Tracking Fitting

Matthew Needham, Christoph Langenbruch, Manuel Schiller

Disclaimer

I helped write the first C++ Kalman filter over 20 years ago. I know the Run 1 and 2 implementation very well. In particular I wrote first versions of the transport tools, implementing all the physics and debugging. My knowledge of Run 3 software and beyond that is more limited.

In this talk I will do my best to tell you what should be done. The actual technicalities are not something I can give much help with

Introduction

- The track fit is critical to judge key detector performance indicators
  - Mass resolution, momentum resolution, impact parameters, vertexing
  - Fitted tracks are input RICH pattern recognition
- Ultimate performance is determined by the measurements, magnetic field and material of the detector
  - No magic here, pattern recognition will evolve but fit with cheated pattern recognition is a solid guide to what is achievable

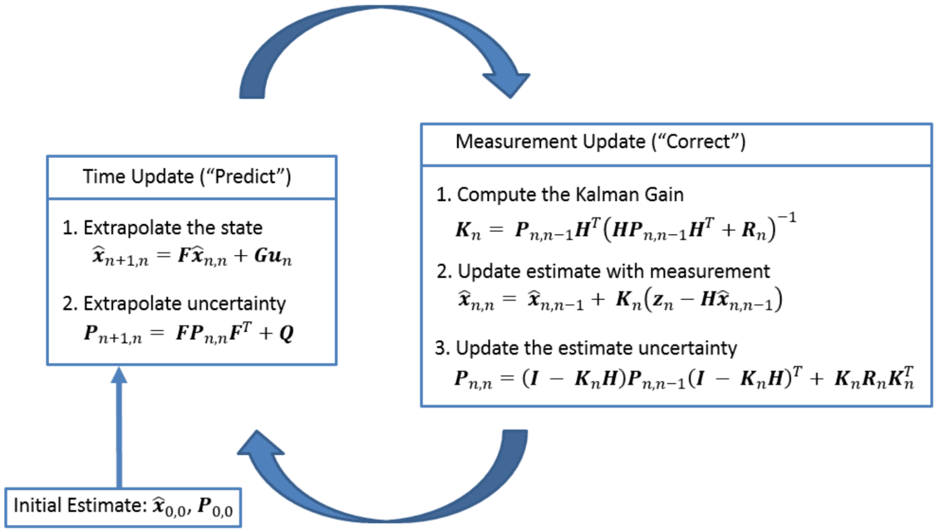
Introduction

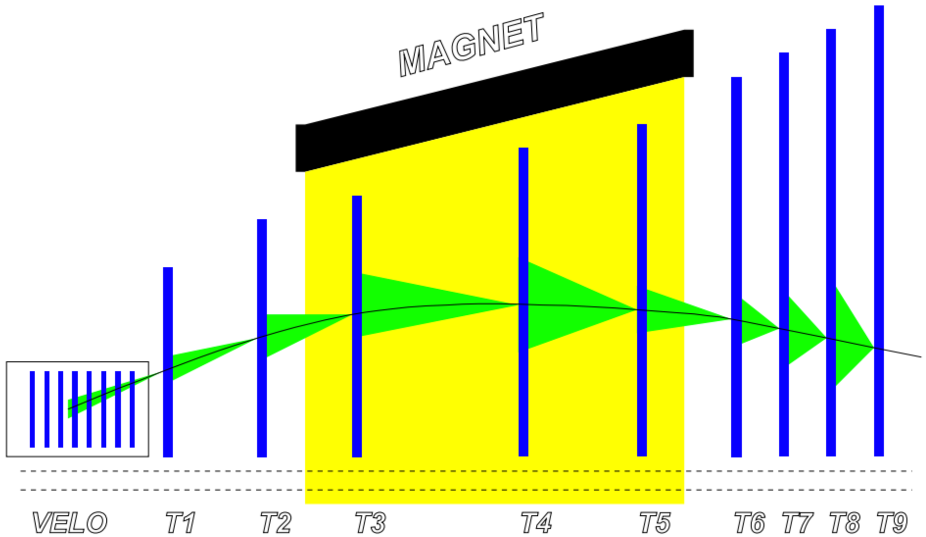
- For first iteration of LHCb from Letter of Intent/Technical proposal time in the 1990s till around Re-optimization TDR in 2000 there no pattern recognition
- Performance was judged from track fit and cheated pattern recognition (assigning hits to tracks using MCTruth) alone
- Can question the realism, but allowed to write PROCEEDINGS OF THE WORKSHOP ON STANDARD MODEL PHYSICS (AND MORE) AT THE LHC
- Yellow book predictions surprisingly good, e.g. based on them in Run 2 precision of 0.012 on sin 2β should have been achieved, compared to 0.015 we actually achieved

