Kalman-Filter Pattern Recognition and Fitting Status

Robert Johnson

U.C. Santa Cruz

May 14, 2020

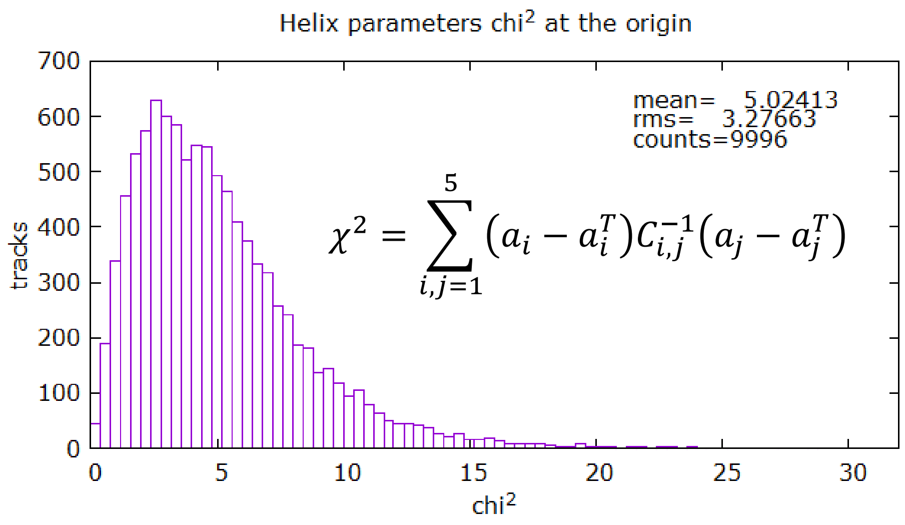
Outline

- What is included in the Kalman package
- Toy MC performance at vertex
- Pattern Recognition overview
- 2016 MC performance
- 2019 MC performance
- 2019 data performance
- Propagation of tracks and covariance in a non-uniform field

org.hps.recon.tracking.kalman

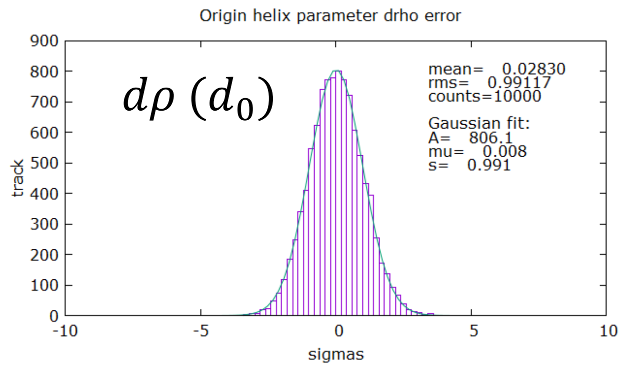
- pdf of LaTeX documentation (not 100% complete yet)
- “toy” MC test code (not loaded into hps-java)
- Linear helix fit
  - For initiating the Kalman Filter and for the combinatoric pattern recognition
- Kalman-Filter track fit
- Pattern recognition
- Propagation of track and covariance in the non-uniform field
  - e.g. to the vertex and to the ECAL
- KalmanInterface.java: code needed to interface to hps-java
- KalmanPatRecDriver.java: driver code for production
  - (older KalmanDriverHPS was used specifically to refit GBL tracks and is not intended for general use)
- KalmanPatRecPlots.java: histograms used up to now for development work

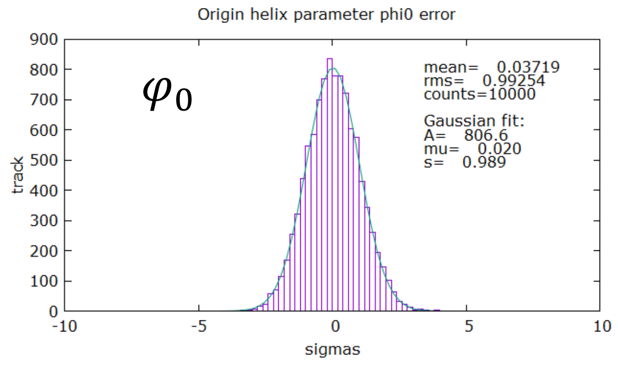
Math test of Kalman-Filter fit with toy MC

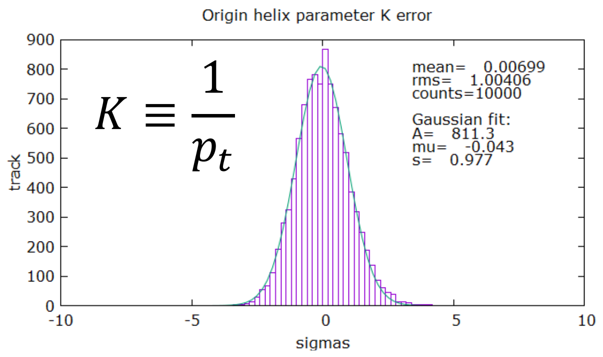


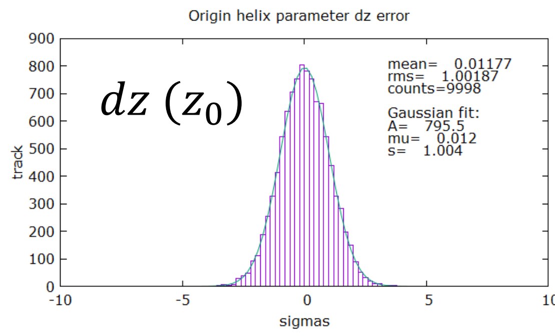
$$\chi^2 = \sum_{i,j=1}^{5} (a_i - a_i^T) C_{i,j}^{-1} (a_j - a_j^T)$$

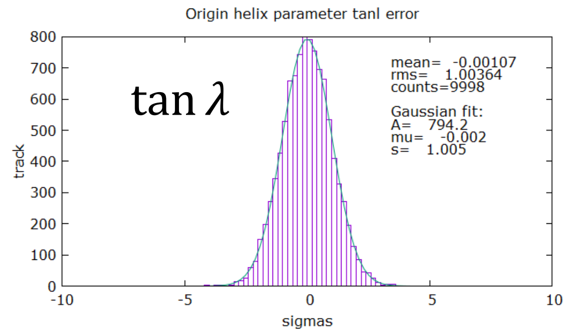
- No pattern Recognition.
- Tests the full covariance matrix, *including propagation to the vertex.*
- For 5 d.o.f., the mean should be 5 and the rms $\sqrt{10} \sim 3.16$
- Also, all 5 helix parameter pull distributions are consistent with zero mean and rms = 1.0





