Kalman Filter Pattern Recognition and Fitting

Robert Johnson

U.C. Santa Cruz

Status: November 19, 2019

Outline

- Objective
- Where and what it is
- The interface to hps-java
- Development and Testing Methodology
- Mathematical Verification with Idealized Simulation
- Pattern Recognition with Idealized Simulation
- Pattern Recognition on Geant-4 MC Events
- Studies of Fit Quality Issues
- Conclusions

Objective

Create a new pattern recognition for HPS that

1. operates from "1-D" hits, for improved efficiency (axial-stereo pairs for making 3-D hits then are not needed in every layer used by the pattern recognition).
2. picks up or removes hits based on a complete fit that accounts for all measurement information and the expected amount of multiple scattering.
3. takes into account the non-uniform nature of the magnetic field.

I am using the *Kalman-Filter formalism* to implement this.

The code is new from the ground up, but I made use of documentation for the KalTest code, as well as the old Fruehwirth paper, to help to implement the mathematics.

Where and What

- Everything is in the package *org.hps.recon.tracking.kalman*
- Currently it is in branch *iss204d*
- Included is a pdf file with extensive *documentation*, although that is a work in progress as the code develops.
- Several of the code classes are dedicated to stand-alone testing and utilities that will never be loaded into hps-java.
- The driver KalmanDriverHPS, written mostly by Miriam, is intended for refitting GBL tracks using the Kalman Filter, for comparison. I don’t think that ultimately this will be used.
- I adapted the code to make a new driver called KalmanPatRecDriver.

The Interface to hps-java

- All of the interface code is collected in the class created by Miriam: KalmanInterface.java.
    - Various transformations of coordinates and helix parameters
    - Loading HPS geometry into the Kalman classes
    - Loading the HPS 1D hits into the Kalman classes (for a given GBL track, for all readout hits, or for MC true hits)
    - HPS B-field map
    - Calling the Kalman fitting or pattern recognition
    - Loading found and fitted tracks back into HPS collections
- Other than the field map, the Kalman code does not access any external libraries.
    - Miriam did some coding magic such that the Kalman code accesses the standard hps-java field map when interfaced but uses a different routine to read in and access the map when running stand-alone.

Development and Testing Methodology

- I did all of the initial development and testing running stand-alone, using an idealized simulation of tracking measurements. This capability is maintained.
  - Runge-Kutta integration of MC trajectories through the HPS field map.
  - Copied silicon wafer locations and orientations from hps-java.
  - Gaussian multiple scattering at each silicon layer.
  - Gaussian smearing of true intersection points to produce hit "measurements."
  - Random noise hits and measurement inefficiency.
- Reasons for doing this:
  - Compiles, loads, and executes very rapidly in Eclipse, with full debugging capability.
  - Provides a rigorous test of the mathematics behind the track covariance matrices and chi-squared.

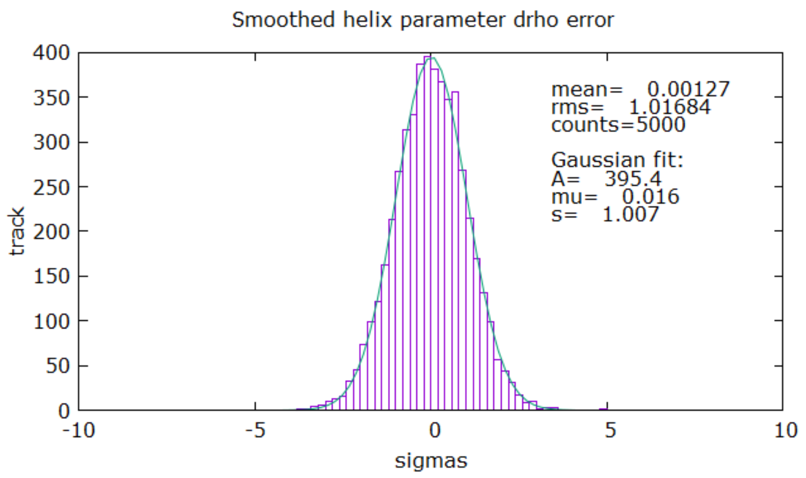
Verification of the Fitting Mathematics

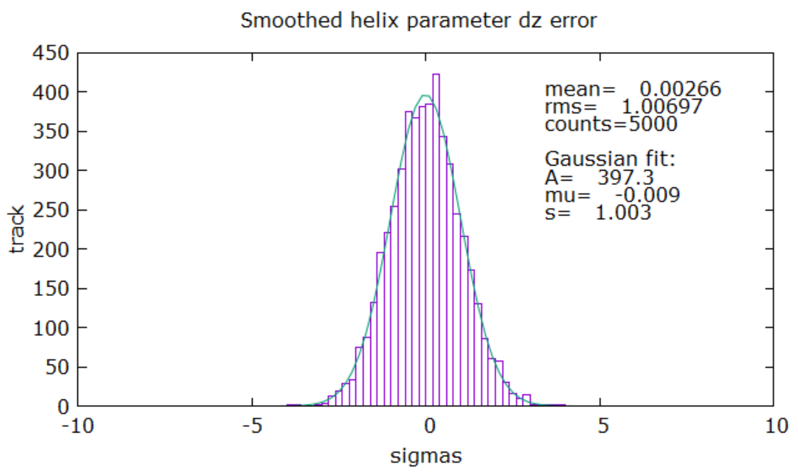
Single ~2.4 GeV tracks in the idealized simulation with no inefficiency and no noise, and with the 2016 B-field.

- Gaussian multiple scattering according to the PDG formula.
- 6-micron rms measurement uncertainty.

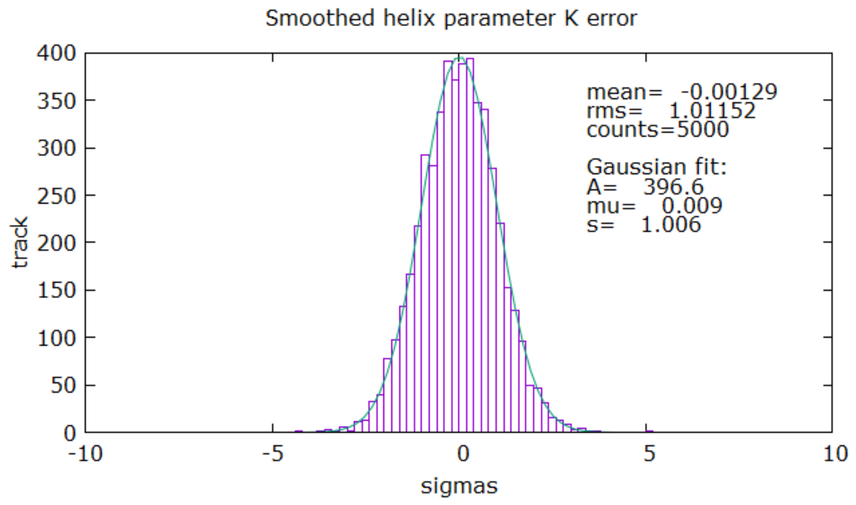
Pull distributions: difference from the MC true input divided, by the uncertainty prediction from the fit covariance matrix.

