Layer definition's Rosetta Stone

or: How to accomodate the new alignment definition with the default one

Marco Scodeggio

CGEM SW Weekly Meeting

The two definitions

New definition (ND)

Introduced for new alignment procedure by Aiqiang

Layers
[0 - 5] (e.g. L1S1 = 0, L2S2 = 3)

Affects the CgemAlignAlg-[xxx] and the CgemGeomSvc-[xxx] packages

Default definition (DD)

Standard definition up to release *CgemBoss6.6.5.f*

Layers [0 - 2]
Sheets [0 - 1]

Used in the QA

The two definitions

New definition (ND)

Introduced for new alignme
procedure by Aiqiang

Layers
[0 - 5] (e.g. L1S1 = 0, L2S2 =

Affects the CgemAlignAlg-[xxx]
and the CgemGeomSvc-[xxx]
packages

Default definition (DD)

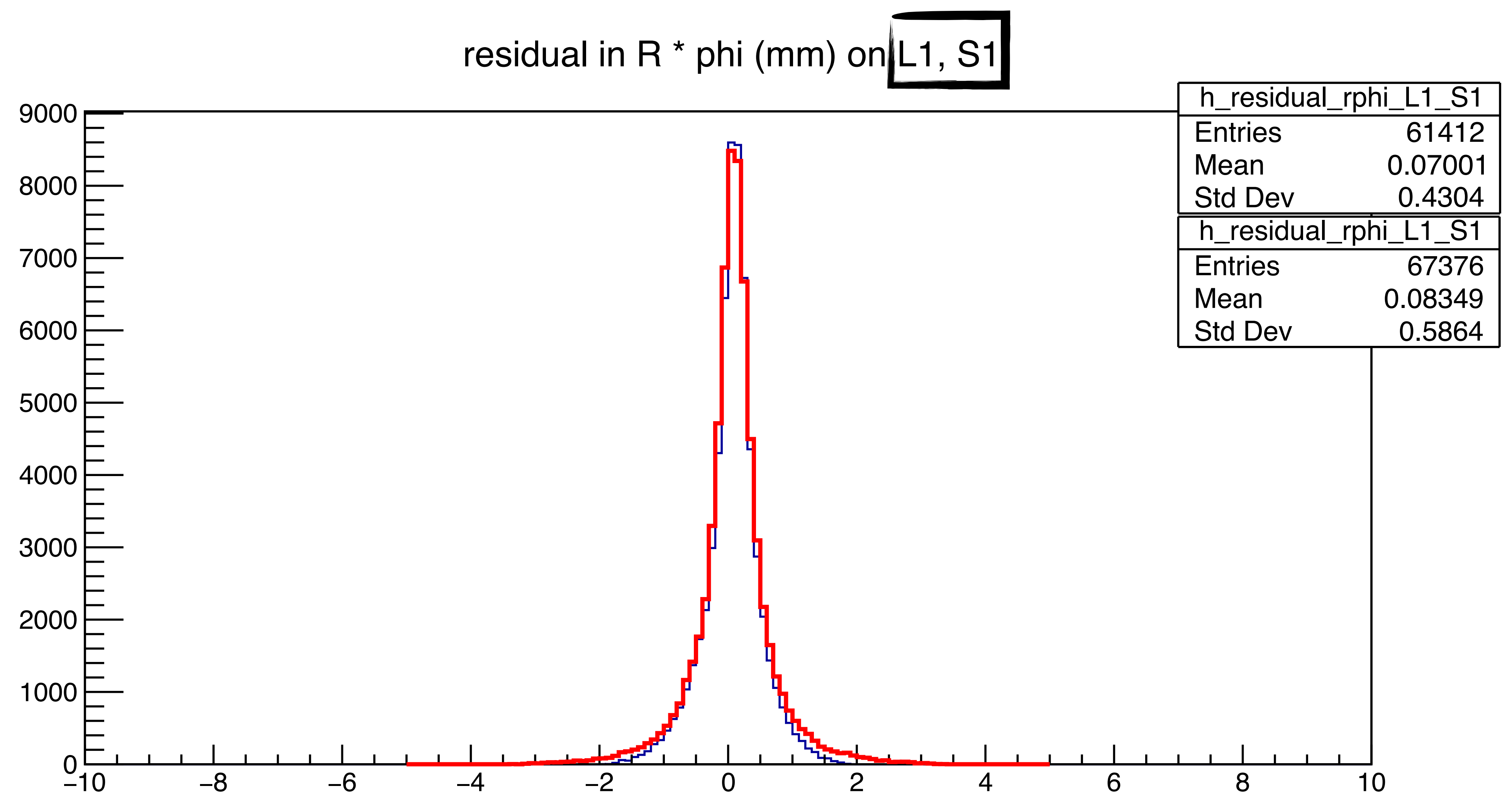
tandard definition up to
release *CgemBoss6.6.5.f*

