Kalman Filter Pattern Recognition

Robert Johnson

U.C. Santa Cruz

Introduction

- Goal: implement a new pattern recognition based on a Kalman filter for track following.
  - Fit to actual SSD measurements, not measurements combined into 3D points. This could improve efficiency.
  - A fast linear fit is used with 3 stereo and 2 axial layers is used to try all reasonable combinations of hits.
  - The linear fit results are used to seed the Kalman filter, starting with the fits that project best back toward the origin.
  - Use a rigorously generated covariance matrix, including multiple scattering, for hit selection and rejection while following track candidates.
  - Try to keep the code lean and fast.
- Brief Status: the Kalman-filter track fit works well in fitting a selection of simulated hits. A pattern-recognition code based on the Kalman filter now exists in trial form.

Introduction

- Thus far this is still a stand-alone program, which I am testing by using a toy Monte Carlo to generate hits with perfectly Gaussian errors and perfectly Gaussian multiple scattering.
  - The point here is to make sure that it works mathematically when applied to hits simulated from a consistent model.
- The geometry used mimics the HPS geometry, with the expected plane spacing and stereo angles.
  - Random SSD plane misalignments are applied, and the code can handle detector planes that are not perpendicular to the beam axis.
- The HPS field map can be used, or a constant field assumed. The field map breaks the mathematical model slightly, because each helix step in the fit assumes a locally constant field.

Initial Helix Guess Generation

- The guess is generated by simultaneously fitting a line in x,z (where x is the beam axis and z the B field axis) and a parabola in x,y to minimize the residuals with respect to the single coordinate measured by each SSD.
- This linear fit is only a mathematically approximate model, even for my idealized Monte Carlo simulation:
  - It assumes a perfectly constant and perfectly aligned magnetic field.
  - It assumes zero scattering (and no energy loss).
  - The parabola is only an approximation to a circle, and even the straight line in x,z is approximate if the helix curvature is non-zero.
- Nevertheless, it generates helix parameters and associated covariance matrix that serve well to start the Kalman filter.

Linear fit, status

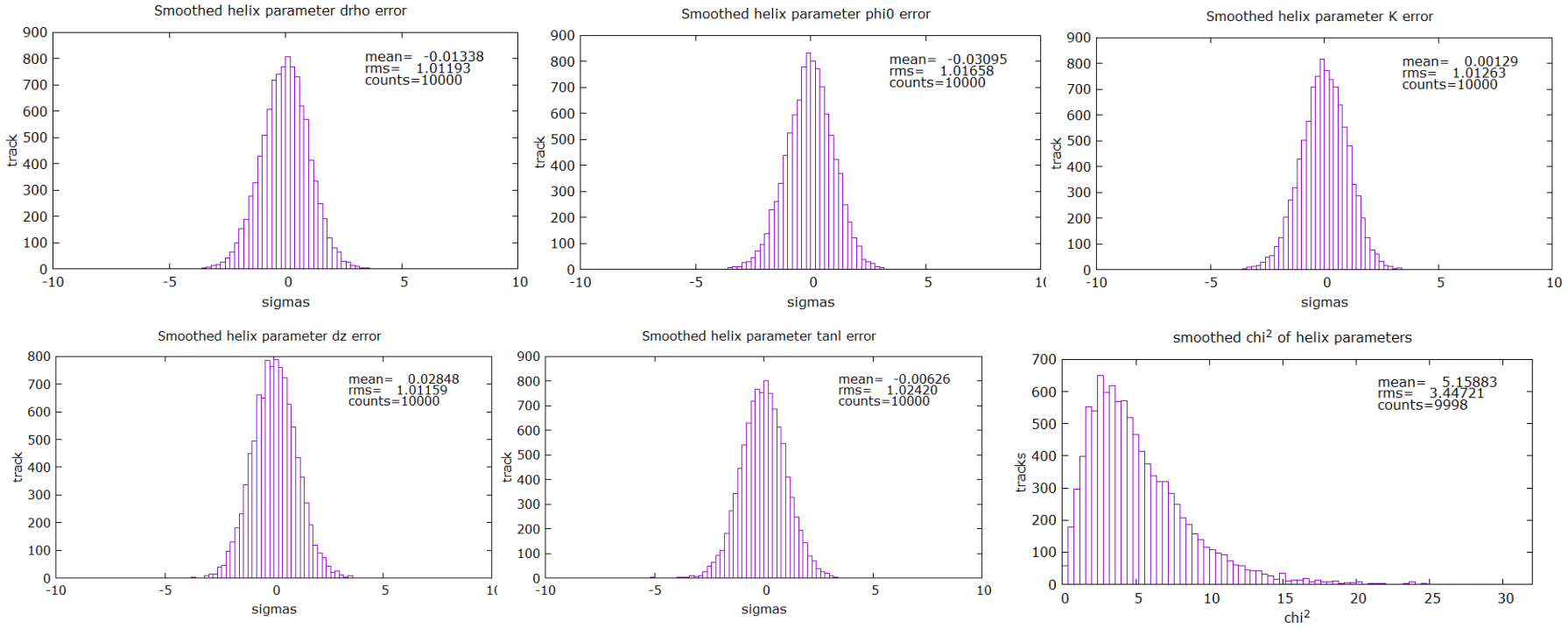
- Just last week I completed the linear fit such that it works properly with known arbitrary offsets and small mis-orientations of the SSD planes.
- The code is updated in Github as branch iss204b.
- With a constant magnetic field and high momentum (e.g. 1 GeV, so minimal scattering) it generates the correct error estimate on each of the five helix parameters.