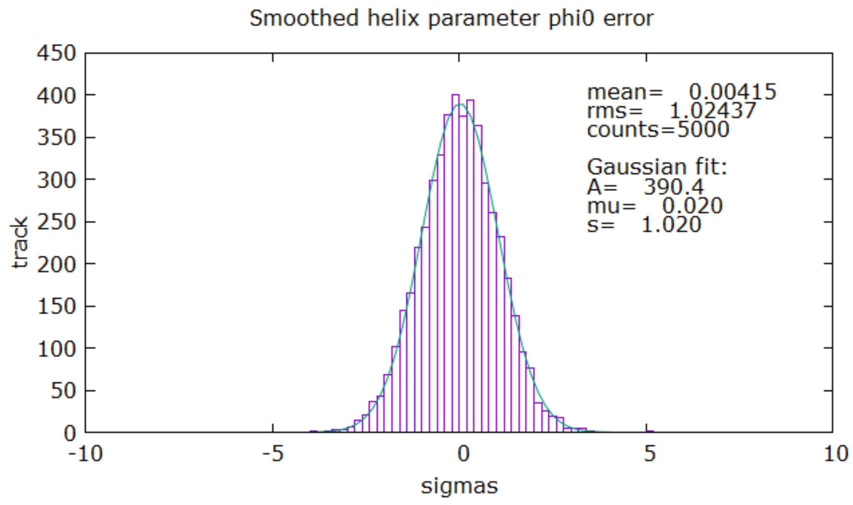
Note that this is done at the origin. The code extrapolates the track and its covariance matrix back to the origin (target vertex) through the non-uniform field by Runge-Kutta integration.

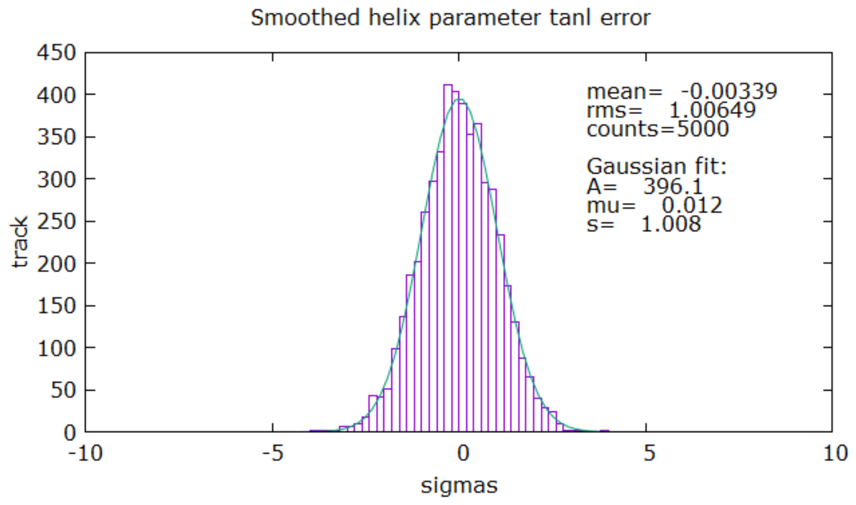




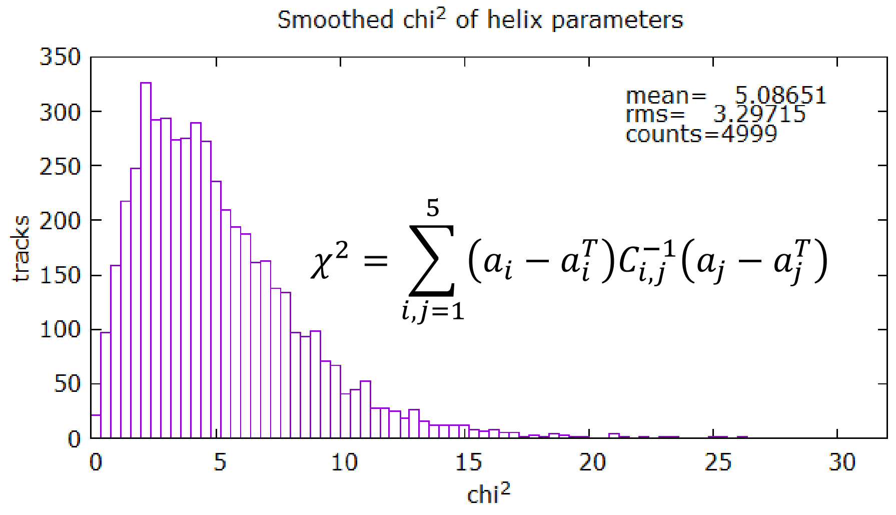
Verification of the Fitting Mathematics







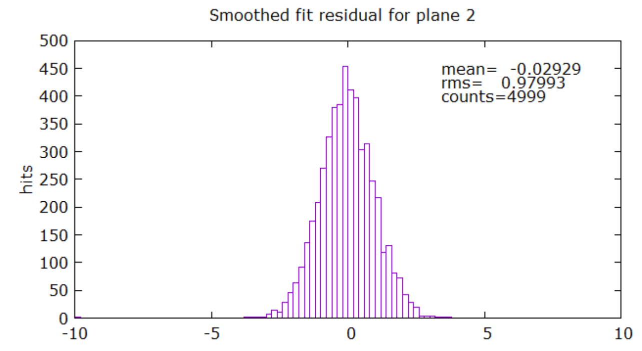
This histogram tests the full covariance matrix. It should by a chi-squared distribution with 5 d.o.f., i.e. $\mu = 5$ and $\sigma = \sqrt{10} = 3.2$

