Subject: Re: Fitting scheme
Posted by killenberg on Fri, 14 May 2010 15:21:02 GMT
View Forum Message <> Reply to Message

Hello Christoph,
hello Jason,

please excuse my delayed reply.

rosemann
residual -- the geometrical distance in x/phi at constant y/r between an associated hit (from track finding) to its fitted track, when it is excluded from the fit distance -- the geometrical distance in x/phi at constant y/r between an associated hit (from track finding) to its fitted track, when it is included from the fit

The resolution is then the geometric mean of both values. (For reference check e.g. Ralf Dieners thesis or physics/0402054)

I was not aware that residual and resolution are used with this definition at Desy. Now I understand why you don't like it the way it is used in the current scheme. Here is how I used it:

residual -- The geometric distance (calculated by a method defined by the actual fitter implementation) of a 3D space point (hit) to a track hypothesis (=track with a certain set of parameters).

This is much more general than the definition you use. Especially with the usage of the term "resolution" I feel a little uncomfortable. For me resolution is the width of the residual distribution, for example the RMS of a histogram, not a value of a single measured hit.

But also restricting the measured distance to x/phi at constant y/r is not general enough:

Hit coordinates are stored in global coordinates. The pads are in local coordinates. For the LP for instance you hit one module in each module row. So you have hits from three different local polar coordinate systems, which are stored in global Cartesian coordinates.
For the Timepix there is no bias of a pad row, the hits are just measurements of x and y. One wants to measure the distance perpendicular to the helix.

The geometric mean is used because it is an unbiased estimator for the resolution. But we always used the mean of the withs of the two distributions (with and without the specific hit used in the fit). I don't know which is correct, probably both methods are equivalent if you calculate it through.

However, there are other methods to calculate an unbiased estimator. For instance one can use the distance calculated with the hit included in the track fit and multiply it by sqrt(n) / sqrt(n-DoF), where n is the number if hits in the track and DoF is the number of degrees of freedom in the track fit.

Another method is to use an unbiased track estimate, for instance from a separate measurement with a hodoscope. Or using MC truth in simulation.

This is why the interface has two functions:

```
EVENT::DoubleVec TrackFitterBase::calculateResiduals(EVENT::TrackerHit *testHit,
EVENT::Track *trackWithoutTestHit);
IMPL::TrackImpl *   fitTrack (EVENT::Track const *seedTrack);
```

The first function calculates the residuals to the track that has been provided. The naming of the variable should definitely been changed, it is somewhere between misleading and wrong. How about "trackHypothesis"? There is also a calculateResiduals function which takes the track and the hit number on the track. It was intended to be a convenience function which calls the other one. But I see that it probably causes only confusion.

The fitTrack does what Christoph wants to do with the simpler "fit object". You put in an seed track (probably also not a good name since it is not a seed for pattern recognition) which serves as a container for the hits and provides start parameters for a numerical minimisation. And you get out the fitted track.

To avoid code duplication a chi^2 fitter will call calculateResiduals() during the minimisation. And the histogramming processor calls it to get the residual values.
In addition, once the chi^2 algorithm is implemented one can inherit from it and just change the calcualteResiduals (for instance from 'perpendicular to the helix' to 'along the pad row'.
A Likelihood fitter does not use the geometrical residuals in the fit process. Here it behaves like Jason's pair proposal: One function for fitting, one for residual calculation.

I always considered the decision which definition of resolution (in the sense of width of residual distribution) to use as part of the analysis. This is why there are several histogramming processors for different definitions. The processor for geometric mean resolution for instance loops all hits, removes
the hit from the track, refits it and then retrieves the residual with and without the hit included in the fit. It has to be done 'manually' and is not in the fitter interface.

According the factory:

I was looking for a way to have only one histogramming processor for each "resolution" definition which works independently from the residual implementation (i.e. track fitter implementation) that was used. The Japanese group proposed that one possibly would use different fitters depending on the track properties (I can imagine using a different fit for particles coming from the vertex). So I would not store the fitter type as a collection
parameter but as a number in the track itself (bits 16-23 in the track type word). So every fitter implementation needed a unique ID, which is the peculiar list of types in the base class.
As the geometric mean processor does refitting and some implementations need additional parameters for this, the factory has an interface which takes LCParameters. The fitter processor stores these parameters in the tracks collection, so you pass its collectionParameters to the factory, together with the ID you retrieved from the track.
A combinatorial Kalman filter might not only want to reject hits but also to add hits that were not associated to the track. To have access to the hits collection (or whatever collection might be needed) a pointer to the event can be passed.
The factory reads out the implementation specific parameters and calls the constructor of the fitter. For every new fitter implementation the factory has to be extended. This is the price you have to pay to have the histogramming processor independent from the fitter implementation.

I admit the scheme is complicated and a bit tangled, but I could not come up with a simpler solution for the functionality I wanted.

Cheers

Martin
