HELIX INTERSECTION BUG?

MIRIAM DIAMOND

NOV 26 2017

HELIX INTERSECTION BUG?

hps-tracking ▸ src/main/java ▸ org.hps.recon.tracking ▸ TrackUtils ▸ getHelixPlaneIntercept(HelicalTrackFit, Hep3Vector, Hep3Vector, double, double) : Hep3Vector

This iterative calculation (for finding helix intercept with tilted sensor plane) called by:

- lcsim-tracking ▸ src/main/java ▸ org.lcsim.recon.tracking.seedtracker ▸ HelixFitter ▸ FitCandidate
  - hps-tracking ▸ src/main/java ▸ org.hps.recon.tracking ▸ MultipleScattering ▸ FindScatters
    - hps-tracking ▸ src/main/java ▸ org.hps.recon.tracking ▸ MultipleScattering ▸ FindHPSScatterPoints
      - hps-tracking ▸ src/main/java ▸ org.hps.recon.tracking ▸ MultipleScattering ▸ getHelixIntersection (*)

- hps-tracking ▸ src/main/java ▸ org.hps.recon.tracking.gbl ▸ MakeGblTracks ▸ makeStripData
- hps-tracking ▸ src/main/java ▸ org.hps.recon.tracking.gbl ▸ GBLOutput ▸ printGBL

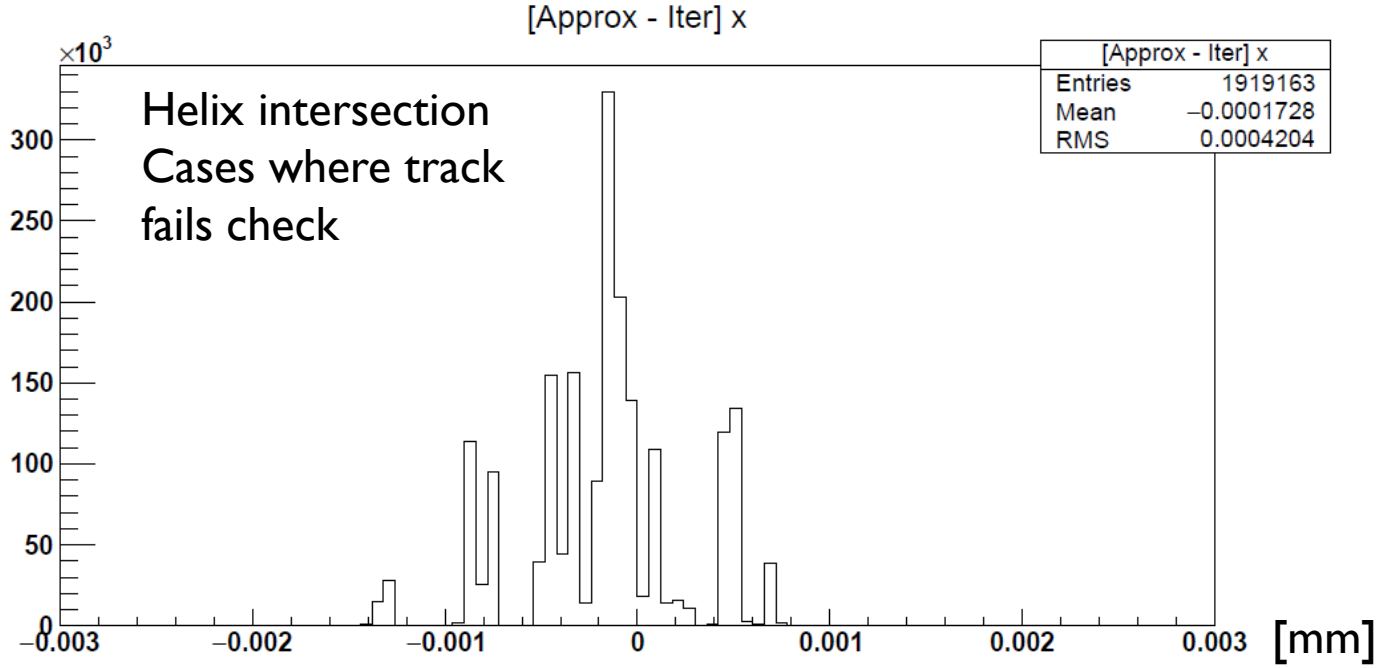
But only getHelixIntersection (*) contains a protective check:

```
// TODO Catch special cases where the incidental iteration procedure seems to fail
if (Math.abs(helix.R()) < 2000 && Math.abs(helix.dca()) > 10.0) {
    if (_debug) {
        System.out.printf("%s: momentum is low (p=%f,R=%f,B=%f) and d0 is big (d0=%f), skip the iterative calculation\n"
```