Layers [0 - 2]
Sheets [0 - 1]

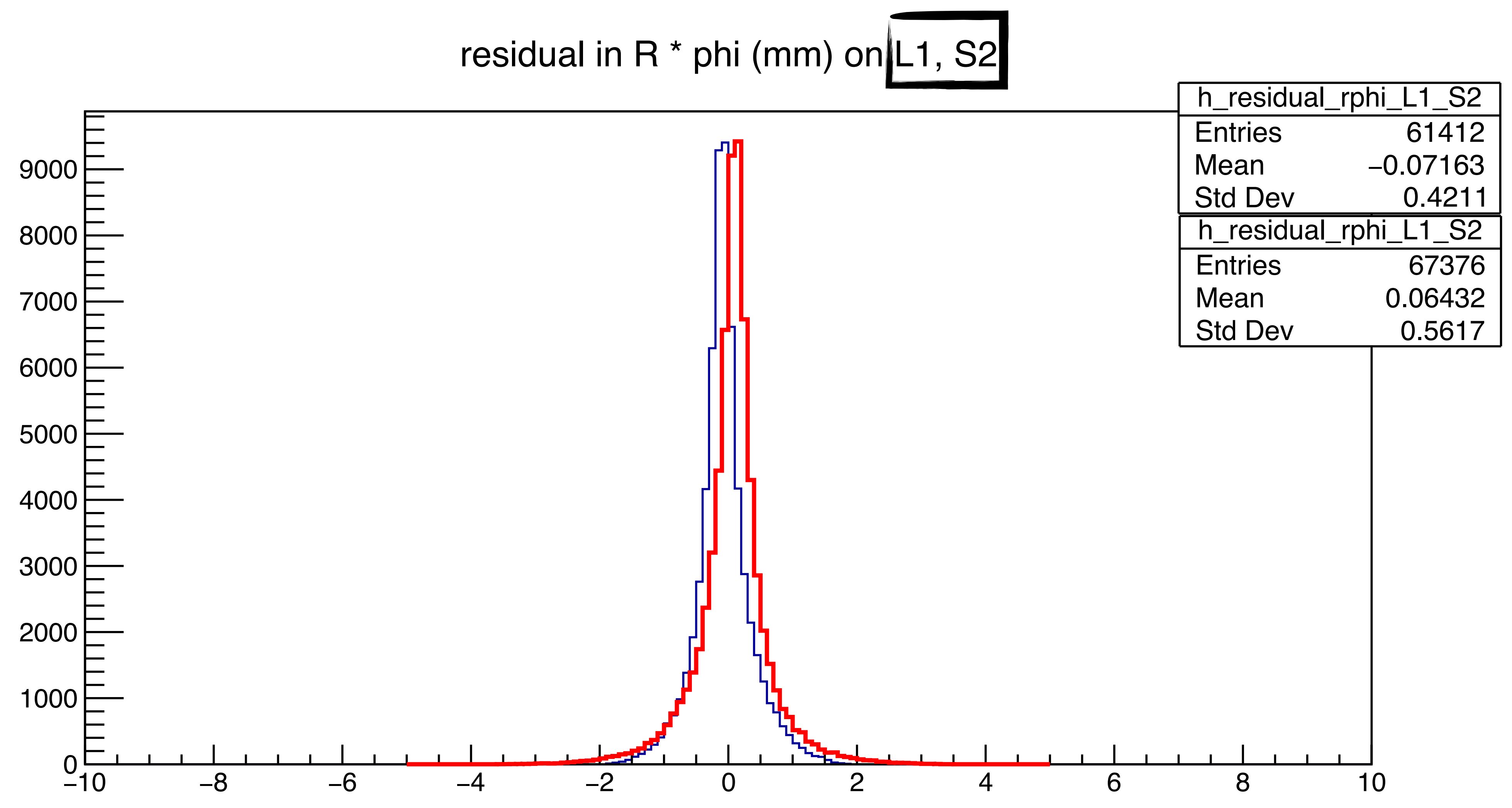
Used in the QA

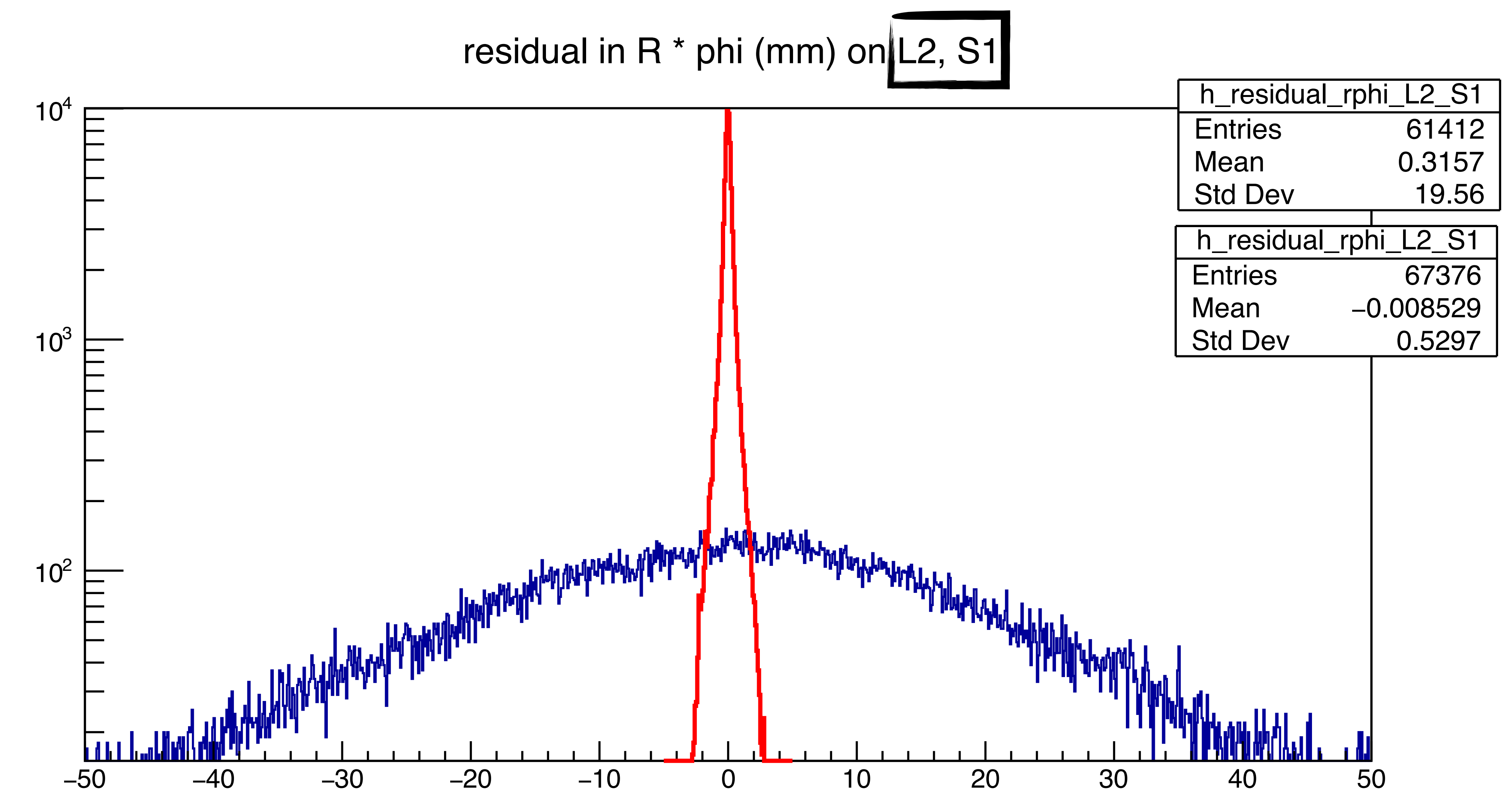
At the present state of the code,
these two definitions **coexist**
and **cause problems**...

