Kalman-Based Pattern Recognition Status

- There is a pull request to put the latest code into Master.
- The latest modifications try to deal with the occasional non-positive-definite covariance matrix of helix parameters.
- I have also made some recent improvements in removal of bad hits.
- Relatively recent modifications deal with low-amplitude hits that typically are from noise
    - Don't use low-amplitude hits when making the 5-hit seeds used to start the Kalman filter. This can save a lot of CPU time in large events that can cripple the performance with enormous combinatorics.
    - Give low-amplitude hits low priority for getting picked up by the pattern recognition. This can potentially help avoid mistakes from picking up a noise hit instead of the true hit.

Negative Covariance

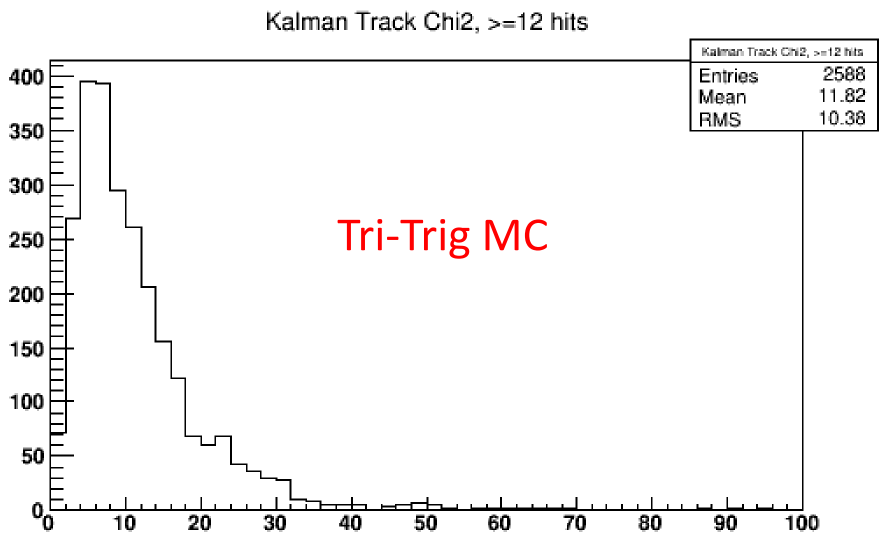
- Literature shows that this is an old problem with the Kalman formalism, going back to the 1960s (satellite tracking).
  - Various numerical instabilities are addressed in the literature, but I have not found significant improvement when trying the simpler solutions. Already the use of 64-bit numbers makes this less of an issue than it used to be.
  - Starting with very large initial covariance values exacerbates the problem, as I have previously reported.
  - In general having very small hit errors relative to residuals that occur due to multiple scattering and mistakes makes the formalism susceptible to this problem.
- Usually a diagonal element is negative, so I have not tried to do a more rigorous and expensive test based on eigenvalues.
  - It may be that the problems occur on tracks that are bad in some way (e.g. wrong hit). However, this supposition needs more study.
  - I am dealing with the remaining problem by “fixing” the covariance and flagging the track internally as “bad.” The question remains whether to delete these before making HPS persisted tracks.

Negative Covariance

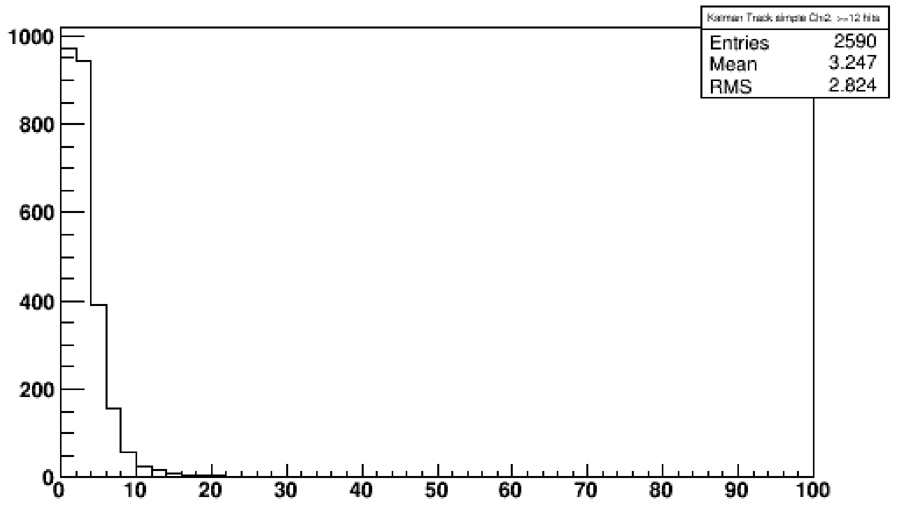
- The rate on final tracks is low and occurs in Monte Carlo as well as in data.
- Almost always it occurs in the Kalman smoothing process, not the filter process.
  - 2019 data: 0.35% of tracks
  - 2019 tri-trig MC: 0.33% of tracks