Kalman Filter Fit Performance

- The linear-fit results are used to start the Kalman filter, and two fit iterations are done.
- The plots below show that the covariant matrix at the end of the smoothing operation is correct.
- The chi-squared of helix parameters (the last plot below) is calculated as

$$\chi^2 = \boldsymbol{e}^T C^{-1} \boldsymbol{e}$$

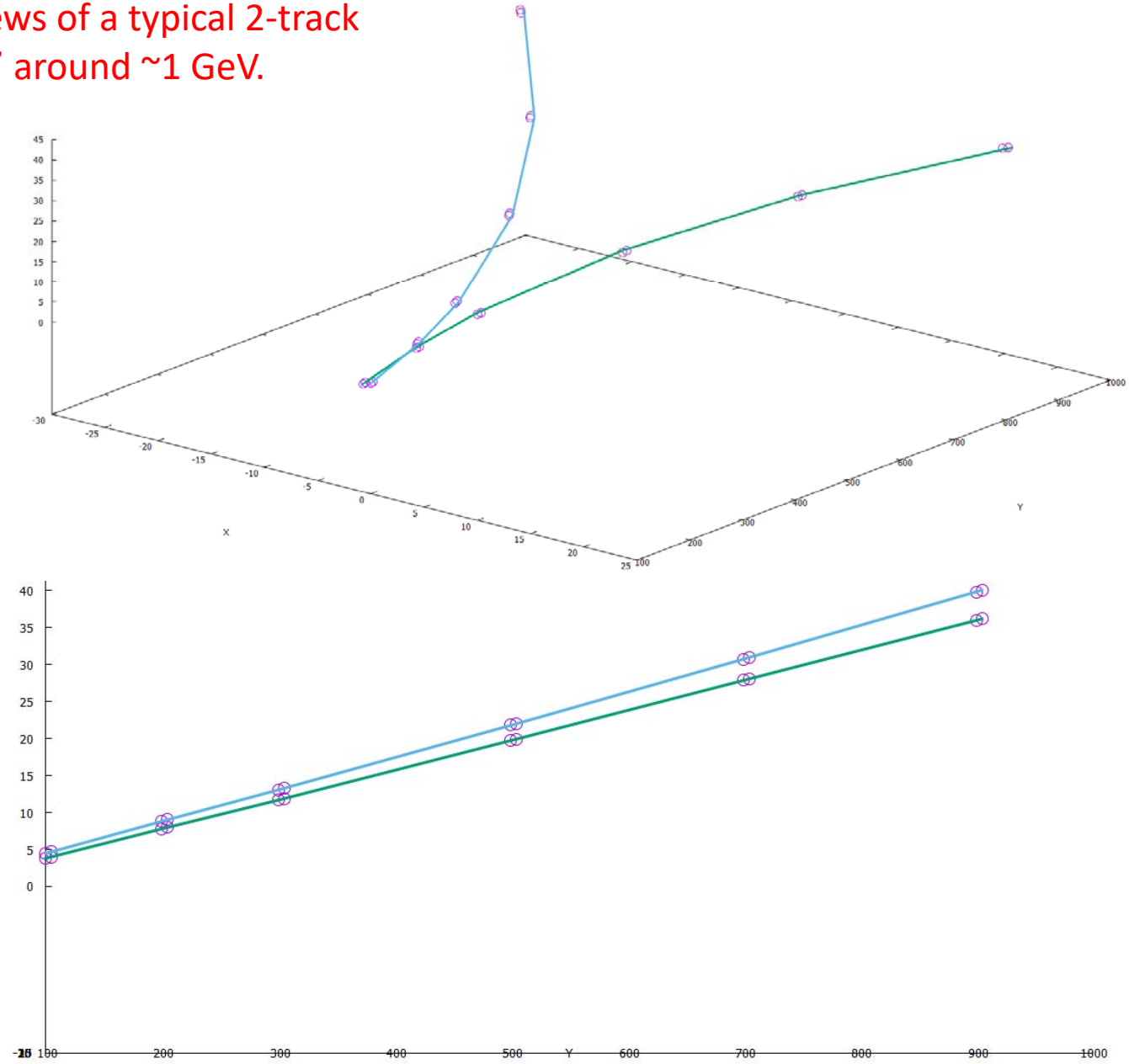
where $\boldsymbol{e}$ is the vector of errors, relative to the true values of the helix parameters, and C is the covariance matrix.