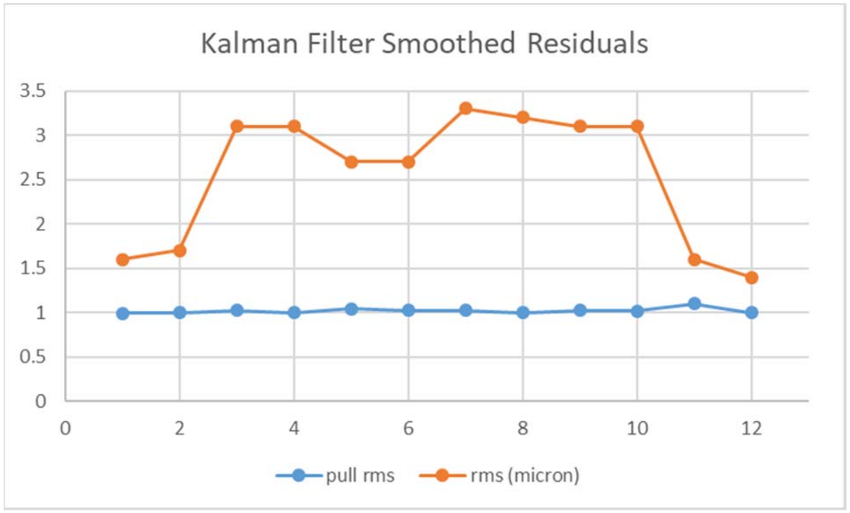


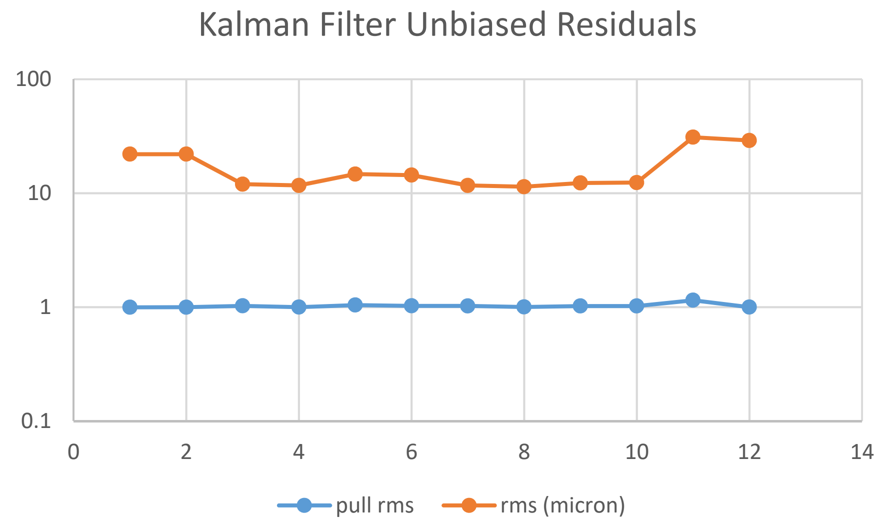




Toy MC residuals (math verification)

- Layer by layer pulls have mean 0 and rms = 1.0
- The code will also calculate *unbiased* residuals layer by layer *without* refitting the track to exclude the selected layer.



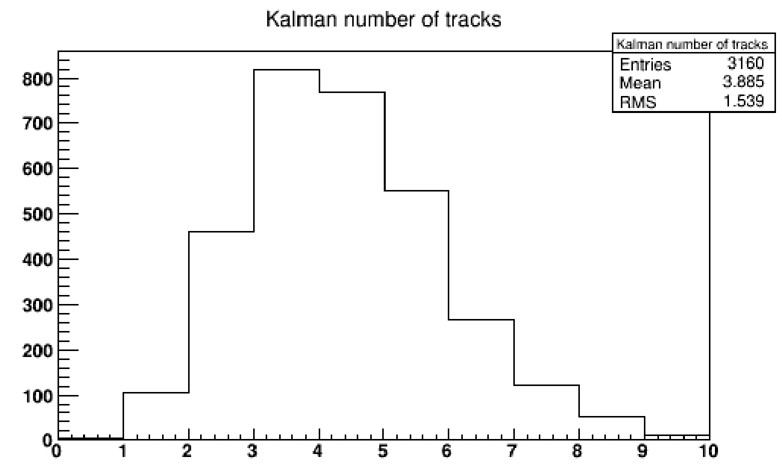


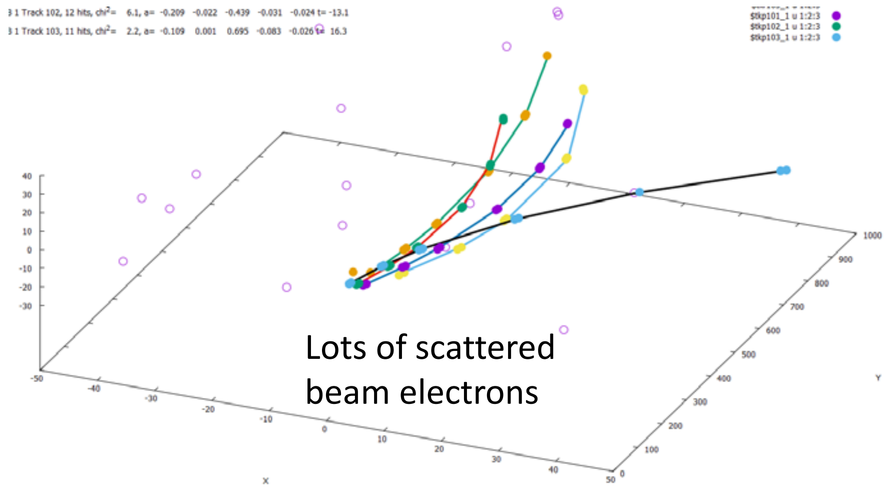
Note: the toy MC assumed the non-uniform HPS B-field, 6-micron hit errors, 12 silicon layers (as in 2016 data) and Gaussian multiple scattering from 0.32 mm thick silicon.

Pattern Recognition Overview

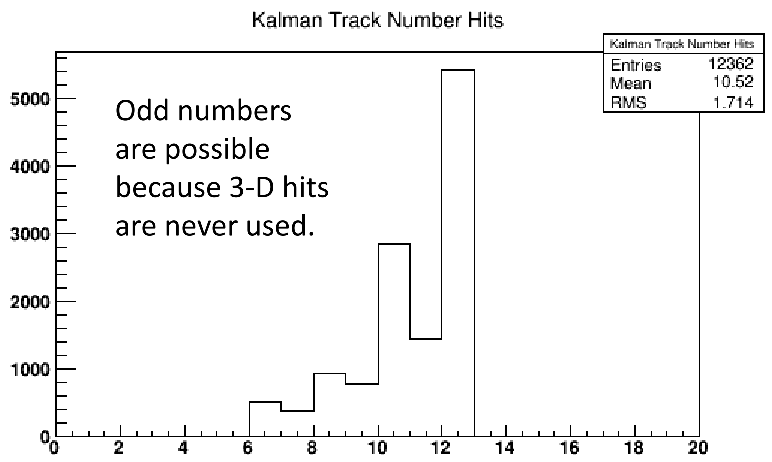
- This is simplified. See the documentation for more detail.
- Two global iterations, with looser cuts in the second iteration, to try to give priority to the best tracks. (This could probably be reduced to one iteration to save CPU time.)
    - Loop over configurable set of “strategies”, each a set of 2 axial and 3 stereo layers.
        - Loop over all combinations of hits from the 5 layers, doing the linear fit to each, to produce “seeds”. There are, in fact, various cuts made here to reduce the number of combinations fit.
        - Use Kalman Filter to propagate sorted seeds to other layers and pick up hits, to produce “track candidates”.
    - Algorithms to evaluate the “good” track candidates to remove redundancies, in favor of the best candidates. In the end, only a limited number (a few) shared hits are allowed, and only if they fit very well to both tracks.
- Final fit of the tracks, plus extrapolation to the origin.