Towards full performance studies

Step 1	Parametric studies, fast simulation (RapidSim/emulation), standalone studies
Step 2	Track fit with cheated pattern recognition Allows to validate (= gain trust in) standalone studies Evaluate ultimate performance for e.g. mass resolution Global physics performance studies
Step 3	Realistic pattern recognition and track fit

The track fit: Kalman filter





Main Ingredients: Hits with errors and extrapolation

Extrapolation needs good knowledge of material and magnetic field

Detector Measurements

VELO: Move from pixels to pixels with timing in Upgrade II. For now we should not worry about fitting time

UT becomes UP: Pixels rather than strips

SciFi: No change from Run 3 to Run 5

MPix: Now pixels in the centre of the T-stations

Long-term we will have detailed digitizations for the pixel detectors but for now MCHits with smearing/some inefficiency ok for pattern/fit studies

Detector Measurements

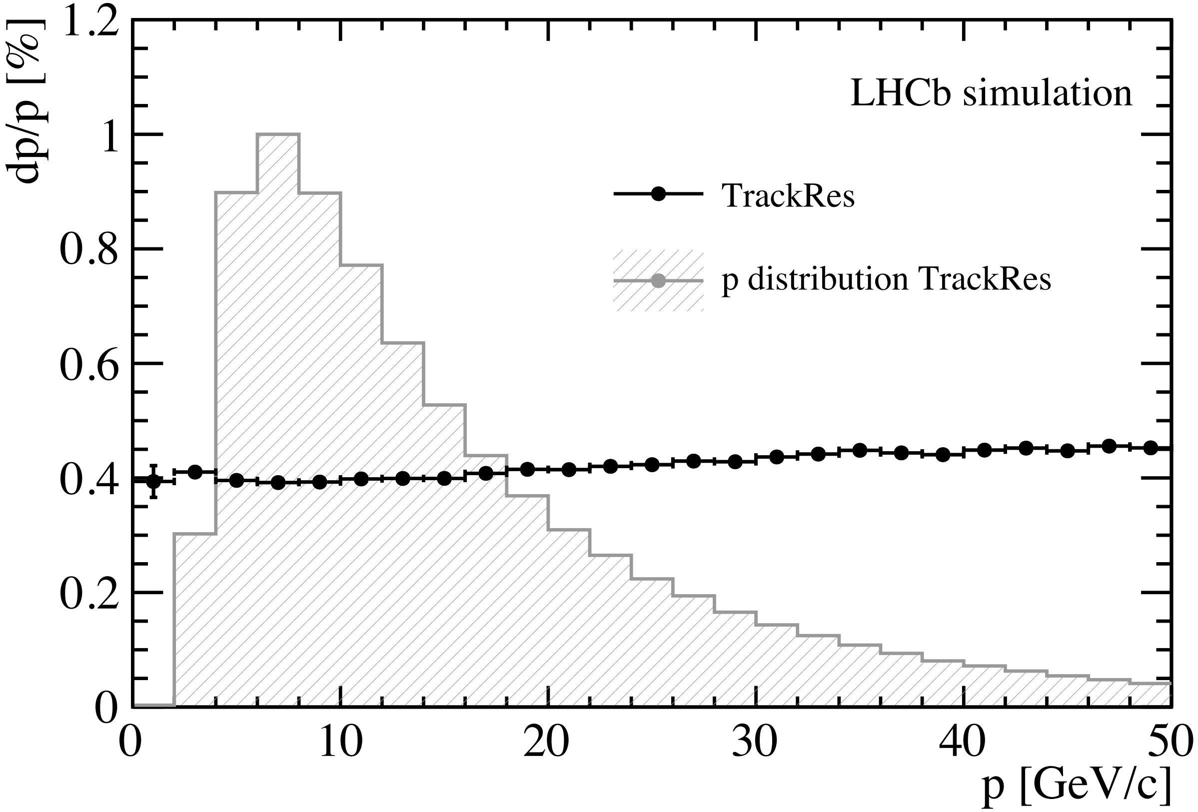
For the track fit rules of the game for measurements very clear

