THREE POINT HELIX CHECK ERRORS

MIRIAM DIAMOND

JULY 24 2017

github issue 126

DOWN THE RABBIT-HOLE

- org.lcsim.recon.tracking.seedtracker.FastCheck ▶ ThreePointHelixCheck triplet-finding for track seeds
  - For each of the 3 hits, calculates contribution to z *error*

    ```
    dztot += _nsig * Math.sqrt(hit.getCovMatrix()[5]);
    ```

    lcsim z
    (hps y)

  - Then

    ```
    //  Add multiple scattering error here - for now, just set it to 1 mm
    dztot += 1.;  dztot += _nsig * MSerror;
    ```

  - Compares *total z error* to *(predicted – actual) z position* of middle hit

    ```
    if (Math.abs(zpred - z[1]) > dztot) return false;
    ```

- Implementing a proper MSerror makes ~no difference to tracking output. Why?
- Because *even without any MSerror*, dztot is far bigger than zpred-z[1], meaning no seeds get thrown out here anyway

DOWN THE RABBIT-HOLE

- Why is this potentially a problem?
  - We do want to avoid throwing out decent candidates at seeding stage, but if we're not throwing out any seeds, we might as well not bother with this check at all
  - Intuitively, dztot *should* be dominated by MSerror. But it *is* dominated by hit errors.
- Why are the hit errors so big?

1. Big _nsig

```
_nsig = Math.sqrt(strategy.getMaxChisq());
```

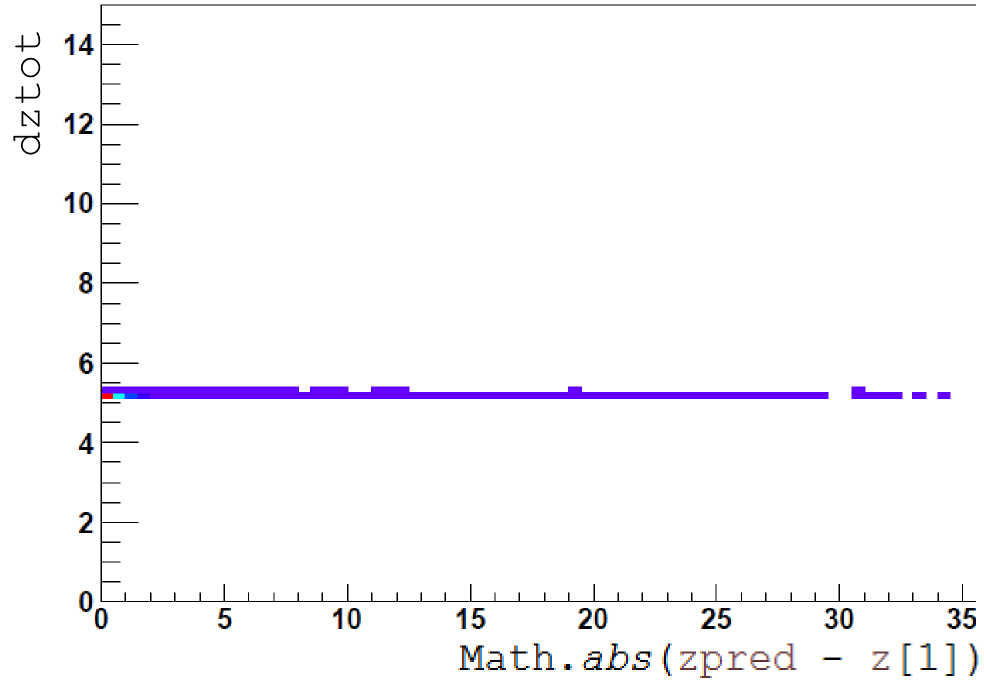
MaxChisq = 100
_nsig = 10

- Decouple _nsig from Strategy.MaxChisq
- MSerror has greater effect at lower _nsig

