Status of the Track Fit and Framework

Matt Needham

- Introduction
- Design
- Performance
  - Pull Distributions
  - Comparison with SICB
  - Timing Study
- What's Missing
- Summary/Conclusions

Introduction

- Framework exists in Gaudi for track reconstruction
- **We have** first version of track fit in Gaudi with same $\sim$ functionality as we have in SICB
- We re-use some SICB code (mainly the in the transport + geometry)

Some hints on how to use it can be found at:

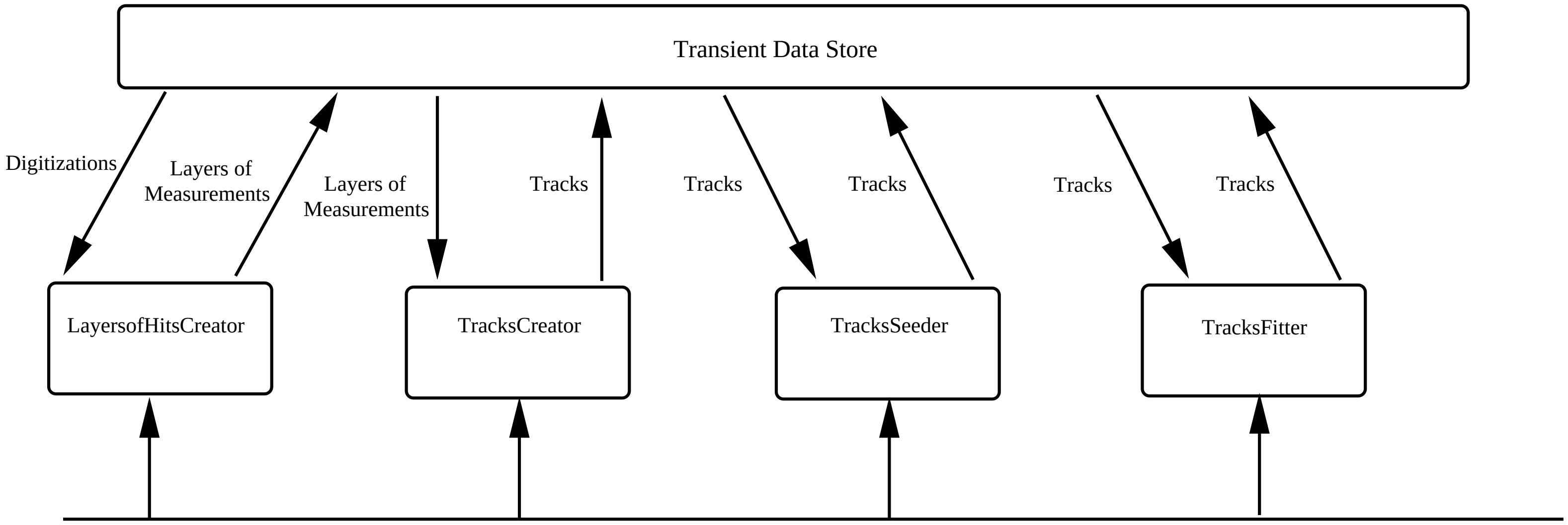
http://home.cern.ch/l/lhcbott/www/tracking/

Talk today concentrates on:

