Recent GENFIT WORK

Tobias Schlüter
LMU München

F2F Meeting (Karlsruhe)
2015-09-01

Overview

Glass Half Full

- we are now on `github`
- much deeper understanding of the RK code
- lots and lots of `git` checkins

Glass Half Empty

- no new features in the RK code (yet)
- no way of syncing `git` repo and `basf2` repo
- still no paper, no manual

Luckily there are other people

- Tadeas Bilka: `RecoHits` and alignment for BKLM
- Nils Braun: energy loss in track fits
- Markus Prim: progress with electron energy loss

github REPOSITORY

M. Prim and myself now use `github` for GENFIT devlopment

- master is still `svn`
- no real way of sync'ing
- I just clone the `git` repo into `basf2/genfit2/code2` and got from there

`https://github.com/TobiSchluter/genfit`

Runge-Kutta studies

I want to determine T_0 in the track fit

- need to understand how to understand how to evaluate Jacobians containing time in the numerical propagation (i.e. RK)
- derived how this works
- then found paper that describes how to evolve Jacobians

Wisdom

``Two weeks of work in the laboratory can save you a mornings trip to the library''
(well, it did take the better part of a day, not two weeks).

Working through this precisely and reading the literature, I noticed one possible issue.

RK STUDIES CTD.

- In the Runge-Kutta we carry the variables $(q/p, \mathbf{x}, \mathbf{T} = \mathrm{d}\mathbf{x}/\mathrm{d}s$ along some path length s. One important point is that the tangent $\mathbf{T} = \mathrm{d}\mathbf{x}/\mathrm{d}s$ is a unit vector.
- But the RK procedure does not guarantee this: for a step of lenghth S it calculates $\mathbf{T}_{n+1} = \mathbf{T}_n + (S \times$ some function), based on the Lorentz-force law and the specific RK scheme.
- So after each step, $\mathbf{T}$ is normalized, viz $\mathbf{T} \mapsto \mathbf{T}/|\mathbf{T}|$

Question

does this normalization destroy the quality of the RK estimation?

Answer

After spending a week developing an RK that behaved better (less non-converged tracks, yay!) and maintained this condition (differential geometry for the win) ...I realized that my ingenious method contained a mistake / incomplete thought and wasn't actually correct

But I could convince myself that the normalization does no harm

```
(* These should resemble sin / cos series *)
v = RKN[vStart = {0, 1, 0, 1, 0, 0}, fMag, h];
Series[v, {h, 0, 7}] // MatrixForm
```

trixForm=

$$\begin{pmatrix} h - \frac{h^3}{6} + O[h]^8 \\ 1 - \frac{h^2}{2} + \frac{h^4}{24} + O[h]^8 \\ 0 \\ 1 - \frac{h^2}{2} + \frac{h^4}{24} + O[h]^8 \\ -h + \frac{h^3}{6} + O[h]^8 \\ 0 \end{pmatrix}$$

```
normalize[v] // Series[#, {h, 0, 5}] & // MatrixForm
```

trixForm=

$$\begin{pmatrix} h - \frac{h^3}{6} + O[h]^6 \\ 1 - \frac{h^2}{2} + \frac{h^4}{24} + O[h]^6 \\ 0 \\ 1 - \frac{h^2}{2} + \frac{h^4}{24} + O[h]^6 \\ -h + \frac{h^3}{6} + O[h]^6 \\ 0 \end{pmatrix}$$

E.g., with or without normalization the fourth-order RK method that we're using reproduces correctly the power series of sine and cosine up to the fourth order for a planar track in a homogeneous field.

A speed-up that did not happen

- with a RK one make on fixed-size step. If one wants to find the function values at some intermediate point, one has to reevaluate from scratch
- during boundary finding, we actually go back and forth a few times
- there is a method of polynomial approximation (``dense RK'') that allows to approximate with the same quality at intermediate points at the cost of (in our case) two additional intermediate evaluations
- I implemented this, no speed gained (though it may become interesting in the future)

RK REWRITE

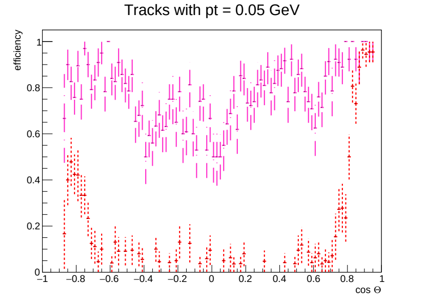
Our current RK code is complicated, difficult to follow

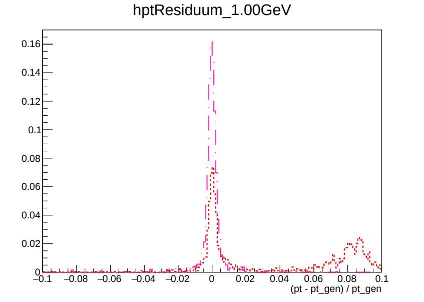
- it is derived from old Fortran code from the age where you could still outsmart the compiler
- expressions are optimized for minimal redundancy, hard to actually see the Lorentz-force in there
- energy loss is evaluated after propagation, this limits propagation steps, and adds several layers of complexity

So, since I had to rewrite this anyway in order to add flight-time to the variables that appear in the Jacobian, I set out to rewrite this, evaluating energy loss in the RK proper.

Two Miserable Weeks

Of course, this wasn't as simple as I hoped: in my synthetic testcases the new implementation works as well or better than the old code (nice). But in `basf2` ...





- efficiciency drops for very low p_T (others look fine), definitely not what's expected if you deal with energy loss correctly
- weird double peaks in momentum: lots of investigation: current status: 1) doesn't affect tracks in the upward 45 degrees 2) the bad tracks don't take material into account, even though the stepper sees it 3) I would not expect an overestimation of momenta if material is missed

Summary

The good

- Runge-Kutta even with normalization is fourth order
- new RK code is shorter, much simpler

The bad

- new RK code loses efficiency at low p_T

The ugly

- weird double peaks
- my contract runs out end of October …