Test on 2016 Tri-trig-beam MC

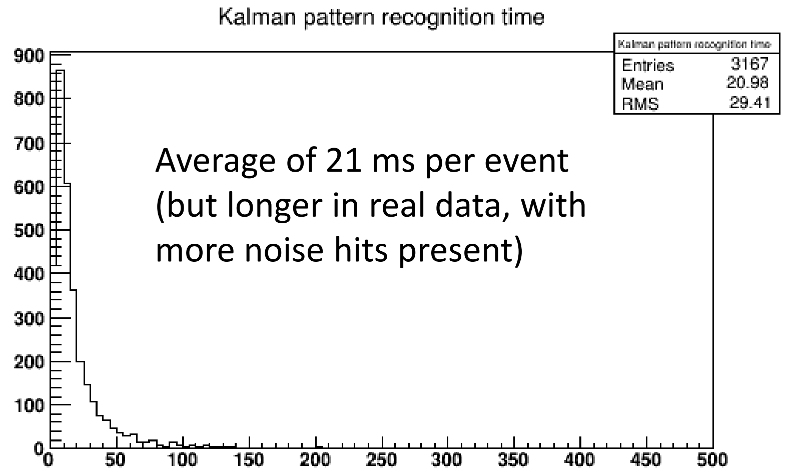



```
3 1 Track 102, 12 hits, chi²=   6.1, a= -0.209  -0.022  -0.439  -0.031  -0.024 t= -13.1
3 1 Track 103, 11 hits, chi²=   2.2, a= -0.109   0.001   0.695  -0.083  -0.026 t=  16.3
```

Lots of scattered beam electrons

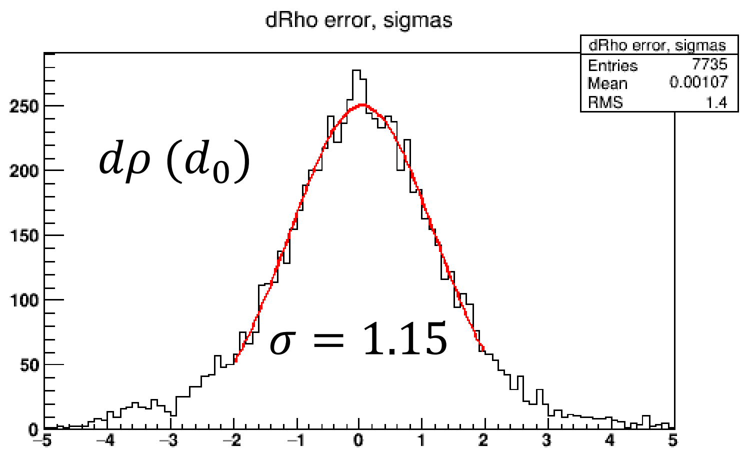


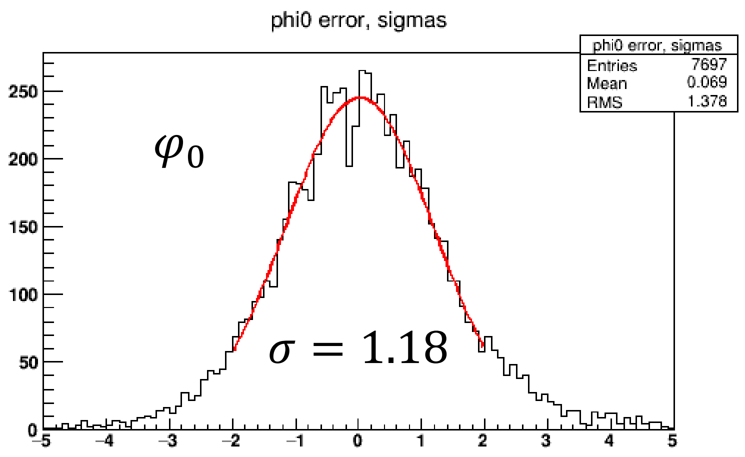
Odd numbers are possible because 3-D hits are never used.

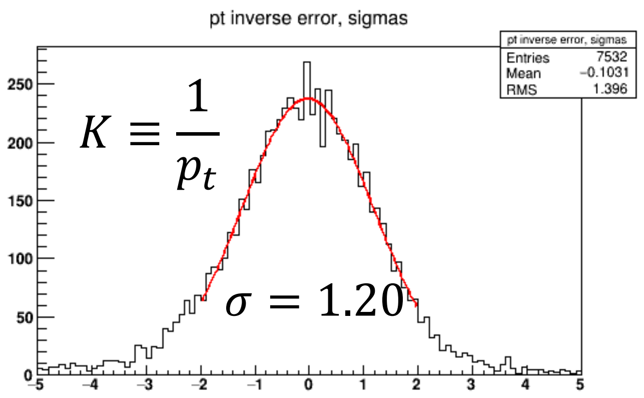


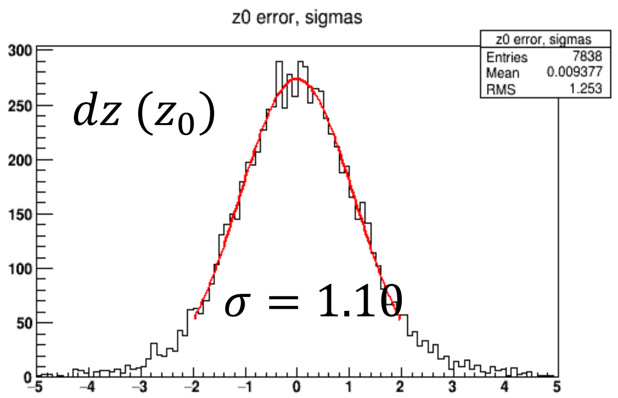
Average of 21 ms per event (but longer in real data, with more noise hits present)

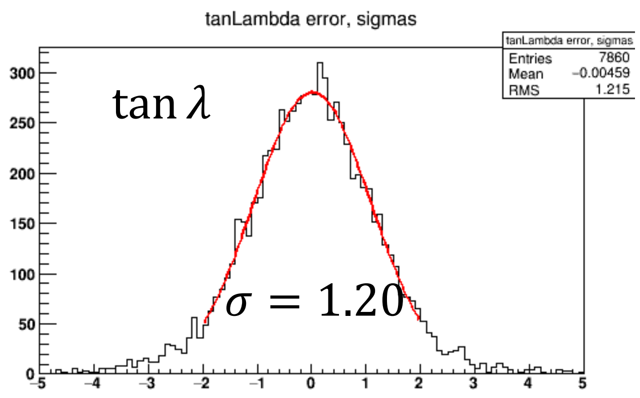
2016 Tri-trig-beam MC, helix param. pulls





