Re-Alignment of 2016 with KF-based Tracks

Tom Eichlersmith

he/him/his

University of Minnesota

eichl008@umn.edu

August 7, 2023

Conclusion

KF-based Alignment is Valid

Full alignment iterations from no-constants and from ST-based alignment constants were able to be done, provide insight, and minimize the χ^2-function.

The 2016 Detector is Good

No noticeable improvements were found - while we could lower the (already small) residuals, this procedure did not improve the momentum resolution in any meaningful way.

- Alignment from No-Constants: ▸ talk at last Ana-Recon workshop
- Alignment from ST-based Constants: this talk

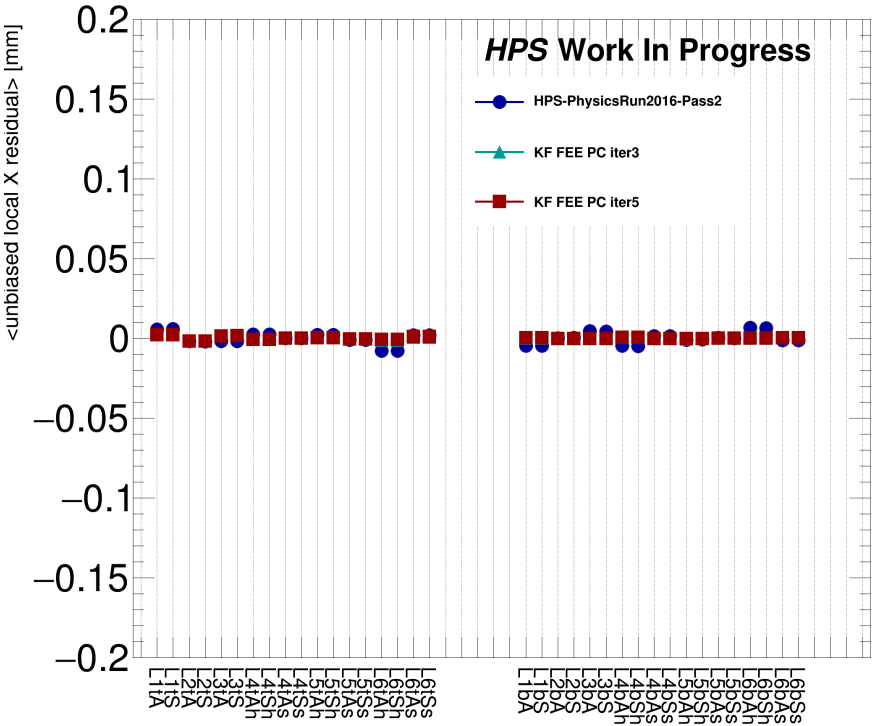
How?

0. Norman produced FEE-skim of 2016 run 7800
   `/sdf/group/hps/data/physrun2016/skim/goldenFees`
1. hps-java and GBL setup
2. hps-java-5.1-74-g4f599a3 The last alignment-related update to hps-java
3. hps-mc:9f6044b A few patches to hps-mc to support 2016 parameters
4. hps-align:v0.3.0 Plotting library

Vocab

- **KF FEE PC** – momentum-constraint (PC) on Full Energy Electrons (FEE) using Kalman Filter (KF) tracks
  - Cut out all tracks not falling within $2.0\text{GeV} < |p| < 2.6\text{GeV}$ and force those tracks to have $|p| = 2.3\text{GeV}$
- **iter3** – *on top* of paper detector, do three alignment iterations floating `tu` for stereo sensors in layers 4, 5, and 6
- **iter5** – two more iterations floating `tu` and `rw` for stereo sensors in layers 4, 5, and 6

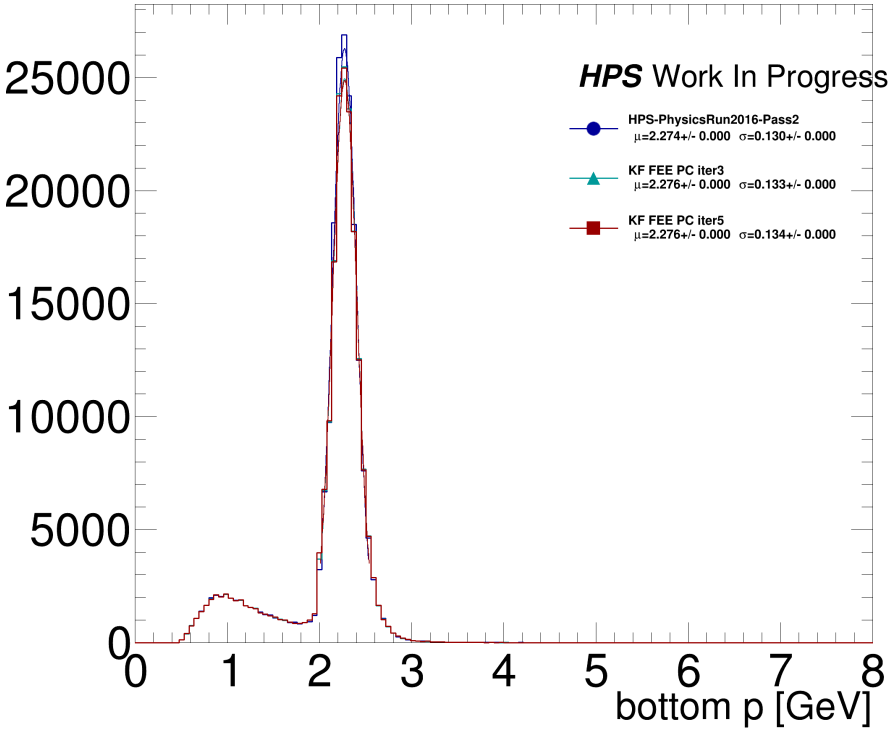
In General, Alignment Constants don't do much...



- Seemingly no trends in residuals across layers
- Pretty tight around zero
- Very similar to what we saw last time *without* using the current detector as base

Let's look in more detail.

No Effect on Momentum Resolution



Detector	μ [GeV]	σ [GeV]
HPS-PhysicsRun2016-Pass2	2.274	0.130
KF FEE PC iter3	2.276	0.133
KF FEE PC iter5	2.276	0.134

Example p in bottom – representative of p distributions with other selections.

- PC does pull average momentum closer to constraint
- Resolution value worsens slightly

Is that it?

...kinda yea...

Apply Beam-Spot Constraint as well?

Did a few iterations on top of these results and did not see improvement of resolution in momentum or vertex.

Sure constraint is powerful enough?

The default for seedPrecision is 10^4 which corresponds (roughly) to 10MeV of momentum precision, so - *yes* - the constraint was powerful enough.

Constrain to Higher Momentum to "drag" Momentum Up?

Could try this but I have no expectation that it would improve the resolution at all.

Float non-stereo and/or different layers?

Started with that and the results *worsened* the momentum resolution.

Questions

Record of Related Bugs Patched 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
2. Including odd-number-hit KF tracks. Previous cut throwing away these tracks to achieve closer parity with ST tracks was removed.
3. 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

4. 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.
5. 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.
6. 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.