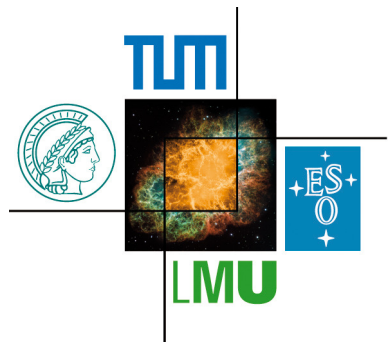
Recent Changes related to GENFIT

Tobias Schlüter (2015-04-21)
LMU München



Recent Developments related to GENFIT

1. fixed bad P-values in track fit
2. standard step length estimator in Runge-Kutta
3. Geant4 material interface speed ups
4. `TrackFitResult` moved to framework `Helix`
5. Electron Tracking Performance Evaluation

And one item I'm not discussing:

6 alternative implementation of Kalman fitters using square-root formalism (mainly crosscheck)

Bad P-values in Track Fit

Ever since we introduced the new `CDCRecoHit` which has the realistic translators, P-values looked bad.

- Ozaki-san found that this is due to the mirror hits being mixed up
- which leads to small additional errors on them (x-t relation different for mirror hits).

The reason: the L/R flags aren't documented and GENFIT's coordinate system made a different choice.

- by re-reading the code I find that the utility function `CDCGeometryPar::getNewLeftRightRaw()` returns `1` for the right hit and `0` for the left hit.
- this is also the input for `RealisticTDCCountTranslator::getDriftLength()`'s argument `bool leftRight` …
- …which has the even more articulate name `ambiguityDiscriminator` in the base class

Of course, the documentation strings for all this are not useful at all. It is never specified what left/right means. Upon request, Ozaki-san refers to a slide in a presentation by Oliver.

How to pass left/right information to genfit

The `CDCRecoHit` can be told which mirror hit is the correct one to choose:

```
/**
 * select how to resolve the left/right ambiguity:
 * -1: negative (left) side on vector (wire direction) x (track direction)
 * 0: mirrors enter with same weight, DAF will decide.
 * 1: positive (right) side on vector (wire direction) x (track direction)
 * where the wire direction is pointing towards +z except for small
 * corrections such as stereo angle, sagging
 */
 virtual void setLeftRightResolution(int lr) { m_leftRight = lr; }
```

The comment is now updated to give a unique definition of the sides. The information is stored in track candidates by using a `genfit::WireTrackCandHit`