Problems arose

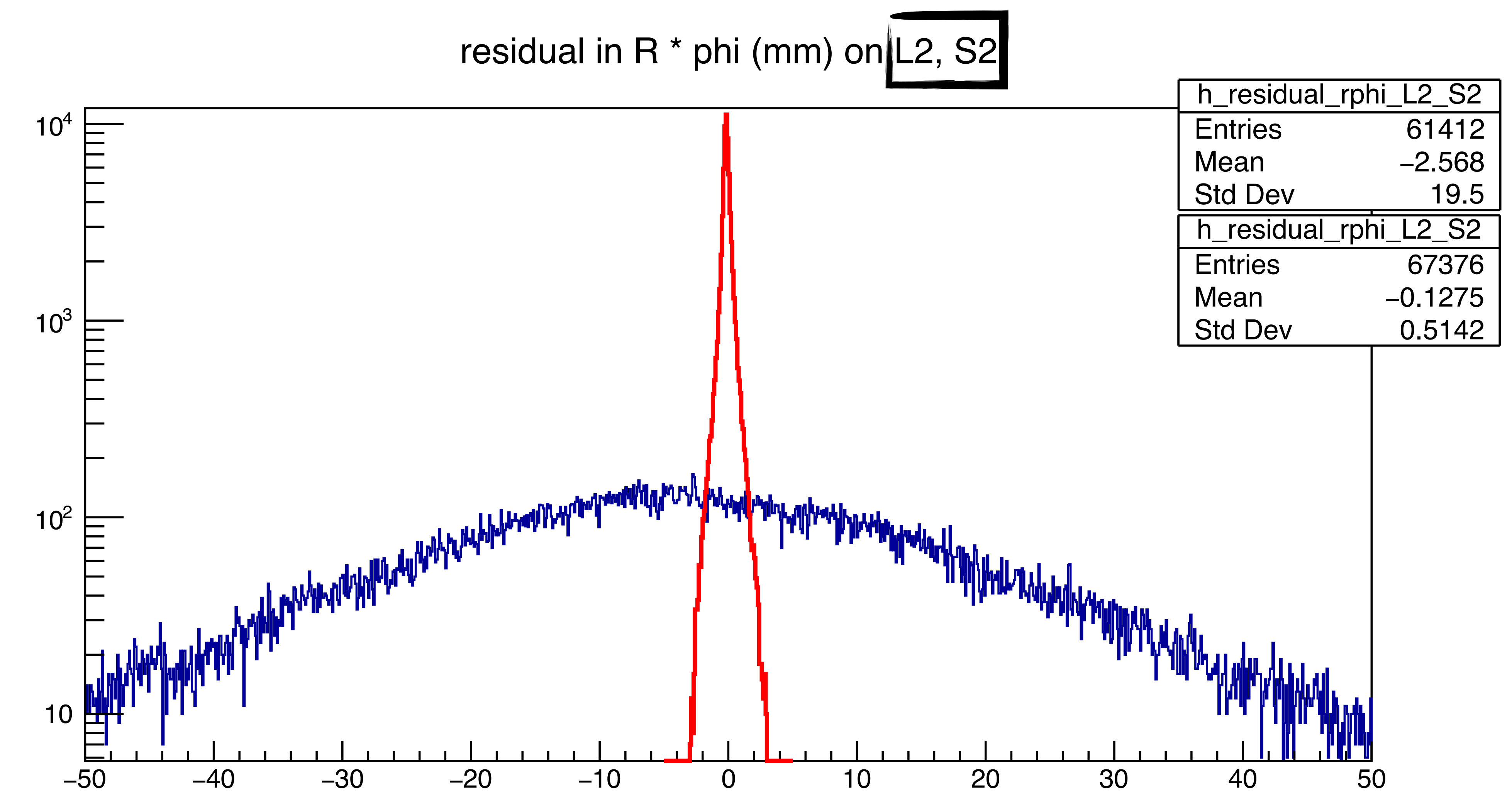


Red: Baseline (665f)
Blue: Testing (665g)

