



TrackLineFitAlg



Bruce Baller
September 10, 2014

Issues & “Solution”

- ▶ Standard track fit algorithms are slow (I assume)
 - ▶ Hits sorted along a trajectory, etc
- ▶ A lot of machinery for short tracks
- ▶ Don't appear to be packaged as standalone algorithms
- ▶ Desirable to do a local fit on a track trajectory to split tracks at kinks
 - ▶ Is this done in Kalman filter fits?
- ▶ TrackLineFitAlg fits a collection of hit parameters to a 3D line using linear algebra
 - ▶ No hit ordering required, presumably fast

Fitting a line in 3D

- ▶ Want to determine the track position and direction (6 parameters) but only 4 unknowns
 - ▶ Can specify an intersection point with a coordinate plane
 - ▶ TrackLineFitAlg : $X = X \text{ Origin}$
 - ▶ Direction vector is unit normalized
 - ▶ TrackLineFitAlg: $(1, dY/dX, dZ/dX)$
- ▶ Why X?
 - ▶ Resolution in X direction is 0.2 – 0.4 mm vs ~1 mm in Y and Z
 - ▶ Simple parameterization of (Y, Z) vs X

Definitions

- ▶ Define a track vector in (X, Y, Z) coordinate system
 - ▶ $T = (Y_0, Z_0, dY/dX, dZ/dX)$
 - ▶ Set X Origin = 0 here for legibility...
 - ▶ Track components
 - ▶ $T_x = X$
 - ▶ $T_y = Y_0 + X (dY/dX)$
 - ▶ $T_z = Z_0 + X (dZ/dX)$
- ▶ Define a hit vector in the wire coordinate system of a plane
 - ▶ W_i is just the wire number of the hit in plane i
- ▶ Track fit: Set $W = A T$ where A is a matrix that transforms the track components into the wire coordinate system

Solution

- ▶ $W_i - W_{i0} = Y \cos(\theta_i) + Z \sin(\theta_i)$
 - ▶ where θ_i is the angle orthogonal to the wire orientation in plane i
 - ▶ W_{i0} is a wire offset in plane i
 - ▶ Get these components from ChannelMapStandardAlg using a (new) WireCoordinate method adapted from IntersectionPoint

$$\begin{array}{c} \text{N Hits} \\ \downarrow \end{array} \underbrace{\begin{bmatrix} W_1 - W_{i0} \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}}_{\underline{W}} = \underbrace{\begin{bmatrix} \cos(\theta_i) & \sin(\theta_i) & X\cos(\theta_i) & X\sin(\theta_i) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{\underline{A (4 \times N)}} \underbrace{\begin{bmatrix} Y0 \\ Z0 \\ dY/dX \\ dZ/dX \end{bmatrix}}_{\underline{T}}$$

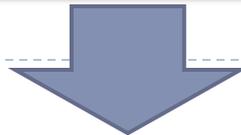
- ▶ Solve for T

TrackLineFitAlg::TrkLineFit

User passes vector of (X, WireID) pairs and an X origin for the fit.
Alg returns TVector3 position and direction and Chisq/DOF

```
void TrackLineFitAlg::TrkLineFit(  
    std::vector<std::pair<double, geo::WireID>>& hits, float XOrigin,  
    TVector3& Pos, TVector3& Dir, float& ChiDOF)  
{  
    // Linear fit using X as the independent variable. Hits to be fitted  
    // are passed in the hits vector in the pair form (X, WireID). The  
    // fitted track position at XOrigin is returned in the Pos vector.  
    // The direction cosines are returned in the Dir vector.  
    //  
    // SVD fit adapted from $ROOTSYS/tutorials/matrix/solveLinear.C  
    // Fit equation is  $w = A(X)v$ , where  $w$  is a vector of hit wires,  $A$  is  
    // a matrix to calculate a track projected to a point at  $X$ , and  $v$  is  
    // a vector ( $Y_0, Z_0, dY/dX, dZ/dX$ ).  
    //  
    // Note: The covariance matrix should also be returned  
    // B. Baller August 2014  
  
    // assume failure  
    ChiDOF = 9999;  
  
    if(hits.size() < 5) return;  
  
    const unsigned int nvars = 4;  
    unsigned int npts = hits.size();
```