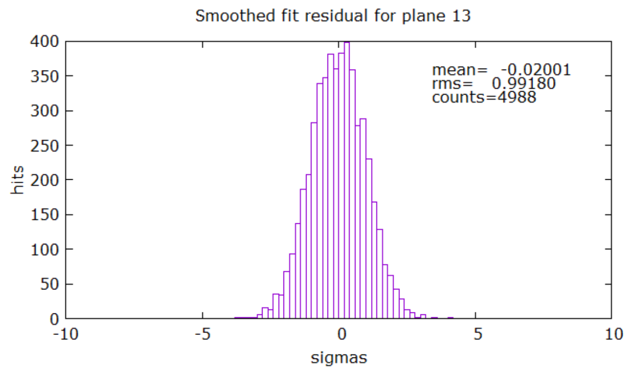


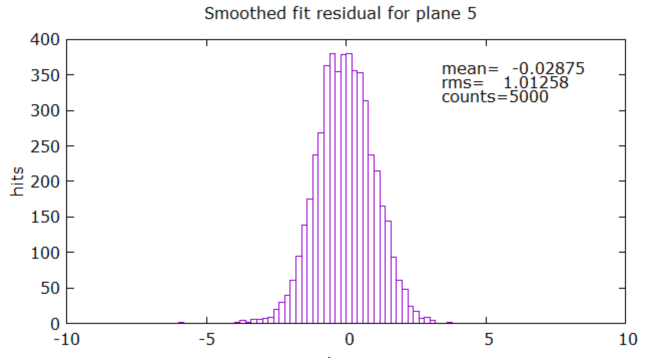
$$\chi^2 = \sum_{i,j=1}^{5} (a_i - a_i^T) C_{i,j}^{-1} (a_j - a_j^T)$$

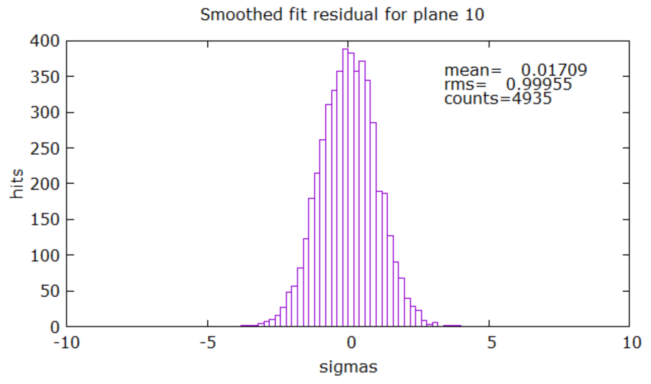
Individual Layer Residuals

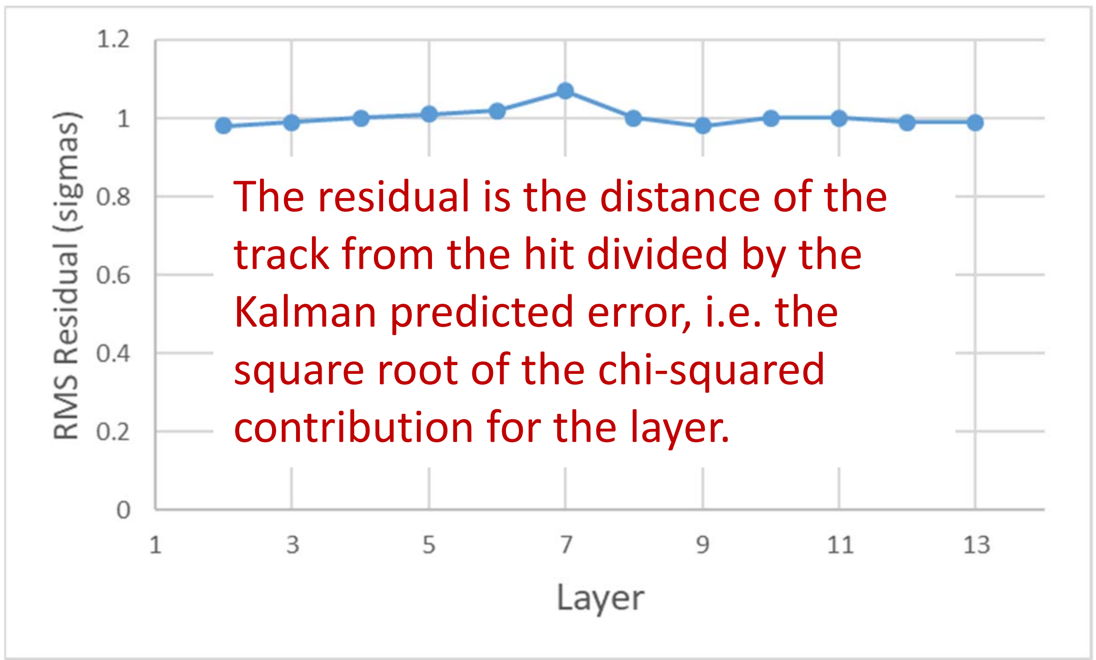
Note: here layers are counted 2 through 13 (with 0 and 1 reserved for the new 2019 tracking layer)