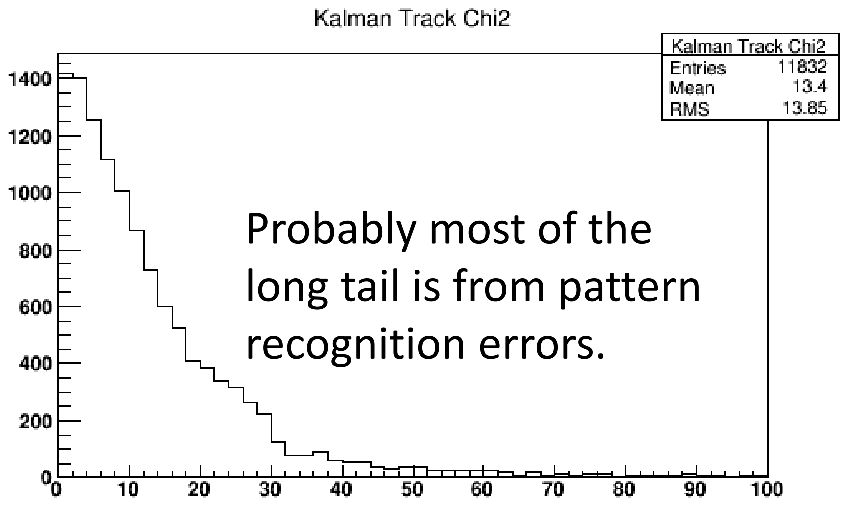




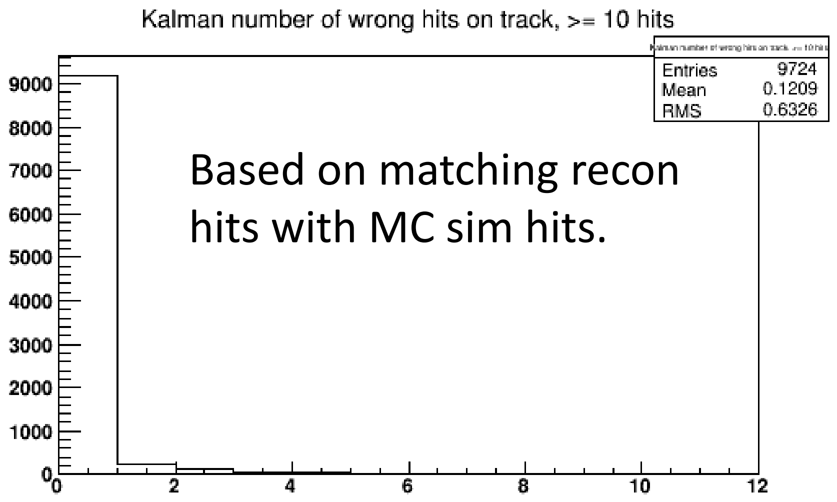


The Gaussian widths of the pull distributions are roughly 10% to 20% too large, compared with the idealized toy MC, but that can probably be explained by non-Gaussian hit errors, hit overlaps, non-Gaussian multiple scattering, tracking errors, and other interaction effects, such as hard scattering and bremsstrahlung. These effects should also explain the non-Gaussian tails.

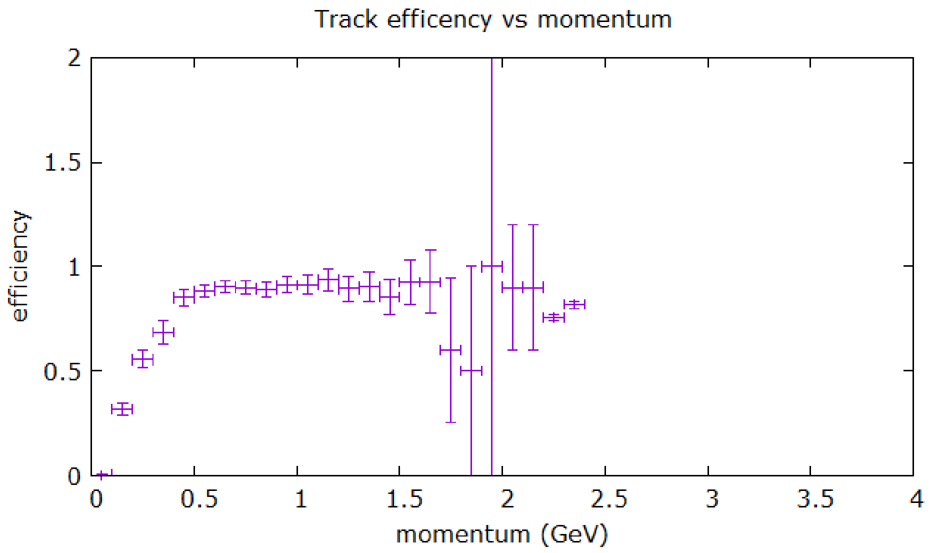
Pattern Recognition, 2016 MC



Probably most of the long tail is from pattern recognition errors.

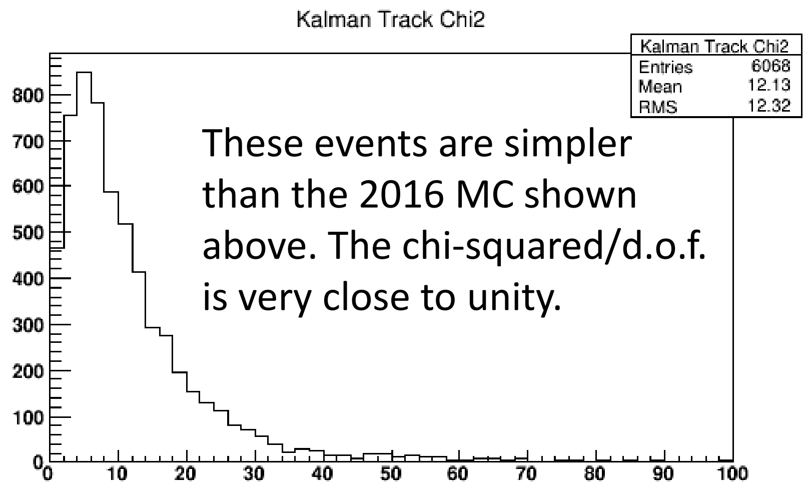


Based on matching recon hits with MC sim hits.

