Effects of the Velo drift on the mass distributions

General remarks and settings

- The goal is to estimate the size of the impact of VELO movements on the reconstructed mass distributions
- Tested on both **Tx** and **Ry** for the VELO C-side**:** effect of Ry proved to be negligible
- Tests run over the full set of B0->D*π MC samples from `/eos/lhcb/user/l/lohenry/ClusterBias/DigiSamples_clusterFix`
  - The samples include the cluster bias fix
- Alignment degrees of freedom: **Tx** and **Rz**
- Alignables: **LongModules**
- Lagrange constraints:
  - Tx, Rz and Szx (shear of x vs z) are constrained to 0 across the whole SciFi
  - The average Tx of the modules in the last layer (T3X2) is also constrained to be 0
- Running using the new **Alignment/master** and the new PrKalman filter

Data selection

Well reconstructed primary vertices with a threshold on the number of reconstructed tracks to reduce combinatorics

```
def VPPrimaryVertices(input_pvs):
    from PyConf.Algorithms import VertexListRefiner
    selected_pvs = VertexListRefiner(
        name='VertexListRefinerVPPrimaryVertices',
        MaxChi2PerDoF=5,
        MinNumTracks=15,
        MaxNumTracks=50,
        MinNumLongTracks=0,
        InputLocation=input_pvs).OutputLocation
    return selected_pvs
```

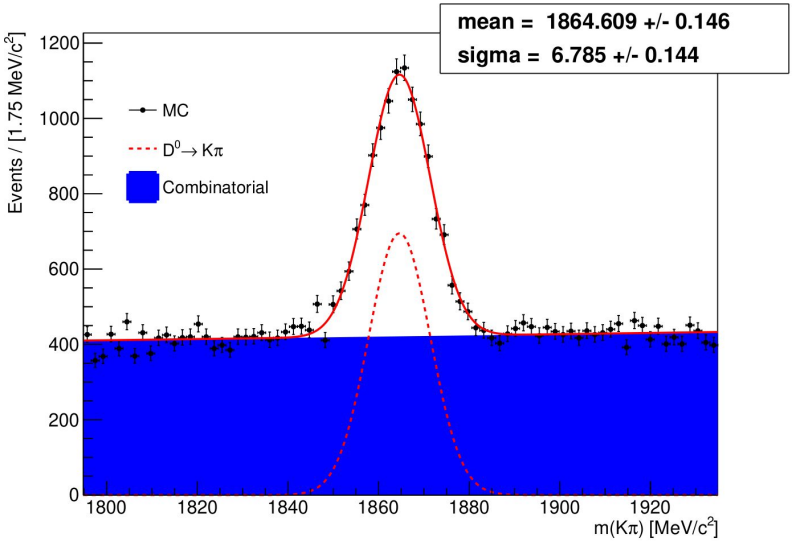
Using only well reconstructed long tracks

```
def GoodLongTracks(input_tracks):
    myTrackSelector = TrackSelector(
        TrackTypes=["Long"],
        MinPCut=5000,
        MaxPCut=200000,
        MinPtCut=200,
        MaxNTHoles=1,
        MaxChi2Cut=5,
        MaxChi2PerDoFMatch=5,
        MaxChi2PerDoFVelo=5,
        MaxChi2PerDoFDownstream=5)
    selected_tracks = TrackListRefiner(
        name='TrackListRefinerGoodLongTracks',
        inputLocation=input_tracks,
        Selector=myTrackSelector).outputLocation
    return selected_tracks
```

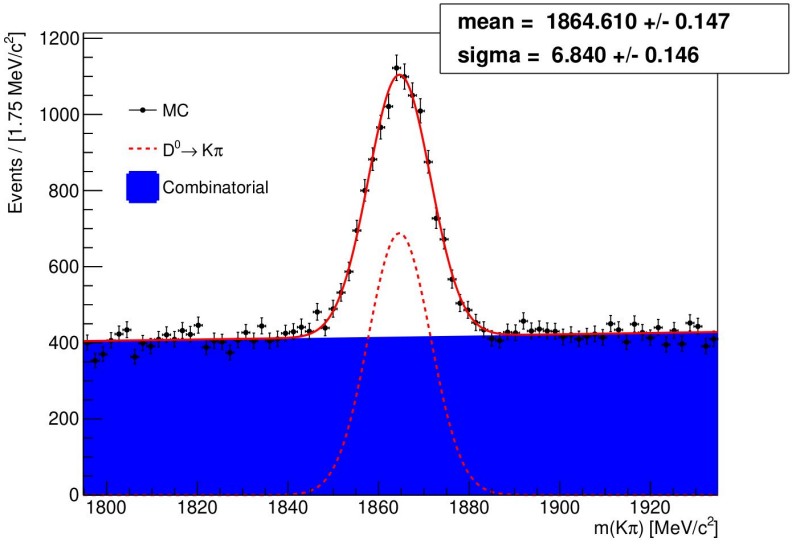
- A further set of cuts looking for **D0 -> Kπ** candidates are applied to the selected tracks and vertices
    - The selection of candidates is aligned with the HLT1 D0 line