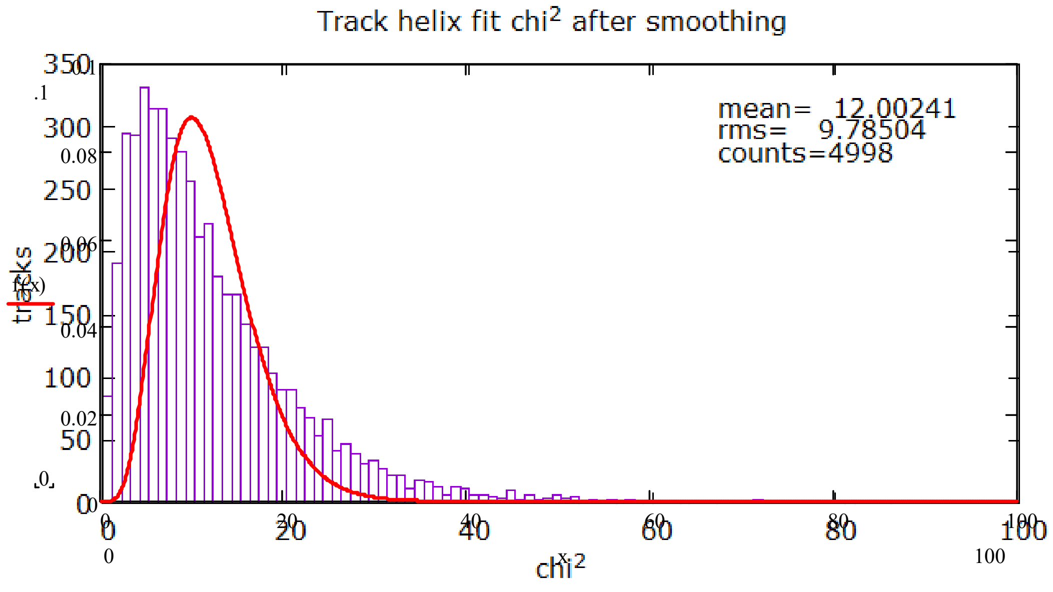


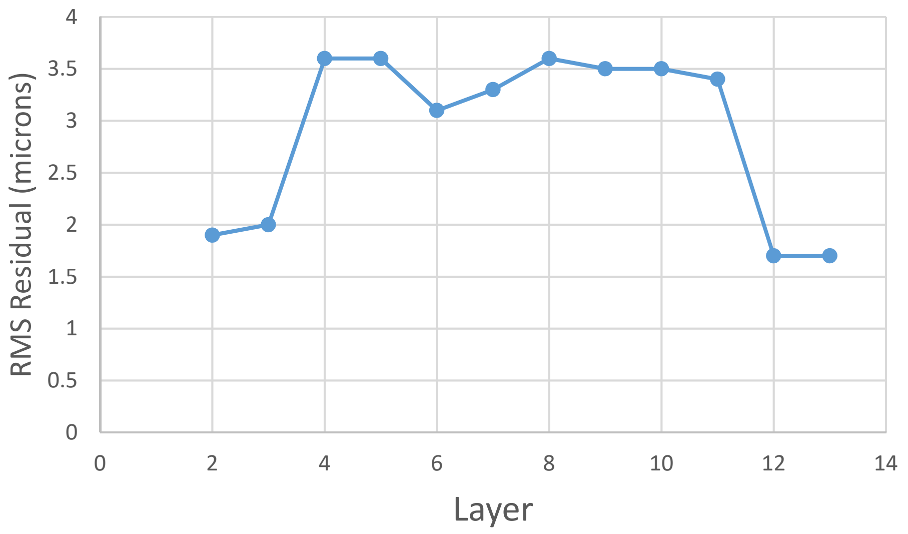




The residual is the distance of the track from the hit divided by the Kalman predicted error, i.e. the square root of the chi-squared contribution for the layer.

Individual Layer Residuals and Track χ^2





The track chi-squared has the expected mean of 12 for 12 d.o.f., but a true chi-squared distribution should have an rms of $\sqrt{24}$=4.9 and looks much more Gaussian.

(I made a simple Kalman Filter to do a purely linear fit to a straight line in two dimensions, and it showed the same behavior of the "chi-squared" statistic: the right mean but too large an rms.)

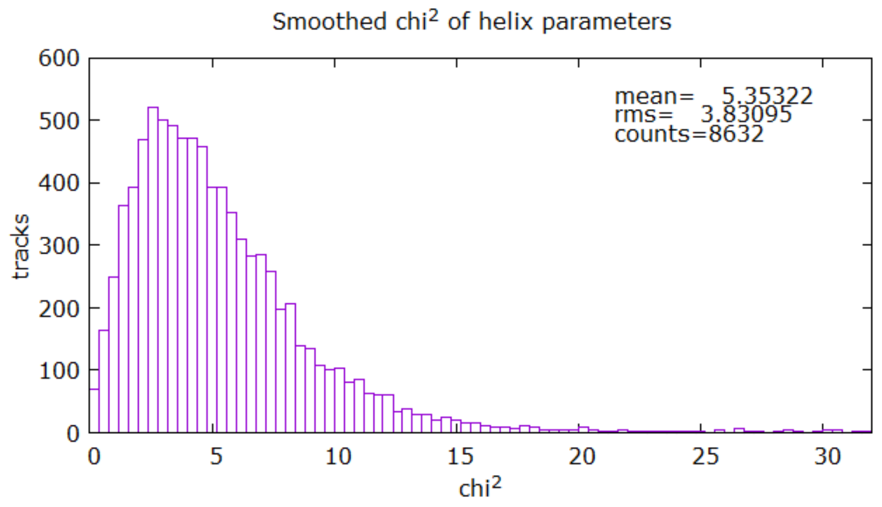
On the left, the biased residuals are much smaller than the 6-micron point resolution, partly due to the freedom afforded by multiple scattering.

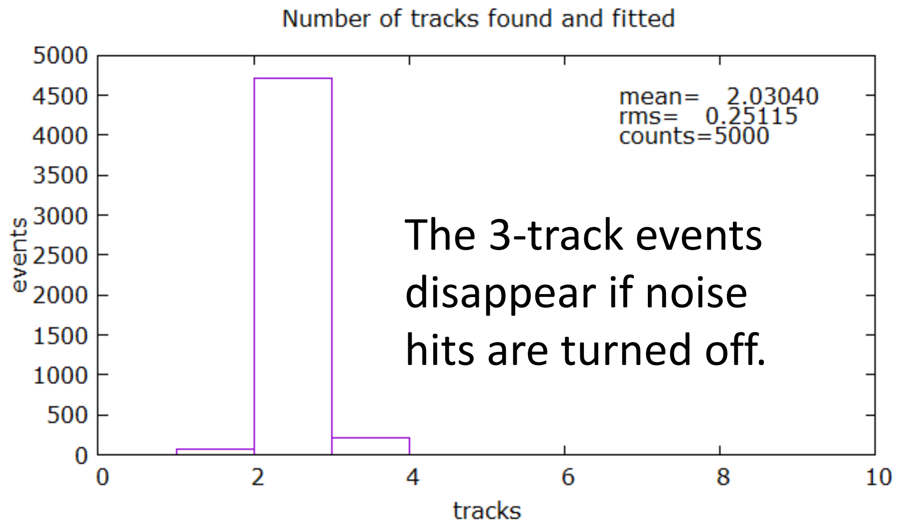
Combinatorial Pattern Recognition

1. Loop over sets of 5 seed hits, two axial and three stereo.
2. Make a 5-parameter 0 d.o.f. linear fit to a line and parabola.
3. Select and sort those that extrapolate near to the vertex.
4. Use the fit to seed the Kalman filter:
   a) Filter to the last layer, picking up the closest hits on layers.
   b) Smooth back to the starting point.
   c) Filter inward to the first layer.
   d) Repeat the fit, filtering to the last layer and smoothing back.
5. Keep candidate tracks of good quality and then arbitrate shared hits (allowing sharing only if the hit has a low chi-squared contribution to both tracks). *No duplicate tracks.*
6. Make another pass to remove hits of large chi-squared contribution, possibly adding others.
7. Final iteration of the fit.