Return crazy Chisq if anything bad happens



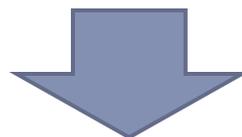
```

TMatrixD A(npts, nvars);
// vector holding the Wire number
TVectorD w(npts);
unsigned short ninpl[3] = {0};
unsigned short nok = 0;
unsigned short iht, cstat, tpc, ipl;
double x, cw, sw, off;
for(iht = 0; iht < hits.size(); ++iht) {
  cstat = hits[iht].second.Cryostat;
  tpc = hits[iht].second.TPC;
  ipl = hits[iht].second.Plane;
  // get the wire plane offset
  off = geom->WireCoordinate(0, 0, ipl, tpc, cstat);
  // get the "cosine-like" component
  cw = geom->WireCoordinate(1, 0, ipl, tpc, cstat) - off;
  // the "sine-like" component
  sw = geom->WireCoordinate(0, 1, ipl, tpc, cstat) - off;
  x = hits[iht].first - XOrigin;
  A[iht][0] = cw;
  A[iht][1] = sw;
  A[iht][2] = cw * x;
  A[iht][3] = sw * x;
  w[iht] = (hits[iht].second.Wire - off);
  ++ninpl[ipl];
  // need at least two points in a plane
  if(ninpl[ipl] == 2) ++nok;
}

// need at least 2 planes with at least two points
if(nok < 2) return;

```

Fit at X origin



```

// need at least 2 planes with at least two points
if(nok < 2) return;

TDecompSVD svd(A);
bool ok;
TVectorD tVec = svd.Solve(w, ok);

ChiDOF = 0;
double ypr, zpr, diff;
for(iht = 0; iht < hits.size(); ++iht) {
  cstat = hits[iht].second.Cryostat;
  tpc = hits[iht].second.TPC;
  ipl = hits[iht].second.Plane;
  off = geom->WireCoordinate(0, 0, ipl, tpc, cstat);
  cw = geom->WireCoordinate(1, 0, ipl, tpc, cstat) - off;
  sw = geom->WireCoordinate(0, 1, ipl, tpc, cstat) - off;
  x = hits[iht].first - XOrigin;
  ypr = tVec[0] + tVec[2] * x;
  zpr = tVec[1] + tVec[3] * x;
  diff = ypr * cw + zpr * sw - (hits[iht].second.Wire - off);
  ChiDOF += diff * diff;
}

float werr2 = geom->WirePitch() * geom->WirePitch();
ChiDOF /= werr2;
ChiDOF /= (float)(npts - 4);

double norm = sqrt(1 + tVec[2] * tVec[2] + tVec[3] * tVec[3]);
Dir[0] = 1 / norm;
Dir[1] = tVec[2] / norm;
Dir[2] = tVec[3] / norm;

Pos[0] = XOrigin;
Pos[1] = tVec[0];
Pos[2] = tVec[1];
} // TrkLineFit()