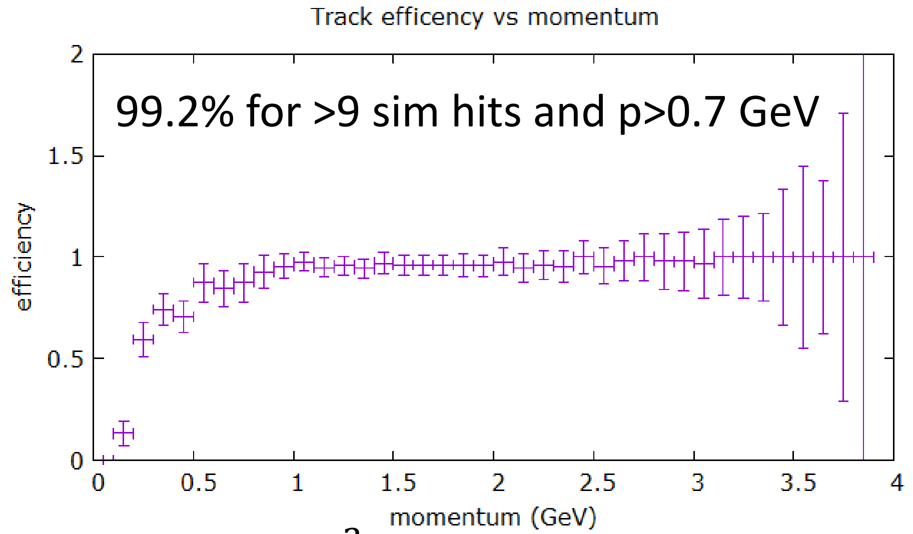


The track efficiency for p>0.7 GeV and ≥ 10 sim hits is 0.943±0.003

2019 Tri-trig MC (two new layers included)

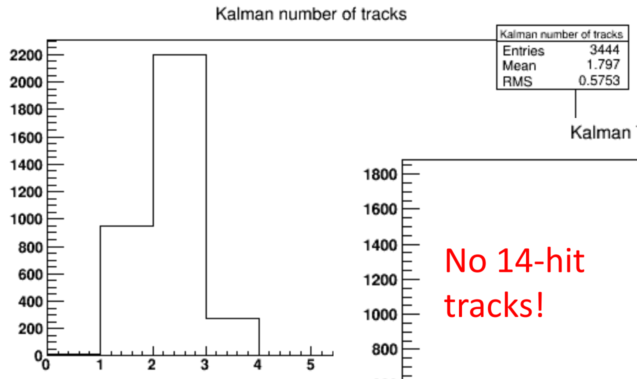


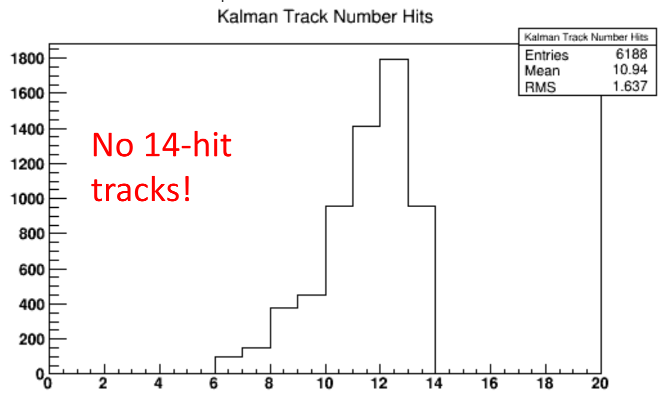
These events are simpler than the 2016 MC shown above. The chi-squared/d.o.f. is very close to unity.



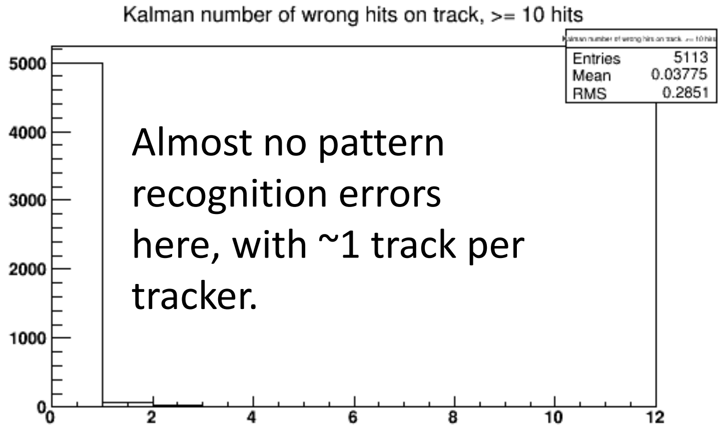
99.2% for >9 sim hits and p>0.7 GeV

Note: for the Kalman formalism 12 hits should give a mean χ^2 of 12, not 7 (not a true chi-squared distribution, I think), as the fit bias is subtracted out of the hit χ^2 contribution.



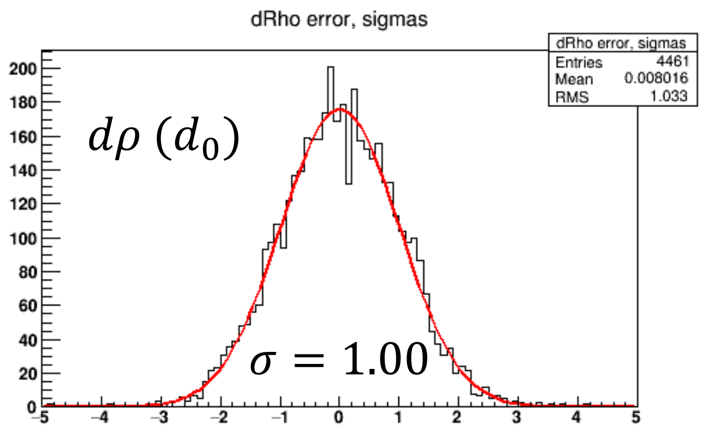


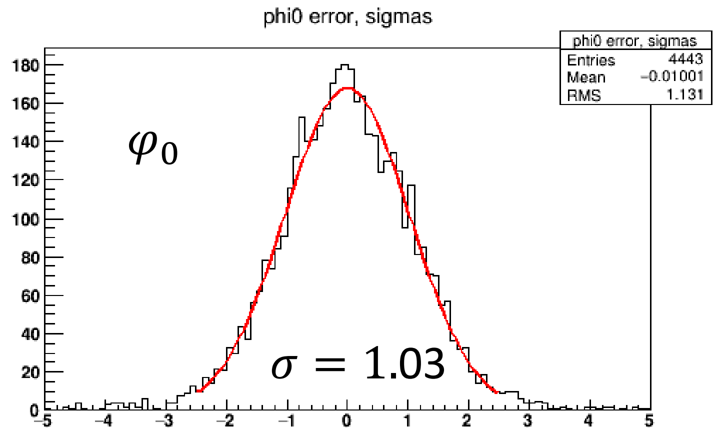
No 14-hit tracks!

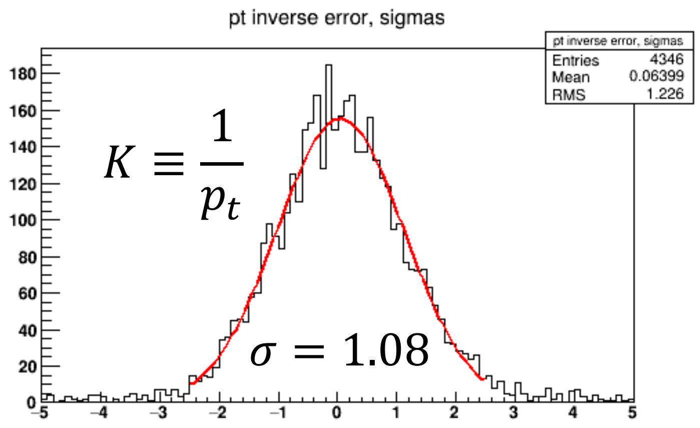


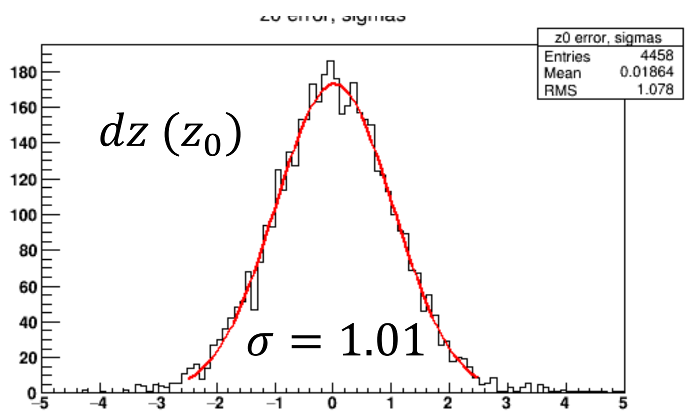
Almost no pattern recognition errors here, with ~1 track per tracker.

