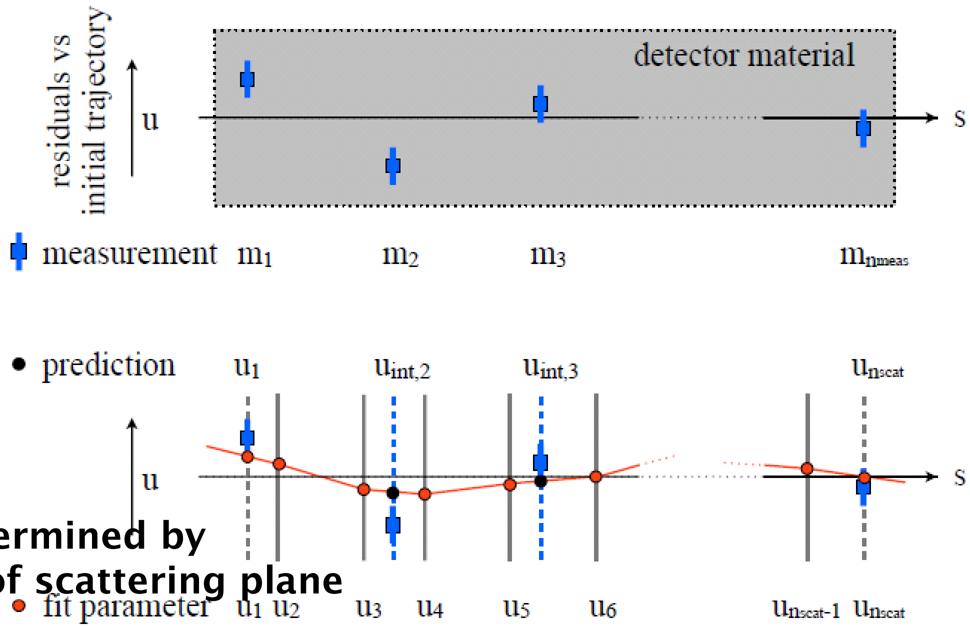

Propagation of track relative to original state
Original State
(X,Y,Tx,Ty,invP)
Global
Non sensitive
material.
Measurement sensor
Hit position
(X,Y) Local

> We fit all offsets and one common curvature correction q/p

- Measurements determined from interp

Prediction determined by interpolation of scattering plane on either side

Prediction determined by interpolation of scattering plane on either side



Prediction determined by interpolation of scattering plane on either side

Pattern Recognition with magnetic field

> **For noise sensor we still have some issues**