```

Use ROOT Single Value
Decomposition

Calculate Chisq

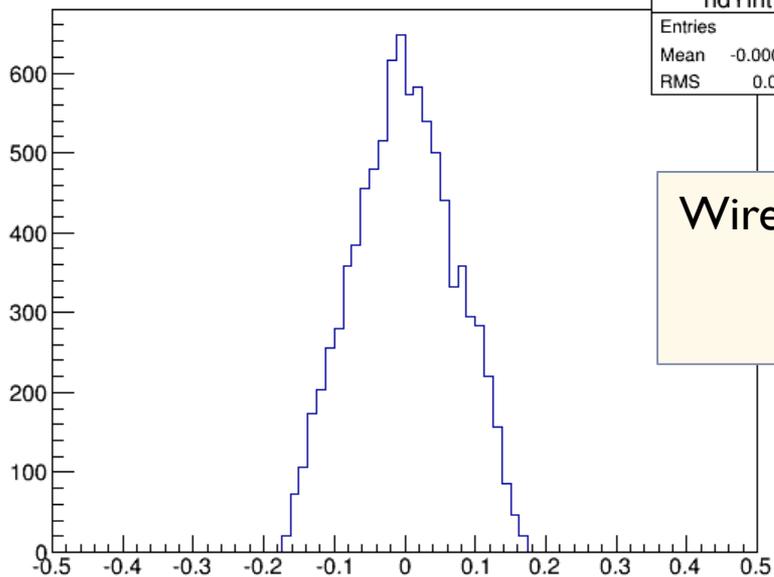
Weight Chisq by wire pitch

Put into TVector3s

Testing

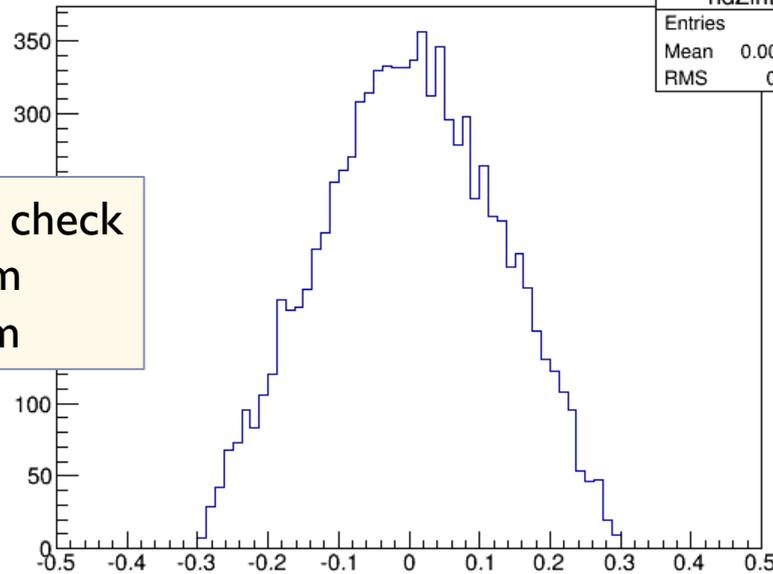
- ▶ **Generate 1000 3D lines – not Geant...**
 - ▶ True origin: $(50 + \delta, 0 + \delta, 20 + \delta)$, $\delta = \text{random}(0.0 - 1.0 \text{ cm})$
 - ▶ True direction: random θ, ϕ (0.1 – 1.5 radians)
 - ▶ Step along the line, calculate (X, Y, Z) and create a hit pair (X, WireID) in all 3 wire planes (ipl)
 - ▶ `Unsigned int wire[ipl] = (0.5 + geom->WireCoordinate(Y, Z, ipl, 0, 0))`
 - ▶ Check `WireCoordinate` using planes 0 and 1
 - ▶ `Geom->IntersectionPoint(wire[0], wire[1], 0, 1, 0, 0, Yint, Zint)`
 - ▶ Calculate $dYint = Yint - Y$, $dZint = Zint - Z$
 - ▶ Expect $\sim 3 \text{ mm} / \text{sqrt}(12)$
 - ▶ Set step size = 0.3 cm, nstp = number of steps
 - ▶ Call `TrkLineFit` with `XOrigin = True X origin`
 - ▶ Compare fitted position and direction with true values

dY_{int} (cm)



| hdYint | |
|---------|------------|
| Entries | 9000 |
| Mean | -0.0006431 |
| RMS | 0.07068 |

dZ_{int} (cm)



| hdZint | |
|---------|----------|
| Entries | 9000 |
| Mean | 0.001054 |
| RMS | 0.1231 |