- Hit class
- Measurement provider
- Heavily templated, lots of glue code work through, but if it compiles it probably works

Christophe has got this working for MPix

- Can fit MPix tracks and Velo-Mpix matched tracks (Run 3 DetDesc/material)
- Need to also go through this for the UP (work but no showstoppers)

Detector Measurements



Momentum resolution for long tracks with MPix from Christoph

Using Run 3 geometry XML DetDesc

Impact at high momentum from pixels is visible

Detector Material

- Momentum resolution dominated by multiple scattering. Critical to get this correct
- Material description comes from the transport service either via detailed or simplified options
- For Upgrade 2 need this to work via DD4HEP geometry
  - We don't have XML geometry
- Technically now in place, being tested by Andrii + Ben
- Summarize in following slides taken from https://indico.cern.ch/event/1386929/contributions/5843802/attachments/2812188/4908546/Multiple_scattering_extrapolation_DSO.pdf

Multiple scattering extrapolation in Detector

https://gitlab.cern.ch/lhcb/Detector/-/merge_requests/471

A. Usachov and B. Couturier

Material correction in the extrapolations

To correct the particle states for multiple scattering in Rec, the *TrackMasterExtrapolator* in Rec invokes *MaterialLocatorBase::applyMaterialCorrection* for scattering or energy loss.

For this purpose there are state correction tools (*IStateCorrectionTool*) :

- *StateThinMSCorrectionTool*
- *StateThickMSCorrectionTool*
- *StateElectronEnergyCorrectionTool*
- *StateDetailedBetheBlochEnergyCorrectionTool*

Those tools uses material properties such as *radiationLength*, but the *StateDetailedBetheBlochEnergyCorrectionTool* also uses properties that are in DetDesc but not in the DD4hep/ROOT TGeo classes (C, X0, X1…)

Parameterize this

$$-\frac{dE}{dx} = 2\pi N_a r_e^2 m_e c^2 \rho \frac{Z}{A} \frac{z^2}{\beta^2} \left[\ln\left(\frac{2 m_e \gamma^2 v^2 W_{\max}}{I^2} \right) - 2\beta^2 - \delta - 2\frac{C}{Z} \right],$$

X

Implementation in DD4hep

The aforementioned properties are defined in the following paper:

M.J. Berger et al. Icru report 37. Journal of the International Commission on Radiation Units and Measurements,
// os19(2):, dec 1984. URL: https://doi.org/10.1093/jicru/os19.2.Report37,
doi:10.1093/jicru/os19.2.report37.

Also used by Geant4. It is possible to compute them using using a set of reference values for each element.

⇒ This was implemented in the Detector project with MaterialHelper.cpp. It is called once a geometry with all its materials has been loaded to decorate the existing materials with the properties I, X0, X1, C,a,m that can subsequently be used by the StateDetailedBetheBlochEnergyCorrectionTool.

This could be done with very limited modifications to the code:

https://gitlab.cern.ch/lhcb/Detector/-/commits/materials_in_dd4hep/?ref_type=heads

Geant4 references

- Builds simple materials from elements in GNistMaterialBuilder
- This map has been moved to DD4HEP materials
- *I effective* is used to compute all other parameters, and **density effect** in Bethe-Bloch formula (with the same formulas as in *DetDesc*)
- Some *I effective* values are different wrt *DetDesc* *(no visible effect)*

