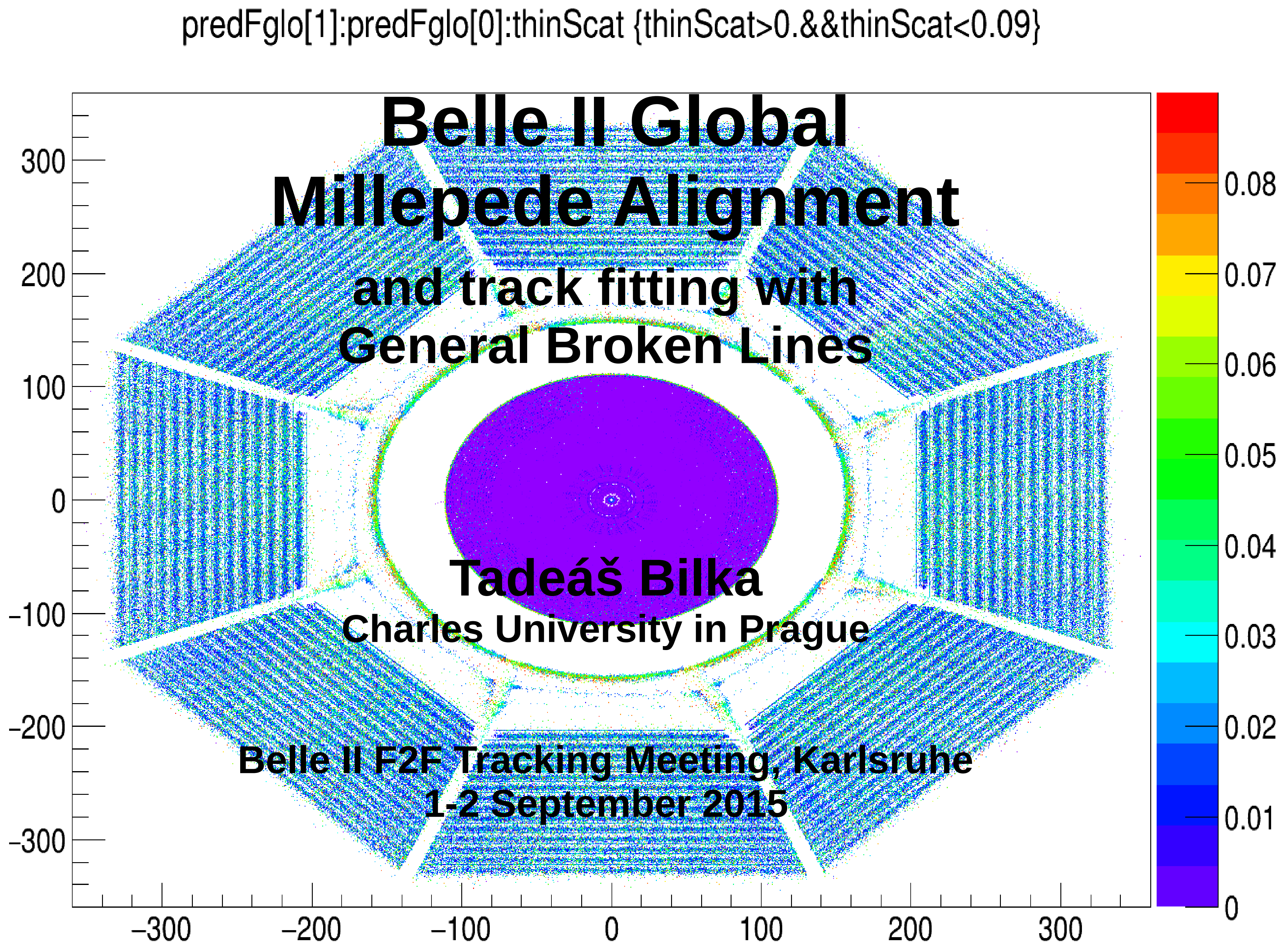
Belle II Global Millepede Alignment

and track fitting with General Broken Lines

Tadeáš Bilka
Charles University in Prague

Belle II F2F Tracking Meeting, Karlsruhe
1-2 September 2015



Overview

- The Calibration Framework
  - Everything runs inside it...
  - But sure it is not too advanced and has some issues...
- Alignment package (Millepede II & GBL integration)
  - That on the other side is quite a powerfull tool already...
  - You want some proof? **VXD alignment test**
  - You want more? **Welcome BKLM!**
- Future plans
- Conclusions

The Calibration Framework

- Tried to implement **granularity=run**, dependencies, iteration, splitting of data collection and calibration/monitoring
- A calibration module can be used multiple times (e.g. each run)
- Main problem in short:
  - Basf2 can mix events from different runs in multi-core processing
  - The first event of a new run is always merged before anything in output path runs
  - Calibration MUST live in output path (But collection is prefered in event path)
- We want to allow: collection in event path, processing of multiple events, arbitrary splitting of input data, splitting of collection and calibration/monitoring (->merging), multi-core processing, ...

Only possible solution for now

- Ilustrative caveat-showing example
  - You process 50 runs in experiment 6 in 1 job, have a module which should calibrate from data of each run and uses 10 histograms for that

  - Your module will be placed 50 times in the path with names like module_6_1_6_1, module_6_2_6_2, …
  - The histograms fill have prefix with module name
  - There will be 500 histograms in DataStore, most of the time only waiting
  - All calibrations will be done at termination (solves mixing of events)

- Performance? Online calibration?
- Question? Comments? Suggestions?

CalibrationFramework.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os
import sys
from basf2 import *
from ROOT import Belle2
import modularCalibration as cal

set_random_seed(100)
set_log_level(LogLevel.INFO)
use_local_database('db/database.txt', '', LogLevel.WARNING)

# Makes list of 2-tuples [(exp, runs[0]), (exp, runs[1]), ...]


def exp_runs(exp, runs):
    if not isinstance(runs, list):
        runs = [runs]
    expruns = []
    for run in runs:
        expruns.append((exp, run))
    return expruns

# Set path and format of DST files and runs, where magnet is off
# If you don't, you should see plenty of failed extrapolations
cal.setDstStoragePattern('alignment/examples/DST_Exp{experiment}_Run{run}.root')
cal.setMagnetOffRuns(exp_runs(1, range(1, 101)))

# createCalibration(name='tadeasTest1.root')#, importCalibration='database.root')

cal.selectRange(1, 1, 2, 100)

cal.selectSample(
    exp_runs(1, range(1, 11)) +
    exp_runs(2, range(1, 11))
)

cal.loadGeometry()
```

CalibrationFramework.py

```
# Generate misalignment in first pass
if not os.path.isfile('calibration_cache.txt'):
    misalignment = cal.VXDMisalignment('VXDMisalignment')
    # misalign everything
    misalignment.genSensorU('0.0.0', 0.01)
    misalignment.genSensorV('0.0.0', 0.01)
    misalignment.genSensorW('0.0.0', 0.01)
    misalignment.genSensorAlpha('0.0.0', 0.001)
    misalignment.genSensorBeta('0.0.0', 0.001)
    misalignment.genSensorGamma('0.0.0', 0.001)

    # Reset misalignment to zero for sensor we fix
    # in PedeSteering (see below)
    misalignment.setSensorAll('4.0.1', 0.0)
    misalignment.setSensorAll('5.0.1', 0.0)
    misalignment.setSensorAll('6.0.0', 0.0)

    cal.generateMisalignment(misalignment)

cal.useMisalignment('VXDMisalignment')
cal.refitGBL()

cal.createPedeSteering(name='PedeSteeringFine',
                       commands=['method diagonalization 3 0.1',
                                 'chiscut 30. 6.',
                                 'outlierdownweighting 3',
                                 'dwfractioncut 0.1'],
                       fix=['4.0.1.0', '5.0.1.0', '6.0.0.0'])  # fix slanted SVD's + whole 6th layer

cal.addMillepedeCalibration(name='Millepede',
                            granularity='data',
                            steering='PedeSteeringFine')

cal.calibrate()
print statistics
```

Dependencies+Granularity

```
cal.createPedeSteering(fix=['4.0.1.0', '5.0.1.0', '6.0.0.0'])  # fix slanted SVD's + whole 6th layer

cal.createPedeSteering(name='PedeSteeringFine',
                       commands=['method diagonalization 3 0.1',
                                 'chiscut 30. 6.',
                                 'outlierdownweighting 3',
                                 'dwfractioncut 0.1'],
                       fix=['4.0.1.0', '5.0.1.0', '6.0.0.0'])  # fix slanted SVD's + whole 6th layer


cal.addMillepedeCalibration(name='MP2_perrun',
                            granularity='1.1.1.1,1.2.1.2,1.3.1.3,1.4.1.4')
cal.addMillepedeCalibration(name='MP2_perdata',
                            granularity='data',
                            dependency='MP2_perrun')
cal.addMillepedeCalibration(name='MP2_perdata_fine',
                            granularity='data',
                            dependency='MP2_perdata',
                            steering='PedeSteeringFine')