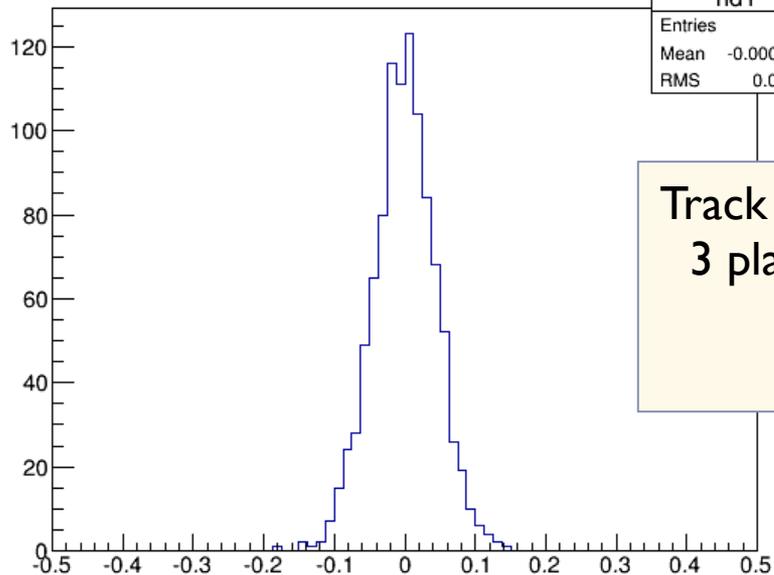
WireCoordinate check

$$\sigma_Y = 0.7 \text{ mm}$$

$$\sigma_Z = 1.2 \text{ mm}$$

$Y_{\text{int}} - Y_{\text{true}}$ (cm)

$Z_{\text{int}} - Z_{\text{true}}$ (cm)



| hdY | |
|---------|------------|
| Entries | 1000 |
| Mean | -0.0007472 |
| RMS | 0.04403 |

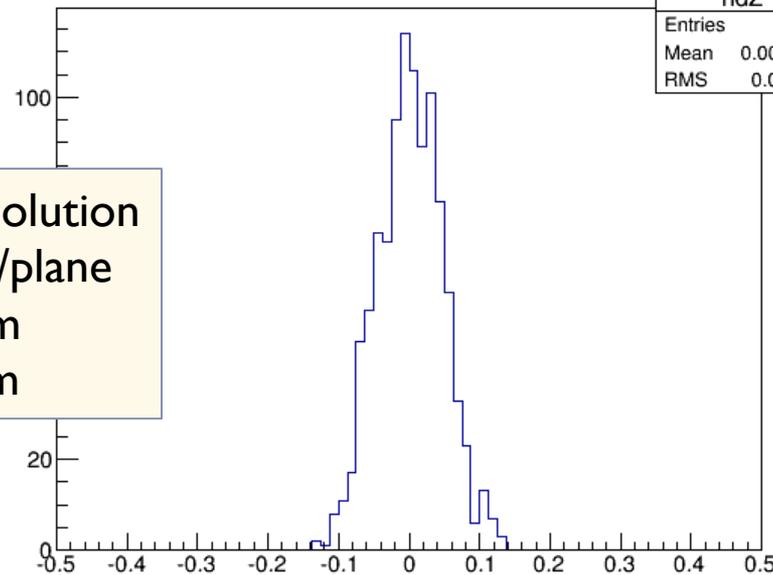
Track position resolution

3 planes x 9 hits/plane

$$\sigma_Y = 0.4 \text{ mm}$$

$$\sigma_Z = 0.5 \text{ mm}$$

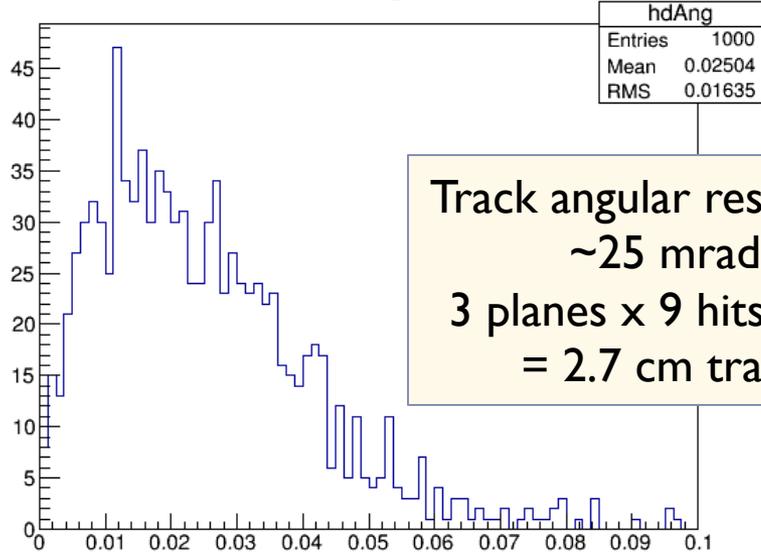
$Y_{\text{fit}} - Y_{\text{true}}$ (cm)



| hdZ | |
|---------|----------|
| Entries | 1000 |
| Mean | 0.001819 |
| RMS | 0.04545 |