I effective ↓

```
AddMaterial("G4_H" ,  8.37480e-5,  1,  19.2, 1, kStateGas);
AddMaterial("G4_He",  1.66322e-4,  2,  41.8, 1, kStateGas);
AddMaterial("G4_Li",  0.534      ,  3,  40. );
AddMaterial("G4_Be",  1.848      ,  4,  63.7);
AddMaterial("G4_B" ,  2.37       ,  5,  76. );
AddMaterial("G4_C" ,  2.         ,  6,  81. );
AddMaterial("G4_N" ,  1.16520e-3,  7,  82. , 1, kStateGas);
AddMaterial("G4_O" ,  1.33151e-3,  8,  95. , 1, kStateGas);
AddMaterial("G4_F" ,  1.58029e-3,  9, 115. , 1, kStateGas);
AddMaterial("G4_Ne",  8.38505e-4, 10, 137. , 1, kStateGas);
AddMaterial("G4_Na",  0.971      , 11, 149. );
AddMaterial("G4_Mg",  1.74       , 12, 156. );
AddMaterial("G4_Al",  2.699      , 13, 166. );
```

```
// Differences between Geant and DetDesc:
// Element 1  (Hydrogen)   : G4: 19.2                DetDesc: 20       (average between gas and liquid)
// Element 5  (Boron)      : G4: 76.0                DetDesc: 92.1085
// Element 7  (Nitrogen)   : G4: 82.0   gas          DetDesc: 108.946
// Element 9  (Fluorine)   : G4: 115.0  gas          DetDesc: 126.572
// Element 11 (Sodium)     : G4: 149.0               DetDesc: 149.7   - small difference
// Element 15 (Phosphorus) : G4: 173.0               DetDesc: 187
// Element 16 (Sulfur)     : G4: 180.0               DetDesc: 199
// Element 17 (Chlorine)   : G4: 174.0  gas          DetDesc: 211
// Element 24 (Chromium)   : G4: 257.0               DetDesc: 295
// Element 53 (Iodine)     : G4: 491.0  condensed    DetDesc: 474      (gas value taken)
```

Materials in DetDesc vs DD4HEP

- Thanks to material dumper, we have DetDesc materials in *json* (+ some manual changes of material names)
- test_Materials_properties compares with those built in DD4HEP

```
MaterialCheck OK     : Kapton X0 0.2|0.2
MaterialCheck OK     : Kapton X1 2|2
MaterialCheck ERROR : Kapton a 0.468177|0.456707
MaterialCheck OK     : Kapton m 3|3
MaterialCheck ERROR : Kapton C 3.65161|3.58472
MaterialCheck ERROR : Kapton I 9.07195e-05|8.87902e-05
MaterialCheck MISSING Could not find material PEEK in DetDesc
MaterialCheck MISSING Could not find material Ecal paper in DetDesc
MaterialCheck MISSING Could not find material Pipe:AlBe in DetDesc
MaterialCheck MISSING Could not find material Pipe:AlCu in DetDesc

6: Missing    :152 missing materials from DetDesc JSON export
6: Error count: 81/1068
```

If material is missing it doesn't necessarily mean that it is used in the geometry

- For some DD4HEP materials there is no match in DetDesc and vice versa

Materials in DetDesc vs DD4HEP

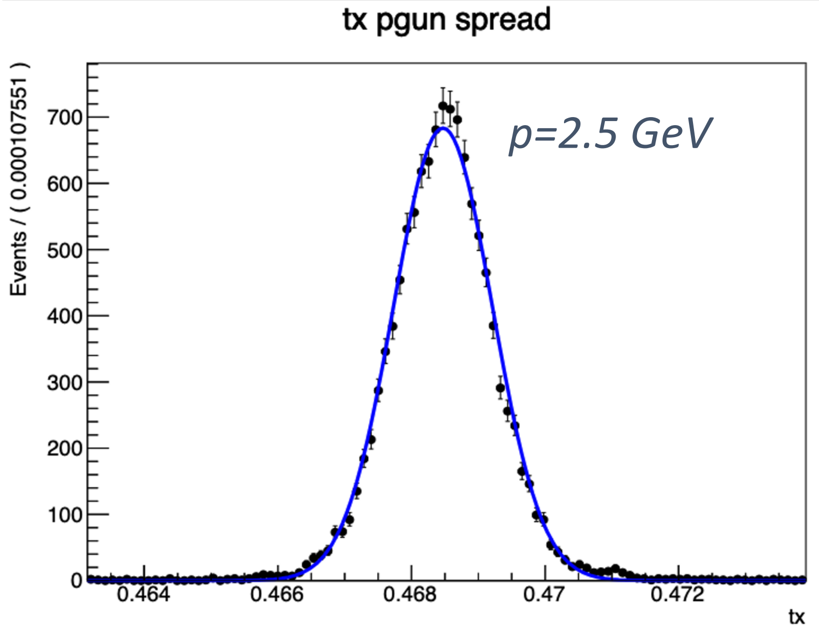
- Catched 2 issues in materials
- Setting units led to extremely large values of density