cal.calibrate()
print statistics
```

VXD Alignment Test

- Full example in alignment/examples
  - generate_samples.sh … GenDST.py
  - calibrationFramework.py
- Sample
  - 100k particle gun events + 300k cosmic rays
    - Only about 100k cosmic muons pass selection:
      - 4 < # hits < 24
      - Fit success & p-value > 0.001 (ideal geometry)
- Fixed 6th SVD layer and all slanted SVD sensors
  - Not misaligned. Used as reference.
  - Slanted - low statistics for cosmics & selection criteria

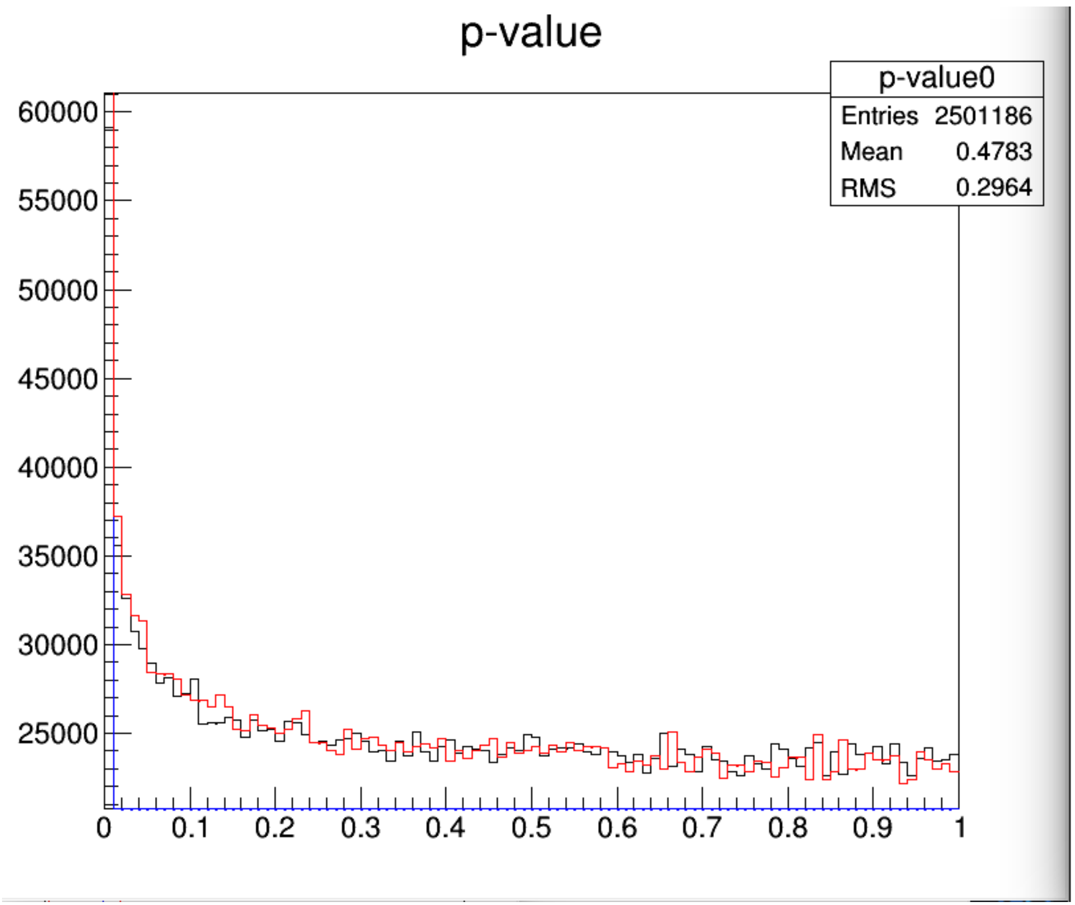
VXD Alignment Test

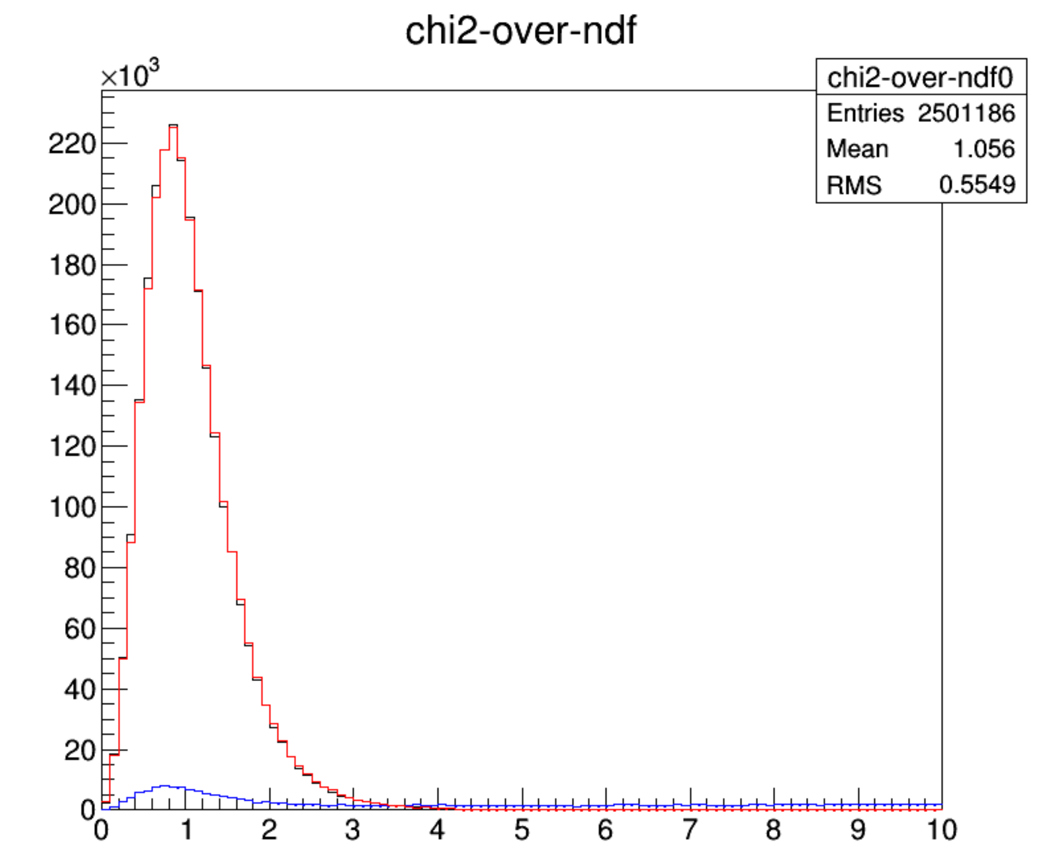
- Generated misalignment: random per each (non-fixed) sensor and param
  - u, v, w … 100 micrometers
  - alpha, beta, gamma … 1 mrad

- On following plots:
  - Black: ideal geometry
  - Blue: misaligned reconstruction geometry