- The Design
- What still needs to be done before the code is released

In the LHCB week a more physics performance related talk will be given

Design -The Algorithms

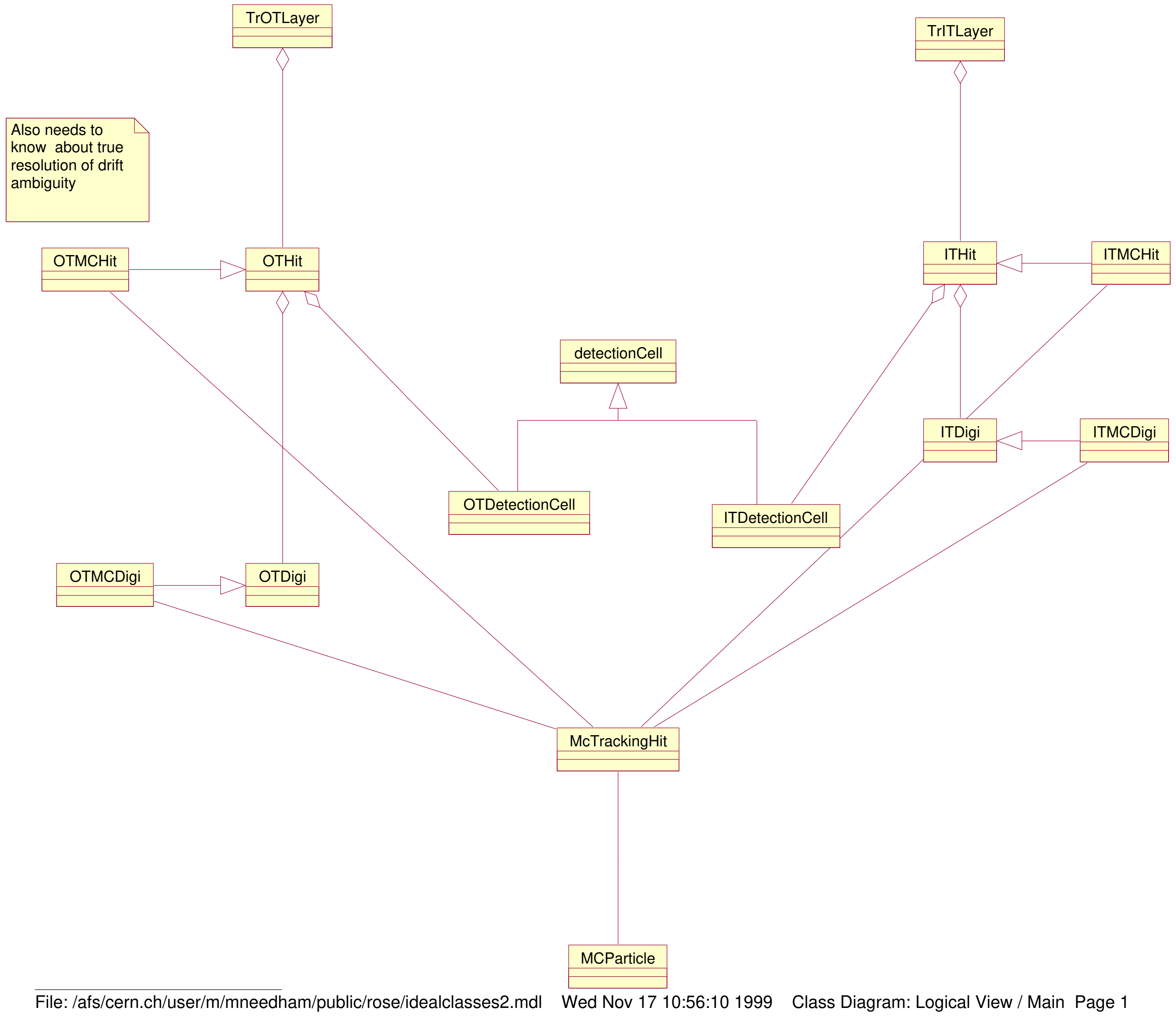


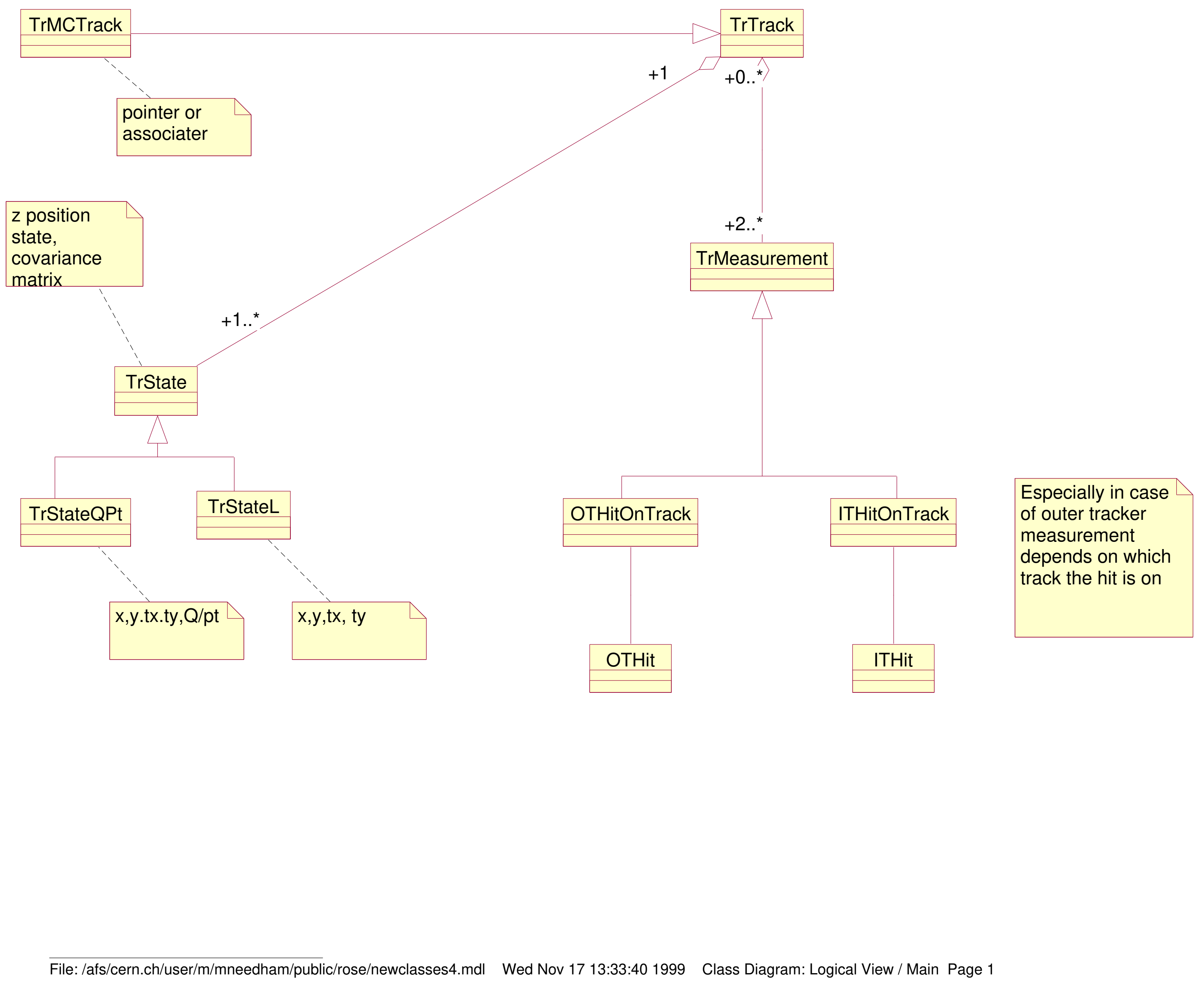
Input (wrapped) SICB digitizations $\Rightarrow$ output reconstructed tracks

Track $=$ collection of measurements and tracks parameters (states) at a set of user defined z positions.

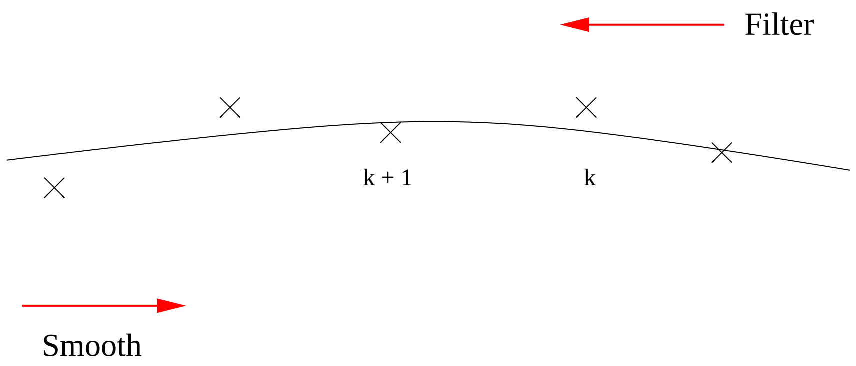
Main points:

- HitsCreator
  - Builds layers of tracking hits from SICB digitizations.
  - SICB geometry used to build database of detection cells
- TracksCreator
  - Builds cheated tracks using Monte Carlo truth information.
  - Select tracks to be fitted set via job options service
- TracksSeeder
  - To begin fit need a seed state $\Rightarrow$ cheated from MC
- TracksFitter
  - Kalman Filter technique
  - Re-use transport - ‘stand-alone’ SICB routine.
  - Uses simplified description of detector material that is built in SICB
  - Can intialize parameters needed by fit from job options.