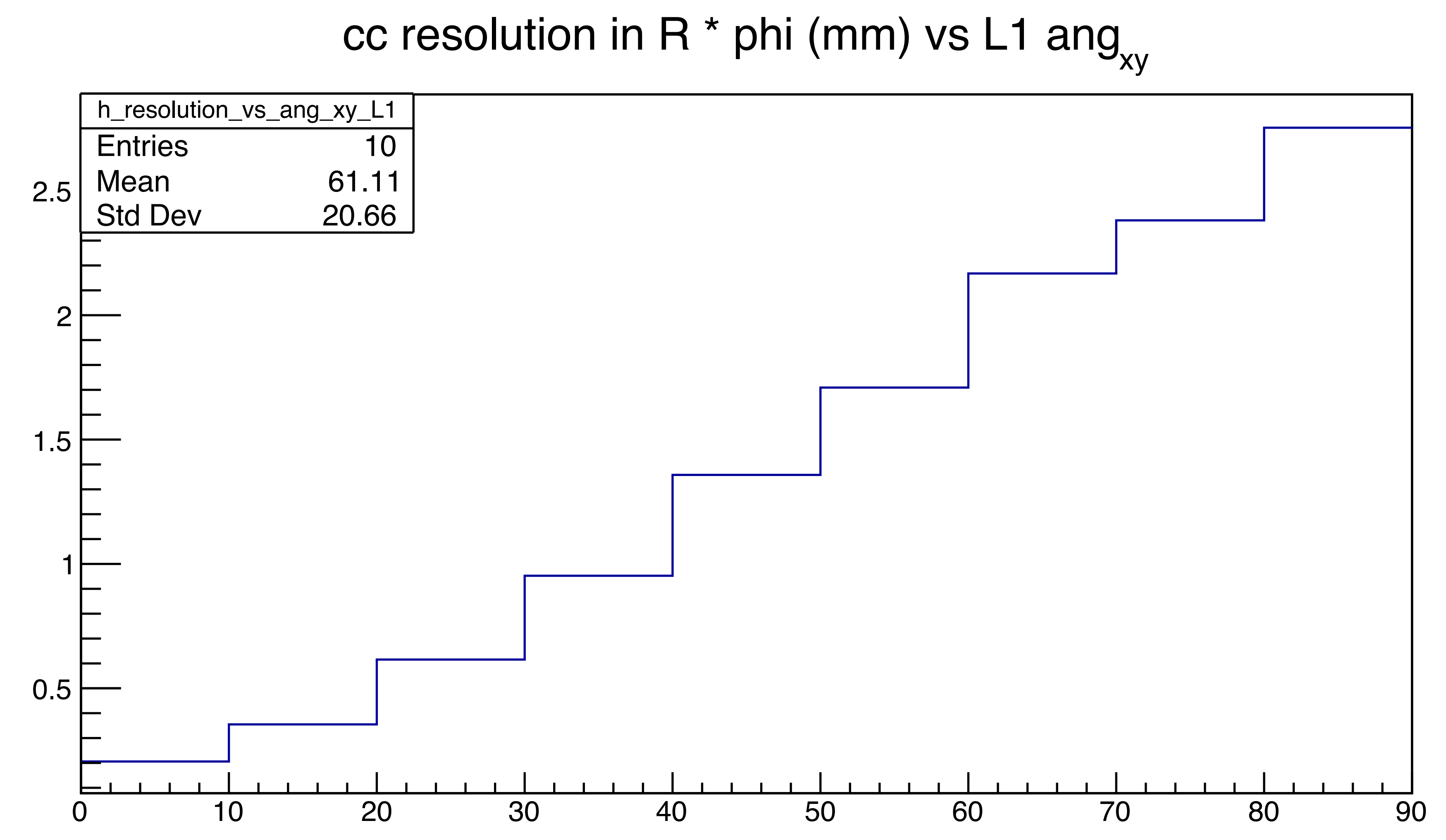




Clearly there is a **problem** in the **definition of Layer 2**, but who is the culprit?