Pattern Recognition

- I have implemented a preliminary combinatorial pattern recognition based on the Kalman Filter track following and fitting.
- Seeds are generated from hits in 2 axial layers plus 3 stereo layers by the linear fit.
  - Multiple different sets of 5 such layers are tried (but probably not the first and last layers, where the field is non-uniform).
  - All hit combinations in the 5 layers are tried and accepted or rejected according mainly to how closely they extrapolate to the origin. (No chi-square cut as this is a 0 d.o.f. fit.)
- The seeds are sorted by quality and then followed one-by-one with the Kalman filter, working first toward the origin.
  - Tracks are allowed to share only a limited number of hits.
  - Shared hits are reviewed at the end and dropped or reassigned if appropriate.
- Final accepted tracks and covariance matrices are extrapolated by Runge-Kutta integration through the non-uniform field to the origin.

Two views of a typical 2-track "event" around ~1 GeV.

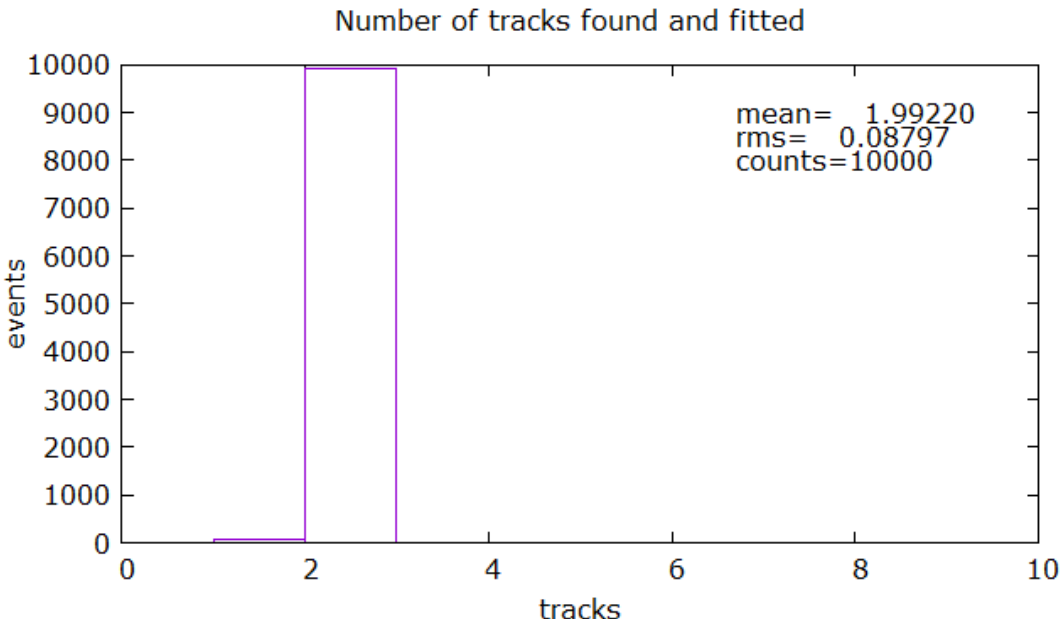


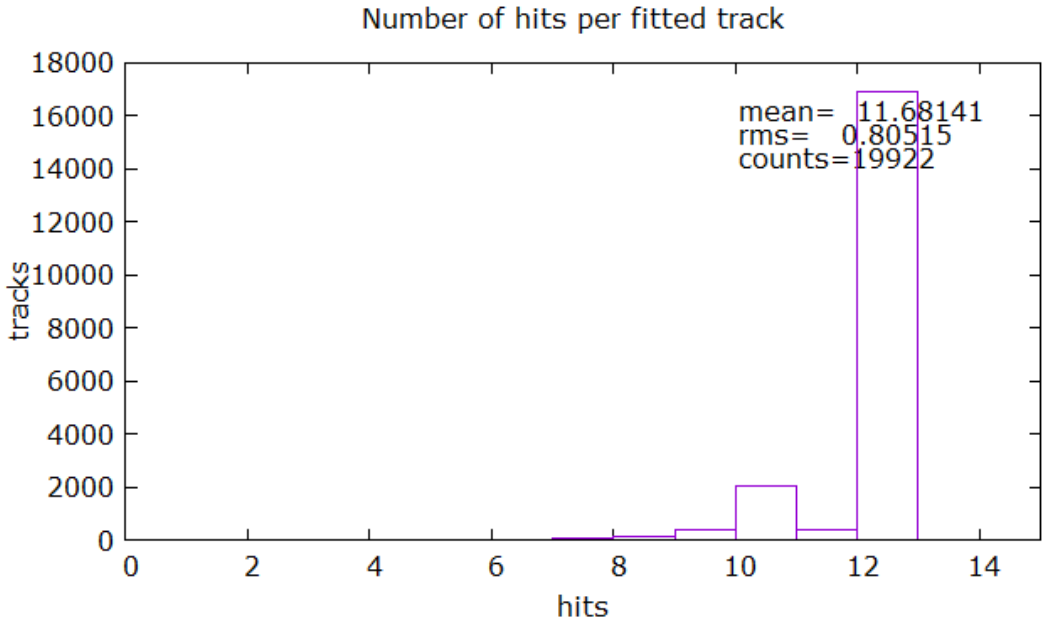
Preliminary test with 10,000 “events” of two particles each, with momentum spread around 1 GeV and launched from the origin.

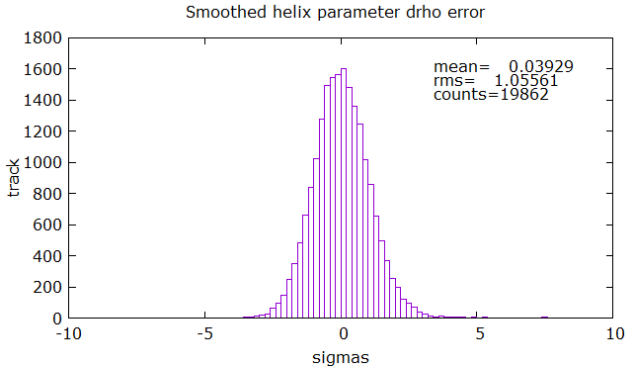
Almost all of the tracks are found, and most have all 12 hits.

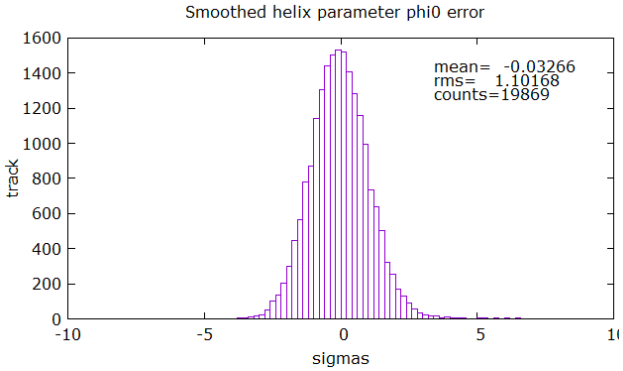
The CPU time was ~1.5 ms per event on my CORE-i7 notebook.

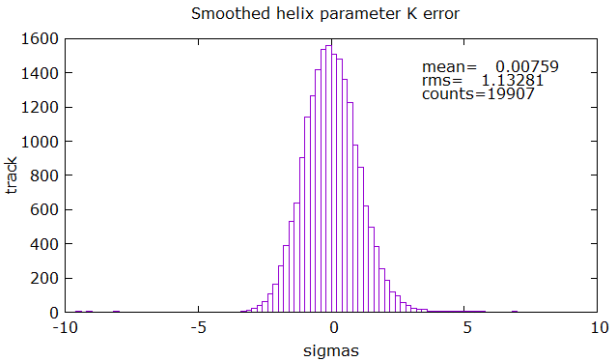
On the following page the reconstructed helix parameters, after swimming to the origin, are compared with the generated values at the origin.