  - Red: reconstruction geometry after Millepede alignment (2nd iteration with complete refit)

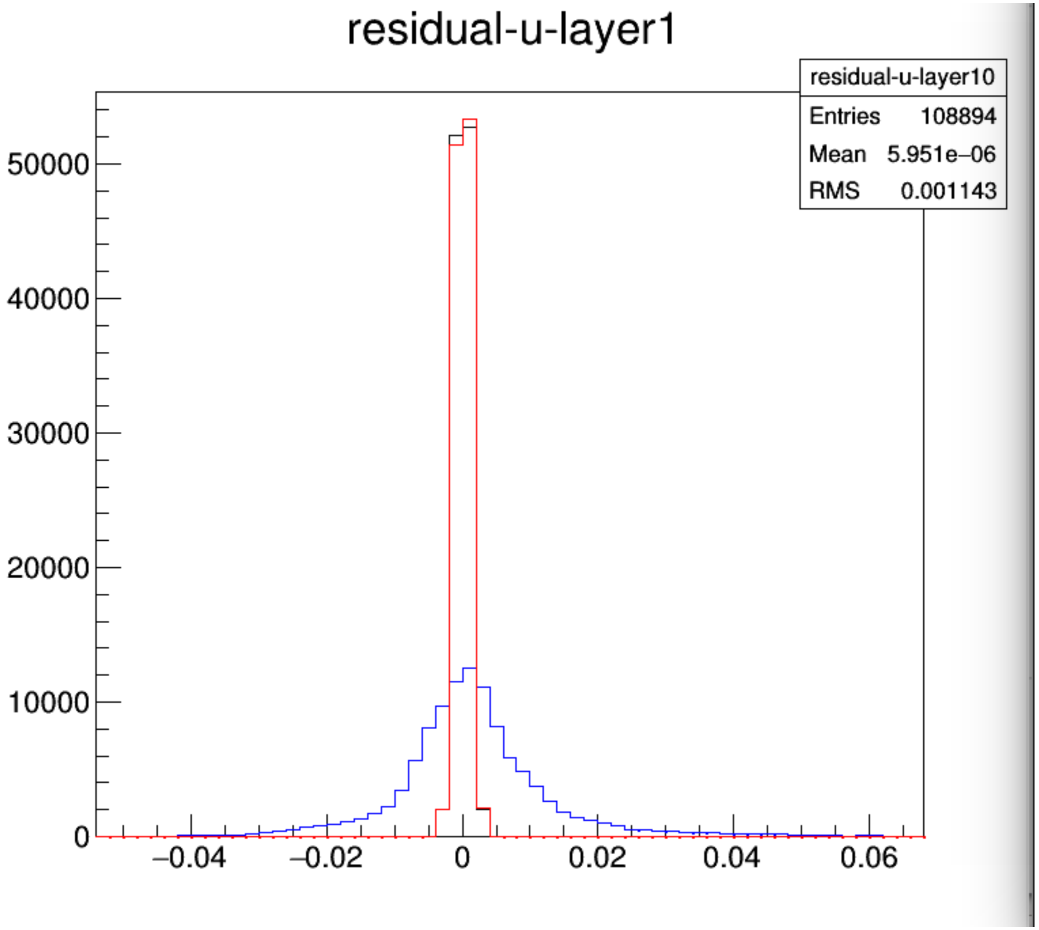
    misalignment – alignment = residual misalignment

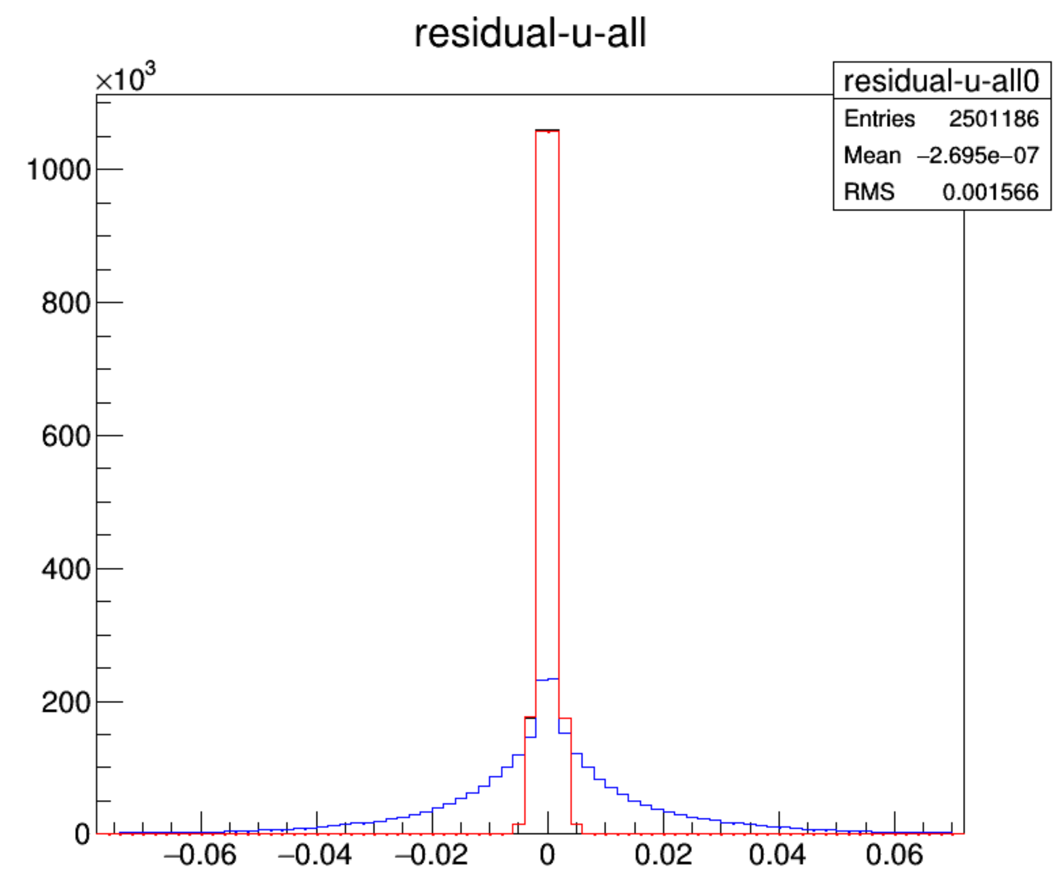
Chi squared


p-value
p-value0
Entries 2501186
Mean 0.4783
RMS 0.2964
60000
55000
50000
45000
40000
35000
30000
25000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

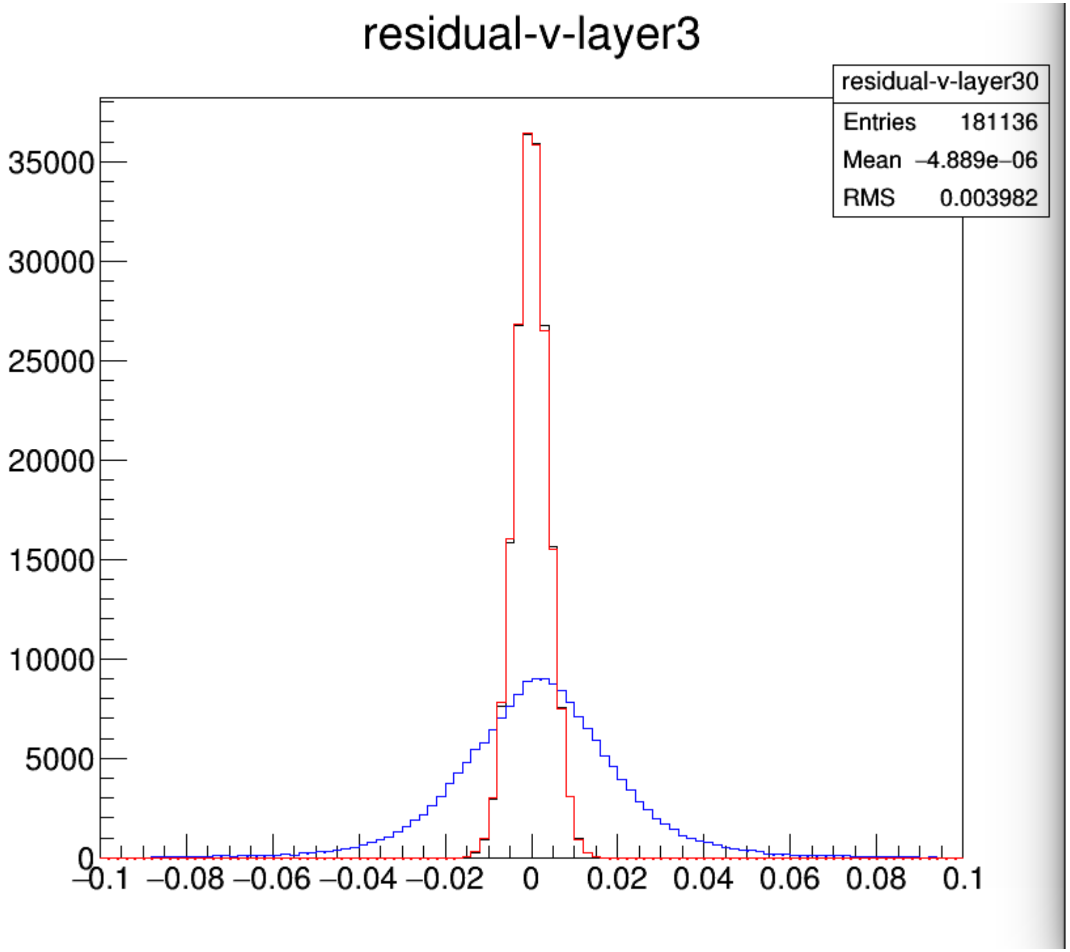

chi2-over-ndf
×10^3
chi2-over-ndf0
Entries 2501186
Mean 1.056
RMS 0.5549
220
200
180
160
140
120
100
80
60
40
20
0
0
1
2
3
4
5
6
7
8
9
10

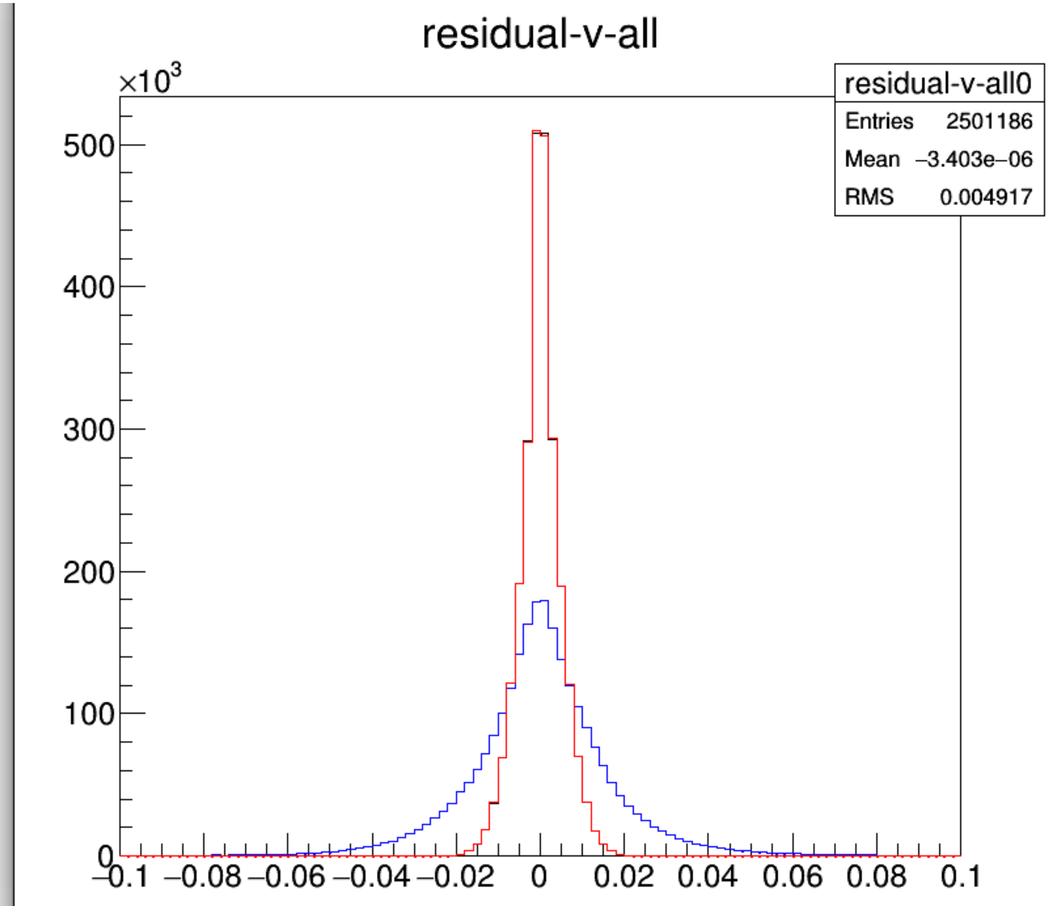
U (R-Phi) Residuals



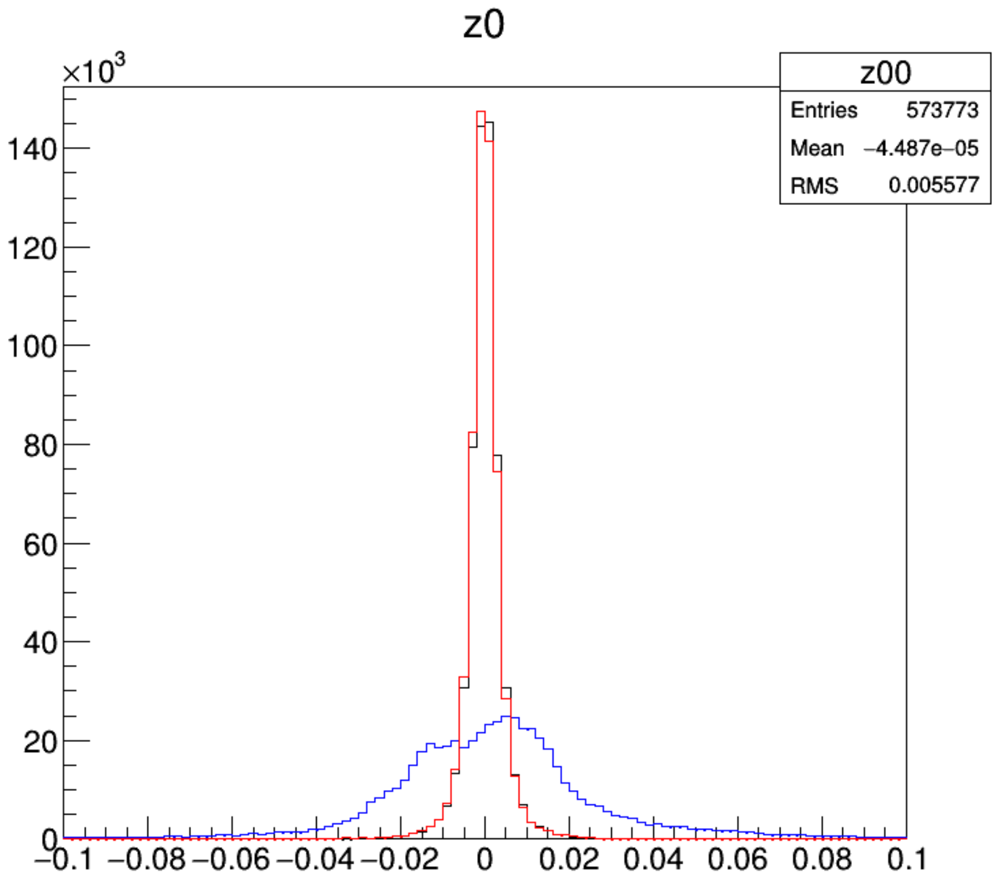


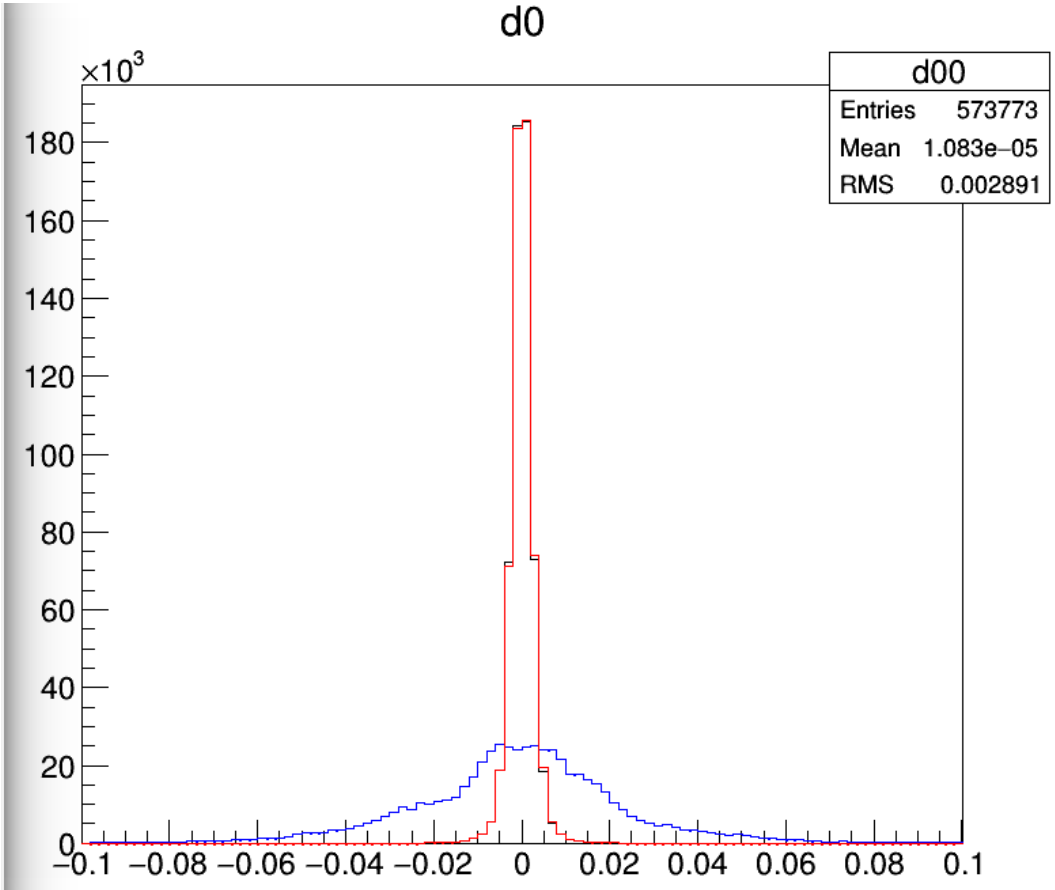
V (Z) Residuals



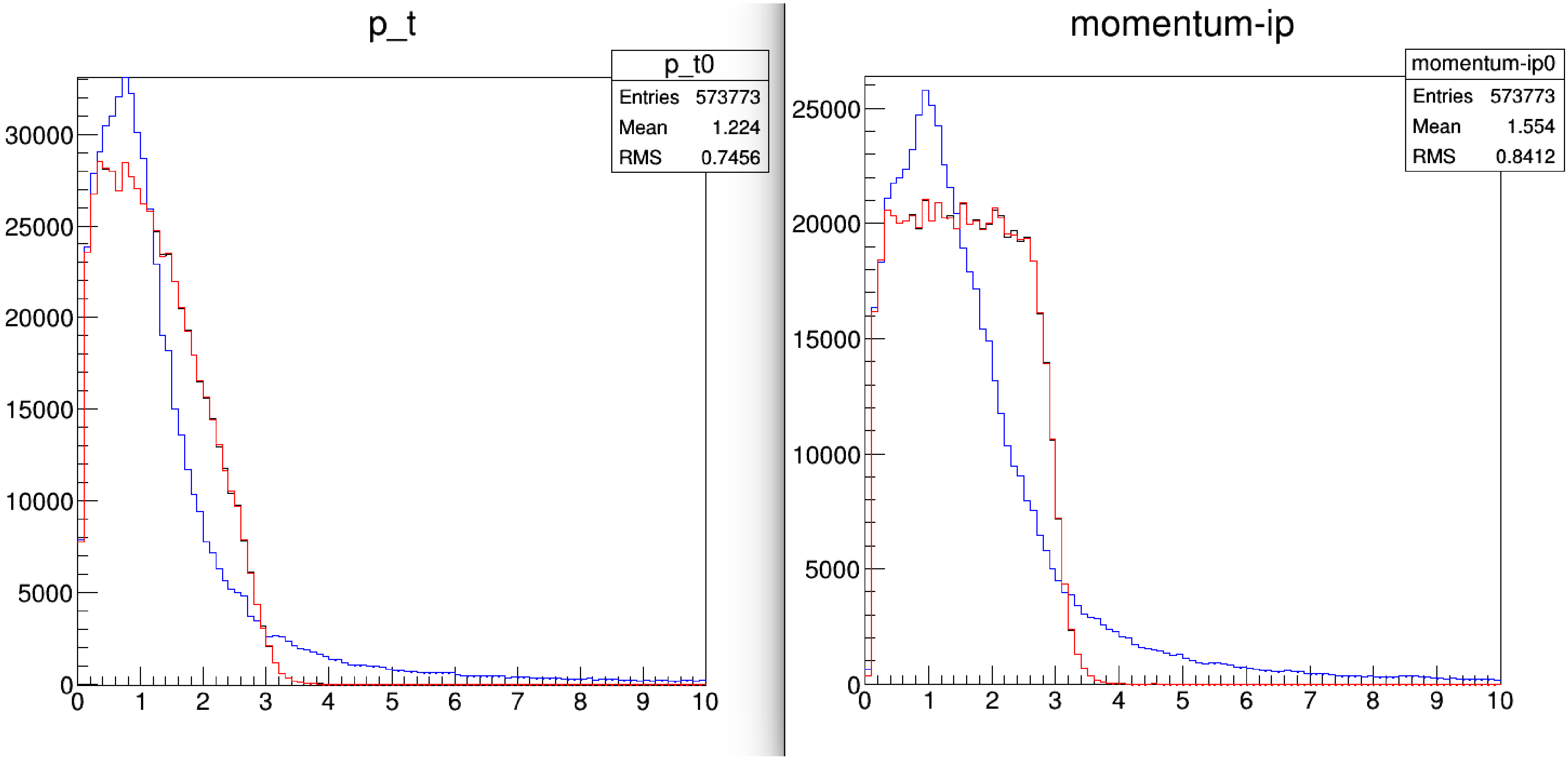