Pattern Recognition Testing, Idealized

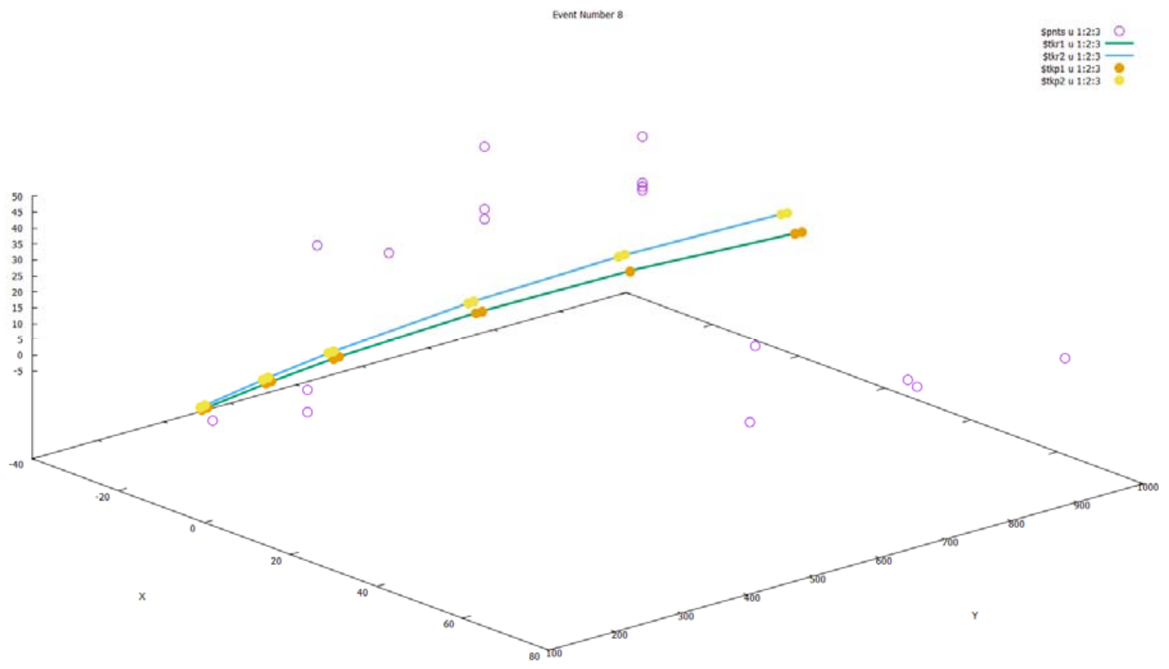
- Generate two closely spaced MC tracks in the lower tracker.
- Assume a 97% hit efficiency.
- Add noise hits with an occupancy that includes a flat component and another that peaks at the inner radius and falls off exponentially.
- Time: 9.4 milliseconds per event (Core-I7 notebook PC), including the event generation.
- Most tracks are found, with helix-parameter quality only slightly degraded relative to the single-track simulation.

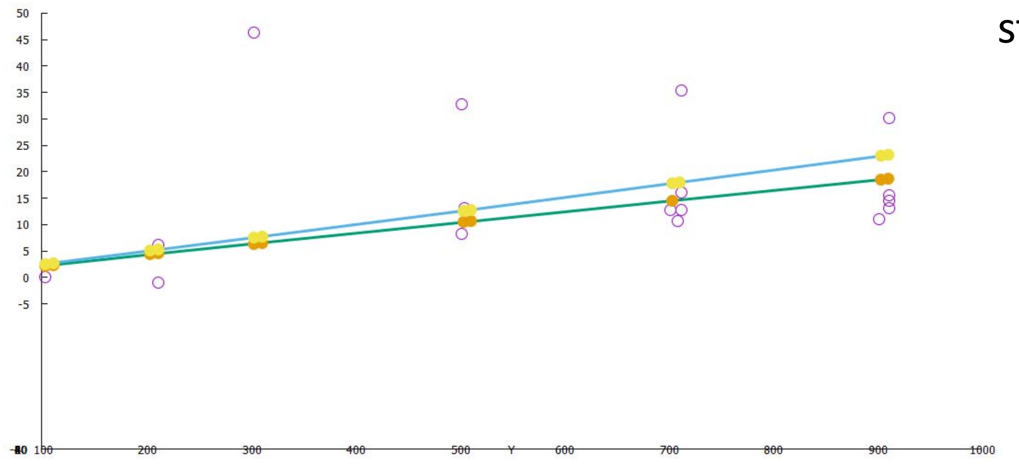




The 3-track events disappear if noise hits are turned off.

