KF-based Alignment Testing

Tom Eichlersmith

he/him/his

University of Minnesota

eichl008@umn.edu

April 12, 2023

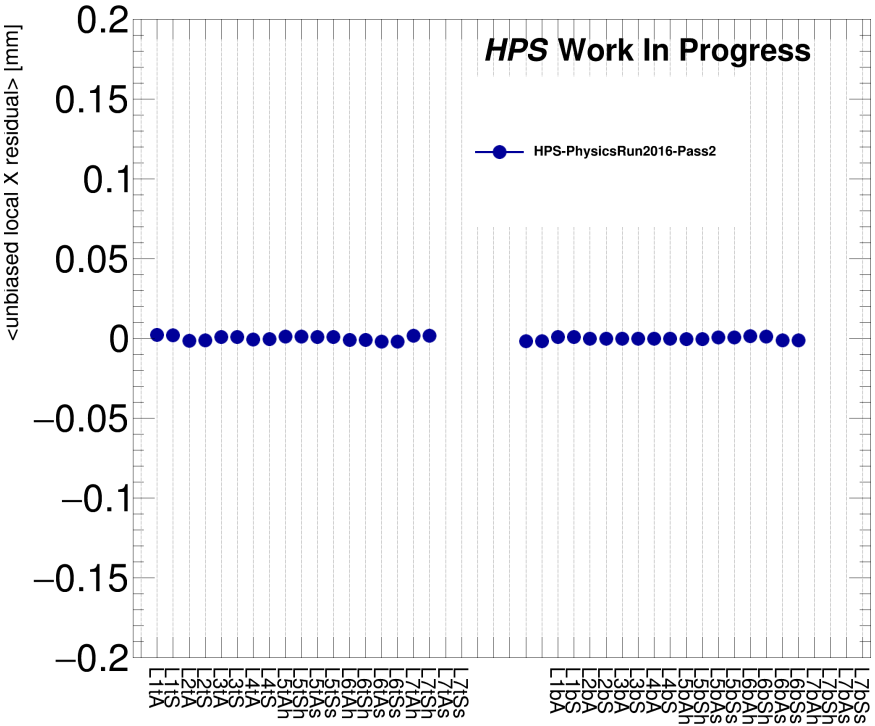
Intro

- **MP** – MillePede, the library (mille) and program (pede) we use for alignment
- **GBL** – General Broken Lines, an algorithm for determining a trajectory including potential scattering within the tracker volumes
- **KF** – Kalman Filter, new method of finding tracks that we wish to use for alignment
- **JNA** – Java Native Access, the method through which we call GBL C++ functions from the java code

Current Goal

Re-familiarize ourselves with 2016 alignment.

In General, Good

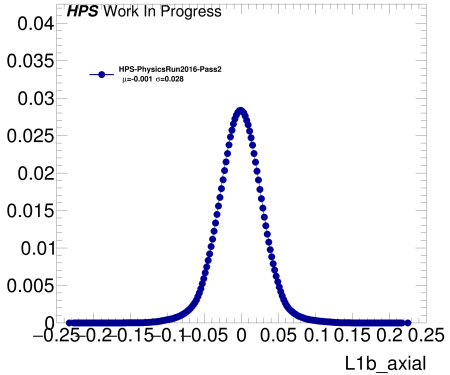


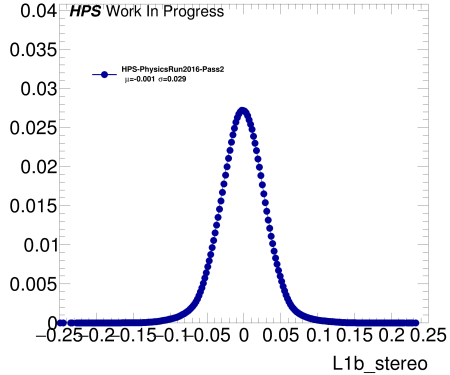
- Seemingly no trends in residuals across layers
- Pretty tight around zero

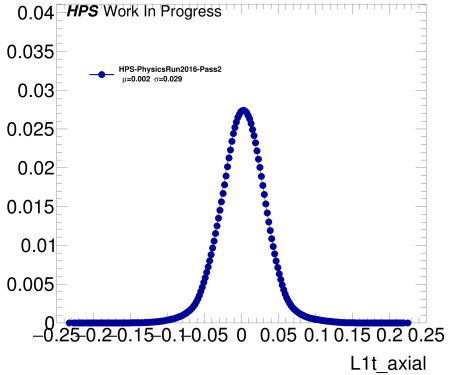
Let's look in more detail.