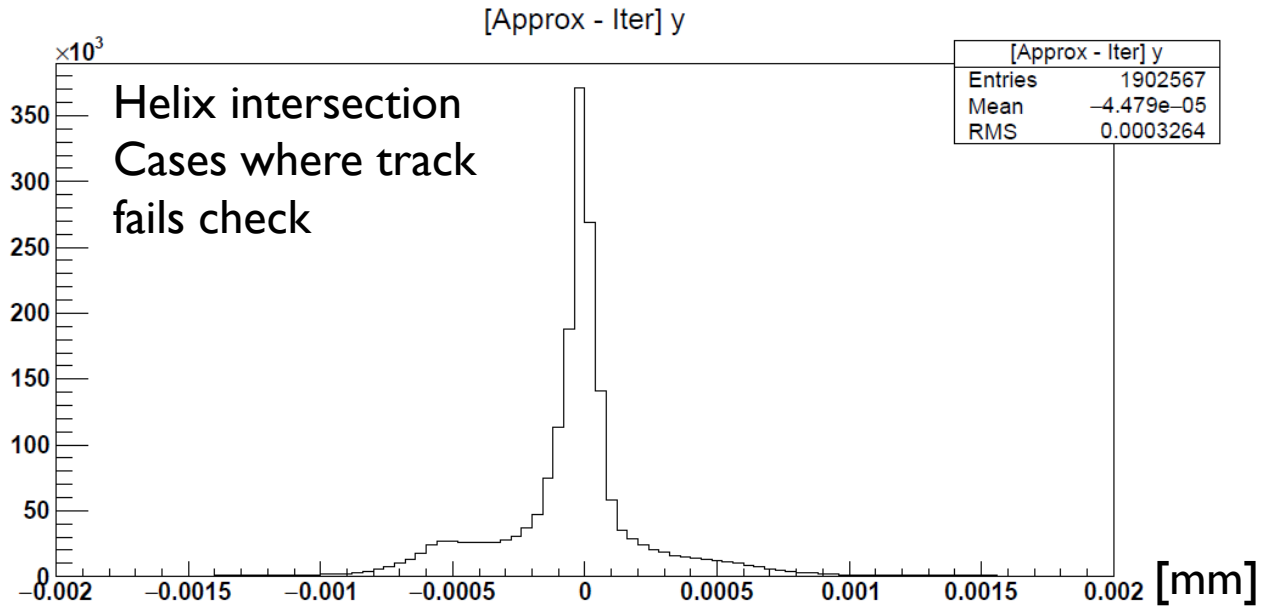
What's up with this check? Does its absence in other calling methods ever matter?

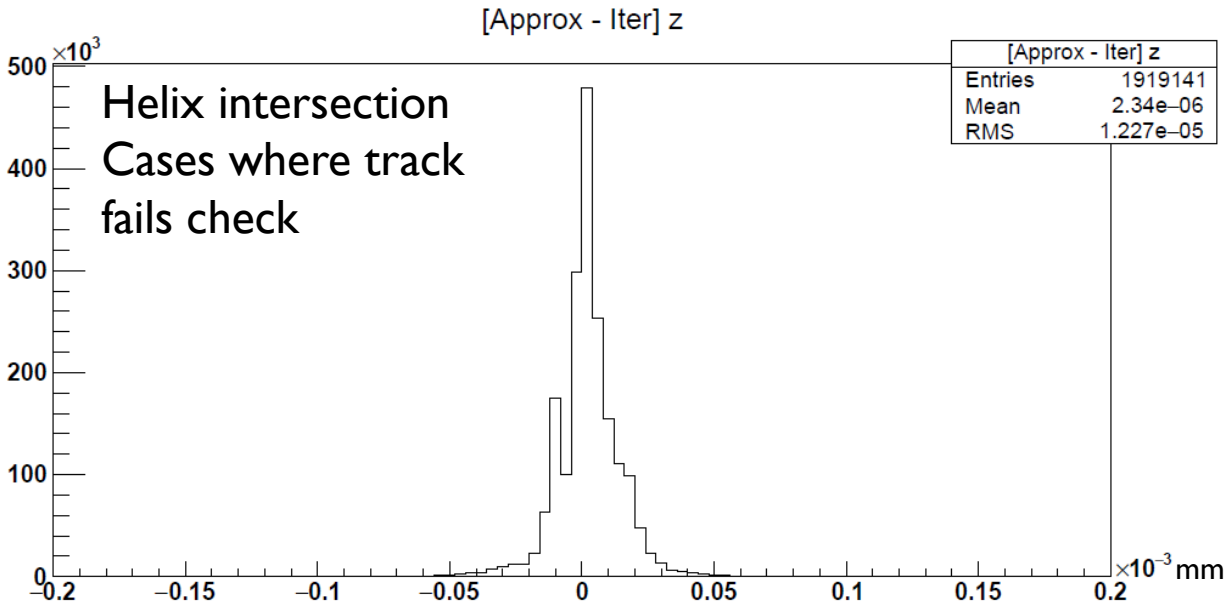
HELIX INTERSECTION BUG?