Vertex estimation in VXD (default particle gun)

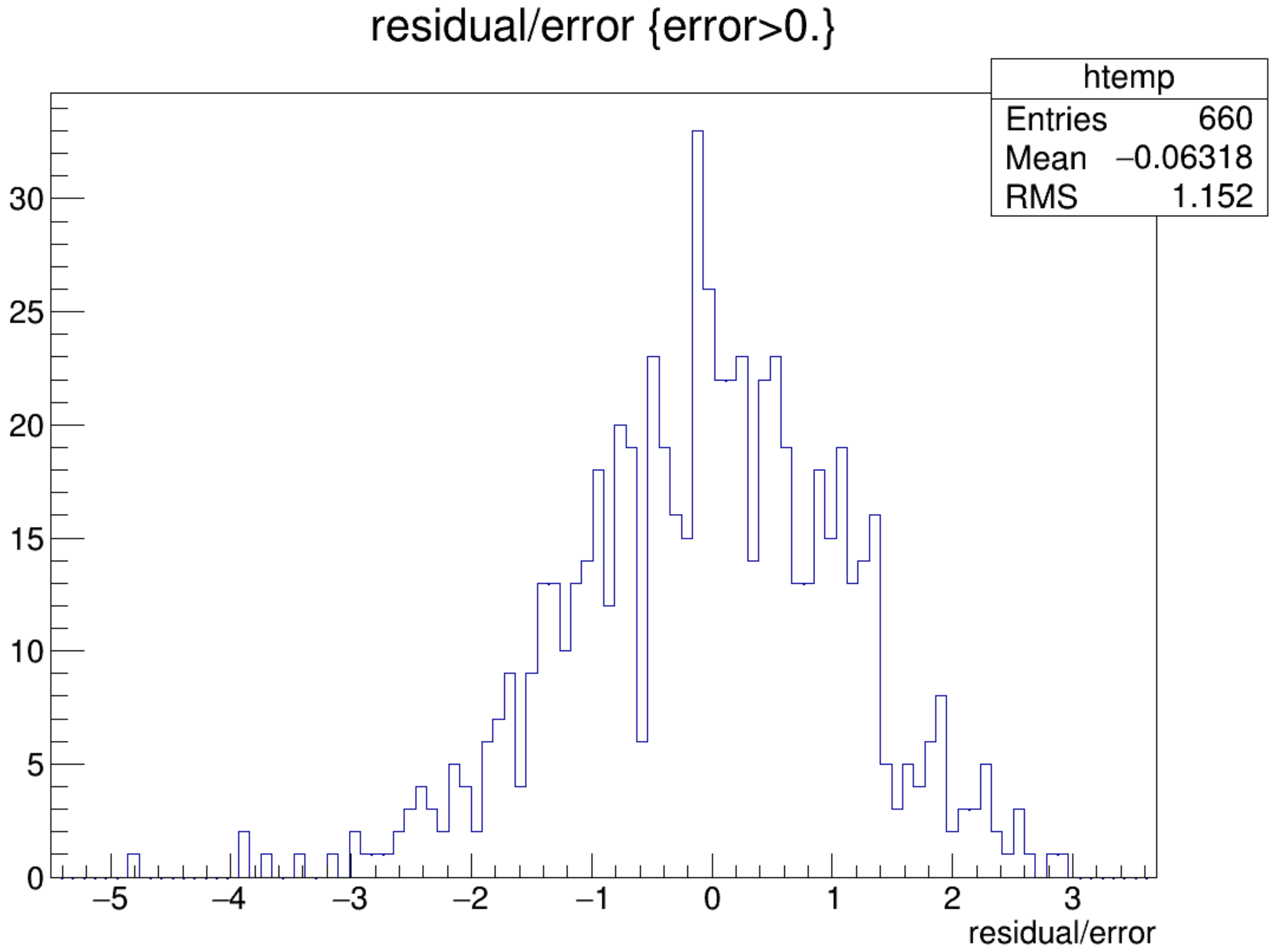




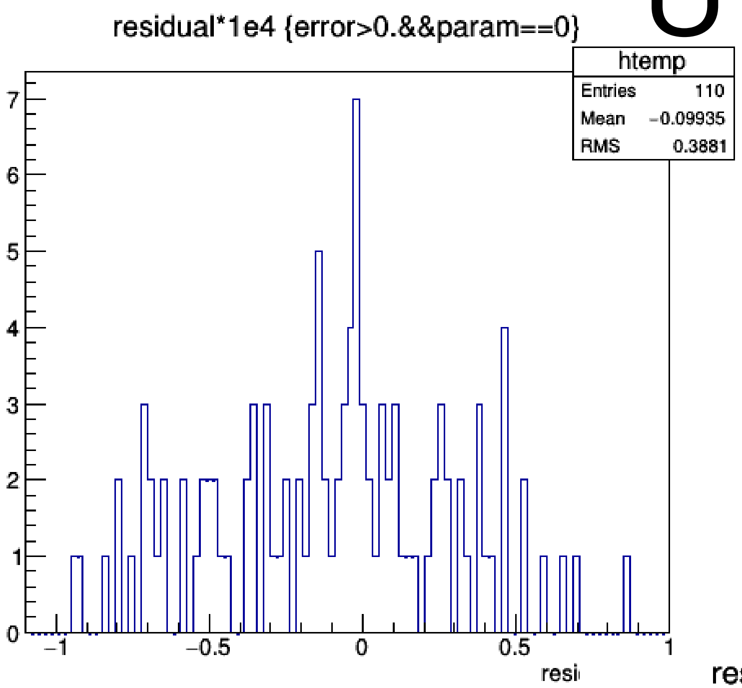
Momentum estimation in VXD

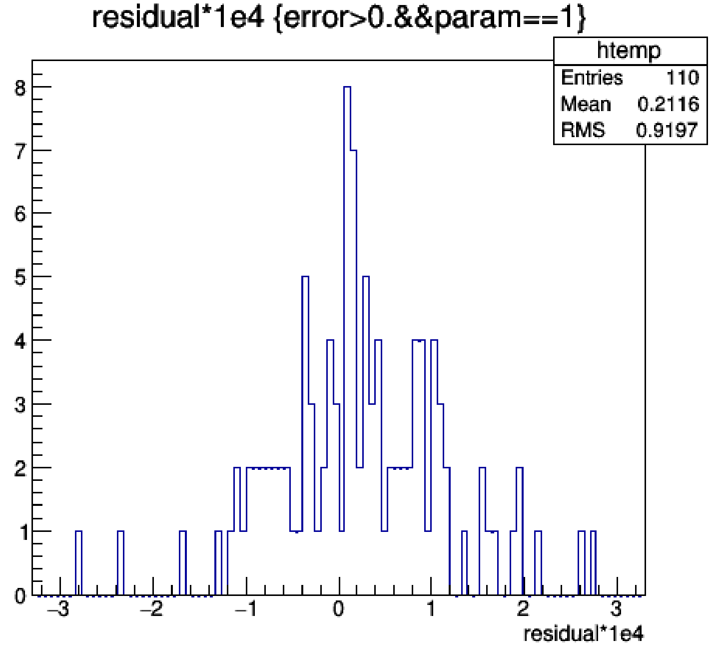


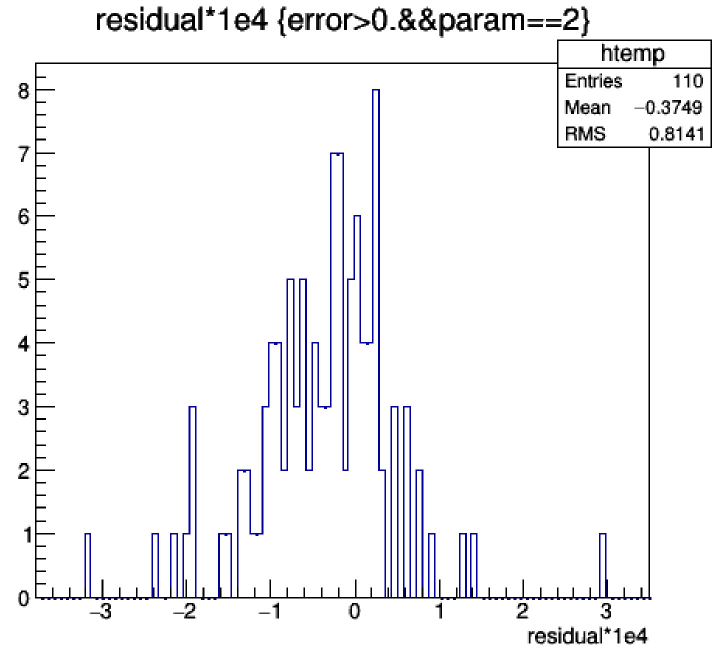
Residual Misalignment Pulls of all params



Residual Misalignment U | V | W [um]

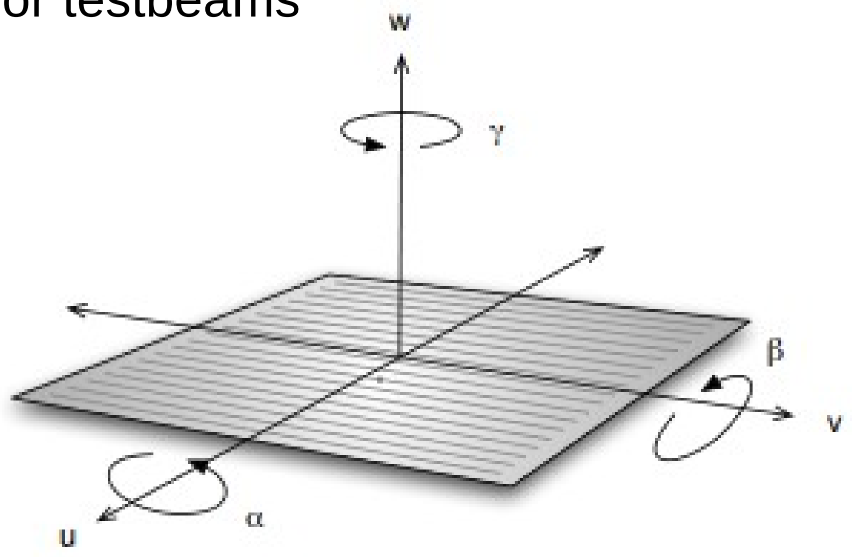






Parametrization of Angles

- In vxd/data/alignment.xml and elsewhere
  - Euler angles of Geant4: zx'z'' convention
- Proposal: zy'x'' convention (Tait–Bryan angles)
  - Better for infinitesimal angles (no pi/2 skips)
  - Simpler to understand and imagine (for planes at least)
  - More intuitive to reach desired rotation „by hand“
    - Geant4 angles are a real nightmare for testbeams



Parametrization of Angles

- The rotation matrix from Tait–Bryan angles