Sample fit results

Tx = 0

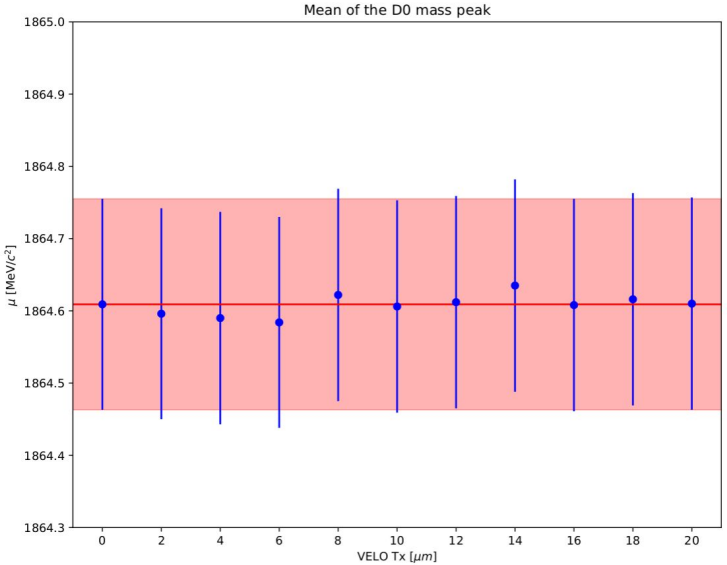


Tx = 20 μm

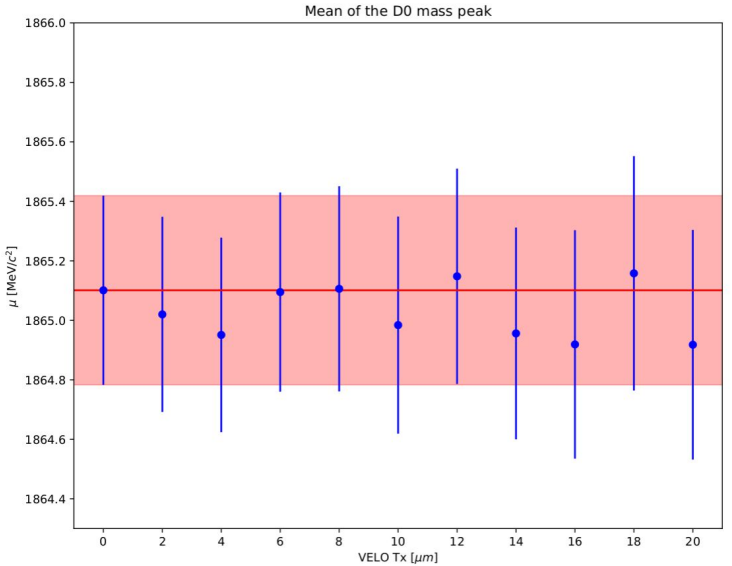


Evolution of the mean

Tx in [0, 20] μm



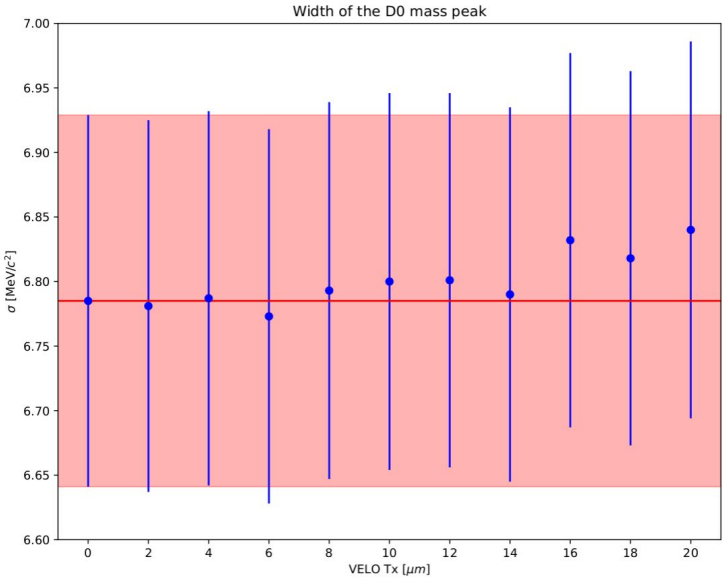
Tx in [0, 200] μm



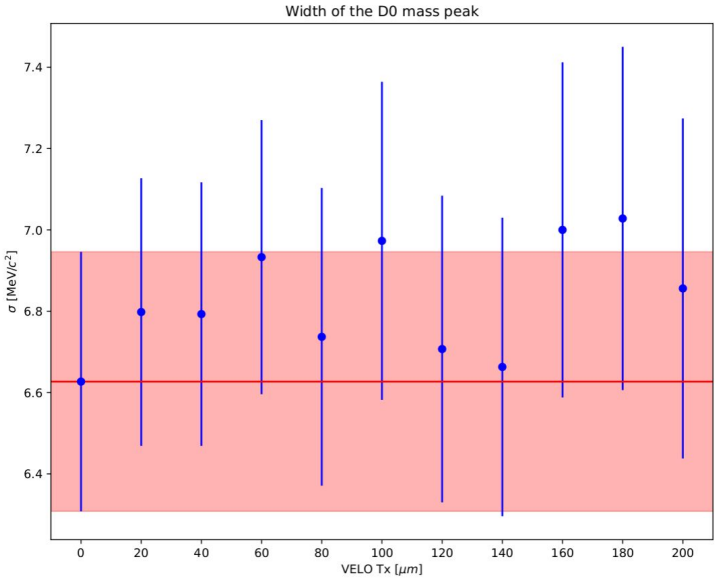
Note: Jobs run for the range [0, 200] μm have less statistics (8k events) to speed up testing

Evolution of the width

Tx in [0, 20] μm



Tx in [0, 200] μm



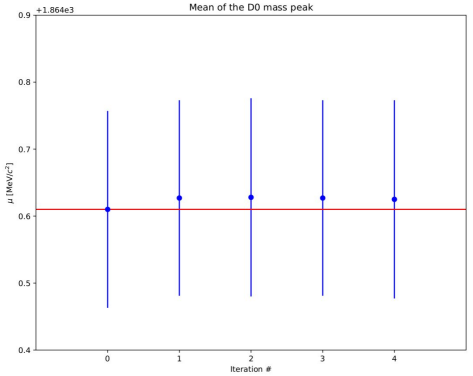
Note: Jobs run for the range [0, 200] μm have less statistics (8k events) to speed up testing