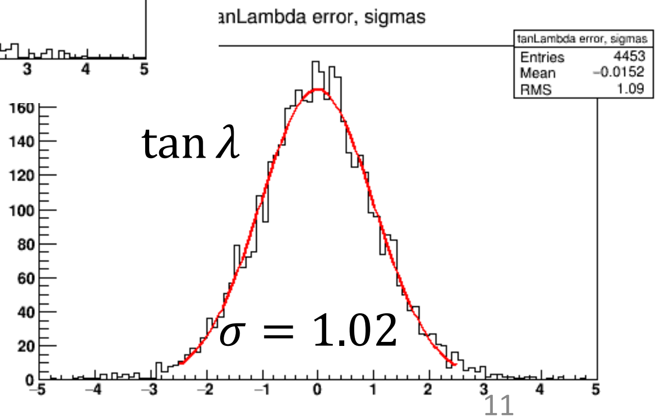
2019 Tri-trig MC





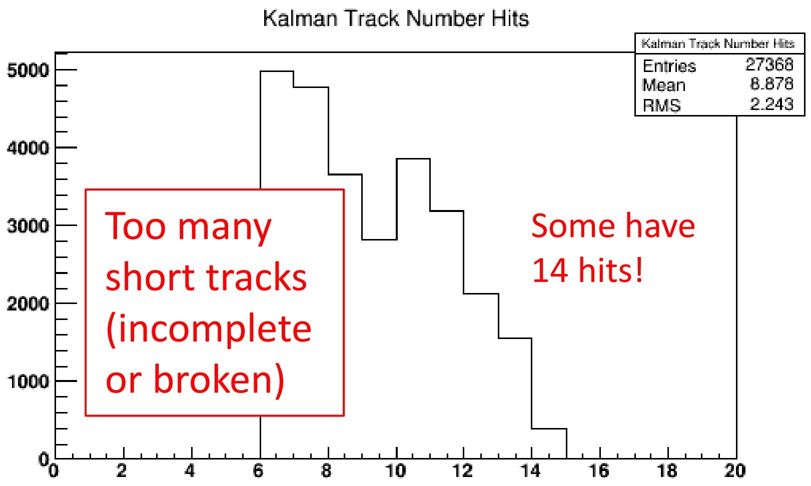






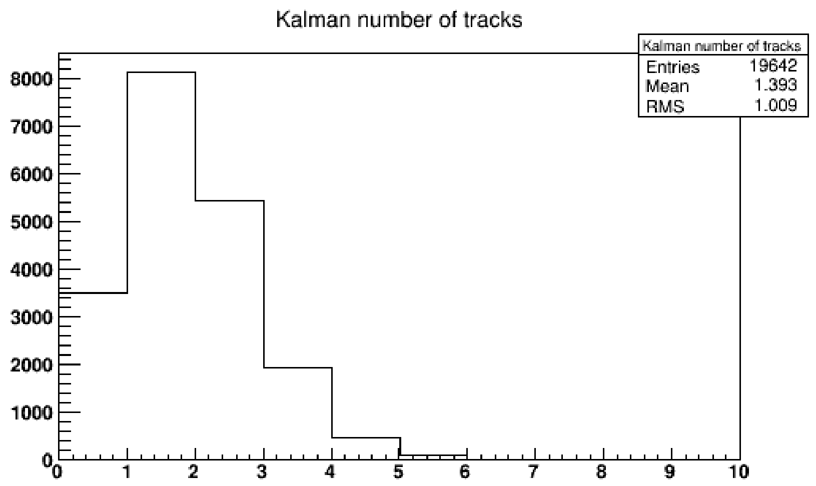
Without the beam background included in the 2016 MC sample, the pull distributions are significantly closer to normal distributions.

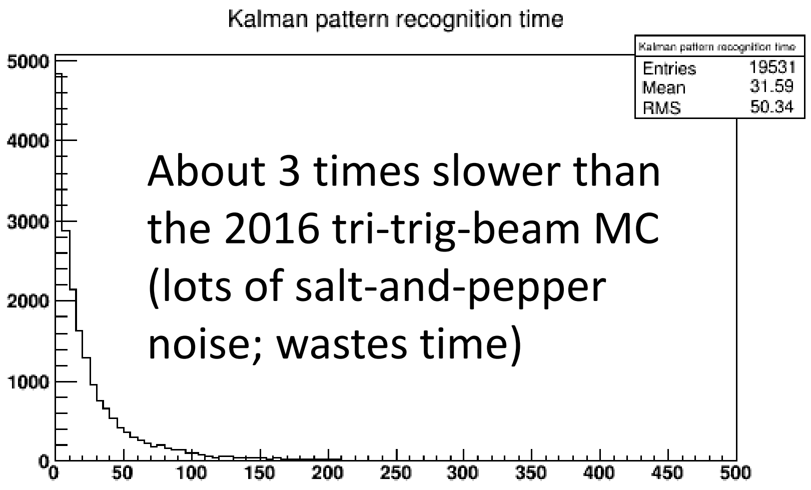
2019 Data (not aligned)



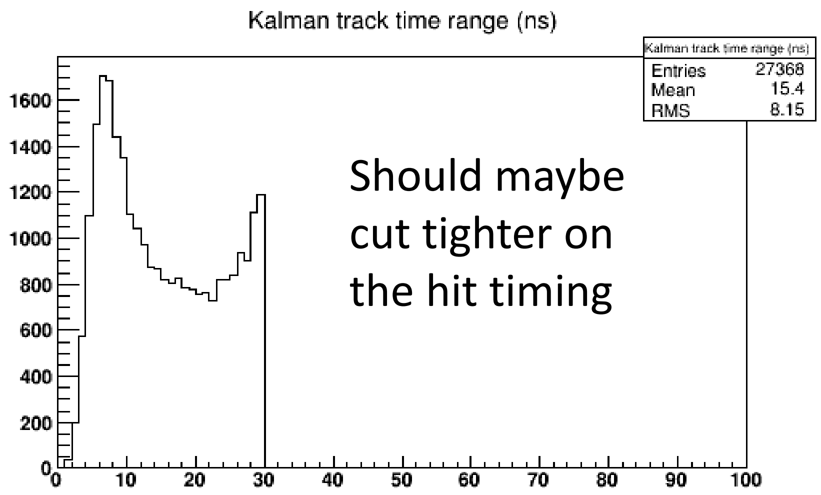
Too many short tracks (incomplete or broken)

Some have 14 hits!



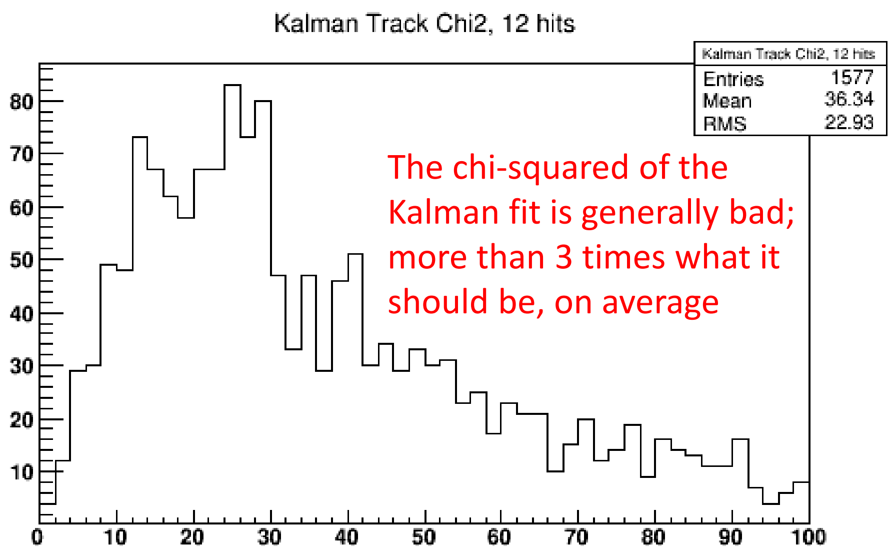


About 3 times slower than the 2016 tri-trig-beam MC (lots of salt-and-pepper noise; wastes time)