- If this check fails, getHelixIntersection uses "approximate" calculation rather than trying iterative method
- What if we remove this check? (Tried ~11K tracks from Run 5772)
  - Didn't crash, even though many of these tracks failed the check
  - Plotted iterative intersection position minus "approximate" , for tracks that failed. Small difference, so iterative result seems reasonable!



HELIX INTERSECTION BUG?





- So if this check ever matters, it must only be very rarely!
- Good news: its sometimes-absence isn't a significant bug
- Bad news: not sure why it was included in the first place
  - Looked in source of algorithm (Avery CBX 98–39), no stated limits on applicability

"MATRIX IS SINGULAR" BUG

- Is it the cause of Norman's bug https://github.com/JeffersonLab/hps-java/issues/243 ?
  - Offending track's HelicalTrackFit:

    d0= 106.8580318055611 7

    phi0= -0.8142691936809715

    curvature= -0.0031131460914910 68

    z0= 1.0396211176748624

    tanLambda= -0.020735453823318976
  - Would indeed be caught by the large-d0 / low-R check; crash only happens in absence of the check
  - But, a much more specialized check would work to catch it …

"MATRIX IS SINGULAR" BUG

```
public static Hep3Vector getHelixPlaneIntercept(HelicalTrackFit helfit, HelicalTrackStripGbl strip, double bfield) {
    Hep3Vector point_on_plane = strip.origin();
    Hep3Vector unit_vec_normal_to_plane = VecOp.cross(strip.u(), strip.v());// strip.w();
    double s_origin = HelixUtils.PathToXPlane(helfit, strip.origin().x(), 0., 0).get(0);
    Hep3Vector intercept_point = getHelixPlaneIntercept(helfit, unit_vec_normal_to_plane, point_on_plane, bfield, s_origin);
    return intercept_point;
}

public static Hep3Vector getHelixPlaneIntercept(HelicalTrackFit helfit, Hep3Vector unit_vec_normal_to_plane,
        Hep3Vector point_on_plane, double bfield, double initial_s) {
    WTrack wtrack = new WTrack(helfit, bfield); //
    if (initial_s != 0)
        wtrack.setTrackParameters(wtrack.getHelixParametersAtPathLength(initial_s, B));
    Hep3Vector intercept_point = wtrack.getHelixAndPlaneIntercept(point_on_plane, unit_vec_normal_to_plane, B);
```

Value in red box is NaN

Core iterative algorithm

"MATRIX IS SINGULAR" BUG

lcsim-tracking ▸ src/main/java ▸ org.lcsim.fit.helicaltrack ▸ HelixUtils ▸ PathToXPlane ▸ PathCalc

- Calculation of value in red box: (helfit is HelicalTrackFit)

```
double x = point_on_plane.x();
double y = helfit.yc() + Math.signum(helfit.R()) * Math.sqrt(helfit.R() * helfit.R() - Math.pow(x - helfit.xc(), 2));
double phi1 = Math.atan2(helfit.y0() - helfit.yc(), helfit.x0() - helfit.xc());
double phi2 = Math.atan2(y - helfit.yc(), x - helfit.xc());
double dphi = phi2 - phi1;
if (dphi > Math.PI)
    dphi -= 2. * Math.PI;
if (dphi < -Math.PI)
    dphi += 2. * Math.PI;
double s = -1.0 * helfit.R() * dphi;
```

point_on_plane : [705.49, -29.009, -32.166]
helfit: x0 77.709743 , y0 73.347357 , xc 311.307572 , yc 293.831721 , R -321.218462
y = NaN , phi1 = -2.385066 , phi2 = NaN , dphi = NaN

Problem: Quantity in orange box is negative

Suggests using a more specialized check: x-xc > R