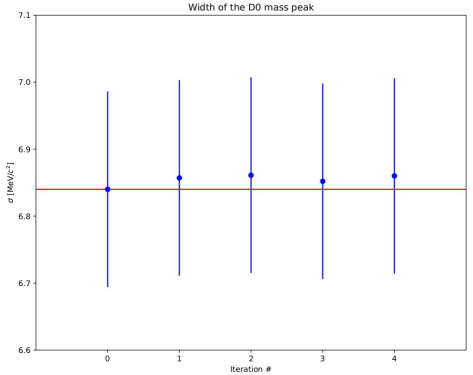
Effect of the alignment algorithm

Mean

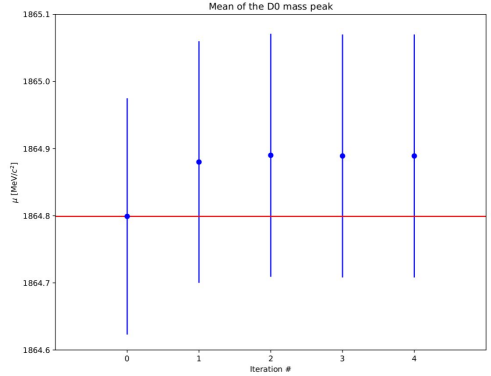
Width

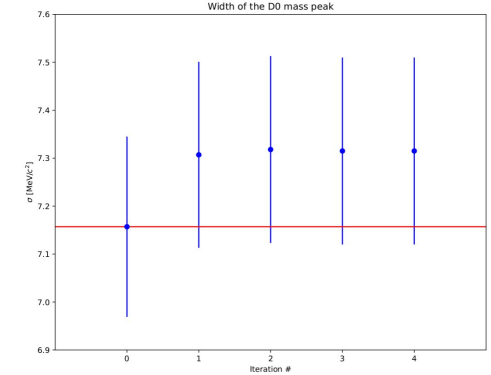
Tx = 20 μm





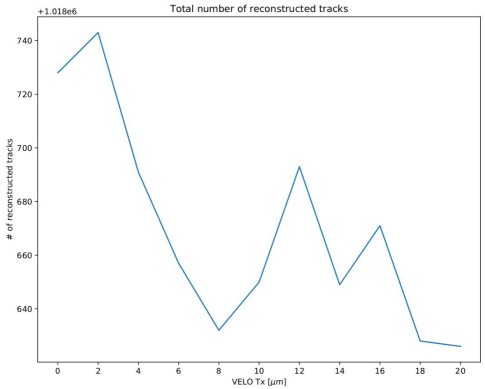
Tx = 20 μm



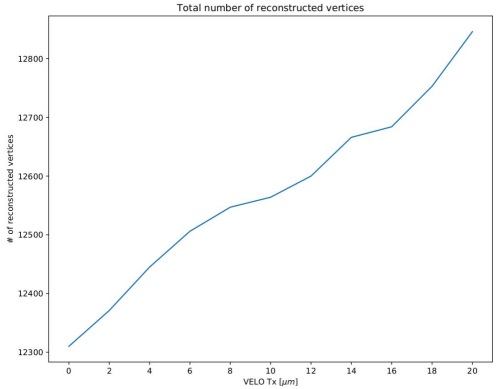


Further checks

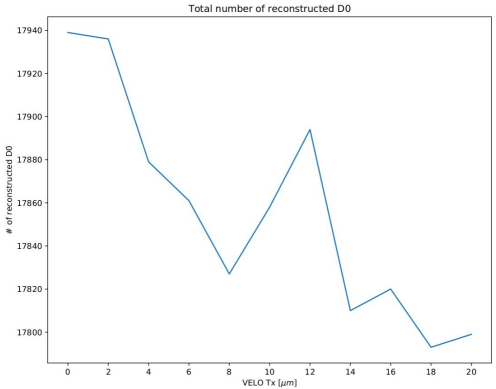
of tracks



of PVs



of D0 candidates



Larger number of PVs due to the appearance of **clone vertices** in the two VP halves when one of the halves is displaced