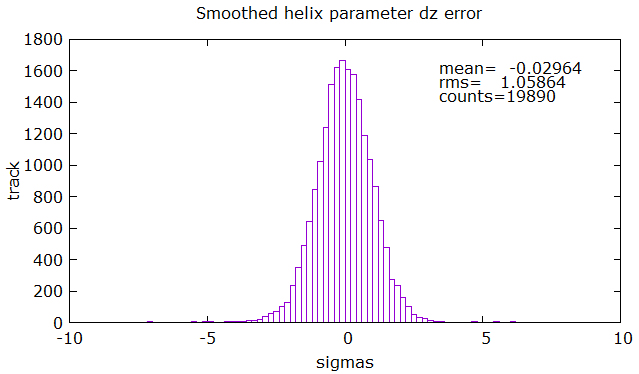


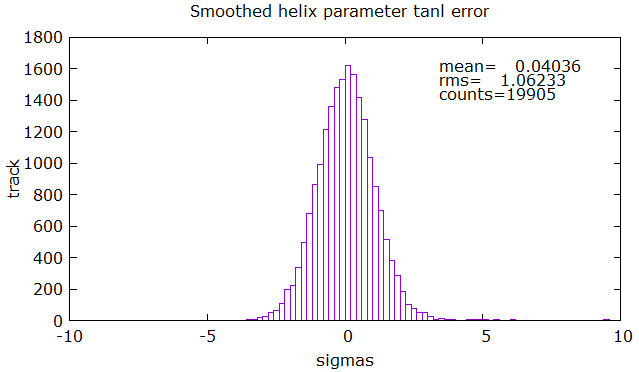




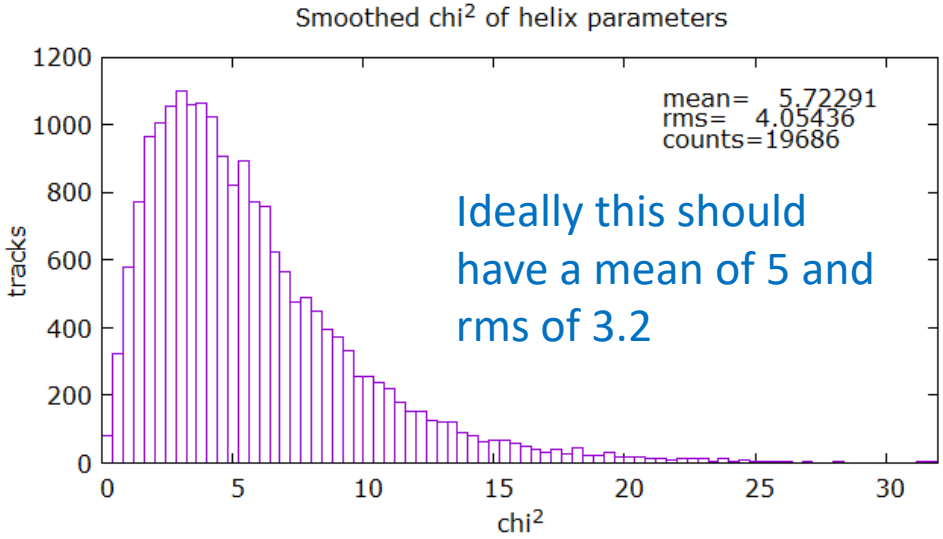








The Runge-Kutta extrapolation of the covariance matrix to the origin is not working perfectly, as the errors appear to be underestimated by several percent (despite being perfect at the first measuring plane).



Code Status

- All of my testing thus far has been done with a simplistic home-made Monte Carlo, with perfectly Gaussian scattering and no bremsstrahlung, etc.
- Miriam has interfaced the Kalman fitting code into hps-java, to test it with the full simulation, but using 3-D hits already pattern-recognized.
  - Testing of this work is in progress after going through one round of corrections to my code, so I have no results to show yet.
- Once that is working and well tested, then we can interface the pattern-recognition code to start with SSD 1-D hits.
  - I hope that by end of summer this will be working, and I will be busy optimizing it using Geant-4 simulated events.

Next Steps

- Work with Miriam to complete the interface with the HPS code.
- Test with MC events and tune the parameters and algorithms to optimize performance and execution speed.
- Tie up some loose ends. For example, “KalTrack.java” could easily include a method to extrapolate to the electromagnetic calorimeter, if needed.
- Test with some real data.