Kalman Track Chi-Squared

- The chi-square of the Kalman fit can be calculated in two ways
  - Rigorous Kalman formula, which accounts for measurement errors and multiple scattering and ideally should have a mean of one per hit.
  - Sum of squares of residuals divided by measurement error. This doesn't account for multiple scattering and in principle has #d.o.f. = Nhits – 5.



Simple sum of squares

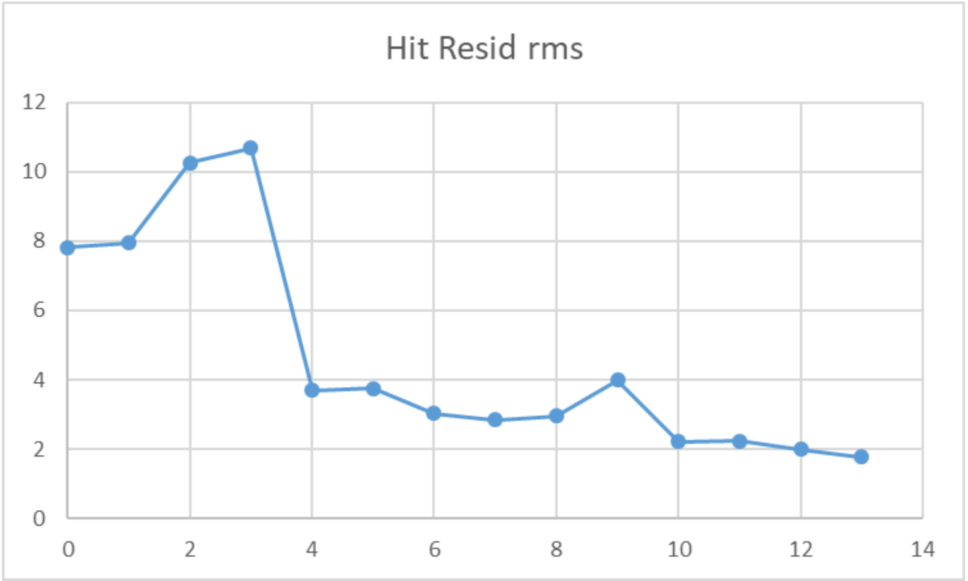


- The mean on the left looks about right, whereas on the right it seems too small (especially given that the simple calculation does not account for scattering).

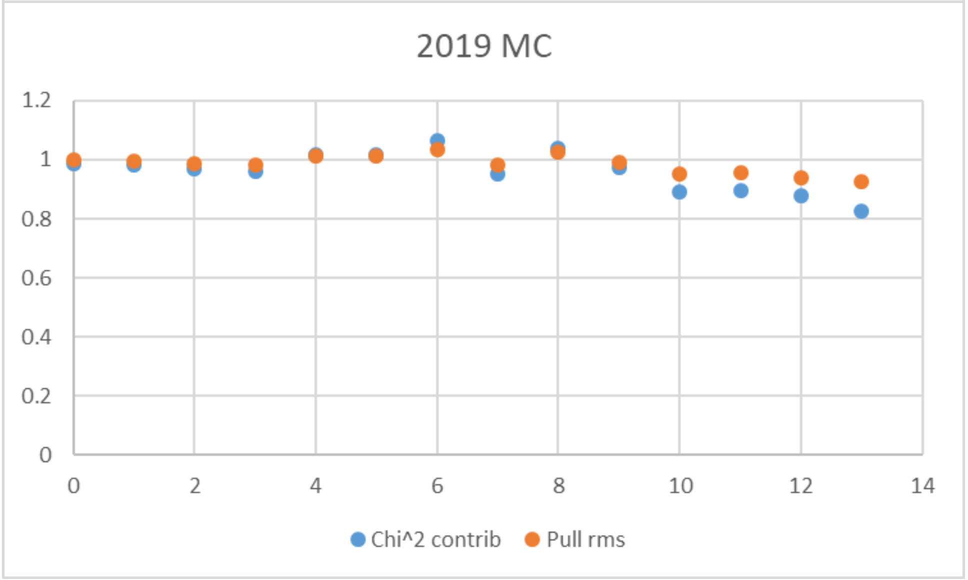
- These are generally very clean tracks, with few pattern recognition errors.