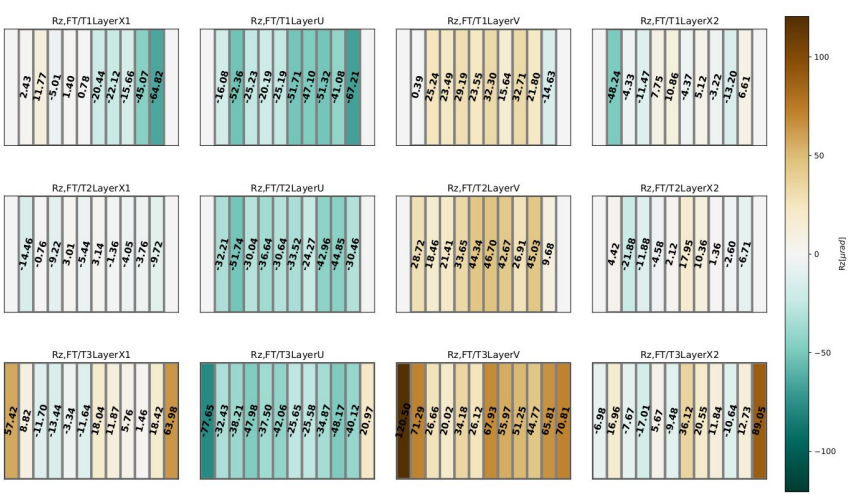
Test of alignment quality

Alignment settings

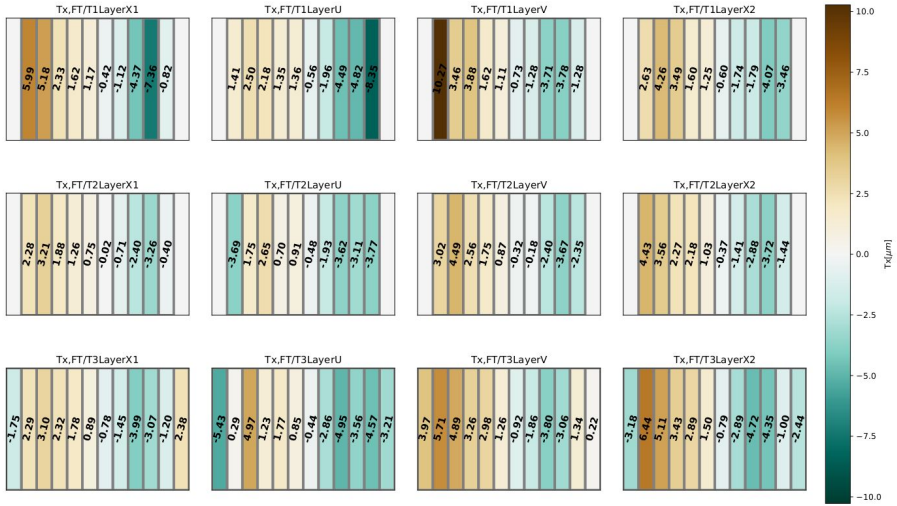
- Initially we wanted to check the impact of the UT on the general alignment algorithm with the new PrKalman filter
- Running over the full set of MC events from `/eos/lhcb/user/l/lohenry/ClusterBias/MinBias/ZehuaFix`
  - The sample includes the fix to remove the cluster bias
- Test are run assuming a fully aligned detector and employing a D0 mass constrain
- Degrees of freedom: **Tx** and **Rz**
- Alignables: **LongModules**
- Lagrange constraints:
  - Tx, Rz and Szx (shear of x vs z) are constrained to 0 across the whole SciFi
  - The average Tx of the modules in the last layer (T3X2) is also constrained to be 0

