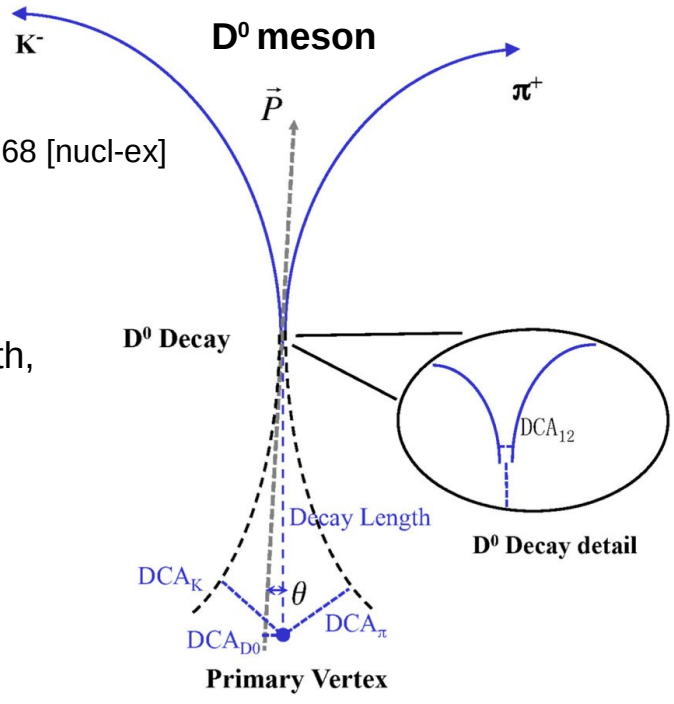
Secondary Vertexing

Shyam Kumar*, Annalisa Mastroserio, Domenico Elia
INFN Bari, Italy

Primary Vertex Reconstruction:

- **CentralTrackVertices**
- AdaptiveMultiVertexFinder (in development: Bishoy)

arXiv:1911.12168 [nucl-ex]



Primary vertex: allows access to the DCA of daughter tracks

Secondary vertex: allows access to more topological variables, e.g., decay length, pointing angle (cosθ), DCA_{D0} or $DCA_{\Lambda c}$ etc.