```
TGeoRotation A;
Double_t m[9];
m[0] =   cos(beta) * cos(gama);
m[1] =   cos(alfa) * sin(gama) + sin(alfa) * sin(beta) * cos(gama);
m[2] =   sin(alfa) * sin(gama) - cos(alfa) * sin(beta) * cos(gama);
m[3] = - cos(beta) * sin(gama);
m[4] =   cos(alfa) * cos(gama) - sin(alfa) * sin(beta) * sin(gama);
m[5] =   sin(alfa) * cos(gama) + cos(alfa) * sin(beta) * cos(gama);
m[6] =   sin(beta);
m[7] = - sin(alfa) * cos(beta);
m[8] =   cos(alfa) * cos(beta);
A.SetMatrix(m);
```

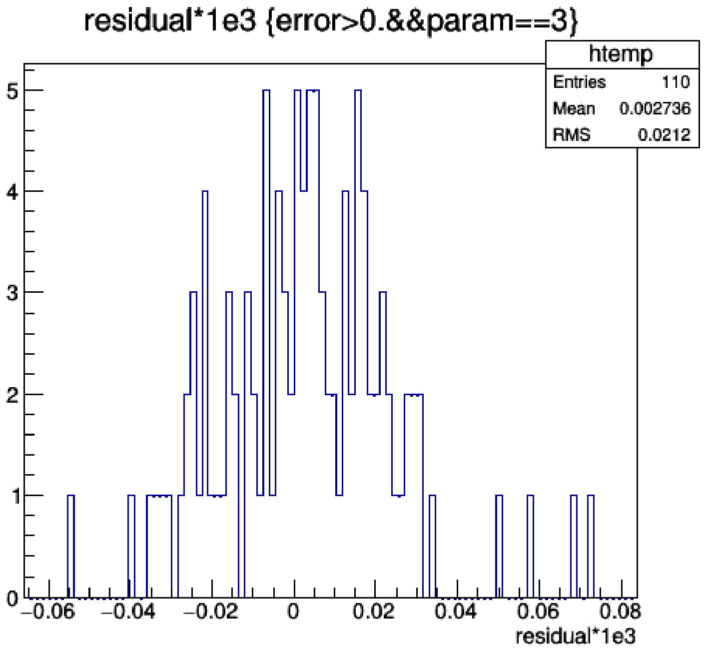
$$-\pi/2 < \beta < \pi/2 \qquad \beta = \arcsin r_{13}$$

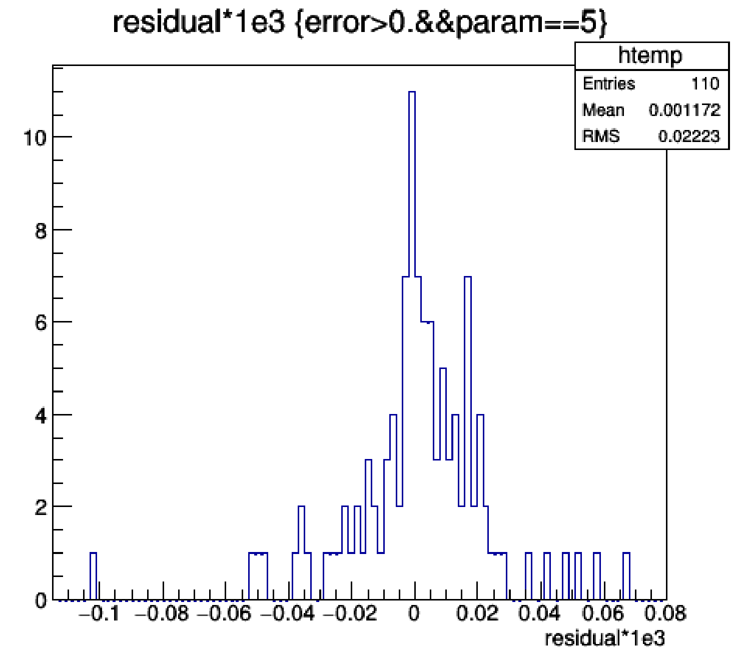
$$\alpha = \begin{cases} \tan^{-1}\frac{-r_{32}}{r_{33}} & (r_{33}\cos\beta > 0) \\ \tan^{-1}\frac{-r_{32}}{r_{33}} + \pi & (r_{33}\cos\beta < 0) \end{cases} \qquad \gamma = \begin{cases} \tan^{-1}\frac{-r_{21}}{r_{33}} & (r_{11}\cos\beta > 0) \\ \tan^{-1}\frac{-r_{21}}{r_{33}} + \pi & (r_{11}\cos\beta < 0) \end{cases}$$

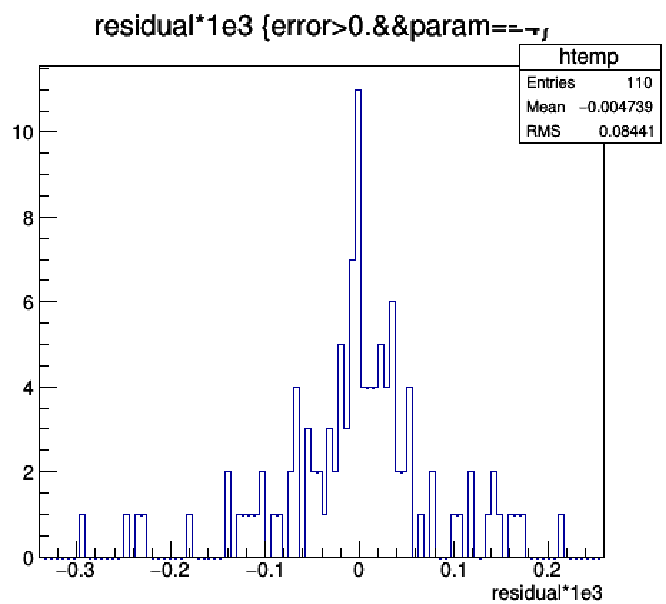
- TGeoRotation can give you Euler angles

$$\phi = \operatorname{arctan2}(A_{31}, A_{32})$$
$$\theta = \arccos(A_{33})$$
$$\psi = -\operatorname{arctan2}(A_{13}, A_{23})$$

Residual Misalignment
Alpha | Beta | Gamma [mrad]







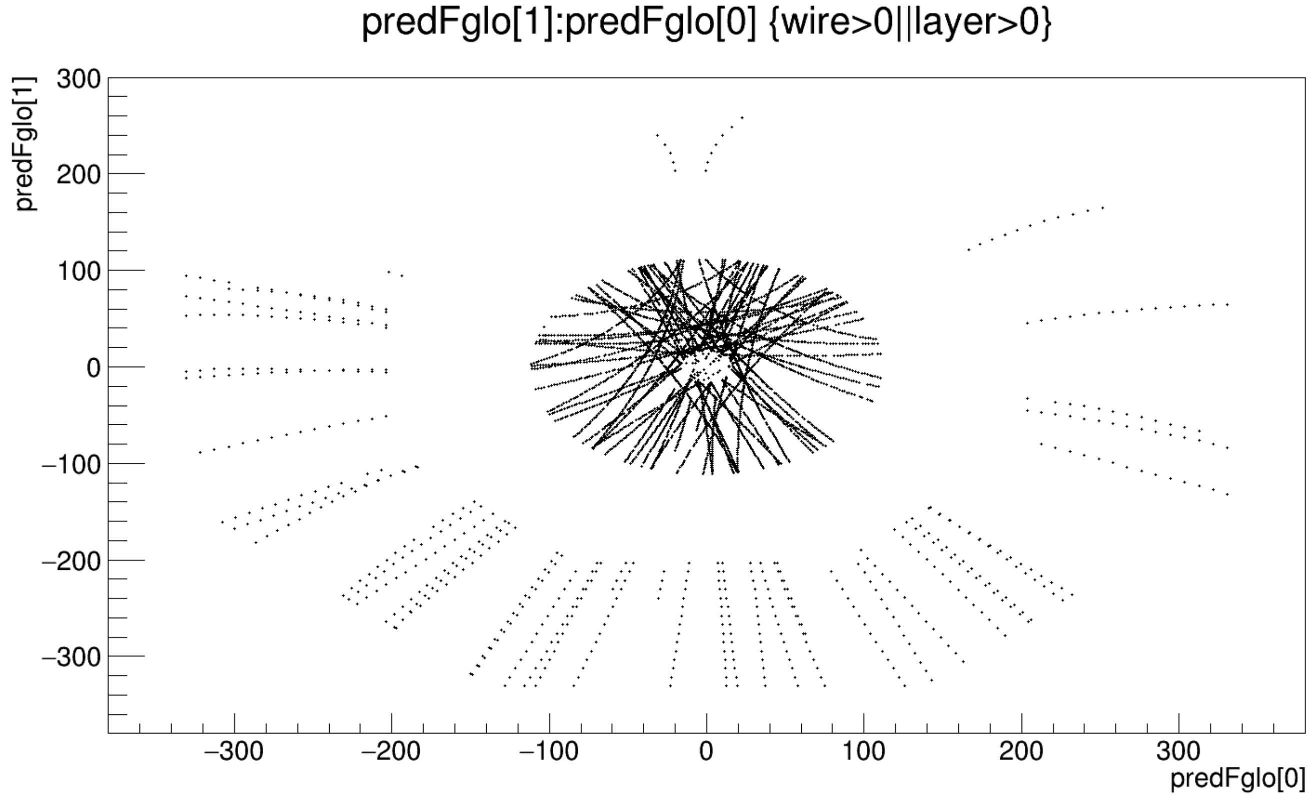
Welcome BKLM!

- The real main message:
  - We have new woman power in alignment!

