




# ILD Tracking Software for the DBD

Steve Aplin  
DESY

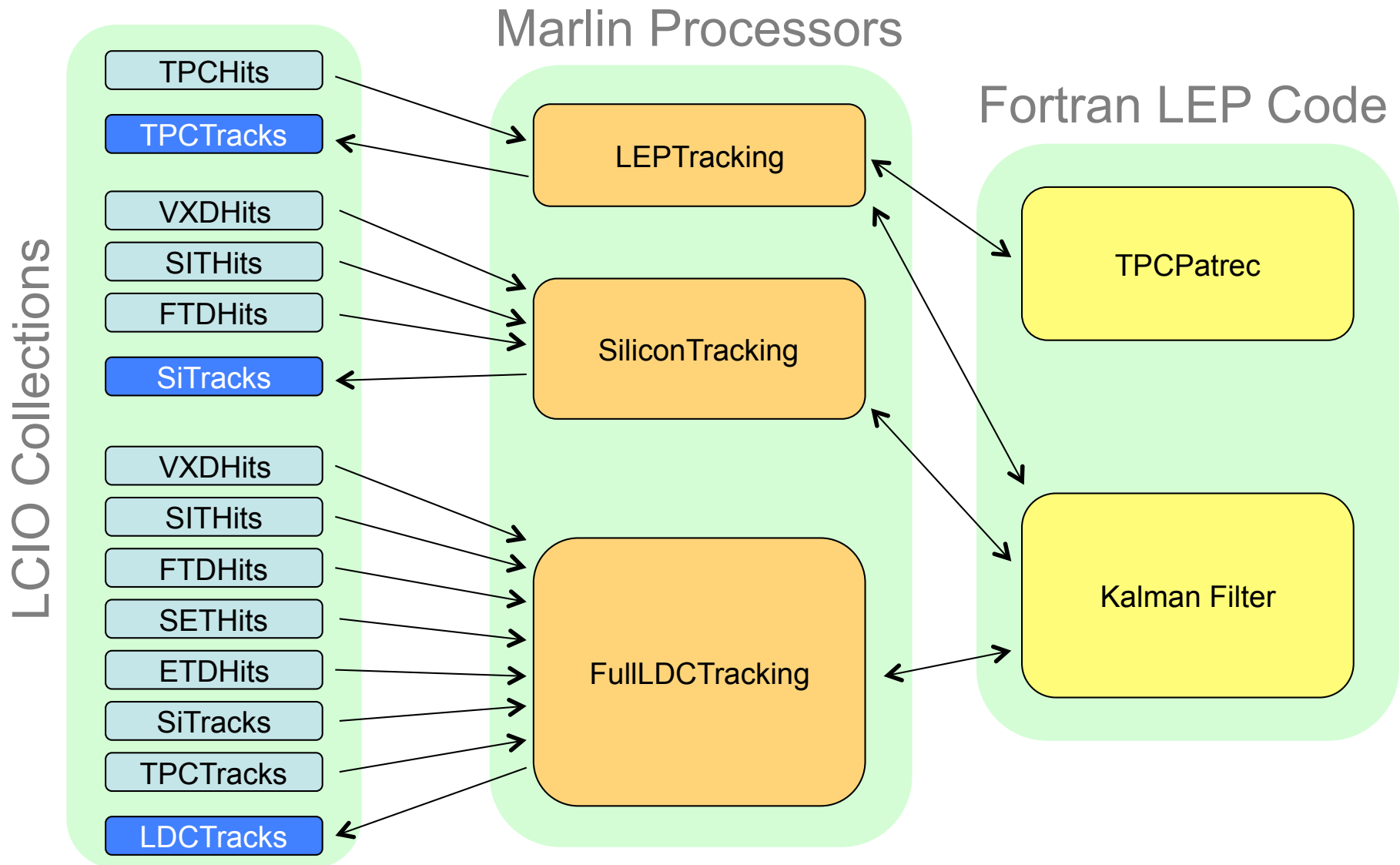
ILD Software Meeting  
27<sup>th</sup> January 2010