Topological Variables

$$\vec{dl} = \vec{SV} - \vec{PV}$$

$$\cos\theta = \frac{\vec{dl} \cdot \vec{p_{D0}}}{|\vec{dl}||\vec{p_{D0}}|}$$

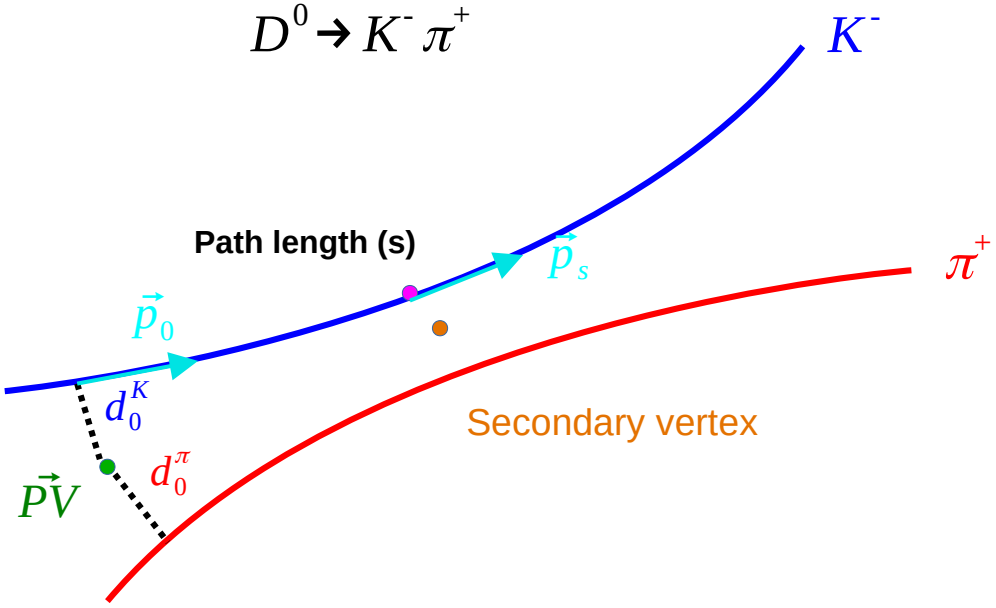
$$DCA_{D0} = |\vec{dl}| \sin\theta$$

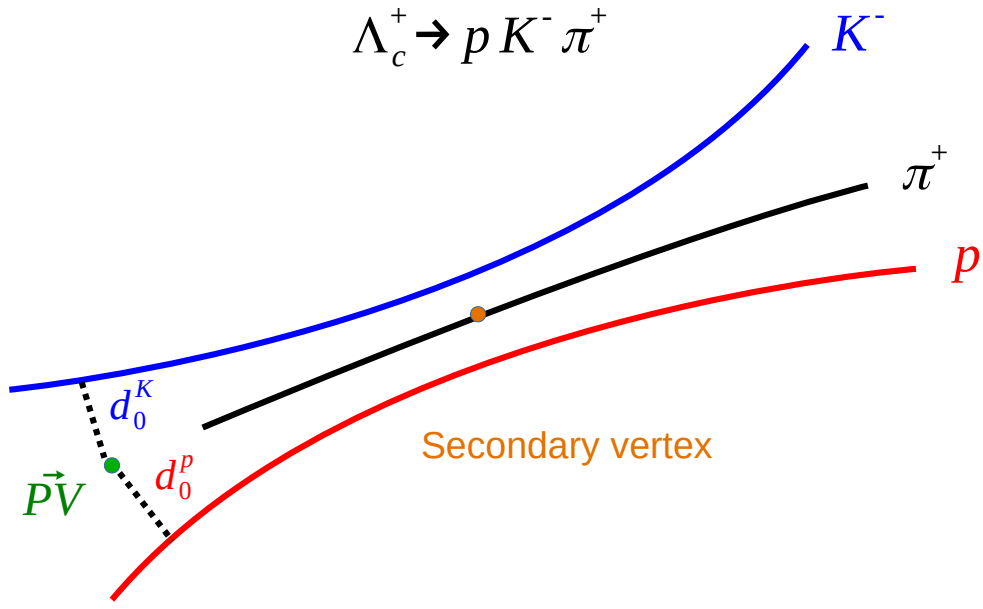
Invariant mass: $m_{D^0} = \sqrt{(E_{K^-} + E_{\pi^+})^2 - (\vec{p_{K^-}} + \vec{p_{\pi^+}})^2}$

Secondary vertex is important for the reconstruction of HF-hadrons

Secondary Vertex Reconstruction

- Perform the combinatorics between two tracks (D^0 meson) and three tracks for Λ_c^+ baryon Λc+ Reconstruction code
- Find the closest space point to the helices properly considering the track errors





Signal pairs: From HF-hadron decay

Bkg pairs: One of them is not from an HF-hadron decay

AdaptiveMultiVertexFinder and KFParticle for secondary vertex, as they account for proper track errors once available

Various approaches for secondary vertex reconstruction

Secondary Vertex Reconstruction (D^0)

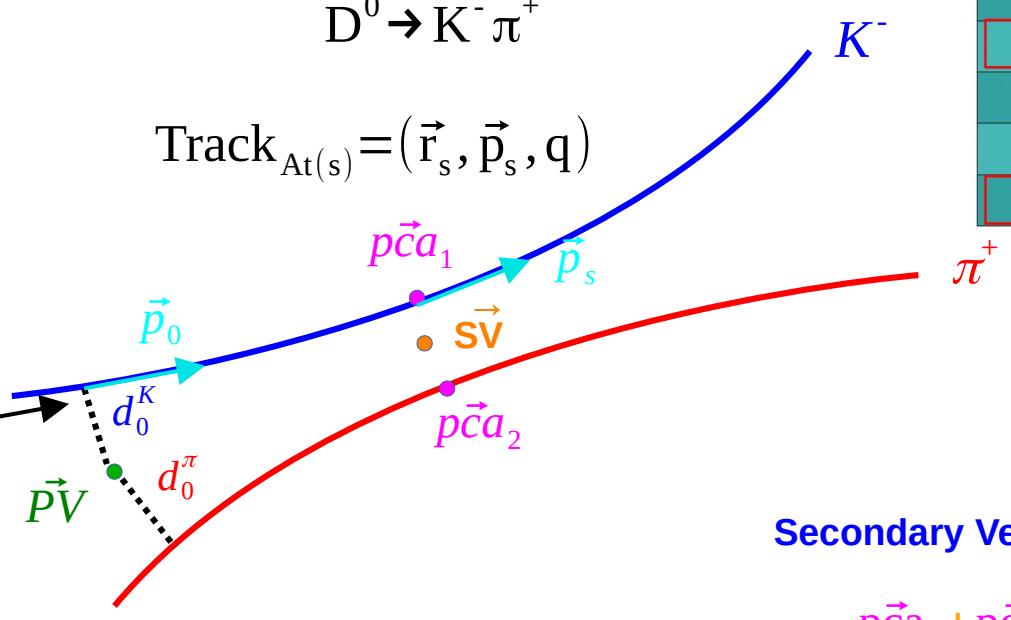
Approach 1 (Analytical ignoring tracking errors) from the STAR helix calculation