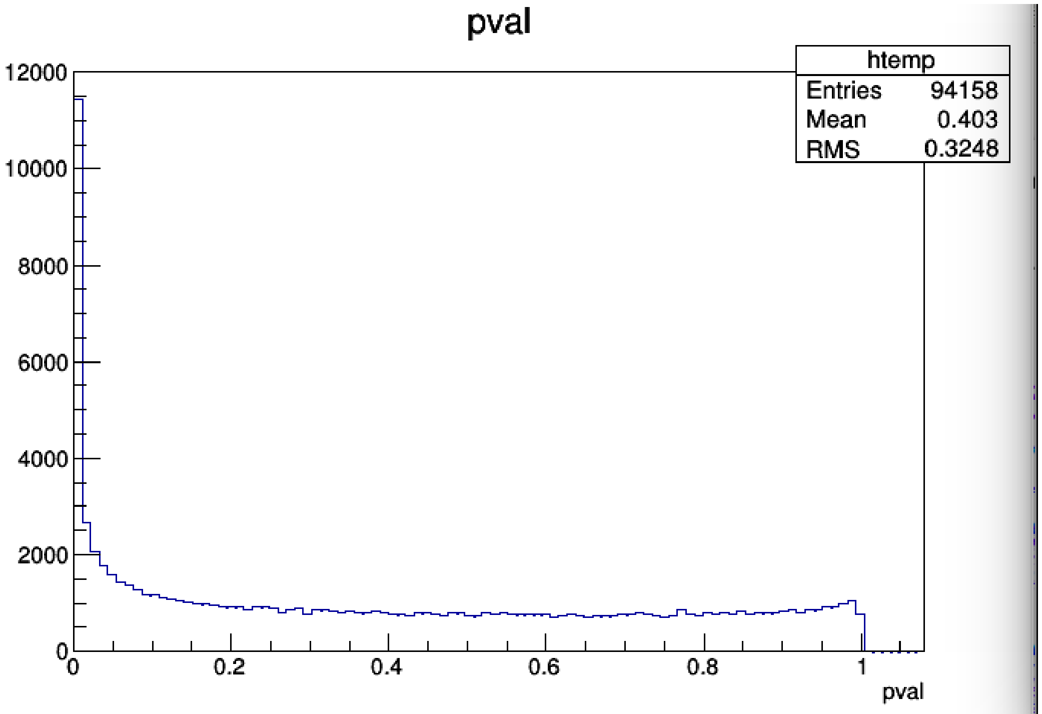
    Guan Yinghui
- Interface to GENFIT(by GBL) (BKLMRecoHit) and updated Muid module
- BKLM in Millepede alignment

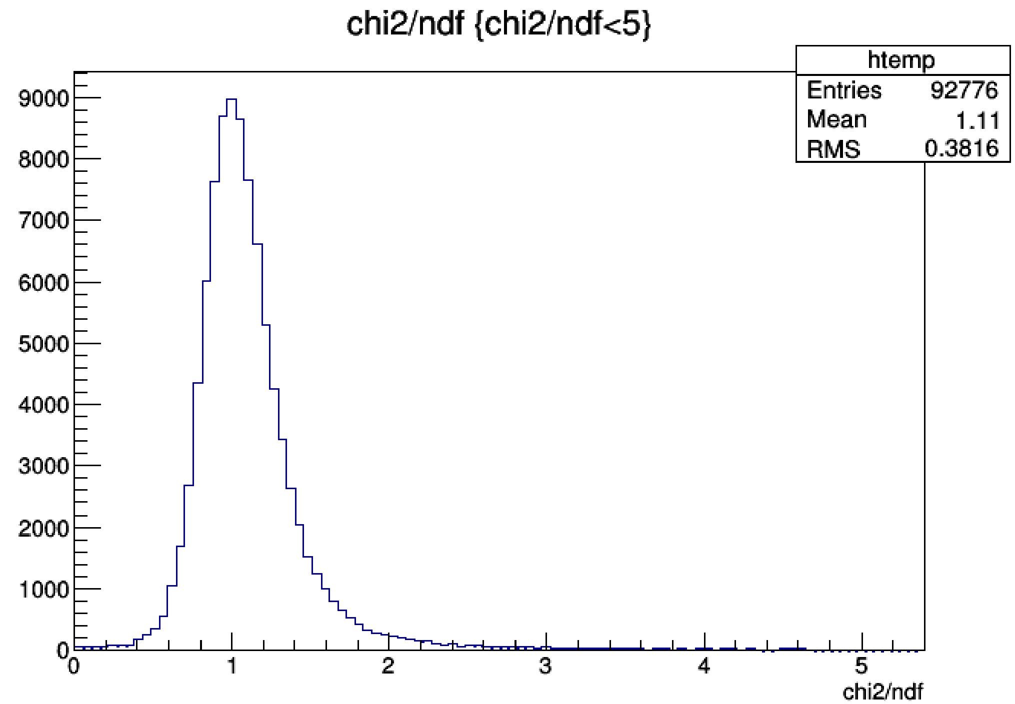
Alignment and add_mc_reconstruction

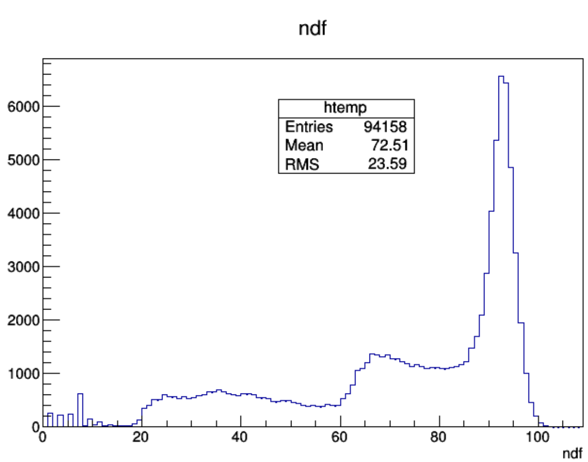
- Full standard (MC) reconstruction has to be finished before alignment of KLM can start
- Without MC, wrong weights in CDC (solution: DAF seed for GBL + take closest measurement)
- Extrapolations to outer detectors for cosmics (B field on/off)
  - Cosmics + add(_mc)_reconstruction only extrapolates one arm

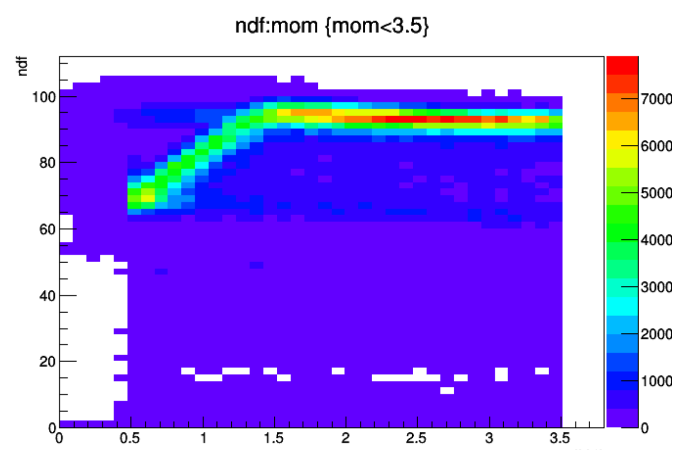


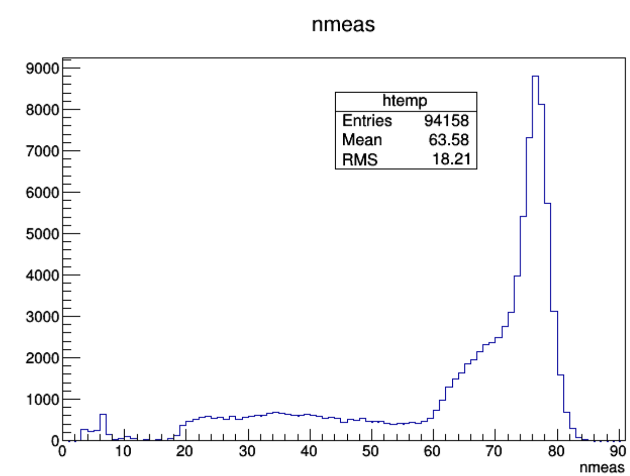
Full MC Reconstruction with BKLM





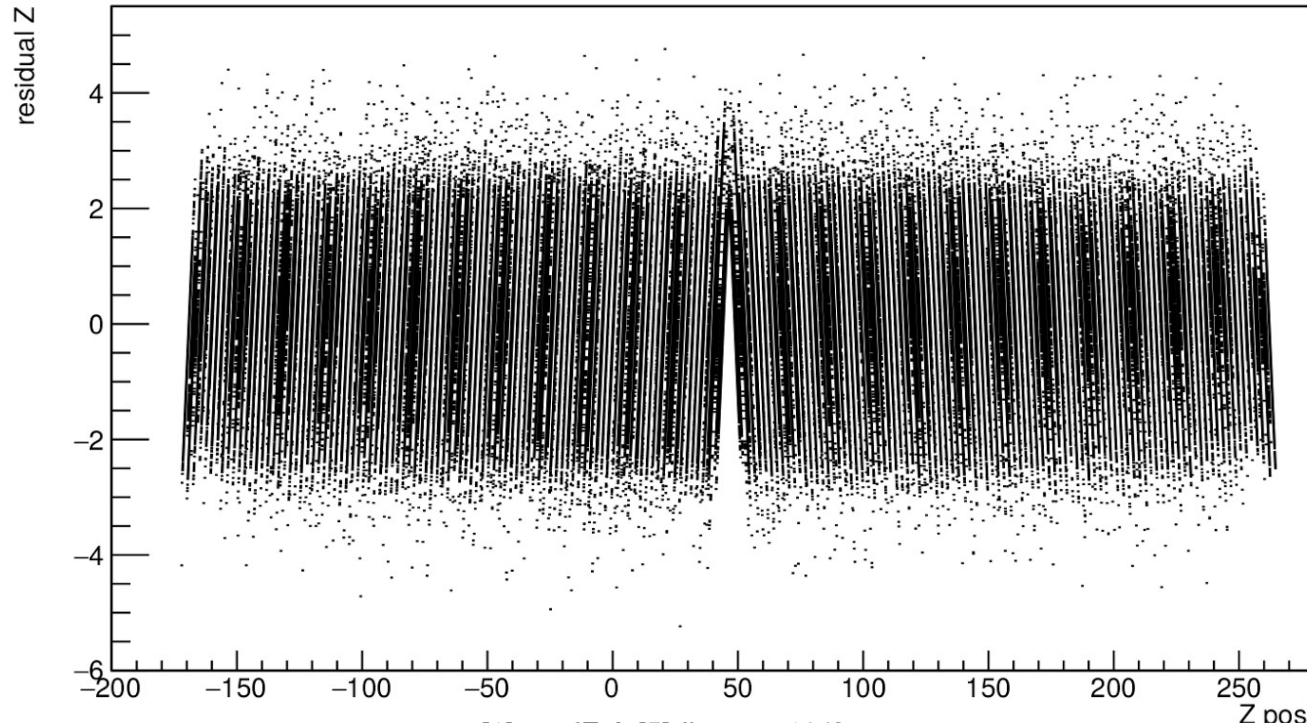


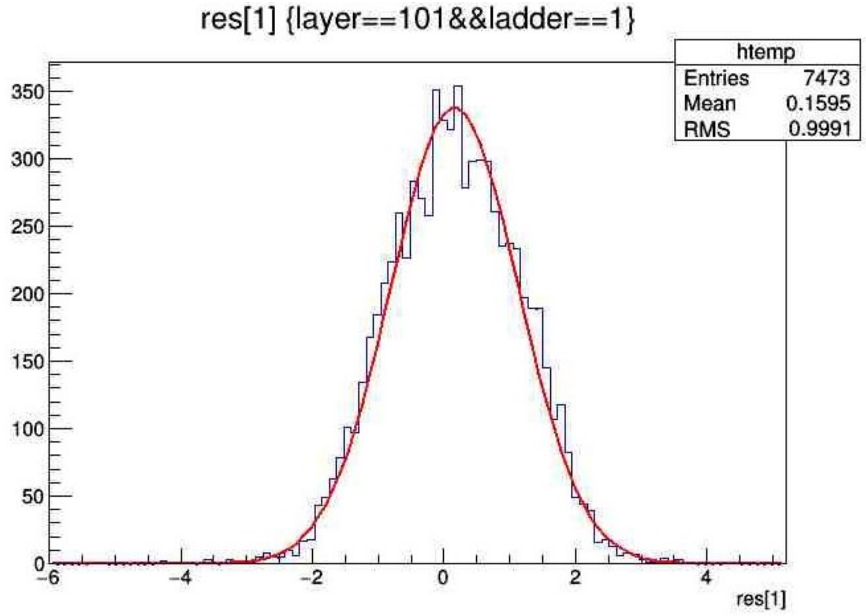


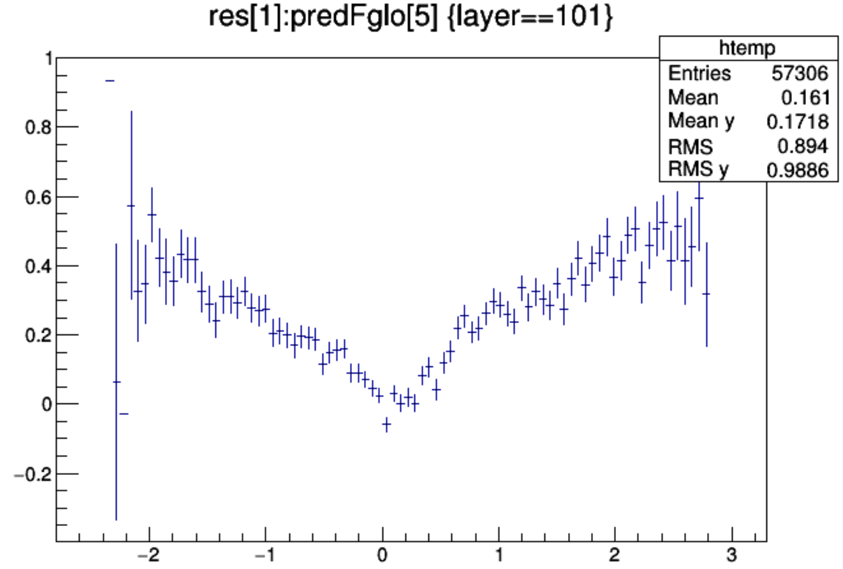


Reconstruction in BKLM

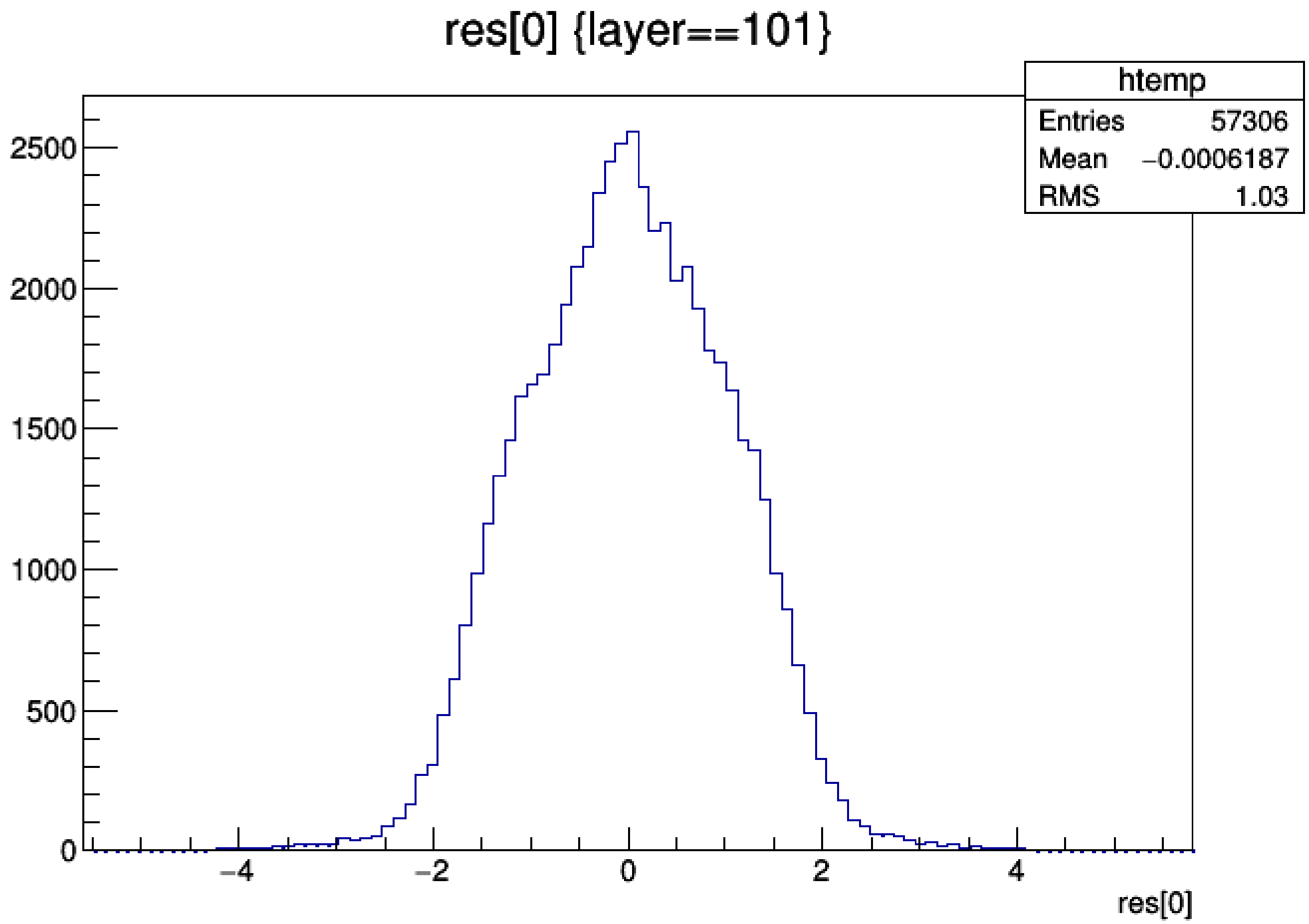
- V (Z) residual
  - Correlation with z-position
  - Bias
- (R-Phi OK)







(R-Phi OK)



res[0] {layer==101}
htemp
Entries 57306
Mean −0.0006187
RMS 1.03
2500
2000
1500
1000
500
0
−4
−2
0
2
4
res[0]

Alignment in BKLM

- From my private hacking test (100k IP muons)
  - sectors as rigid structures

    (no internal movement of layers allowed)

    Precision is better than 200um
  - movements of all modules

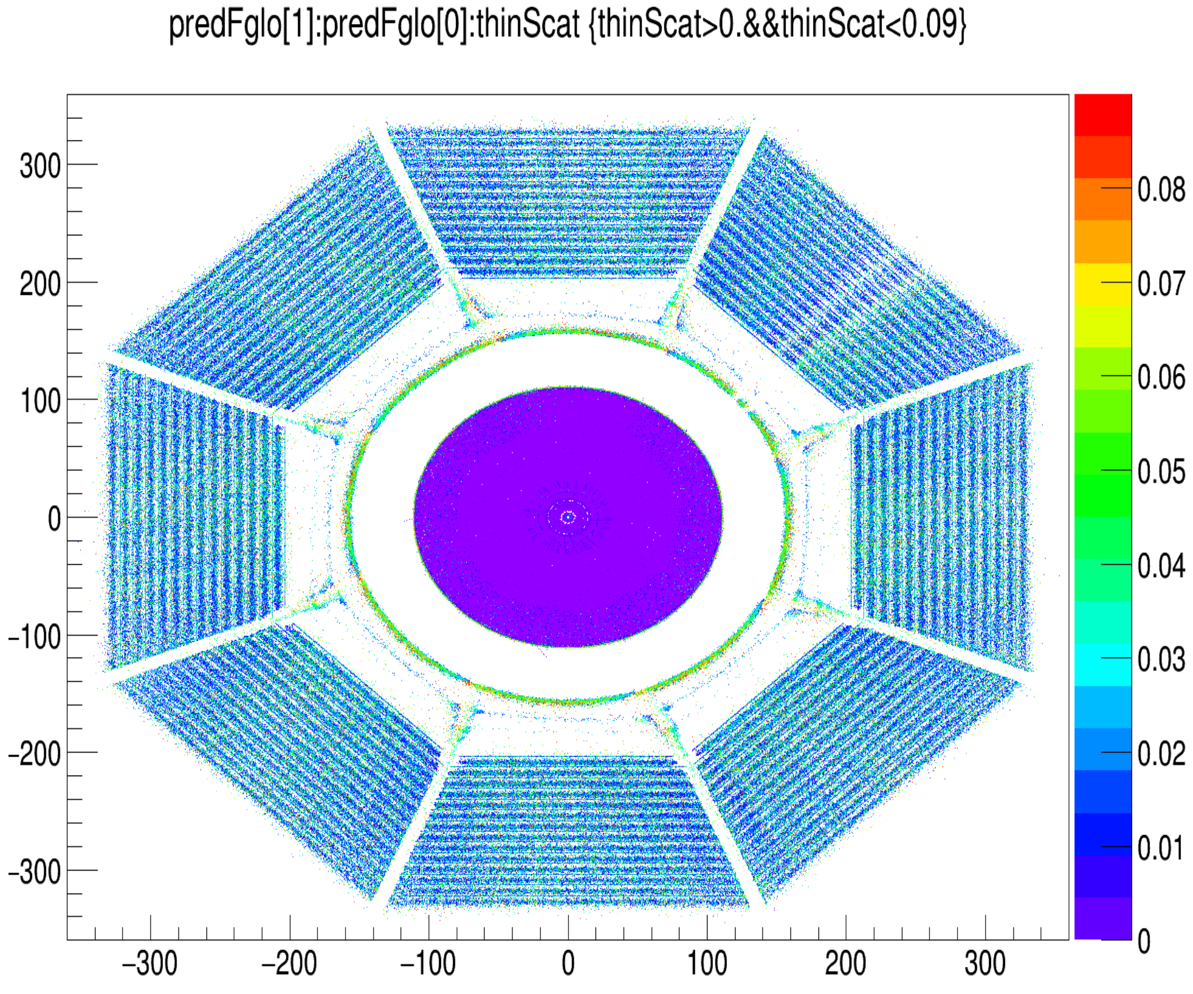