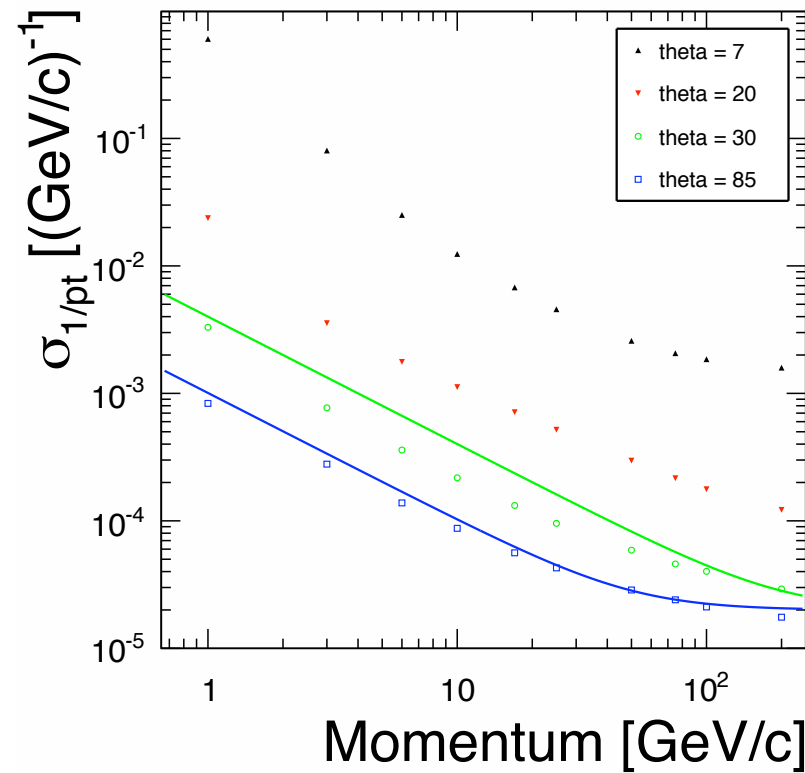
# Current use of Fortran and C++

- All Track fitting done is done in Fortran
- TPC Pattern recognition is done in Fortran
- VXD+SIT+FTD Pattern recognition is done in C++
- Final Track Building + Extra Hits + SET + ETD – C++
- Final Track fitting is done in Fortran