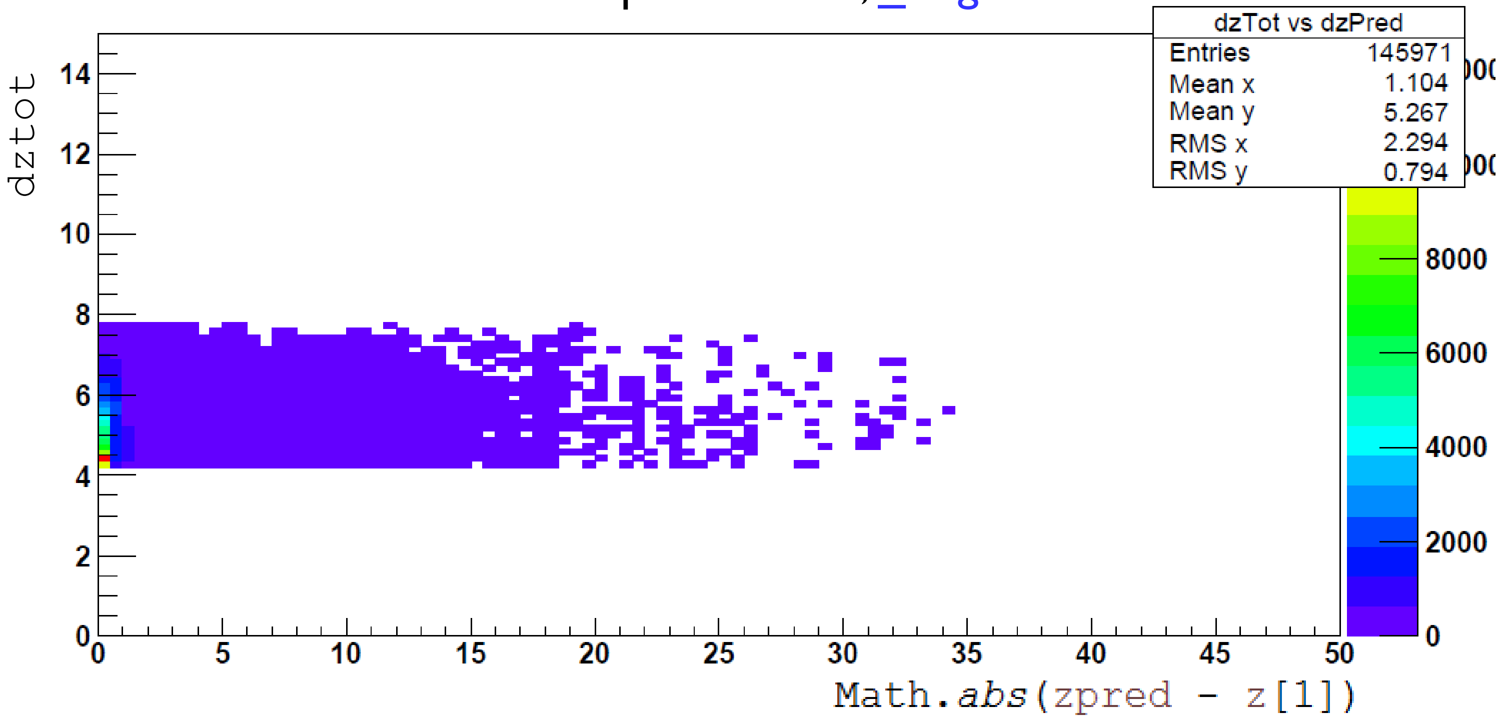
FIXING THE CODE

- But even at _nsig=1, dztot is still big

MSerror = 1.0, _nsig = 1



Proper MSerror, _nsig=1



DOWN THE RABBIT-HOLE

2. Big hit.getCovMatrix()[5]

Not corrected for track direction

- Typically ~2
- For 3 hits, added linearly: $3\sqrt{2} \approx 4.5$
- Typical uncorrected covariance matrix:

$$\begin{pmatrix} 0.3 & -11.5 & -0.6 \\ -11.5 & 400.5 & 20.0 \\ -0.6 & 20.0 & \boxed{2.0} \end{pmatrix}$$