TrOTLayer
TrITLayer
Also needs to know about true resolution of drift ambiguity
OTMCHit
OTHit
ITHit
ITMCHit
detectionCell
ITDigi
ITMCDigi
OTDetectionCell
ITDetectionCell
OTMCDigi
OTDigi
McTrackingHit
MCParticle
File: /afs/cern.ch/user/m/mneedham/public/rose/idealclasses2.mdl Wed Nov 17 10:56:10 1999 Class Diagram: Logical View / Main Page 1


TrMCTrack
TrTrack
pointer or associater
+1
+0..*
z position state, covariance matrix
+2..*
TrMeasurement
+1..*
TrState
TrStateQPt
TrStateL
OTHitOnTrack
ITHitOnTrack
Especially in case of outer tracker measurement depends on which track the hit is on
x,y.tx.ty,Q/pt
x,y,tx, ty
OTHit
ITHit
File: /afs/cern.ch/user/m/mneedham/public/rose/newclasses4.mdl Wed Nov 17 13:33:40 1999 Class Diagram: Logical View / Main Page 1

Smoothing



- Needed to get best track state at all measurement points
- Not just turning around and filtering in the other direction. To smooth measurement k need:
  - Transport from $k \rightarrow k+1$
  - Filtered state at k
  - Predicted state from filter step at $k+1$
  - Smoothed state at $k+1$
- Bundle together above as ‘Node’
- Assume no smoothing in pattern recognition phase
- Fit track-by-track
- Don’t need to keep track state at all measurements
- Therefore - copy and store needed information locally.

Common Design Features

- Tools vs Algorithms
  - At the moment treat extrapolators, update step of kalman filter as subalgorithms
  - But e.g. Extrapolator needed by more than one algorithm (track fit/ pattern recognition/ RICH)
  - Tools rather than algorithms ?
- Store hits in Layers (Gaudi ObjectVectors)
  - Every event create/delete containers
  - Better(?) to create once at intialization time, empty every event

Checks/Comparision with SICB

All checks made with 2 samples of inclusive b events generated with SICB v116

- Believe code works correctly
- Pulls at last measurement point/ χ^2 probability distribution look ok

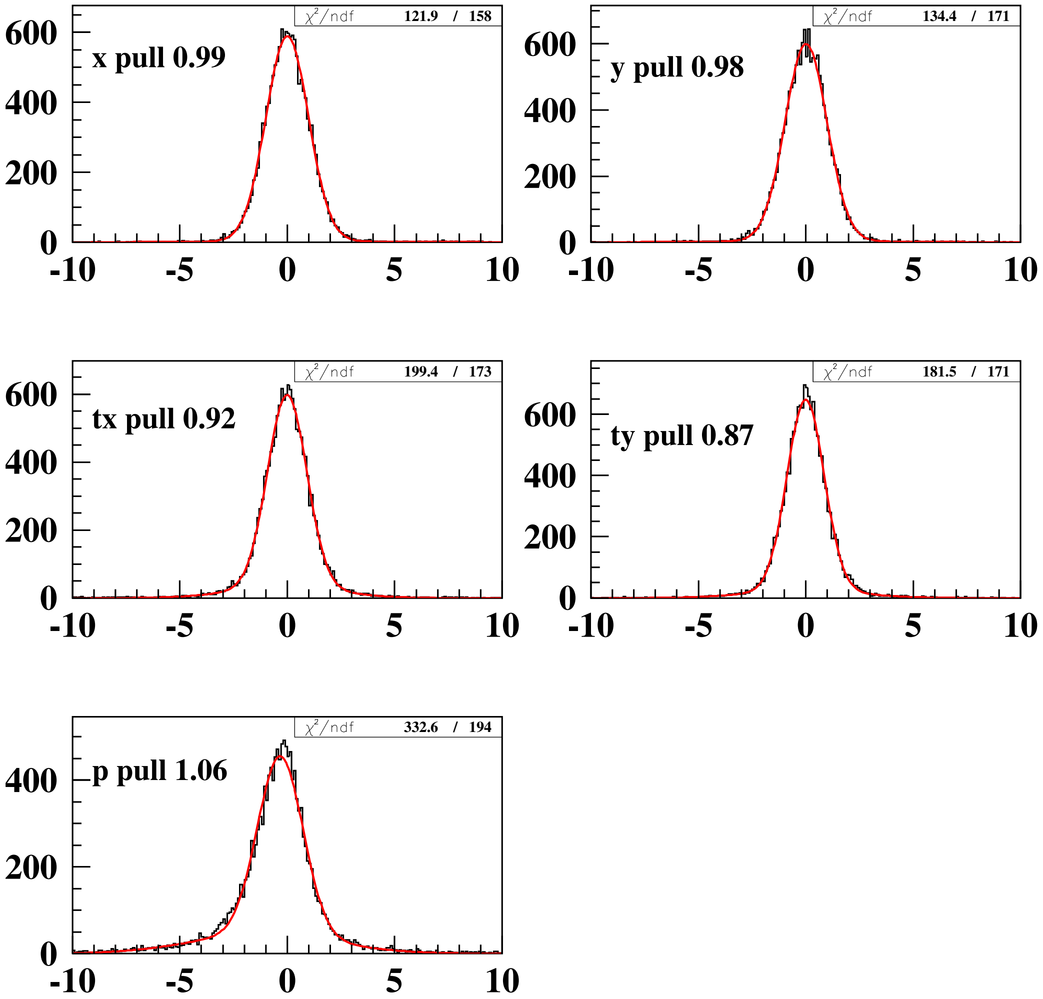
  Compare with SICB (see plots)