2019 Tritrig MC Hit Residuals by Layer

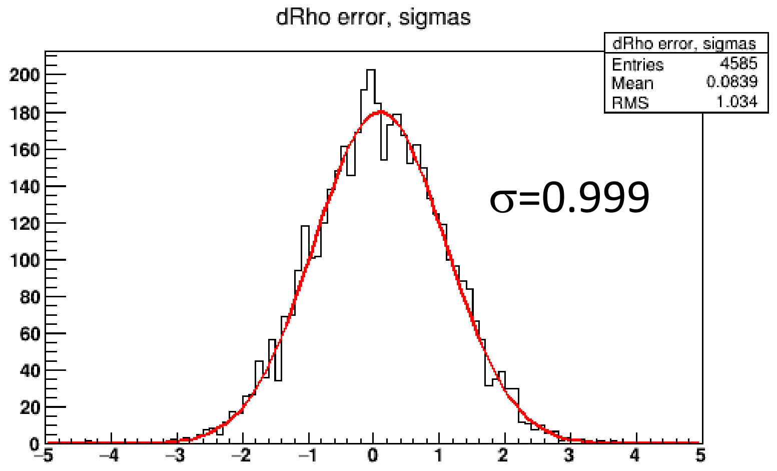
Note how much larger the residuals are in the new layers than in the old.

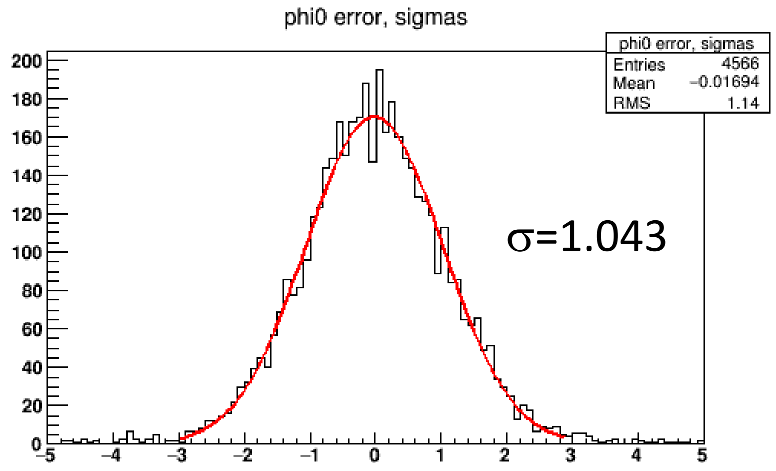


The residuals pulls and the mean contributions to the chi-squared are about right.

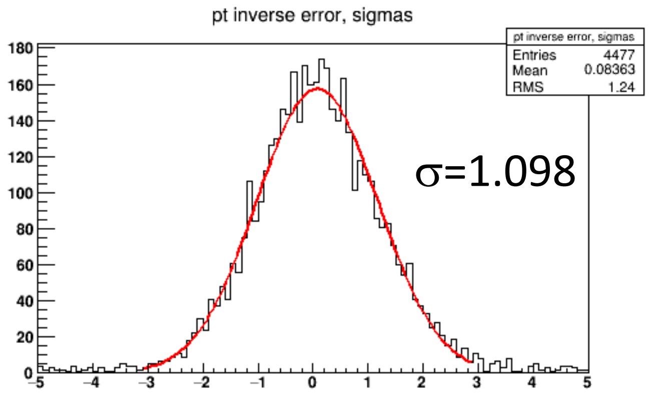


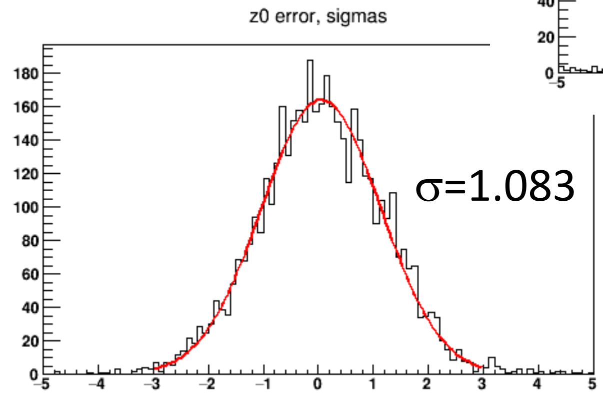
2019 Tritrig MC Helix Param Pull Distributions