New P-value distribution

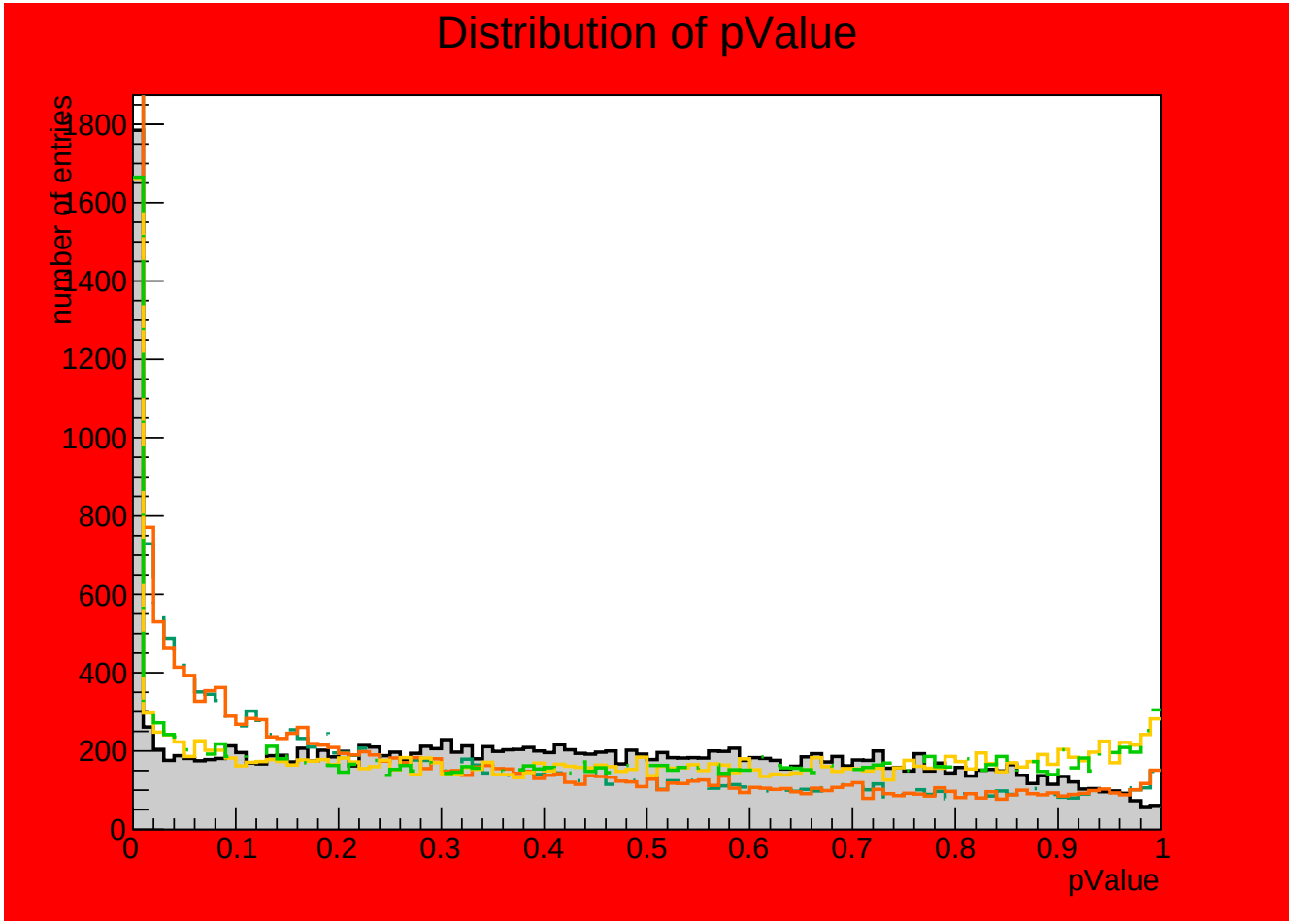


Figure: P-value distribution. Yellow and light green: fixed. Other colors: previous builds. Black: reference.

Step Length Evaluation in Runge-Kutta Propagation

Extrapolation proceeds in steps whose lengths are limited by various possibilities:

- extrapolation goal reached
- too much energy loss
- material changed
- Runge-Kutta error estimate too large

In the homogeneous field, the step length is rarely limited by the RK, yet the RK step length estimator showed up as 1 % on the profile.

- the RK step length estimator used `pow(error_estimate,1/3.08)`, and the value is calculated very often
- this is slow
- and the exponent is non-standard (should be 1/5 for a fifth order RK)
- we did not manage to reconstruct the history of that exponent
- changed to standard value, and don't calculate always

Led to no change in performance except CPU (as expected for the homogeneous field).

Geant4 material interface speed ups

Material lookups using Geant4 were hideously slow, adding 25% to the execution time of the `GenFitterModule` when using Geant4 for material lookup instead of ROOT's `TGeo`.

- Geant4 makes lots of assumptions about how its material lookup machinery is used, and it keeps a lot of state information to accelerate the foreseen usage pattern
- if the assumptions are not met, this can lead to wrong materials being returned or even crashes (which are bugs, according to the documentation it should be more tolerant)
- so GENFIT has to be extra careful in using it

All the required backing up and re-searching costs time, so I had to minimize the number of material lookups.

- the algorithm I had used for `TGeo` turned out to be overly pessimistic, making lots of unnecessary lookups in some rare cases
- and these completely dominated the execution time

Improvements

Improved the algorithm:

- if the track prematurely crosses a boundary (i.e., before reaching the estimated distance along the initial direciton), the code previously and stupidly took a half-length step, then looked from there (and thus taking a lot of ever-smaller steps if the boundary made a large angle WRT the track, as in the type of geometries depicted below)

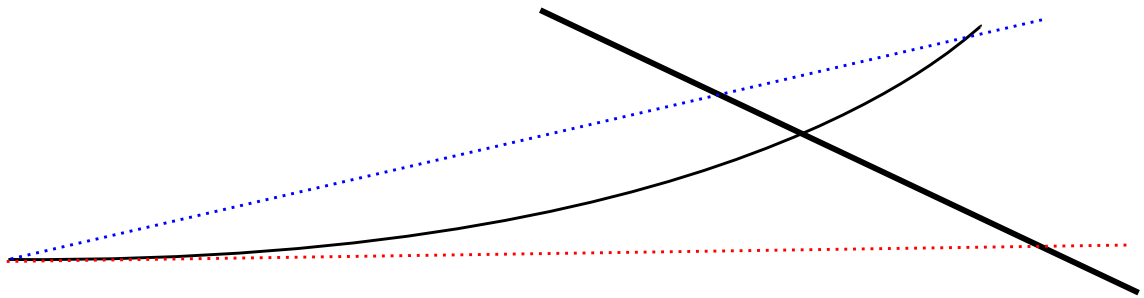


Figure: Material lookup: due to the slant of the boundary (thick black) the initial lookup (red) significantly overestimates the distance to the boundary. In this case, the new algorithm looks along the direction taken (blue) to guess a new step distance. This works, because I have already convinced myself that the distance between the blue line and the trajectory is smaller than the tolerance for the material lookup precision.