  Results:

  - Number of reconstructed tracks $\sim$ same
  - x,y pulls agree
  - Momentum pull/resolutions $\sim$ same , slightly more tail in Gaudi
  - Slopes $\Rightarrow$ pulls worse in SICB than Gaudi

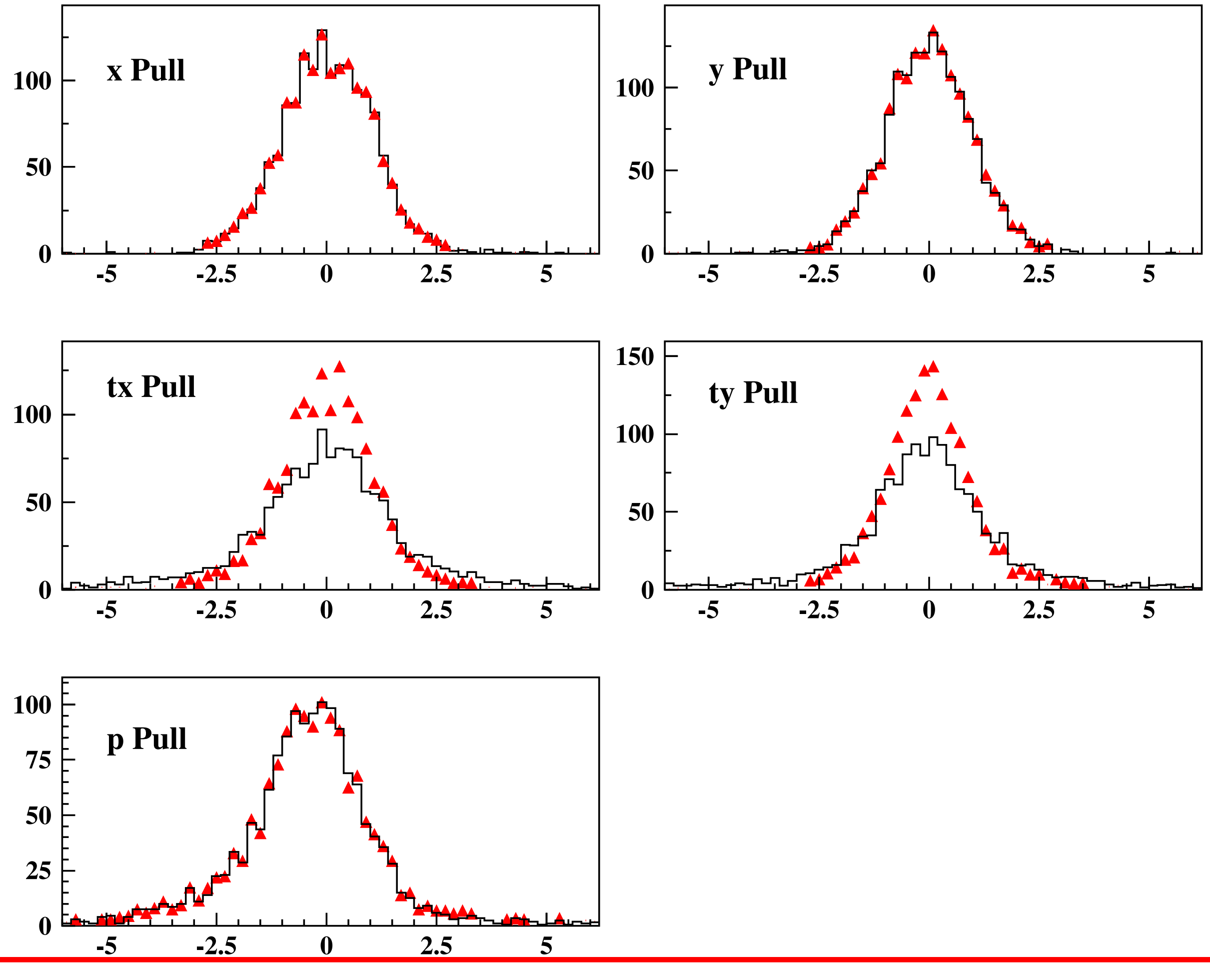
Pulls:

- 300 inclusive b events
- Minimum Momentum cut at 1 GeV, electrons excluded
- Double Gaussian fit to each distribution
- Quoted pull is the smaller σ of the two Gaussian



Comparison with SICB

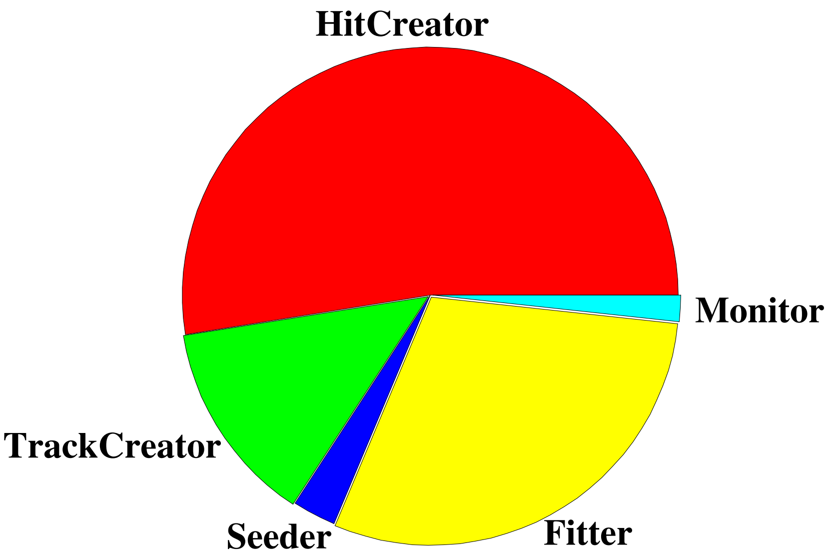
Compare fitting **SAME** 100 inclusive b events with SICB (solid line) and Gaudi (triangles)



Timing study

All numbers obtained on shiftlhcb

- CPU time to fit one track ~ 0.14 s. Using the compiler optimization options ~ 0.11 s
- Corresponds to ~ 7 s for an inclusive b event
- Where time go studied with gprof / simple ‘stopWatch’ class



Major offenders:

- Finding detection cell in database time consuming. All timing improvement when use optimizer comes here ! Speed up by adding more hierarchy to database.
- Runga-Kutta in Track fitter

Whats still missing

In order to have a reasonably complete ‘package’ still need:

- Vertex detector hits $\Rightarrow$ available but incomplete
- Digitizations $\Rightarrow$ on there way
- Smoother $\Rightarrow$ being debugged/tested
- Documentation $\Rightarrow$ started
- Associators $\Rightarrow$ not essential

Also need to:

- Move to v220 SICB , new release of Gaudi
- Code compiles and runs on Linux, rsplus. What about NT ?
- Minor tiding up of code/repackaging

Are we supposed to provide back compatibility for older versions of SICB ?

Summary and Conclusions

- Track fit in C++ works
- All distributions look reasonable
- Differences between SICB/ Gaudi still need to be understood
- Time per event ~ 7 s
  - Room for optimization.....
  - But... smoothing will slow things down
  - Moving to new magnet design will slow things down (need to iterate fit)
- Need to move to new version of SICB/Gaudi
- Could aim for first publically available version of the code $\sim$ early next year