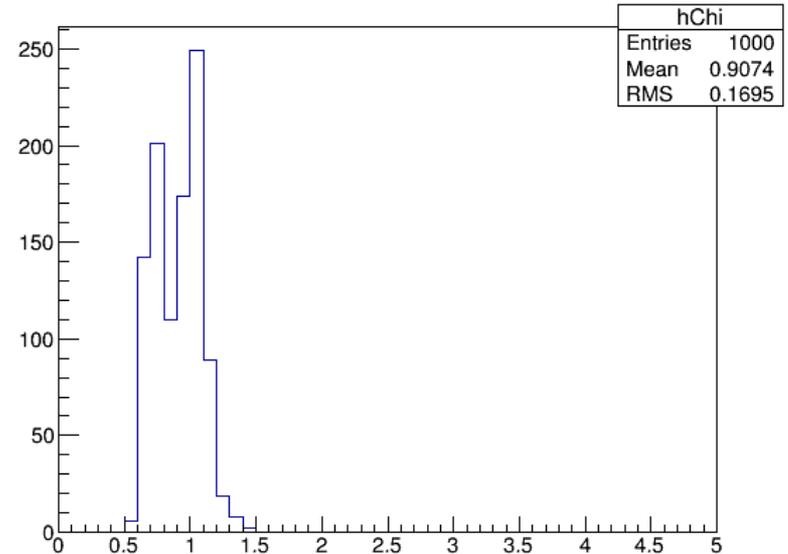
$Z_{\text{fit}} - Z_{\text{true}}$ (cm)

dAng

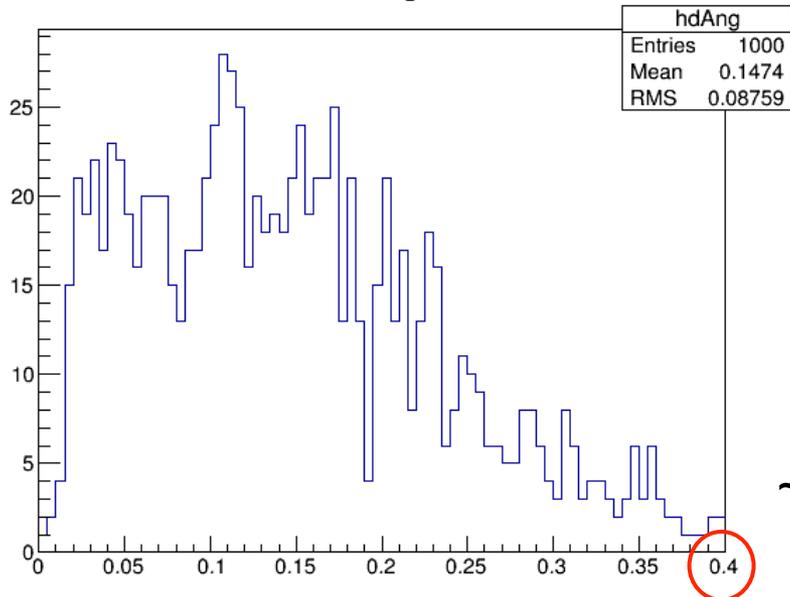


Track angular resolution
~25 mrad
3 planes x 9 hits/plane
= 2.7 cm track

ChiDOF

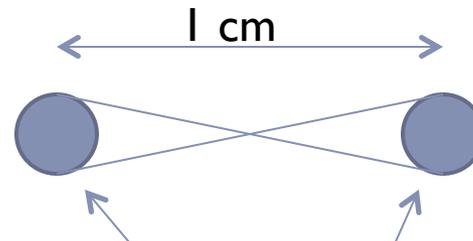


dAng



Track angular resolution
~150 mrad
3 planes x 3 hits/plane
= 1 cm track

~1mm



Expect angular resolution
1 mm / 5 mm ~
200 mrad

Note scale change

YZ error ellipses

Plans & Ruminations

- ▶ Simple code, works as expected with a unit(?) test
- ▶ Would like to return a covariance matrix
 - ▶ Need to figure out how to do this with SVD matrices
- ▶ A need for a polynomial fit, e.g. for short curly stopping tracks?
- ▶ An alternate (fast) fitter for cosmic ray tracks?