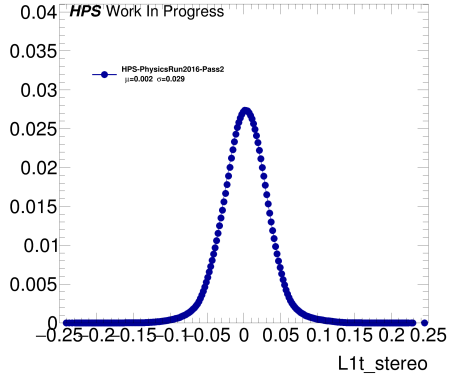
Plain Unbiased Residuals

Looking for translation along sensitive direction (tu) misalignment
Details not important,
long story short:
Unbiased residuals look Gaussian and are close to 0.
Layer 1 sensors shown to the right, other layers are included in tar-ball uploaded on the agenda.



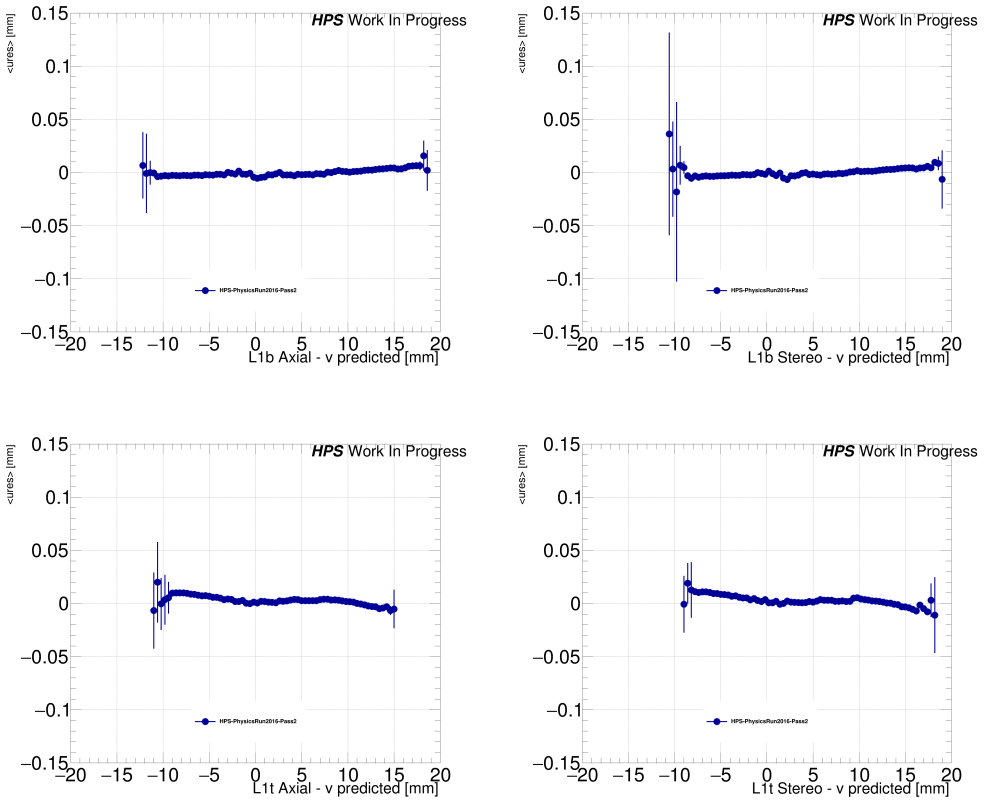






Unbiased Residuals against v

Looking for rotation about normal to sensor plane (rw) misalignment
Details not important,
long story short:
Hard pressed to find obvious rotation, could maybe find a trend.
Layer 1 sensors shown to the right, other layers are included in tar-ball uploaded on the agenda.