Problems arose

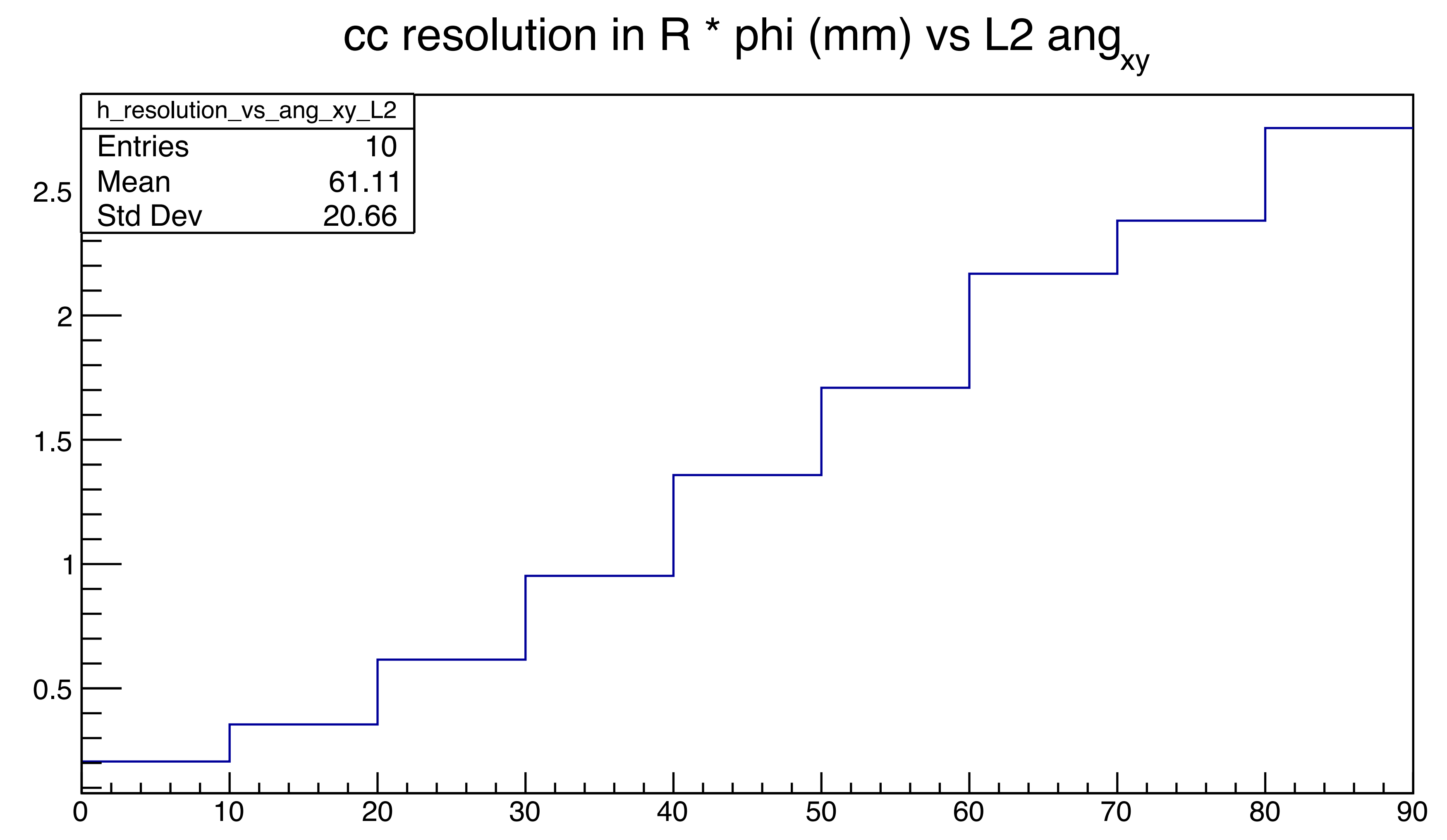


Testing release

CgemBoss6.6.5.g

Clearly there is a **problem** in the **definition of Layer 2**, but who is the culprit?

In CgemLineFit/…/TestTrack.cxx
```
bool TestTrack::ComputeIntersection
```



Let’s get to the code

CgemLineFit/…/TestTrack.cxx

L865
```
bool TestTrack::ComputeIntersection(int layerid,
                                    double &x1, double &y1, double &z1, double& phi1, double &v1,
                                    double &x2, double &y2, double &z2, double& phi2, double &v2,
                                    double &angCR_xy, double &angCR_yz) {
```

[...]

layerid takes the DD of layers and is fed into `getPointAligned` and `getPointIdealGeom`

```
if(align_flag==true) gotit = midplane->getPointAligned(layerid, linefit, posup, posdown, phivup, phivdown);
     else gotit = midplane->getPointIdealGeom(layerid, linefit, posup, posdown, phivup, phivdown);
```

[...]

}

Let’s get to the code

CgemGeomSvc/CgemGeomSvc-00-00-37/…/CgemMidDriftPlane.cxx

```
bool CgemMidDriftPlane::getPointIdealGeom(int layer, StraightLine pLine,
                                          HepPoint3D& posUp, HepPoint3D& posDown,
                                          double phiVUp[], double phiVDown[]){

     int layer_geo = int(layer/2);
```

Takes the ND to convert it back to the DD

```
bool CgemMidDriftPlane::getPointAligned(int layer, StraightLine pLine,
                                        HepPoint3D& posUp, HepPoint3D& posDown,
                                        double phiVUp[], double phiVDown[]){

     int layer_geo = int(layer/2);
```

```
bool CgemMidDriftPlane::getPointAligned_New(int layer, StraightLine pLine,
                                            HepPoint3D& posUp, HepPoint3D& posDown,
                                            double phiVUp[], double phiVDown[]){

     int layer_geo = int(layer/2);
```

A solution

A possible one given the idea of uniformity inside the QA

Keeping in mind the paradigm of uniformity within a package,
it was opted to let `ComputeIntersection` externally the same while changing the call
to
`CgemMidDriftPlane::getPointAligned`
into
`CgemMidDriftPlane::getPointAligned_QA`

A solution

A possible one given the idea of uniformity inside the QA

Why?

Keeping in mind the paradigm of uniformity within a package,
it was opted to let `ComputeIntersection` externally the same while changing the call
to
`CgemMidDriftPlane::getPointAligned`
into
`CgemMidDriftPlane::getPointAligned_QA`

A solution

A possible one given the idea of uniformity inside the QA

Keeping in mind the paradigm of uniformity within a package,
it was opted to let `ComputeIntersection` externally the same while changing the call to
`CgemMidDriftPlane::getPointAligned`
into
`CgemMidDriftPlane::getPointAligned_QA`

Why?