Example Event from the Idealized Simulation



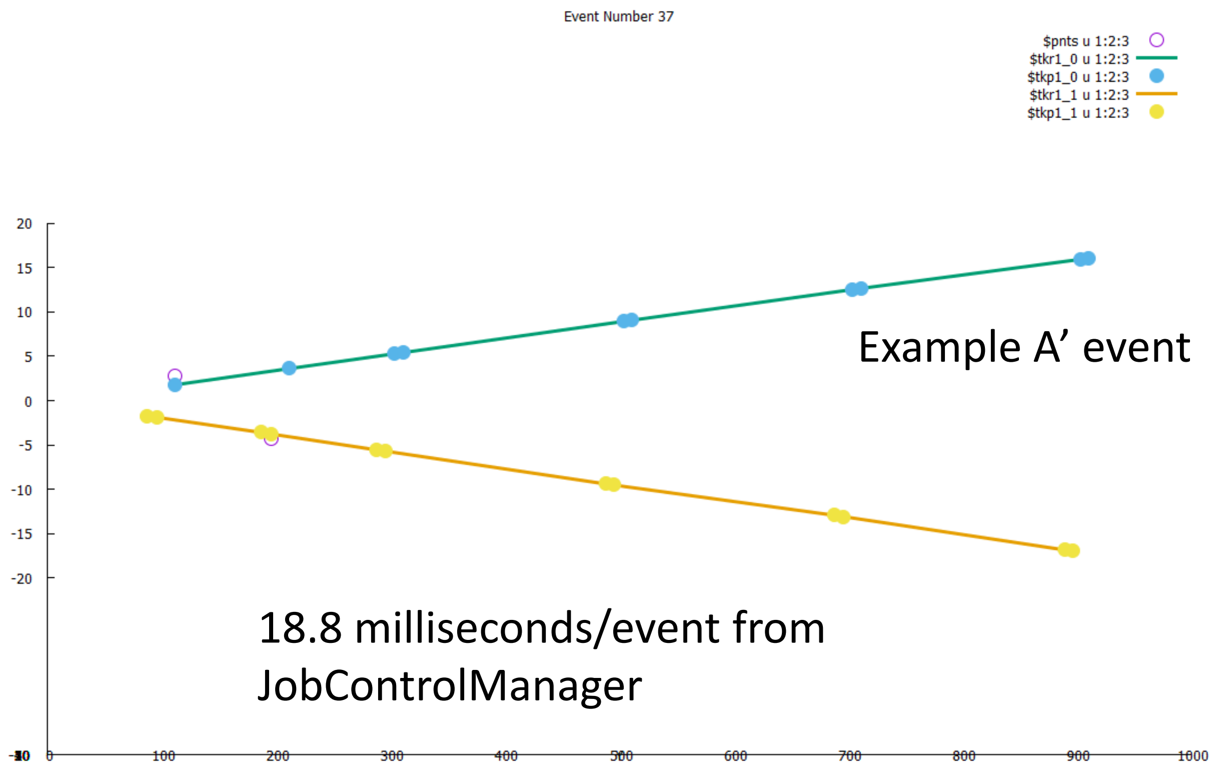


Note that noise hits are drawn at the strip center.

True hits are drawn at the MC true location along the strip and the simulated hit position perpendicular to the strips.

Trials with HPS Monte Carlo Events

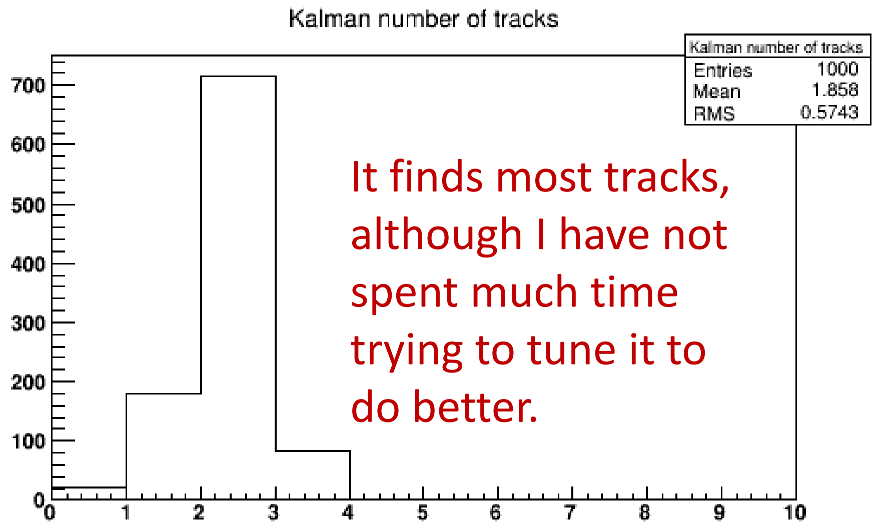
- The pattern recognition works similarly with full-blown Monte Carlo A’ signal events. In fact, they tend to be easier, as there is usually only a single track in each detector.
- The problem is that the Kalman-Filter track fit chi-squared is nearly always very large. This is still not understood.