Implementation Issues

Difficulties:

1. trying to keep Geant4 from looking in places that I didn't ask it to look at (it tries to be smart, keeps a lot of state, and sometimes doesn't back up correctly if asked)
2. if the fit runs astray, it can happen that we look up material outside the world volume – fine according to the docs, but nevertheless crashes sometimes

So I had to be super-careful in how the code goes back and forth and how it recovers from lookups outside the world volume.

Results

Results:

- Geant4 material lookups are not as fast as `TGeo`, but with the improved algorithm the number of lookups explodes much less frequently than before
- genfit's Runge Kutta together with the new lookup algorithm is actually faster than Geant4's own Runge Kutta track following code, while Genfit's precision requirements are much more stringent by default
- most time is spent in `G4PolyCone::Inside()` – we don't however have many polycones in the tracker volume (beam pipe, CDC wall AFAICT), so replacing these may be an easy speed up

`TrackFitResult` moved to framework `Helix`

The code in `TrackFitResult` didn't support straight lines, whereas the framework `Helix` had numerical trouble calculating perigee parameters for tracks given really close to the IP.

- Oliver improved the calculations in the framework `Helix`
- and this is now used by the `TrackFitResult`

Two additionaly bugs found in the process:

1. the code putting together the covariance matrix in the `TFR` didn't produce a symmetric matrix. Fixing this led to errors in `test_analysis`. So the test actually tested for a wrong condition.
2. the Streamer version history of `TFR` is extremely convoluted and goes something like $1 \to 2 \to 1 \to 2 \to 3 \to 4 \to 3 \to 4 \to 5 \to 6$. The two versions 4 are incompatible, and there's no way to reconciliate them. Lesson: NEVER decrease the streamer version, even if you back up to an older version of the data structure, because this will only lead to trouble when the number is incremented the next time

Electron Tracking Studies

Markus Prim looked into the quality of electron tracks. Since his talk has a completely unrelated title, I briefly summarize what he found:

- turning off Bremsstrahlung corrections in Genfit's material handling significantly improves the precision of the reconstruction

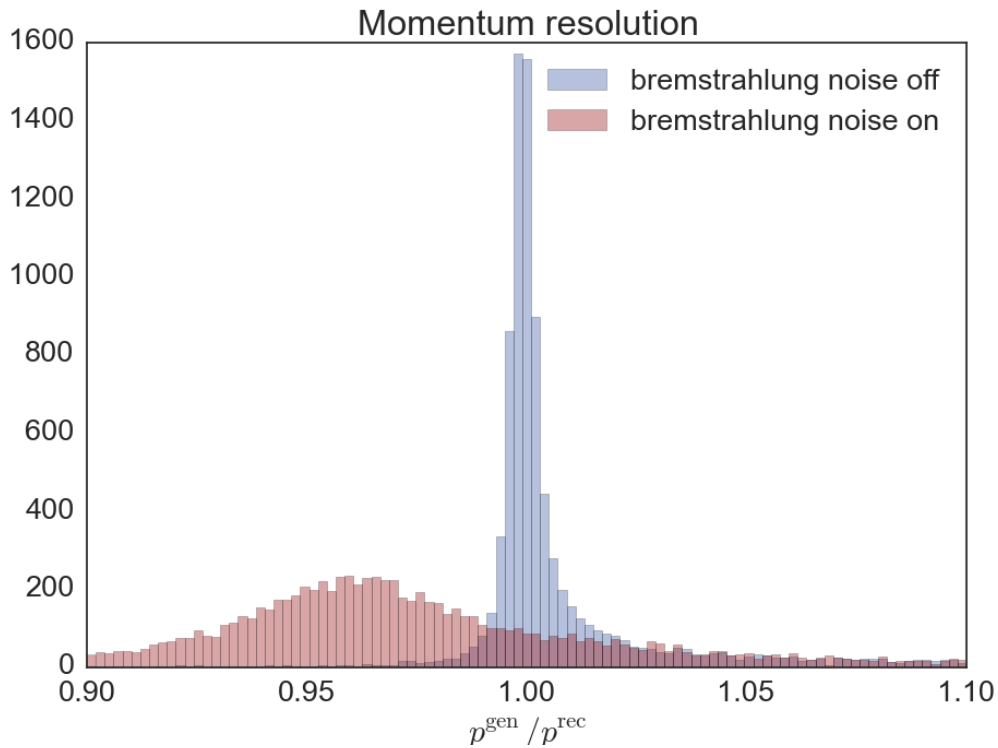


Figure: blue without Bremstrahlung noise and energy loss, red with Bremsstrahlung corrections