In **TestTrack.cxx** we use the DD

```
      int layer = CgemID::layer(ident);
      int sheet = CgemID::sheet(ident);
      int layerid = (*iClusterCol)->getlayerid();
      int sheetid = (*iClusterCol)->getsheetid();
      track_layerid[iclus]   = tcluster->getlayerid();
      track_sheetid[iclus]   = tcluster->getsheetid();
```

A solution

A possible one given the idea of uniformity inside the QA

Keeping in mind the paradigm of uniformity within a package,
it was opted to let `ComputeIntersection` externally the same while changing the call to
`CgemMidDriftPlane::getPointAligned(int layer_id, …)`
into
`CgemMidDriftPlane::getPointAligned_QA(int layer_id, int sheet_id, …)`

Reverting what it was done: Now we convert the DD to the ND

```
int layer =  2*layer_geo + sheet;
```

leaving the ND in all the function referring to the alignment

Conclusions

After some off-line discussions with Aiqiang, regarding the solution he propose (the ND -> DD), this different solution proposed was concocted (DD -> ND) in order to keep uniformity within the QA

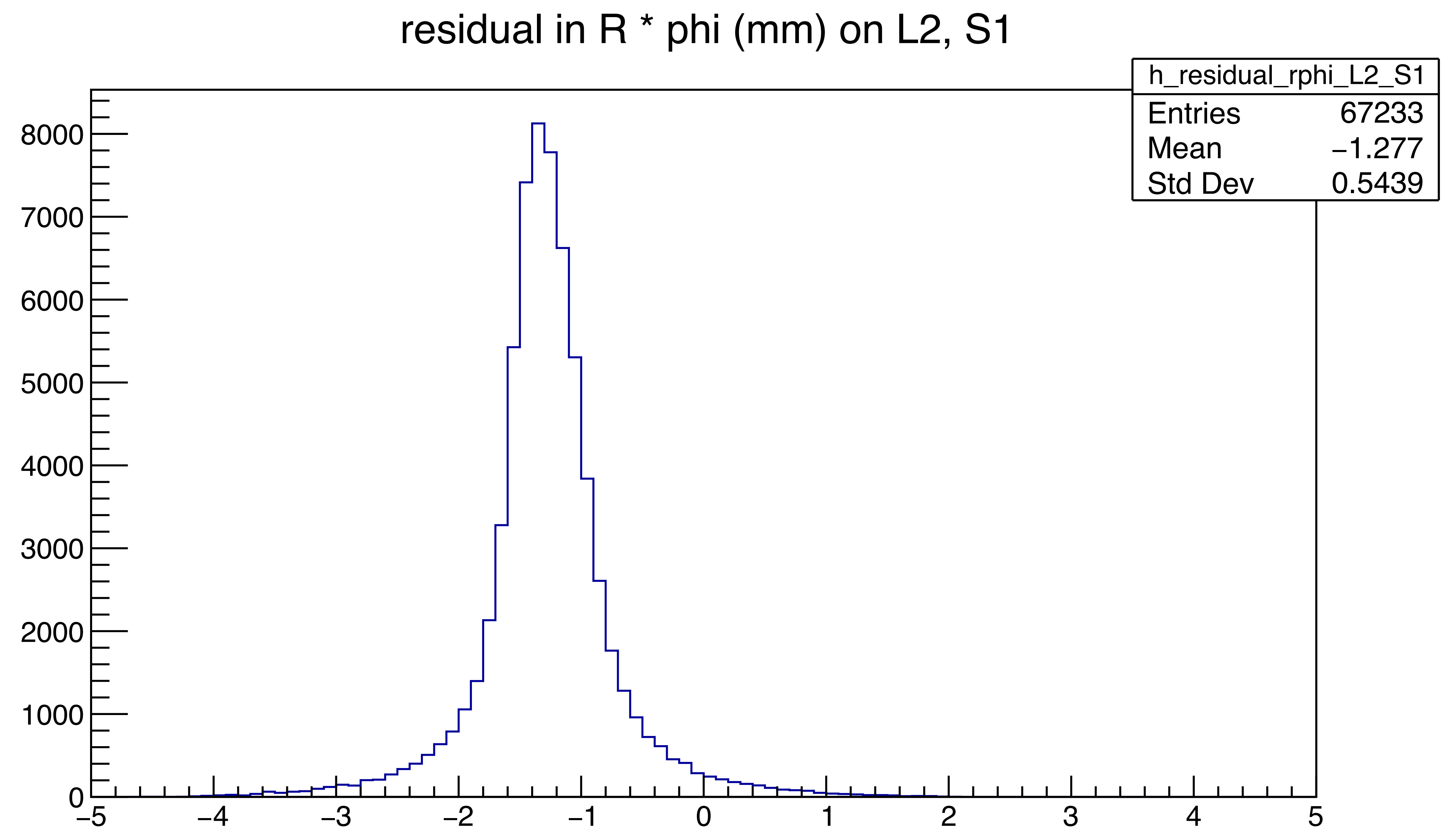
“My solution” came from a not having a clear picture of the Aiqiang’s one (this is on me completely), which sparked more tests on my side and this will of uniformity