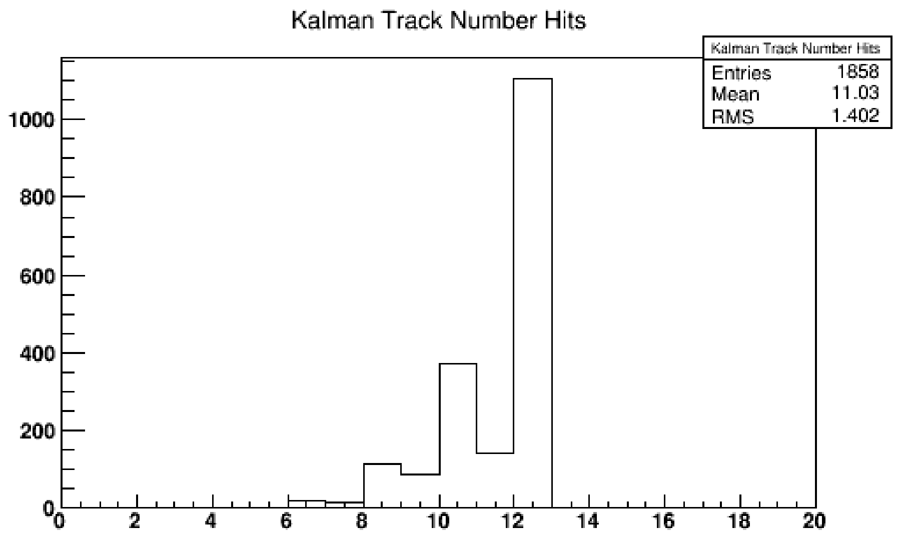
Example A’ event

18.8 milliseconds/event from JobControlManager

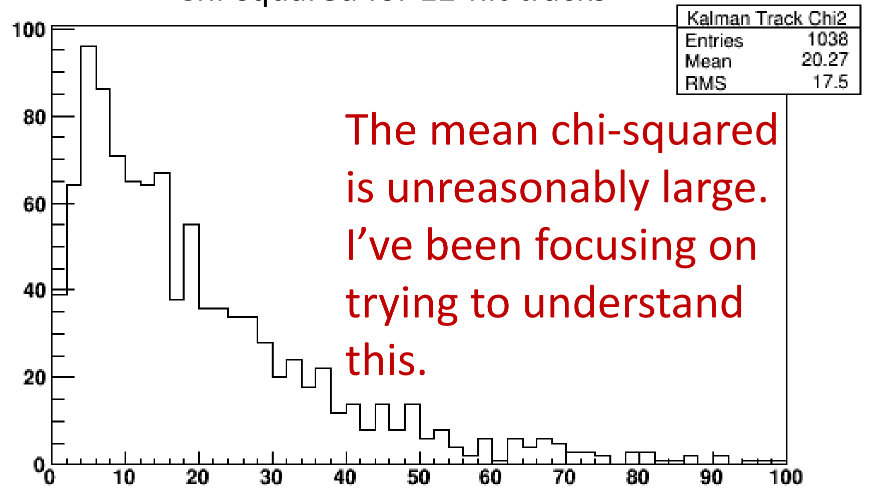
Test with 2015 A' Monte Carlo Events



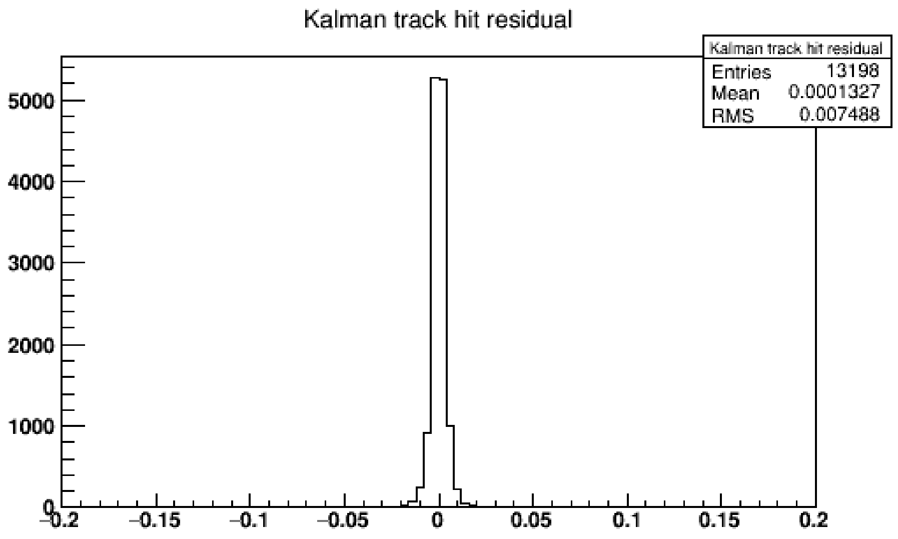
It finds most tracks, although I have not spent much time trying to tune it to do better.



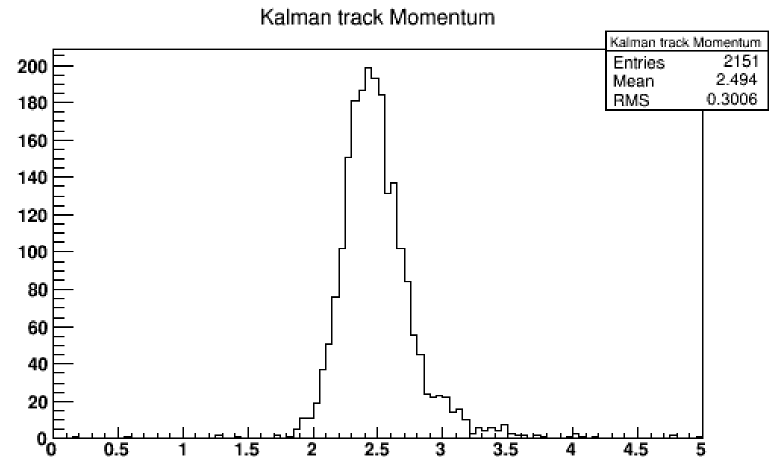
Chi-squared for 12-hit tracks

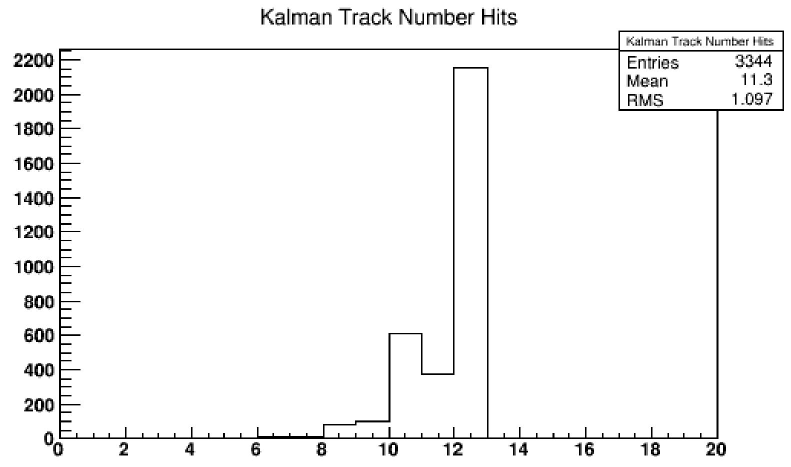


The mean chi-squared is unreasonably large. I've been focusing on trying to understand this.

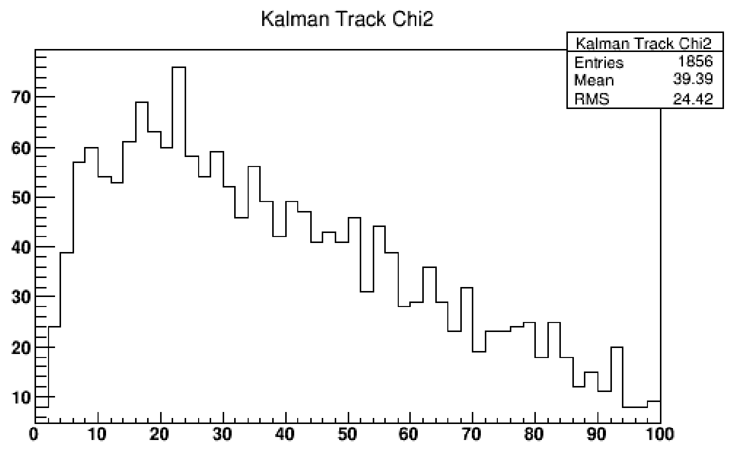


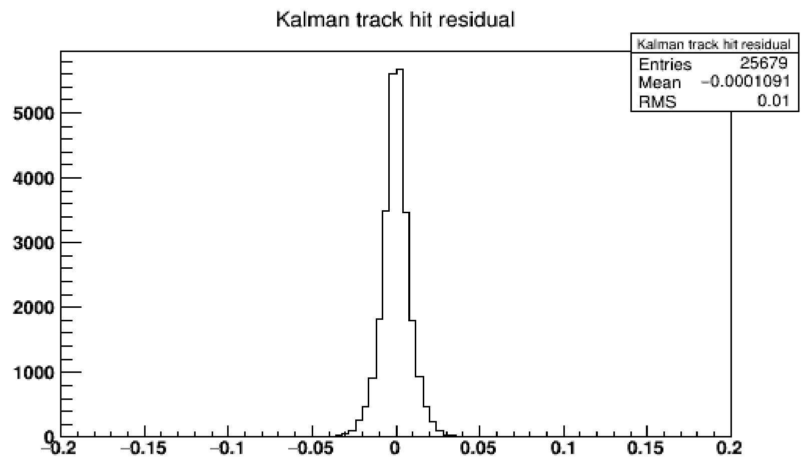
Test with 2016 Single-Track Muon MC Events