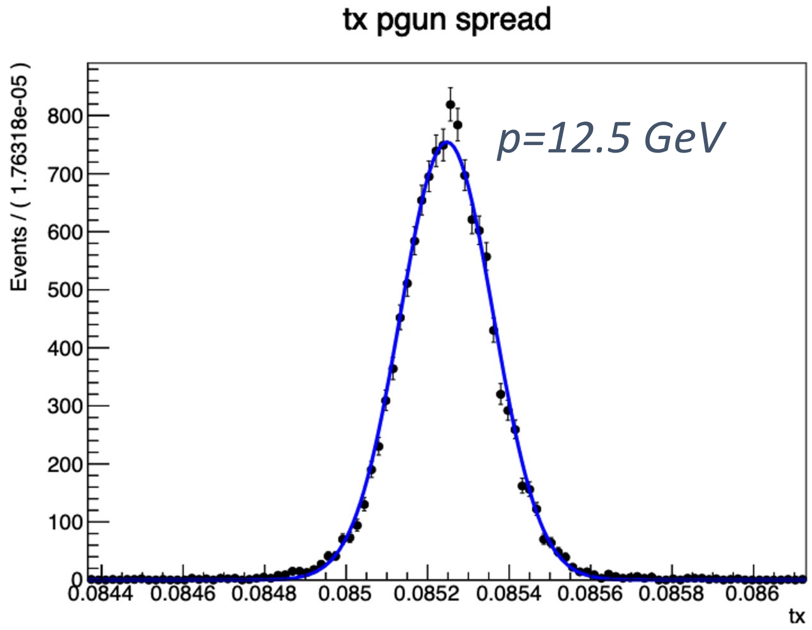
```
      <!-- ### Pipe Mount Brass ### -->
      <material name     = "Pipe:Brass">
-       <D type="density" value="8.53*g/cm3"/>
+       <D type="density" value="8.53" unit="g/cm3"/>
        <fraction ref = "Cu" n = "0.70"/>
        <fraction ref = "Zn"   n = "0.30"/>
      </material>

      <!-- ### Pipe Mount AW2219 ### -->
      <material name     = "Pipe:AW2219">
-       <D type="density" value="2.84*g/cm3"/>
+       <D type="density" value="2.84" unit="g/cm3"/>
        <fraction ref = "Al" n = "0.9168"/>
        <fraction ref = "Cu" n = "0.0680"/>
        <fraction ref = "Fe" n = "0.0030"/>
```

Test with particle gun

- Setup to run particle gun in a loop for different *tx, ty, qop* ***(both DetDesc and DD4HEP)***
  https://gitlab.cern.ch/bcouturi/gaussino-pgun/-/merge_requests/1

- There is low momentum background from secondary particles
  → removed by momentum cut and fitting
- *There are some rare corner cases when distribution is not gaussian*





Tracking performance with *expected-2024* sample

- With *expected-2024* conditions the **drop in efficiency is smaller - 3.5%** (compared to 10% before)
- Some important changes in the geometry?
- Resolution check shows **not zero pulls in *x, y* and momentum**

INFO Long/x pull	: mean = -1.232 +/- 0.300, RMS = 2.546 +/- 0.152
INFO Long/y pull	: mean = -2.955 +/- 0.086, RMS = 1.747 +/- 0.090
INFO Long/tx pull	: mean = 0.000 +/- 0.017, RMS = 1.086 +/- 0.016
INFO Long/ty pull	: mean = 0.002 +/- 0.016, RMS = 1.059 +/- 0.015
INFO Long/p pull	**: mean = -0.467 +/- 0.027, RMS = 1.703 +/- 0.021**
INFO Long/probChi2	: mean = 0.327 +/- 0.005, RMS = 0.327 +/- 0.003
INFO Long/x resolution / mm	: RMS = 127.329 +/- 19.138 micron

Long/y resolution / mm: RMS = 132.978 +/- 12.207 micron
Long/dp/p: mean = -0.0018 +/- 0.0002, RMS = 0.0096 +/- 0.0002