Tests with UT

Rz

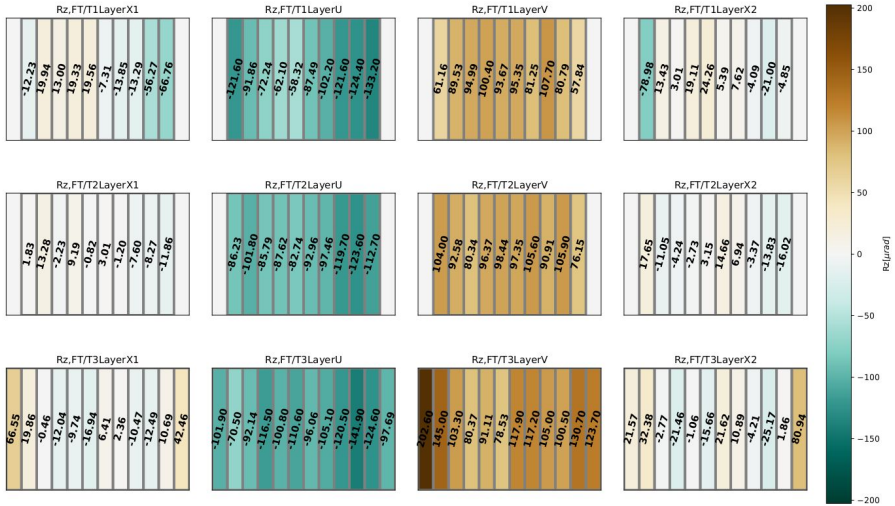


Tx

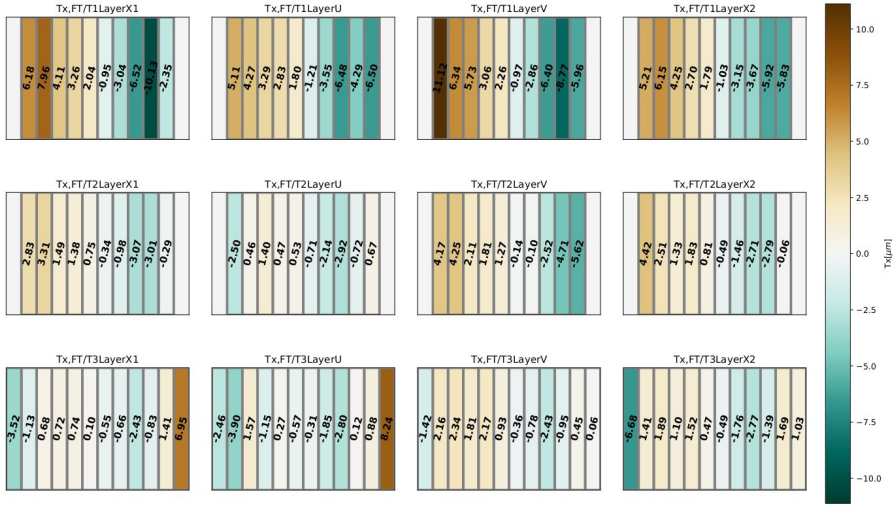


Tests without UT

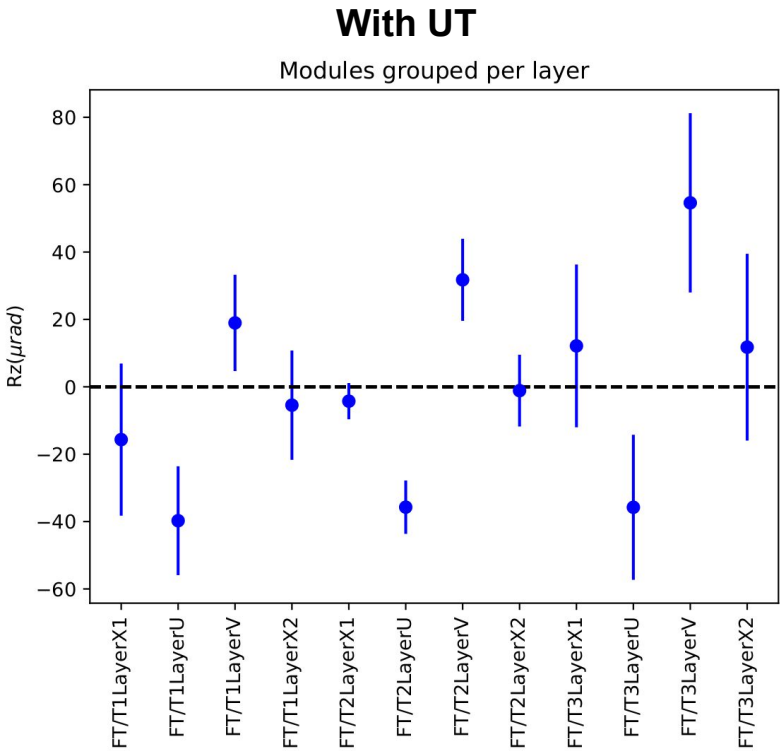
Rz

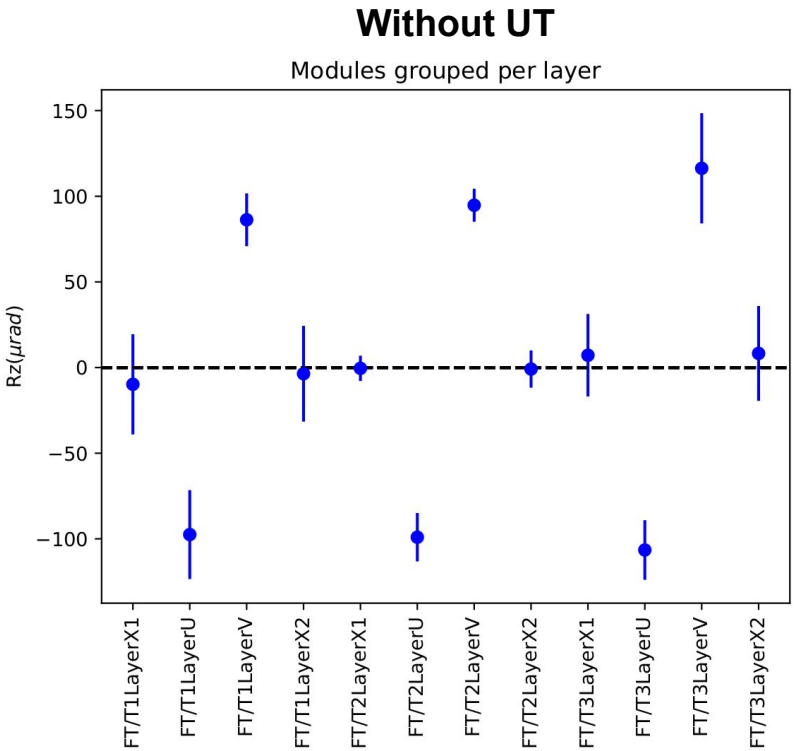