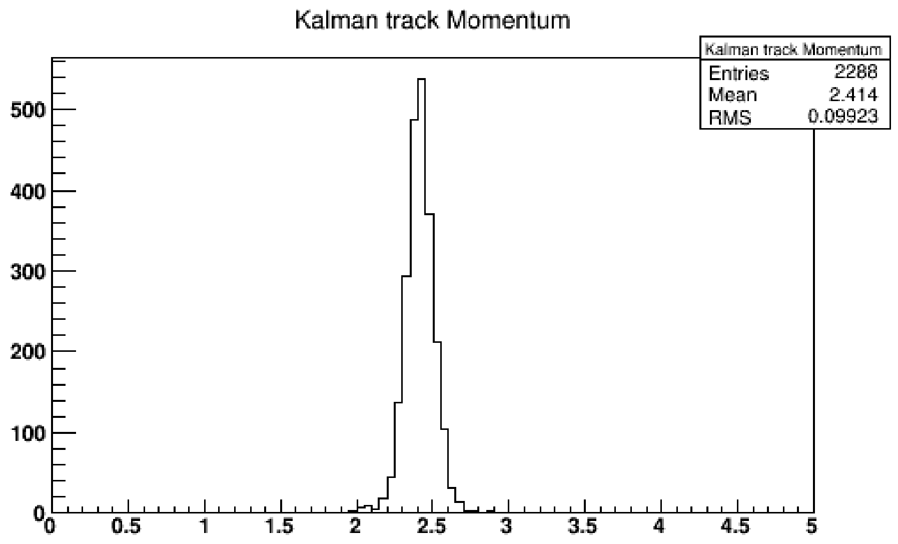


I decided to look at Monte Carlo muons in order to avoid questions about effects of bremsstrahlung, but here the chi-squared is even worse, perhaps related to the larger momentum in 2016. (Suggests it is not a multiple-scattering issue?)



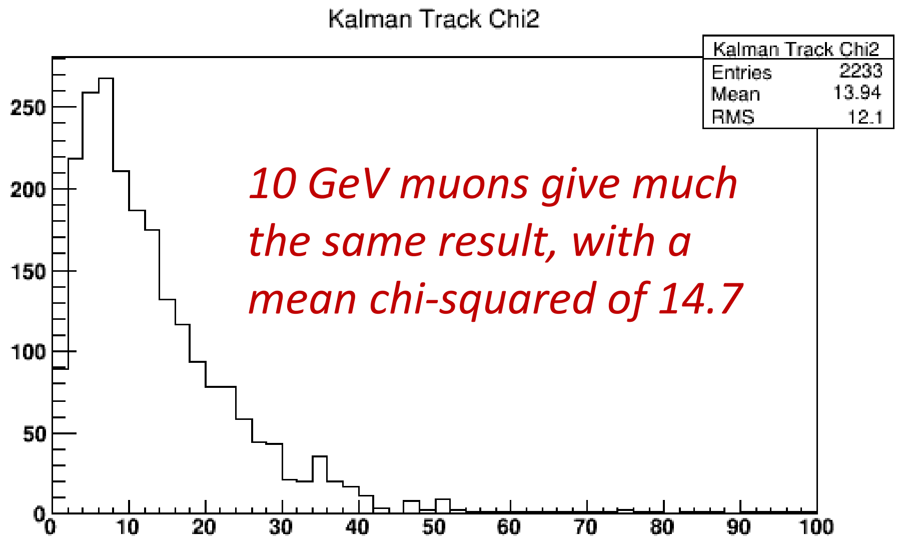


Single-Track Muons Using MC True Positions

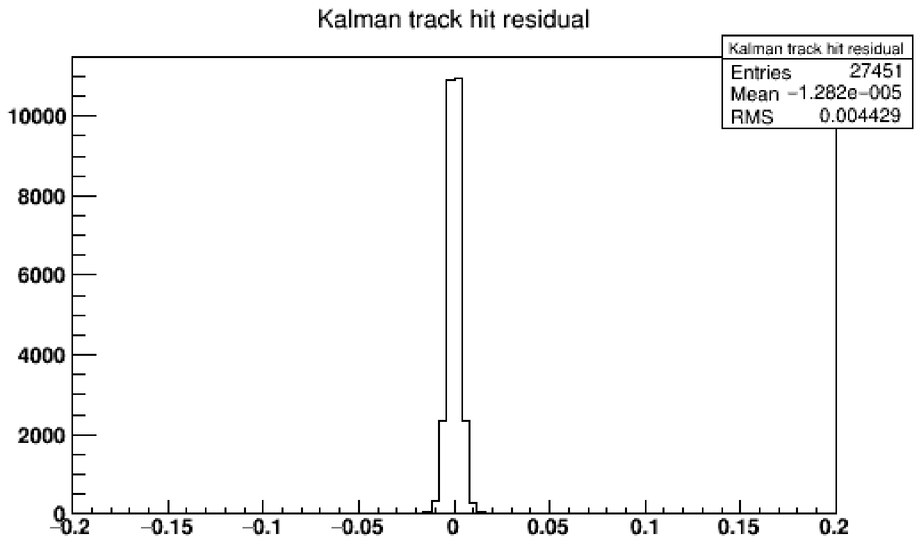


In this test I loaded the *MC true hit* locations into the Kalman pattern recognition, with only a 6-micron Gaussian smearing of the detected coordinate.

The results are much better, suggesting that the problem is not with scattering effects but rather with the loading of the simulated hits into the Kalman classes.



10 GeV muons give much the same result, with a mean chi-squared of 14.7



Conclusions

- This is still very much a work in progress.
- I am convinced that internally the Kalman-Filter code and the seed code work well and give mathematically correct results.
- The pattern recognition based on the Kalman Filter works well.
    - Given the minimal effort put into tuning it so far, there is a lot of potential for improvement.
- However, I now susptect that there are subtle errors in feeding the HPS SVT hits and/or the HPS geometry to the Kalman filter classes.
    - My priority now is to run this down and correct it.