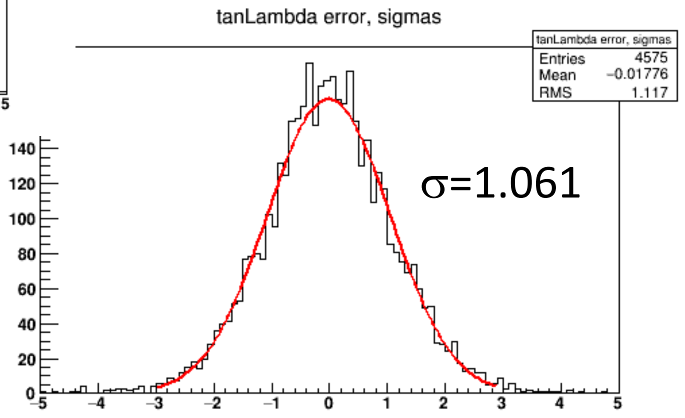




These are a bit on the wide side, but not too bad.



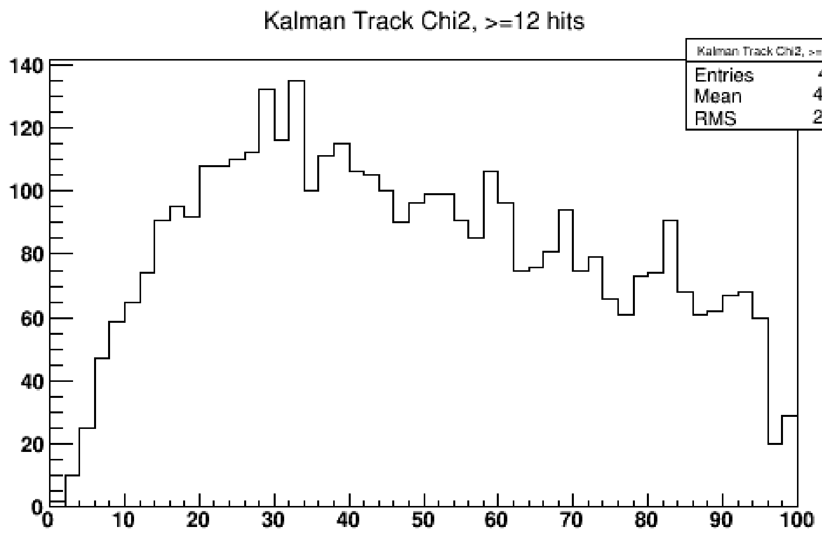




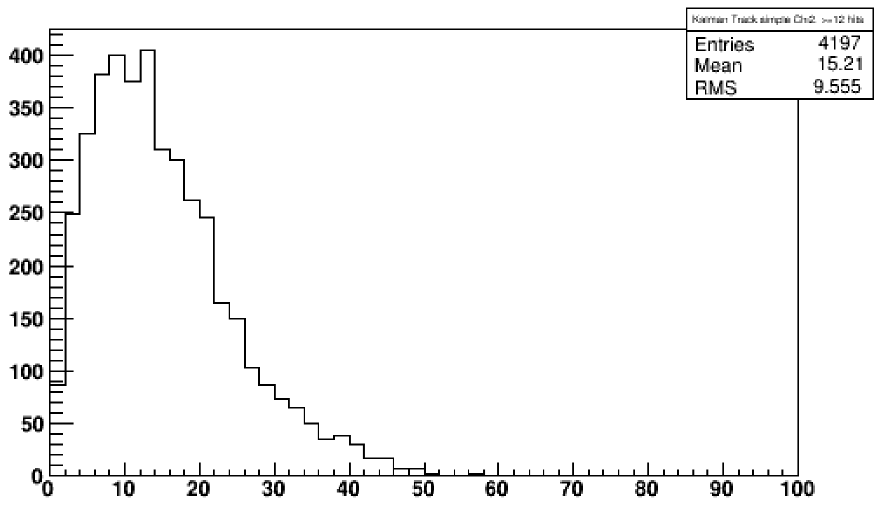
Kalman Track Chi-Squared

But for 2019 data the chi-squared mean is much too large.

- Am I not using the correct alignment?



Simple sum of squares


Kalman Track simple Chi2, >=12 hits
Entries 4197
Mean 15.21
RMS 9.555