Tx

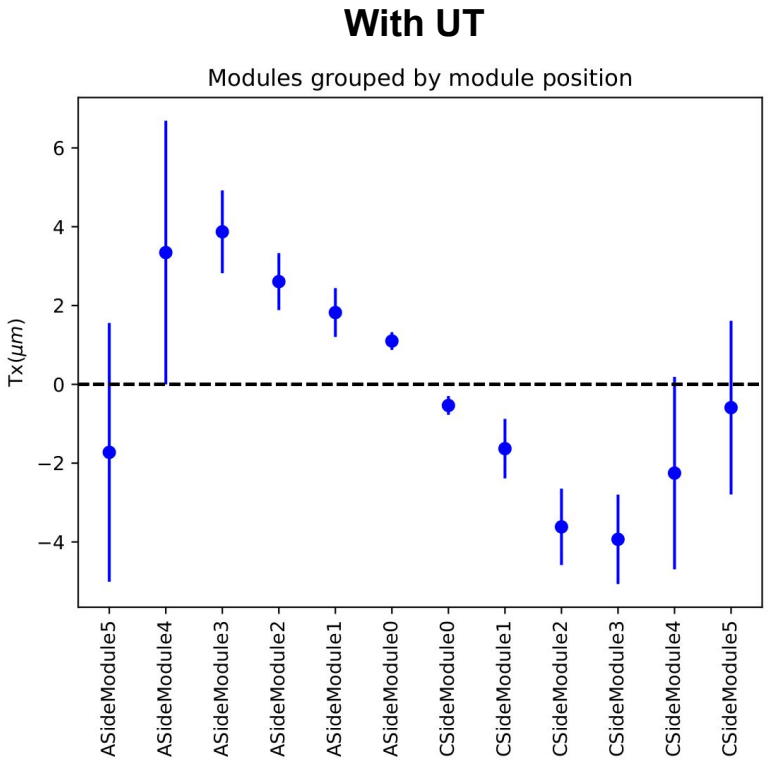


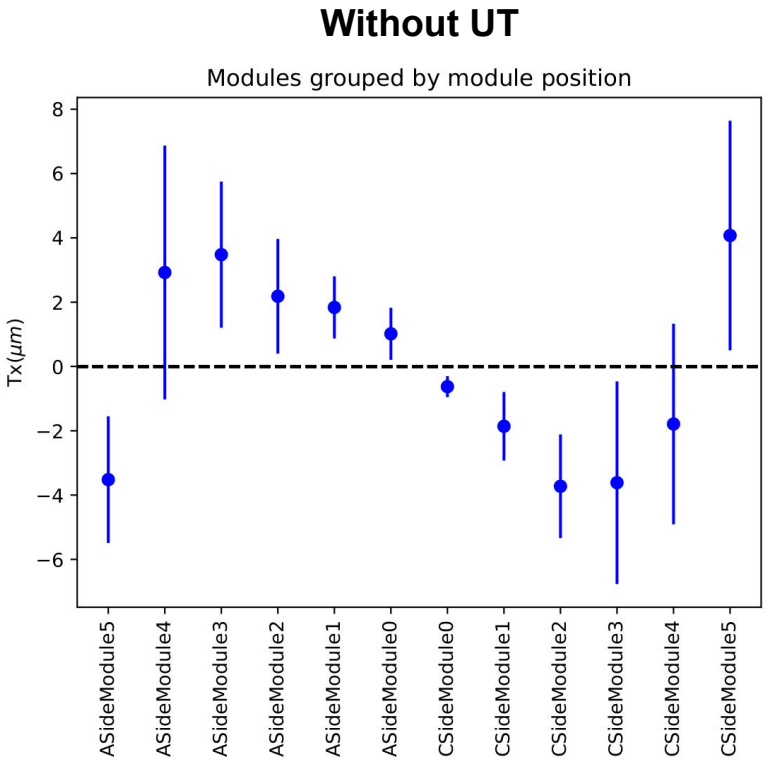
Rz across SciFi layers





Tx across SciFi module positions





On going work

- Null tests so far are run using the DetDesc detector geometry. I'm currently re-running them using DD4Hep to check for the presence of bugs
- I am looking into possible biases present in the reconstruction sequence that could explain these results