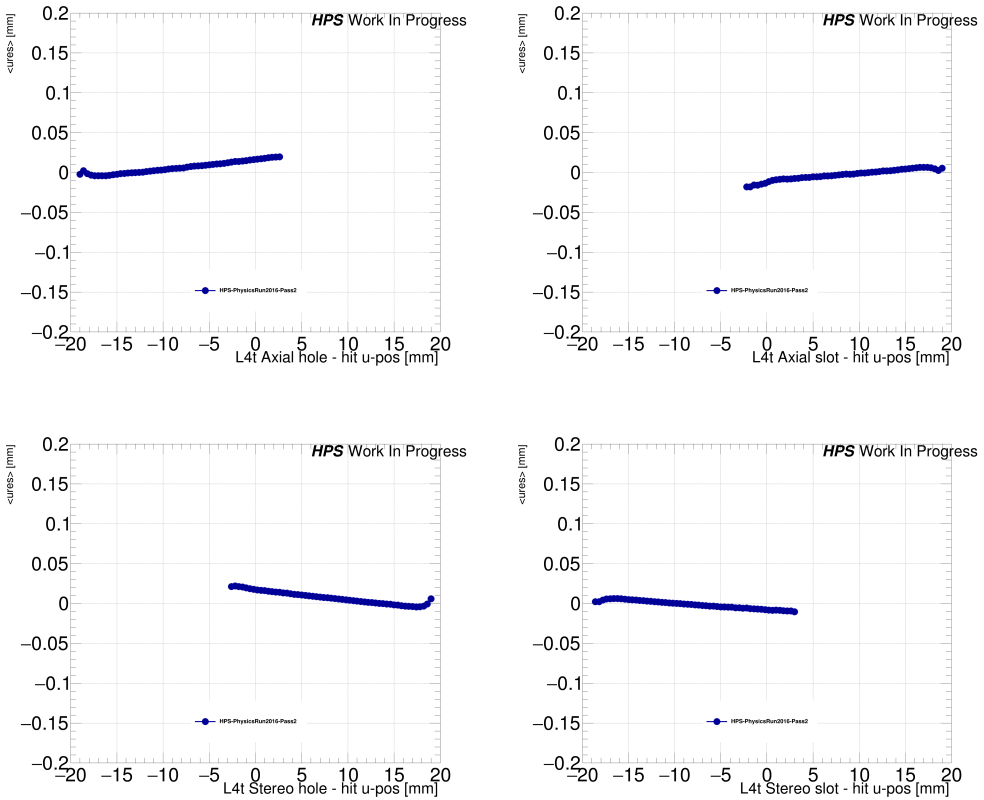
Unbiased Residuals against u

Looking for rotation around in-senstive direction (rv) or translation along normal (tw) misalignment (suggestions – not well understood)

Details not important, long story short:

Clear, linear trends in $\langle ures \rangle$ vs u plots. Different sensors have different trends but they all have a trend.

Layer 4 Top sensors shown to the right, other layers are included in tar-ball uploaded on the agenda.



Potential Discussions

$\langle ures \rangle$ vs u trend

What causes this? How to resolve this?

Look through other plots

I could scroll through all of the plots available in the uploaded tar-ball.

Questions

Known Bugs on hps-java:master

1. Second Memory Leak: Tom found a two more edge cases that the GBL-bound objects were allocating and were not getting cleaned up: (a) the GblPoint objects were being copied into the GblTrajectory and then not cleaned and (b) if the trajectory did not pass the χ^2 cut to be written to the output file, it was not being cleaned. ▸ PR 958 in hps-java

Bugs Patched Since Last Time

1. Including odd-number-hit KF tracks. Previous cut throwing away these tracks to achieve closer parity with ST tracks was removed.
2. Composite Align Bracket Placement. PF noticed the closing bracket on composite align was much too late, moving it up to where it is supposed to be allowed for running on 2016 data where we don't use the "alignable detector elements".

Record of Related Bugs Patched on hps-java:master

1. First Memory Leak: PF deduced that java was not cleaning up the GBL trajectories created by the C++ library and accessed via JNA. This was causing the alignment driver to eventually crash the program on systems with a "small" memory allocation for the JVM.
2. Unintentionally Dynamic Cut: The way "hits" are counted is different between ST and KF tracks. ST tracks use two sensors to construct 3D points. This means whenever we want at least N hits in an ST track, the same cut for KF tracks is $2N$. The alignment driver was mistakenly multiply the cut by 2 *on each track* eventually overflowing and returning to a cut of 0. This meant a lot of poor, low-hit-count tracks were being included in alignment when attempting to use KF.
3. Null Trajectories: Sometimes GBL fails to construct a trajectory from a track. This seems to be happening with some low-quality KF tracks, but I need to investigate more. For now, I simply avoid computing the residuals if the GBL trajectory fails to construct so the entire run doesn't crash.