    Precision on this sample is on level of 100's of micrometers, better or equal to 1mm.
- Millepede interprets bias as misalignment in V(Z)

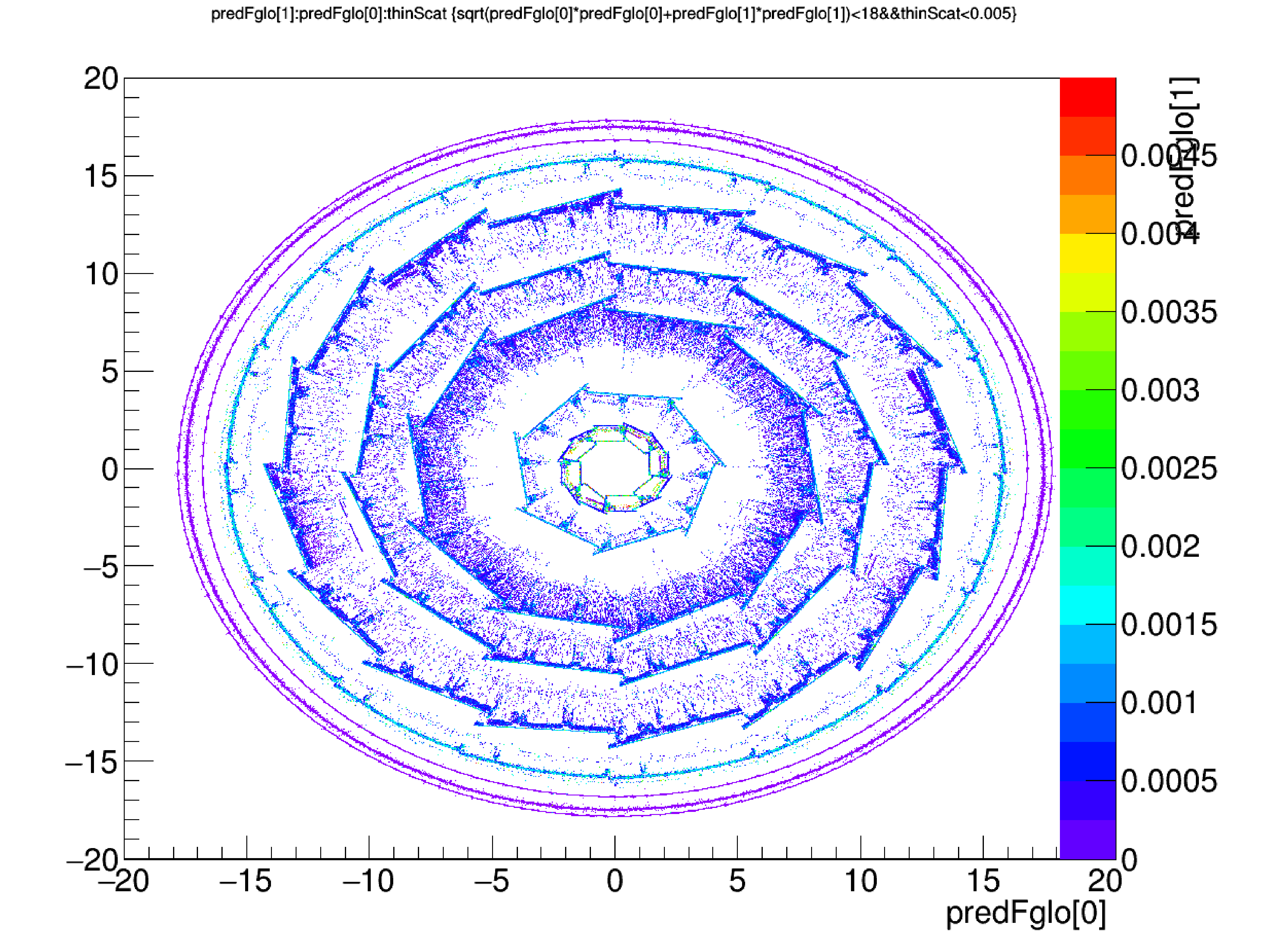
Misalignment/Alignment

- Misalignment + alignment parameter access (Database)
- Only at tracking level in RecoHits
  - Shift/rotate virtual planes (+ hit local positions for deformations)
- Hierarchy + constraints
  - General treatment by alignment group
  - But sub-detectors need to implement misalignment of larger structures and expose parameters
  - Sub-detectors provide:
    - Identification of structures (VXD ladder 1 in layer 1: 1.1.0, CDC 1.1.511?)
    - Relations: mother->daughter (VxdID->VxdID, WireID->WireID, …) and for each:
  - Transformation T: [x,y,z]_mother = T * [x,y,z]_daughter
- Common interfaces need to be decided everywhere

GblMultipleScatteringController

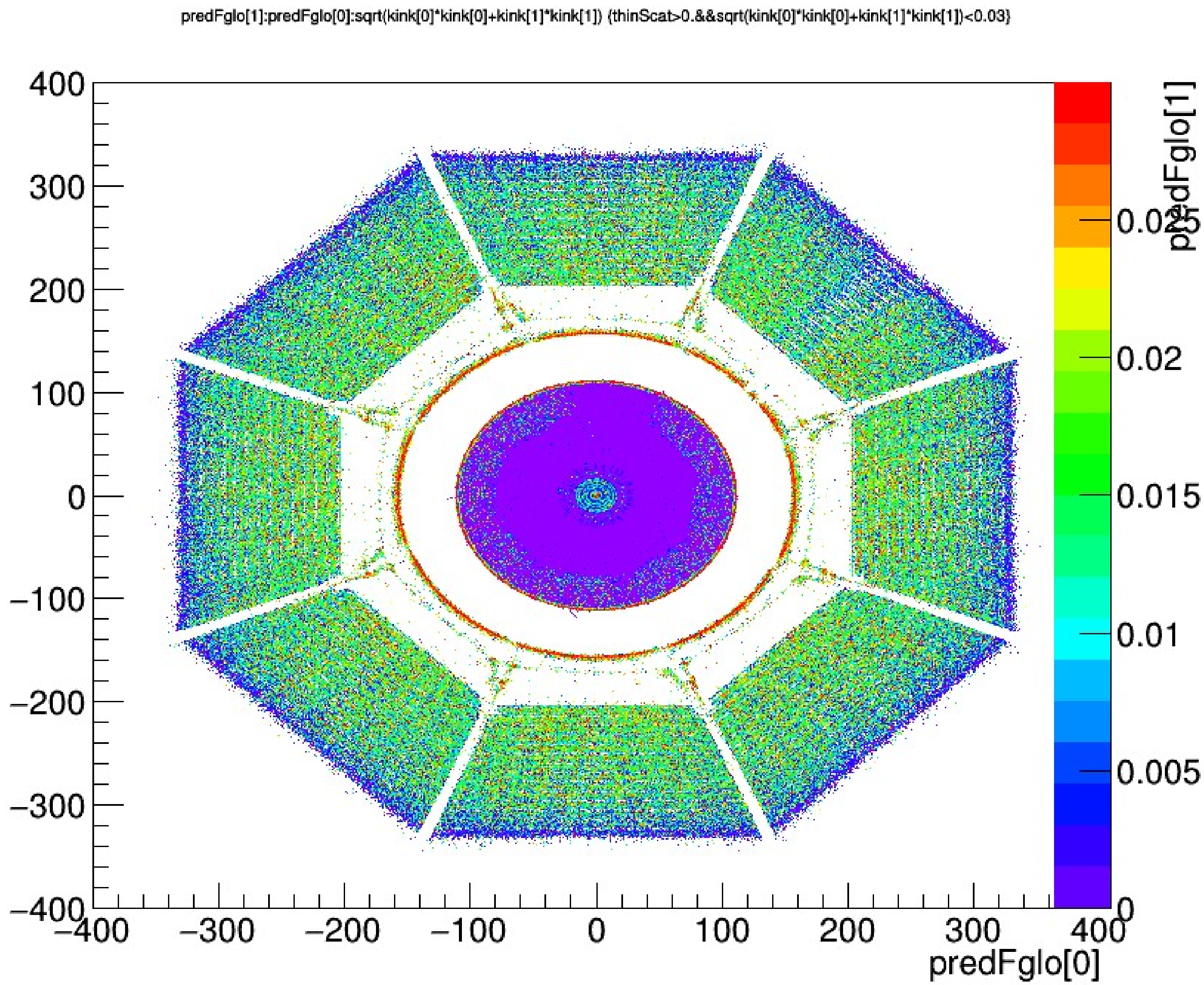
- Thick scatterers by default
- A scatterer at (and between) each (pair of) measurement
- Turn of MS inside CDC
  - gGeoManager





predFglo[1]:predFglo[0]:thinScat {sqrt(predFglo[0]*predFglo[0]+predFglo[1]*predFglo[1])<18&&thinScat<0.005}
20
15
10
5
0
−5
−10
−15
−20
−20
−15
−10
−5
0
5
10
15
20
predFglo[0]
predFglo[1]
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0

Fitted Kinks



Future plans

- BKLM reconstruction improvements, EKLM
- Calibration Framework?
- What do you want first? (sorted by my time needed)
  - Run(even intra) dependence in Millepede (at least initial implementaion → testbeam!!!)
  - Lorentz shift calibration in VXD
  - Sensor deformations for VXD (2nd order)
  - Hierarchy (for all detectors?) + constraints
  - CDC derivatives (alignment + x-t calibration + ?)
  - J/Psi→mumu
- Long term tasks (no clue how to do that:-)
  - Incorporation of (time-dep.) survey mesurements (global reference) and presigmas
  - Time-dependent constraints

Conclusions (and some questions)

- We can already do quite good VXD alignment
- We have four(!) different subdetectors
  - BKLM: promising test results, work in progress
  - VXD+BKLM „almost same“. Most special work necessary for CDC (alignment+calibration)
- CDC layer x,y shifts alignment already tested (application of alignment?)
  - X-t relation?
- We need (a lot of) contributions from sub-detectors
  - Parameters, misalignment/alignment, hierarchy, precisions and possible misalignment, database …

Last Words

- I commited some code which will be thrown away quite soon as a first attempt to integrate database/alignment/misalignment etc. and test it.
- All code is in svn for now. Missing documentation will be added soon (but...)

  Have a look if interested

Thank you for attention!
Questions? Comments?