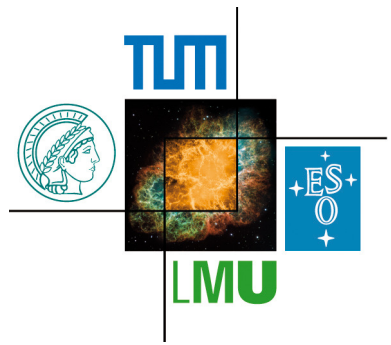
Recent Changes in GENFIT

Tobias Schlüter (2013-05-13)
LMU München



Recent Developments

1. ROOT 6 reade
2. new boundary finder
3. ROOT I/O rewrite
4. precision in Jacobians recovered
5. stuff for TPCs
6. perigee parametrization (not GENFIT proper)

ROOT 6 preparation

Very simple:

- allow `rootcling` as alternative to `rootcint` for dictionary generation

That's it.

New Boundary Finder

Tracking needs to take material into account

- multiple scattering corrections
- energy loss

Therefore we need to be able to efficiently find whether the material changes during an extrapolation step.

- I did not understand the original code
- it actually was wrong

New Boundary Finder II

New algorithm, inspired by Geant4. With inputs $(s_{\max}, \delta, \epsilon)$ (maximum step length, maximum length for stright-line step, maximum tolerated deviation of trajectory from straight line), we iterate statrting with $s = 0$.

1. if $s_{\max} < s + \text{safety}$, then step by $s_{\max}$ (safety is the shortest distance to any boundary)
2. if current step length $< \delta$, then step straight, return
3. if the upper bound on the deviation is $> \epsilon$ try with shorter step (but not less than safety)
4. if the volume changed after propagating by current step length, try again with shorter step (or safety)
5. no boundaries crossed, hence we can advance and start over
6. if we stepped by safety and the material changed, then the current step length takes us to the boundary

ROOT I/O rewrite

GENFIT uses smart pointers, hence we cannot use ROOT's autogenerated I/O

- originally, I wrote I/O code that duplicated some of the `shared_ptr`s, where the user could also have their own references (namely `DetPlane`)
- BUT ROOT does its own pointer caching during I/O
- this can interfere with the temporary objects I used during I/O
- this cache can be reset
- BUT I have no idea what damage this can do to the rest of I/O

Precision of Jacobians

Tadeas found some bad numerics during extrapolations. These could bias the alignment.

- the extrapolation code uses a 7D representation of the track
- but we were propagating the Jacobians in the 5D representation of the `RKTrackRep`'s external interface

After studying all of the remaining projections in the code

- the code now propagates the Jacobians in 7D

and the issue appears fixed (cf. Tadeas's talks)

TPC stuff

Invented a novel way of dealing with spacepoint (3D) hits.

- but no relevance to Belle2

Perigee Parameterization

The extrapolation code doesn't take us to the numerically exact Perigee. This lead to bad behavior of the code converting from $(\boldsymbol{x}, \boldsymbol{p})$ to the perigee parameters.

- fixed by using a six-parameter helix which besides the perigee parameter also includes an angle along the helix
- this way every point along the helix can be reached
- or, reversing, one can get the perigee parameters starting from every poitn along the helix (well, any point less than 90 degrees away)

I also implemented the Belle 1 helix including numerical error propagation as a crosscheck. Numerics confused me for a while until I realized that the `TrackFitResult` uses `float` for its intermediate calculations. Which is a very bad idea, and Markus will fix it.