Conclusions

After some off-line discussions with Aiqiang, regarding the solution he propose (the ND -> DD), this different solution proposed was concocted (DD -> ND) in order to keep uniformity within the QA

“My solution” came from a not having a clear picture of the Aiqiang’s one (this is on me completely), which sparked more tests on my side and this will of uniformity

The solution seems to work, more tests are needed…

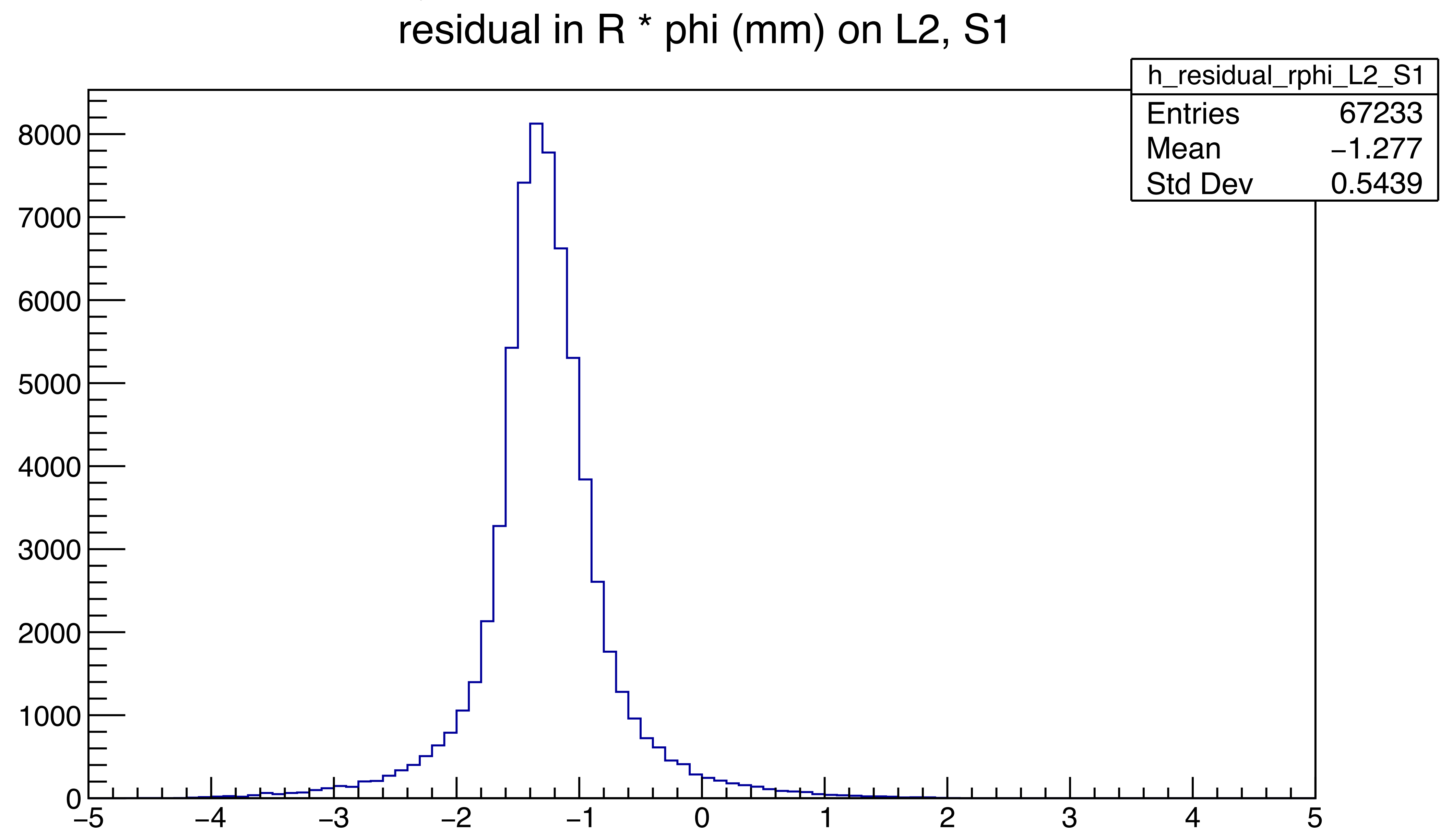


Conclusions

Why so off-centred? More corrections are needed?

The solution seems to work, more tests are needed...

In CgemLineFit.cxx it seems that, for `getPointIdealGeom` or `getPointAligned_New`, the virtual layer (i.e. ND) is used... maybe something else is needed?


residual in R * phi (mm) on L2, S1
h_residual_rphi_L2_S1
Entries 67233
Mean −1.277
Std Dev 0.5439