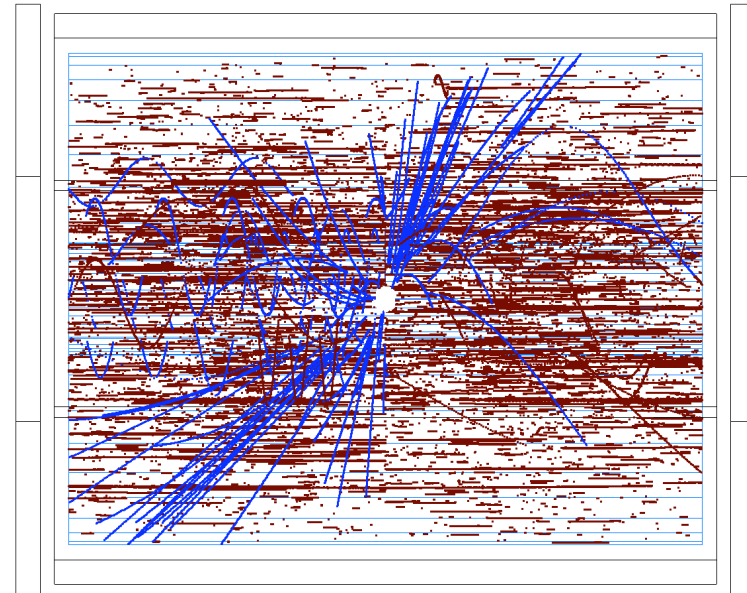
# Tracking Software used in LOI



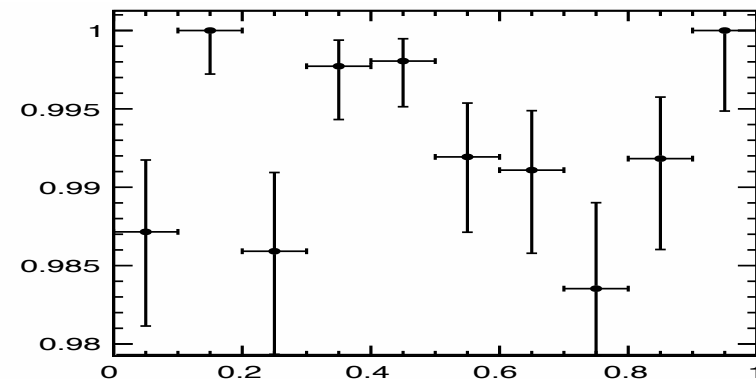
# Tracking Software used in LOI



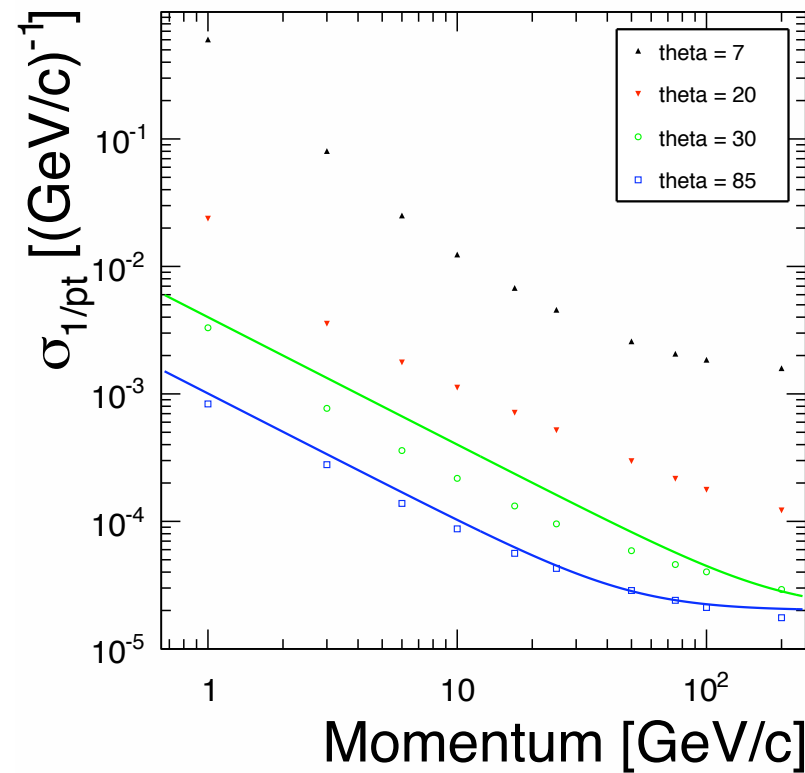
—  $\sigma_{1/pt} = 2 \times 10^{-5} \oplus 1 \times 10^{-3} / (pt \sin \theta)$   
—  $\sigma_{1/pt} = 2 \times 10^{-5} \oplus 1 \times 10^{-3} / (pt \sin \theta)$



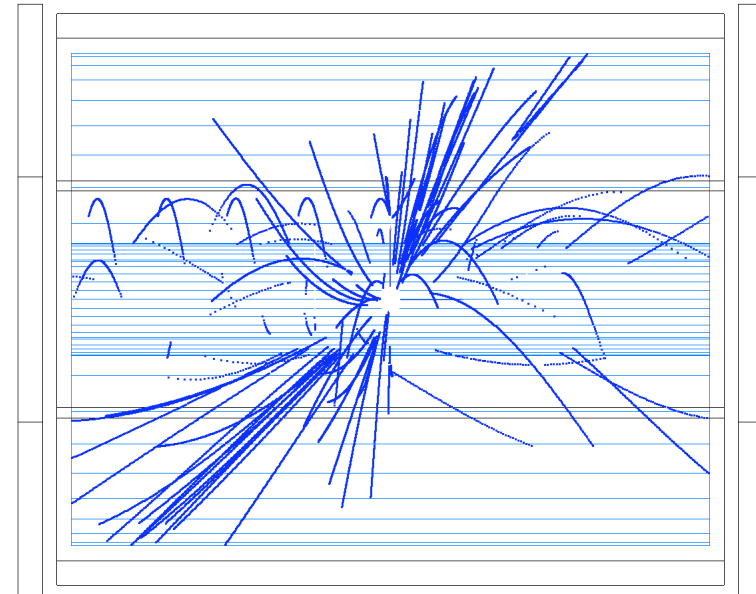
TPC Tracking Efficiency (Good Tracks) vs  $\cos \theta$  ( $p > 1 \text{ GeV}$  and  $N_{\text{Hits}} > 30$ )