Typical corrected covariance matrix:

$$\begin{pmatrix} 6e-5 & -2e-3 & -1e-4 \\ -2e-3 & 0.08 & 4e-3 \\ -1e-4 & 4e-3 & \boxed{2e-4} \end{pmatrix}$$

DOWN THE RABBIT-HOLE

lcsim-tracking ‣ src/main/java ‣ org.lcsim.fit.helicaltrack ‣ HitUtils ‣ CovarianceFromOrigin(HelicalTrackStrip, HelicalTrackStrip) : SymmetricMatrix

```
public static SymmetricMatrix CovarianceFromOrigin(HelicalTrackStrip strip1, HelicalTrackStrip strip2) {
    //  Assume the particle is coming from the origin, so x2 = gamma * x1
    //  gamma = Origin2 . w_hat / Origin1 . w_hat
    double gamma = VecOp.dot(strip2.origin(), strip2.w()) / NonZeroDotProduct(strip1.origin(), strip1.w());
    //  Calculate the seperation between the sensor planes
    double separation = SensorSeperation(strip1, strip2);
    //  Calculate v1hat . u2hat, which is equivalent to sin(alpha) where alpha is the stereo angle
    double salpha = getSinAlpha(strip1, strip2);
    //  Calculate the scale factor:  factor = (1 + gamma)^2 / 4 * sin^2(alpha)
    double factor = (1 + gamma)*(1 + gamma) / (4. * salpha*salpha);
    //  Calculate v * v^T for strips 1 and 2
    Matrix v1 = v2m(strip1.v());
    Matrix v2 = v2m(strip2.v());
    Matrix v1v1t = MatrixOp.mult(v1, MatrixOp.transposed(v1));
    Matrix v2v2t = MatrixOp.mult(v2, MatrixOp.transposed(v2));
    //  Find measurement uncertainties for strips 1 and 2
    double du1 = strip1.du();
    double du2 = strip2.du();
```

Questionable du() values: issue 135

DOWN THE RABBIT-HOLE

lcsim-tracking ▸ src/main/java ▸ org.lcsim.fit.helicaltrack ▸ HitUtils ▸ CovarianceFromOrigin(HelicalTrackStrip, HelicalTrackStrip) : SymmetricMatrix

```
//  Calculate the uncertainty in the unmeasured coordinate due to not knowing the track direction
//  by assuming phat . u has an uncertainty of 2/sqrt(12) so dv = 2 / sqrt(12) * seperation / sin(alpha)
double dv = Math.abs(2. * separation / (Math.sqrt(12) * salpha));
//  Don't let dv by greater than the strip length / sqrt(12)
double dv1 = Math.min(dv, (strip1.vmax()-strip1.vmin()) / Math.sqrt(12.));
double dv2 = Math.min(dv, (strip2.vmax()-strip2.vmin()) / Math.sqrt(12.));
//  Calculate the covariance matrix.
//    From resolution:  cov = factor * (v2 * v2^T * du1^2 + v1 * v1^T * du2^2)
//    From direction:                + (v1 * v1^T * (dv1/2)^2 + v2 * v2^T * (dv2/2)^2
Matrix cov1 = MatrixOp.mult(factor * du2*du2 + 0.25 * dv1*dv1, v1v1t);
Matrix cov2 = MatrixOp.mult(factor * du1*du1 + 0.25 * dv2*dv2, v2v2t);
Matrix cov = MatrixOp.add(cov1, cov2);
return new SymmetricMatrix(cov);
```

phat . u uncertainty
~5x too big?

DOWN THE RABBIT-HOLE

3. Adding up all the contributions linearly
   - Should we add them in quadrature instead?

Options:

A. "Make seeding cuts great again" to throw out some seeds
   - Look at distributions of (phat.u) to get proper uncertainty for it
   - Revisit strip.du() values (issue 135)
   - Perform dedicated studies to decide value of _nsig

B. Decide it's OK to keep all seeds
   - Simply eliminate ThreePointHelixCheck since it's not accomplishing anything