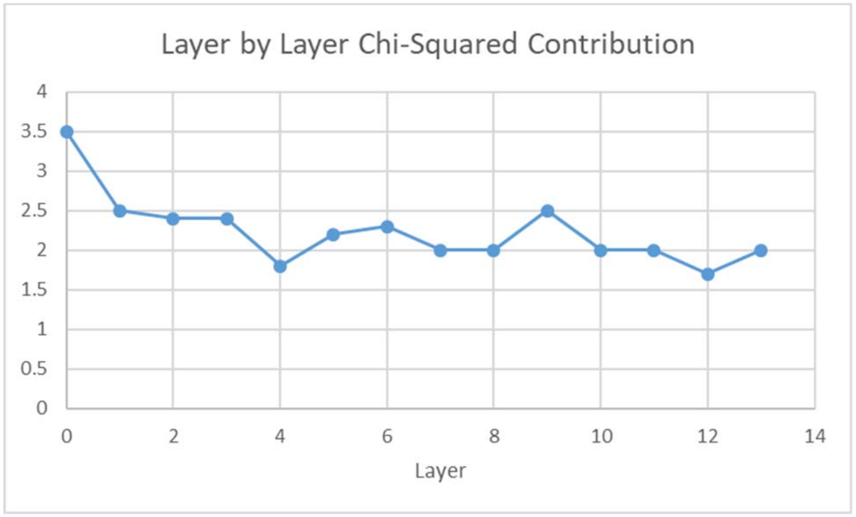


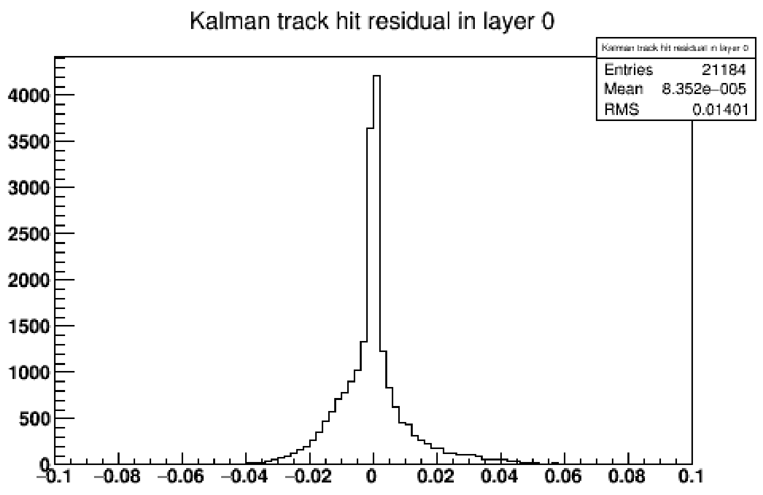
Should maybe cut tighter on the hit timing

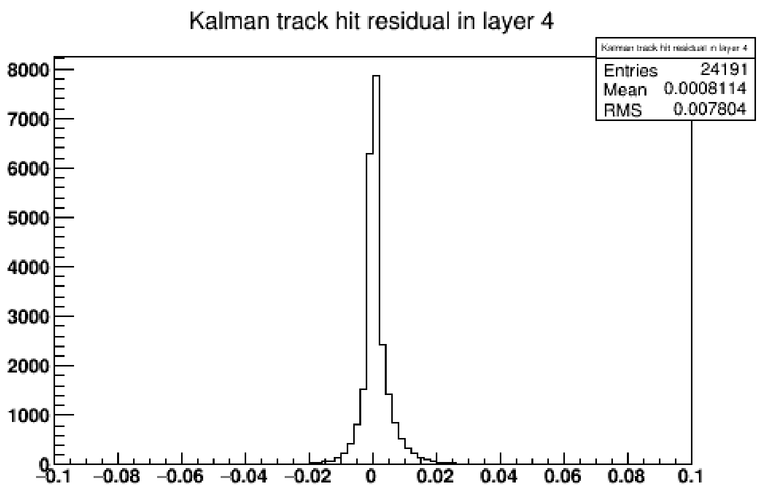
2019 Data (not aligned)



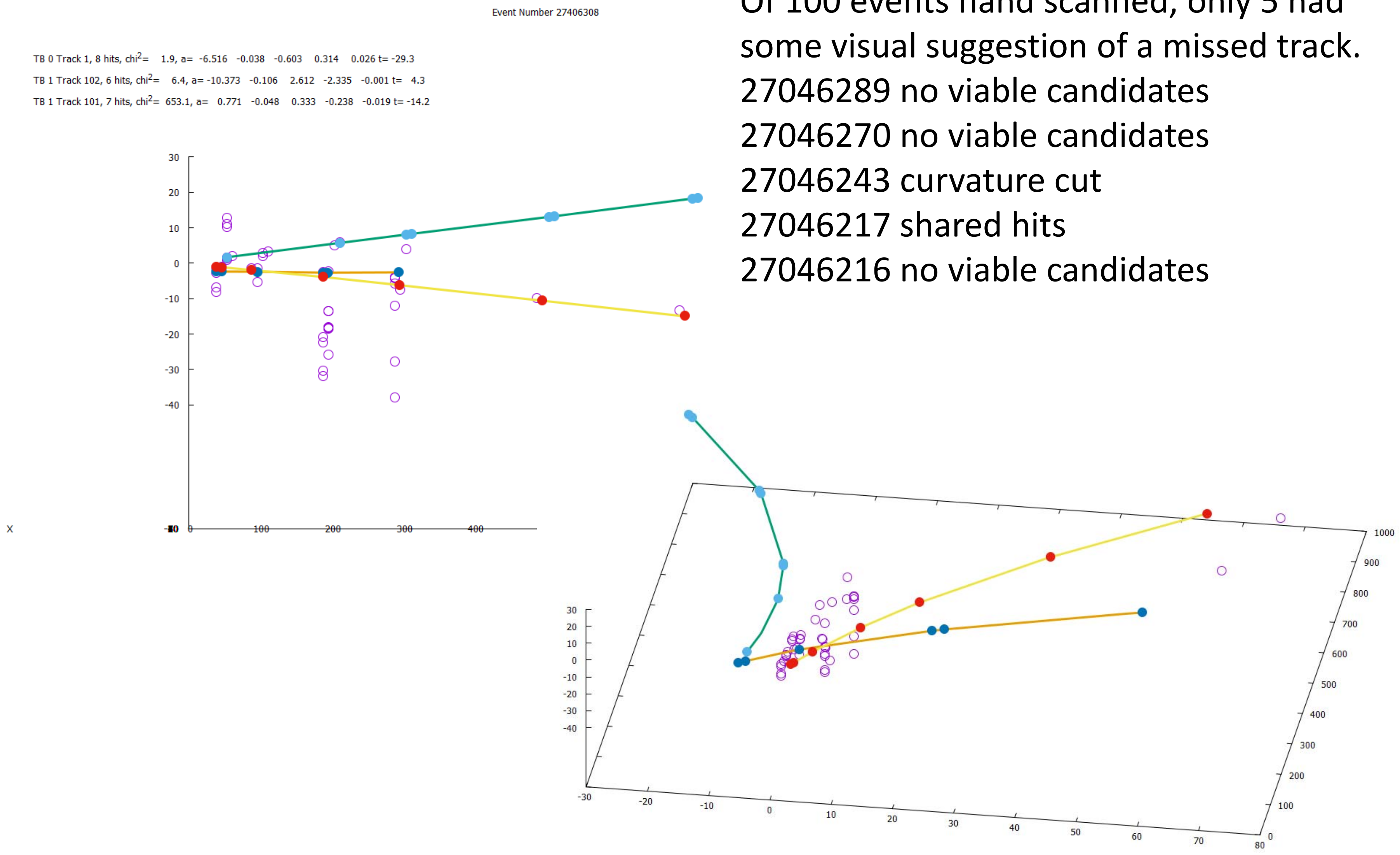
The chi-squared of the Kalman fit is generally bad; more than 3 times what it should be, on average







