The Trajectory Class

Ties Behnke, for Thomas Kraemer, DESY

- What is a trajectory
- Implementation proposal
- Status

The work presented here has been done to a large extend by Thomas Kraemer, who could not be at this meeting personally

Thomas has written a short note on this, which will become available soon

Trajectory: Why

LCIO data model:

- Tracks
- Cluster
- Reconstructed Objects

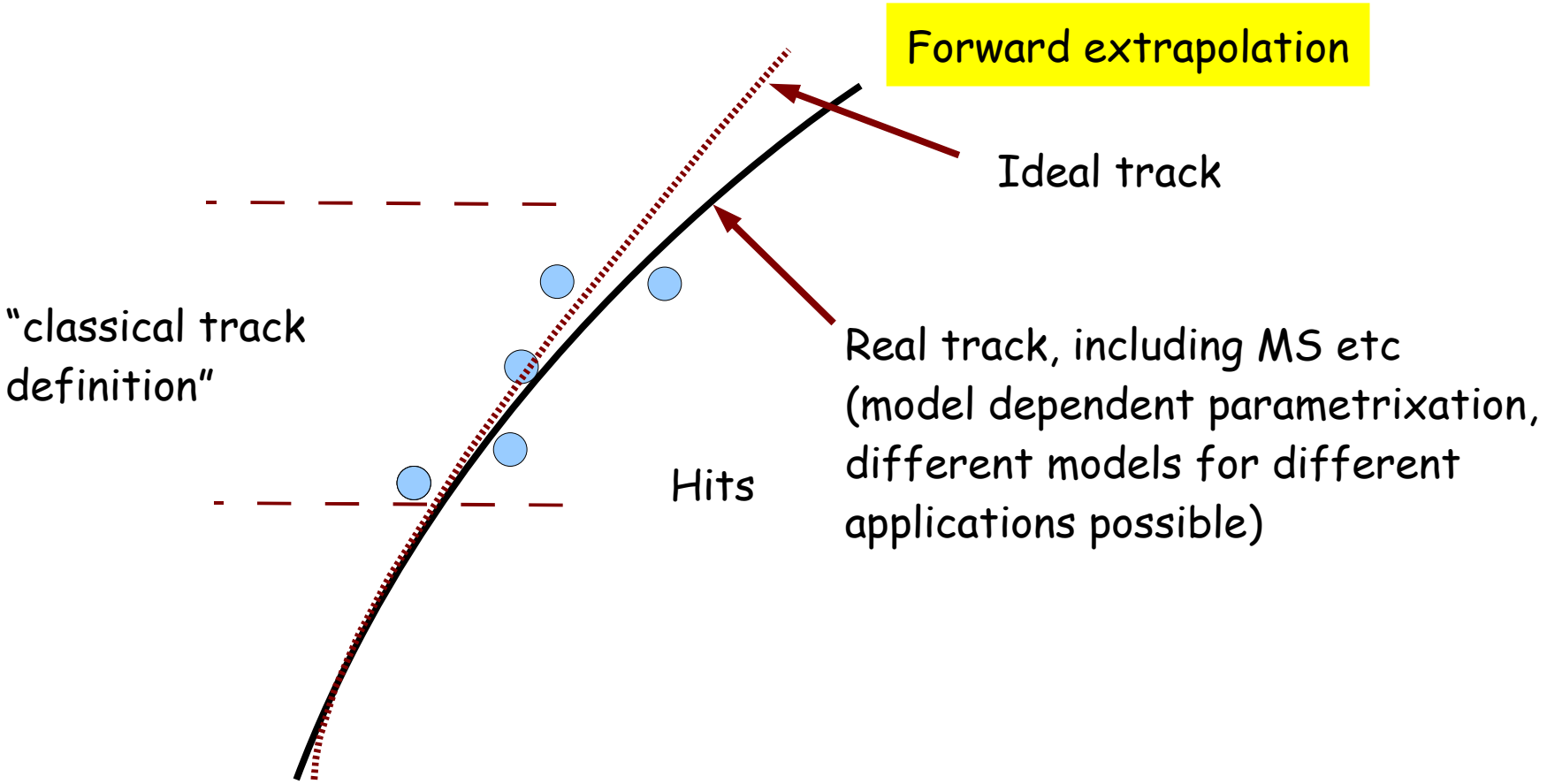
LCIO defines only data entries,
data and function are separated
(well motivated by the goal of LCIO)

Convenient reconstruction needs more:

- Generalized track
- Extrapolation
- Interpolation
- Flexibility

Trajectory as a generalized concept

The Trajectory



Particle Flow in particular needs a very powerful trajectory class

The Proposal

Define a "Trajectory" interface which

- Describes a very generic path of a particle in space
- Starts from the hits in the (generalized) tracker
- Includes the results from a "Fit" to the hits.
  - Simple implementation with "classical" helix fit
  - More complex implementations possible (energy loss parametrizations, ...)
- System should know about the relevant geometries
- System should be able to extrapolate in both directions along the track into other sub-detector
- System should allow "navigation" through the detector (intersections, etc)

Status

First very simple implementation (Helix, straight line) is available,

but implementation is not complete

To be discussed:

- details of the proposed class
- Relation between the Trajectory class and the LCIO classes
- Relation to helper functionality (fitter...)
- Interface to geometries

Trajectory/ LCIO

Trajectory (at least at the moment) is not meant to change the LCIO data model

It is a "second generation", derived class, which can be built from LCIO information.

It will not be (by default) persisted