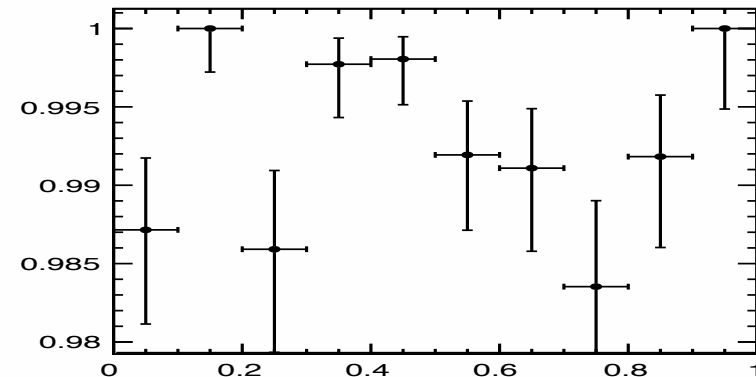
# Tracking Software used in LOI



—  $\sigma_{1/pt} = 2 \times 10^{-5} \oplus 1 \times 10^{-3} / (pt \sin \theta)$



TPC Tracking Efficiency (Good Tracks) vs  $\cos \theta$  ( $p > 1 \text{ GeV}$  and  $N_{\text{Hits}} > 30$ )



# Outstanding Issues

- The description and treatment of strip detectors
- Pattern recognition in the silicon trackers, esp. FTD
- TPC Pattern recognition is based on single BX's
- The use of a Homogenous B-field
- 1TeV not yet thoroughly tested
  
- Code is generally messy and hard to maintain
- The Fortran code makes modification difficult, as well as presenting memory issues, esp. w.r.t. background studies

# Requirements for a replacement

- Tracking Software needs to be used for mass production this means being able to reconstruct millions of challenging events, whilst maintaining an efficiency of better than 99%, including up 1TeV
- Use more realistic measurements, i.e. move away from 3D space points and towards measurements on surfaces, e.g. pixels, strips.
- Able to survive the very high background
- Able to cope with a non-uniform magnetic field

n.b. here I will not address fast sims, i.e. LiCToy and SGV

# Use of Atlas Tracking Code

- Atlas Tracking Code contains a well defined Tracking and Geometry EDM, which are used by a set of Track Fitting Interfaces, with implementations of a Kalman Filter, CKF, DAF, together with a geometry navigator.
- We spoke to Atlas Tracking Authors at CHEP 09 where they initially expressed an interest in seeing their code reused. Presently it is not clear what the position of the Atlas management is in regard to this proposal.
- To evaluate the code I am presently trying to build a prototype of the Atlas tracking code for a TPC, outside of the Gaudi and Athena frameworks, ideally in Marlin
- This was initially with the aim of not significantly modifying the codebase, e.g. use simplified header files, typedefs and #define statement wherever possible.



# Use of Atlas Tracking Code

- The Code is well structured, well written and also very well documented.
- Unfortunately the dependence on Gaudi and Athena appears to us to be quite involved.
- In order to proceed we would need to find solutions concerning the use of Gaudi Algtools, as well as for the replacement of the use of specialized Athena Container classes, which directly concerns memory management.

# Use of Atlas Tracking Code

- With regard to using the Atlas Tracking code, presently the most appealing option would appear to be the following:
  - use the core of the Atlas Tracking Event Data Model. This has a well defined set of base classes, describing measurements and geometrical surfaces independent of Gaudi and Athena.
  - use the core mathematical classes and operations, which are also independent of Gaudi and Athena.
  - this would then allow us to use the algorithms used within the Atlas Tracking code by re-implement the algorithms and interfaces in our own framework.
- We aim to reach a decision on how to proceed by early March.

# Tracking Activities within LCTPC

- KalTest
  - written by Keisuke Fujii et al.
  - ROOT based Kalman filter C++ library
  - Being brought into MarlinTPC by the LCTPC Group
  - Currently implementing a Combinatorial Kalman Filter
- Broken Lines Fitting Algorithm
  - written by V. Blobel at DESY
  - written in Fortran
  - Claus Kleinwort is currently working on wrapping this up in C++ for CMS, and hopes to then be able to use it for LCTPC

# Summary

- We Need to come to a decision quickly about how to proceed
  - We Need to collect the limited man power and focus efforts
    - This means ensuring that interfaces are established so that work presently being done can be brought together efficiently
  - See what needs to be done in LCIO concerning tracking EDM
  - Geometry will be a key problem to solve
- 
- Within the AIDA EU Grant proposal, Tracking has been specified as a major part of the Software Work-package