Example 2019 Event


Event Number 27406308

```
TB 0 Track 1, 8 hits, chi^2=    1.9, a=  -6.516   -0.038   -0.603    0.314    0.026 t= -29.3
TB 1 Track 102, 6 hits, chi^2=    6.4, a= -10.373   -0.106    2.612   -2.335   -0.001 t=   4.3
TB 1 Track 101, 7 hits, chi^2=  653.1, a=   0.771   -0.048    0.333   -0.238   -0.019 t= -14.2
```

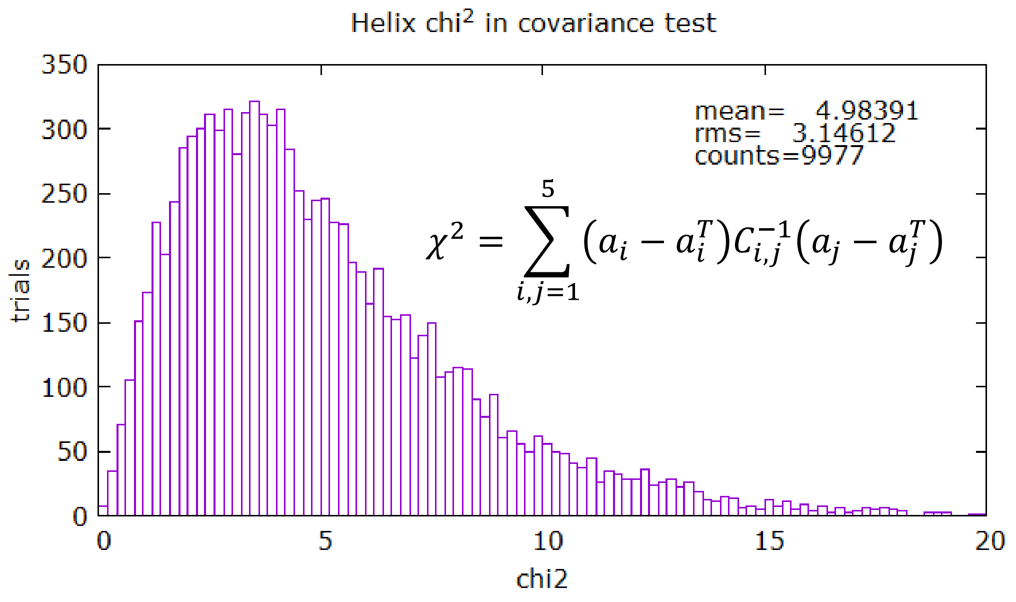
Of 100 events hand scanned, only 5 had some visual suggestion of a missed track.
27046289 no viable candidates
27046270 no viable candidates
27046243 curvature cut
27046217 shared hits
27046216 no viable candidates

Propagation of Tracks and Covariance

- New general routine that should work for any track state with helix parameters and covariance (it was used in all of the work above to propagate from the 1st hit layer to the target position).
- The track is propagated to a given plane, in the non-uniform field, by 4th order Runge-Kutta integration. At the desired end, the helix parameters are re-derived.
- The covariance is propagated by a series of finite pivot transforms. At each step the coordinates are rotated to align with the local field.
- A multiple scattering contribution is added to the covariance when passing through a layer of silicon (e.g. going from 2nd layer to target).
- In recent weeks I've tested this extensively in Monte Carlo to verify it mathematically.

Test of covariance propagation through silicon

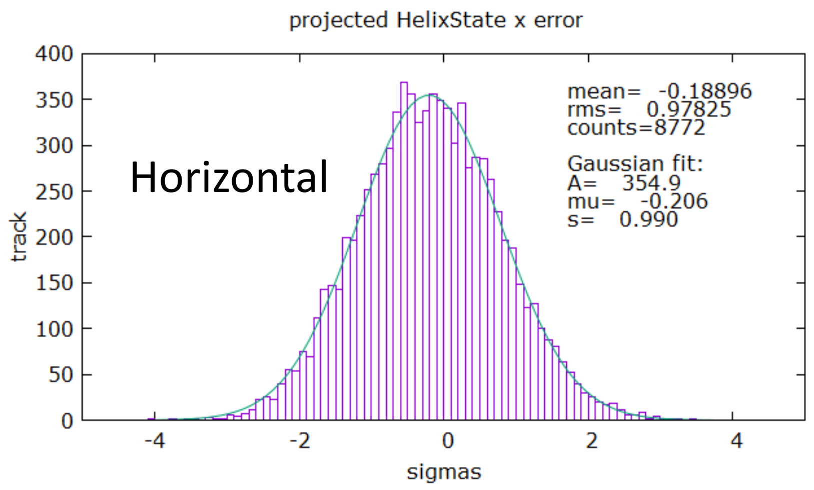
- Start with a track state at 1st layer and propagate it and its covariance to the last layer.
- Generate 10,000 vectors at layer 1 according to the full covariance and propagate them to the last layer by Runge-Kutta + Gaussian MCS.
- Compare each with the propagated track.

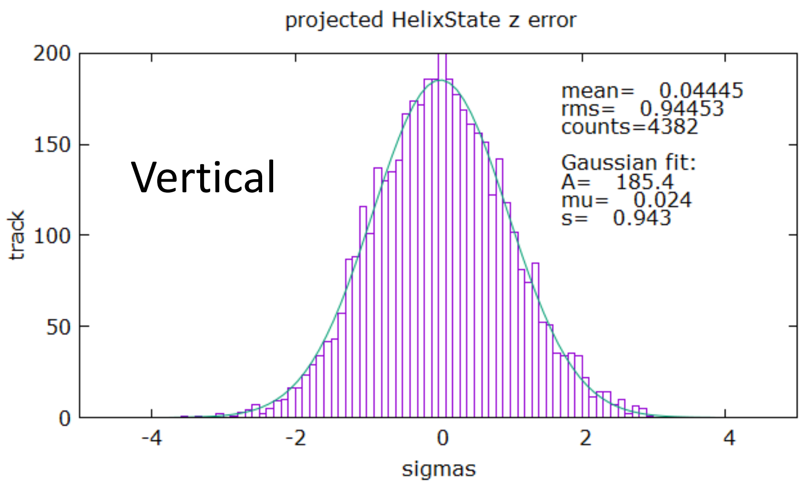


The chi-squared distribution at the left is perfect, verifying that the covariance C at the last layer is correct.

Toy MC Test of Propagation to the ECAL

- Propagate toy MC tracks from the last layer to the ECAL.
- Calculate from the propagated covariance the errors on the 2-D point of impact at the ECAL face.
- Compare with the MC truth.
- With uniform B the pulls come out exactly normal, but vertically it's a little bit off with the non-uniform field:





Conclusions

- The Kalman package could be important for 2019 data in which there are missing layers.
- PF has shown significant speed advantages over the existing pattern recognition.
- The track-state propagation code may be useful for more rigor in vertex analyses.
- The package is complete and working well at this point.
- I will continue to tune it as I get more experience studying the data. PF and Alic are also working on this.