- Also many differences in other counters for not fitted tracks as well
  (MR by Miroslav https://gitlab.cern.ch/lhcb/Moore/-/merge_requests/2907)

Ideal track creator

- Assign hits to tracks based on MCHits (cheated pattern recognition)
- Allows to fit tracks, get resolutions for RICH, mass resolutions
- Some work needed here with Pixel measurements ready but straightforward

Summary

- Work ongoing on pixel measurement providers
  - MP works, UP to be done
  - Can run fit with MP (smeared MCHits) , using Run 3 geometry (XML DetDes
- A lot of progress to having track fit working with DD4HEP
  - But not quite there yet
- Merged into U2 branches by Tim Evans
- More studies and help/effort needed

What needs to be done?

- Situation is very similar to 2008 – 2010 when we had to verify the geometry of the first LHCb
- Then what happened (driven by subdetector experts)
  - Eyeballing – checking that the numbers for all materials used made sense
  - Radiation scans – do they agree with expectations from analytic calculations/standalone
  - Weighting the detector, does the weight in the simulation of detector elements agree with physical expectation
- After a lot of work and iteration we were sure of the numbers in XmlDDDB , sure detector in simulation matched reality at % level
- Similar program needs to be launched now
- From other side debugging from track reconstruction

Backup

Parameterised scatters: DD4HEP vs DetDesc

- Scattering corrections can be parametrised for transitions, e.g. from VP hit to next VP hit and so on
- Helps to locate the differences in materials

energy loss in MeV

```
"VPHit-ClosestToBeam"     : [2.153, 0.816, 0.799, 0.893],     # 28.38%  smaller
"VPHit-VPHit"             : [1.272, 0.659, 0.541, 0.508],     # 10.09%  smaller
"VPHitFoil-VPHitFoil"     : [2.072, 0.640, 0.525, 0.862],     # 5.25%   larger
"UTHit-VPHit"             : [35.099, 0.442, 0.315, 13.201],   # 27.24%  larger
"UTHit-UTHit"             : [1.758, 0.637, 0.505, 1.009],     # 37.28%  larger
"FTHit-VPHit"             : [76.301, 0.387, 0.263, 31.451],   # 10.37%  smaller
"FTHit-UTHit"             : [20.187, 0.688, 0.589, 12.136],   # 208.73% larger
"FTHit-FTHit"             : [6.549, 0.608, 0.492, 5.978],     # 472.61% larger
"EndVelo-VPHit"           : [24.033, 0.418, 0.368, 6.264],    # 271.09% larger
"BegRich1-EndVelo"        : [4.000, 0.650, 0.611, 1.328],     # 47.30%  smaller
"EndRich1-BegRich1"       : [7.165, 0.572, 0.495, 4.095],     # 19.84%  larger
"UTHit-EndRich1"          : [2.988, 0.709, 0.669, 2.151],     # 30.44%  larger
"EndUT-UTHit"             : [1.759, 0.591, 0.483, 1.181],     # 19.66%  larger
"EndUTBeamPipe-EndRich1": [10.780, 0.621, 0.537, 6.697],      # 10.64%  smaller
"BegT-EndUT"              : [13.168, 0.623, 0.558, 7.102],    # 218.76% larger
"BegT-EndUTBeamPipe"      : [8.362, 0.616, 0.548, 5.205],     # 74.97%  smaller
"FTHit-BegT"              : [3.487, 0.896, 0.871, 3.114],     # 426.01% larger
"BegRich2-FTHit"          : [3.862, 0.344, 0.285, 3.510],     # 593.68% larger
"EndRich2-BegRich2"       : [121.141, 0.725, 0.644, 56.657],  # 418.13% larger
"BegT-EndRich1"           : [20.290, 0.419, 0.285, 12.129],   # 53.83%  smaller
"BegT-UTHit"              : [15.538, 0.560, 0.455, 8.442],    # 157.61% larger
"EndUTBeamPipe-VPHit"     : [57.976, 0.497, 0.378, 21.599],   # 24.53%  larger
"FTHit-EndUT"             : [18.253, 0.740, 0.677, 10.834],   # 271.79% larger
"FTHit-EndUTBeamPipe"     : [13.308, 0.741, 0.676, 8.866]     # 58.94%  smaller
```

***comparison of energy loss wrt DetDesc**