$$D^0 \rightarrow K^- \pi^+$$

$$\text{Track}_{\text{At(s)}} = (\vec{r}_s, \vec{p}_s, q)$$

s: path length

$$\text{Track}_{\text{DCA}} = (\vec{r}_0, \vec{p}_0, q)$$



Particle	Mass (GeV/c^2)	cτ (μm)
$D^\pm$	1.869	312
D^0	1.864	123
$B^\pm$	5.279	491
B^0	5.280	456
Λ_c^+	2.286	60

Secondary Vertex

$$\vec{SV} = \frac{\vec{pca}_1 + \vec{pca}_2}{2}$$

Vertex position SV = (v_x, v_y, v_z)

https://doi.org/10.1103/PhysRevLett.124.172301

STAR experiment

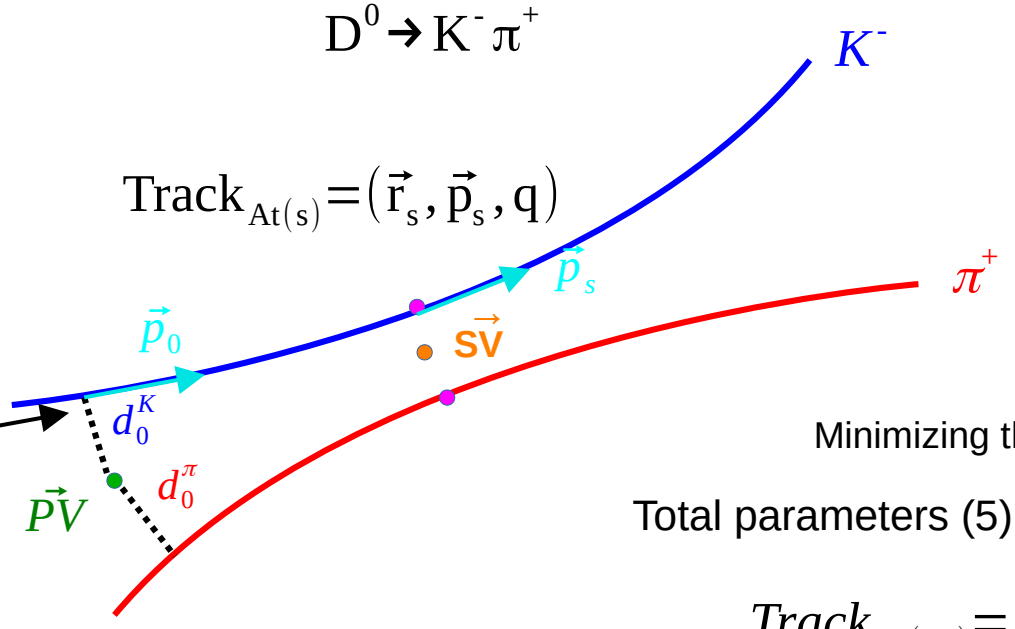
The $\Lambda_c^\pm$ decay vertex is reconstructed as the mid-point of the distance of closest approach (DCA) between the three daughter tracks. To improve separation of signal

Secondary Vertex Reconstruction (D^0)

Approach 2 (ignoring tracking errors)
Minimization of square of distance

Vertex position SV = (v_x, v_y, v_z) parameters

$D^0 \rightarrow K^- \pi^+$



$Track_{At(s)} = (\vec{r}_s, \vec{p}_s, q)$

s: path length

$Track_{DCA} = (\vec{r}_0, \vec{p}_0, q)$

Minimizing the distance

Total parameters (5): (v_x, v_y, v_z, s_1, s_2)

$$Track_{At(s1)} = (\vec{r}_{s1}, \vec{p}1, q1)$$
$$Track_{At(s2)} = (\vec{r}_{s2}, \vec{p}2, q2)$$

From fit:

Parameters

1. $\vec{SV} = \frac{\vec{pca}_1 + \vec{pca}_2}{2}$

2. $\vec{SV} = (v_x, v_y, v_z)$

Minimize $d^2 = (\vec{r}_{s1} - \vec{v})^2 + (\vec{r}_{s2} - \vec{v})^2$

Secondary Vertex Reconstruction (D^0)

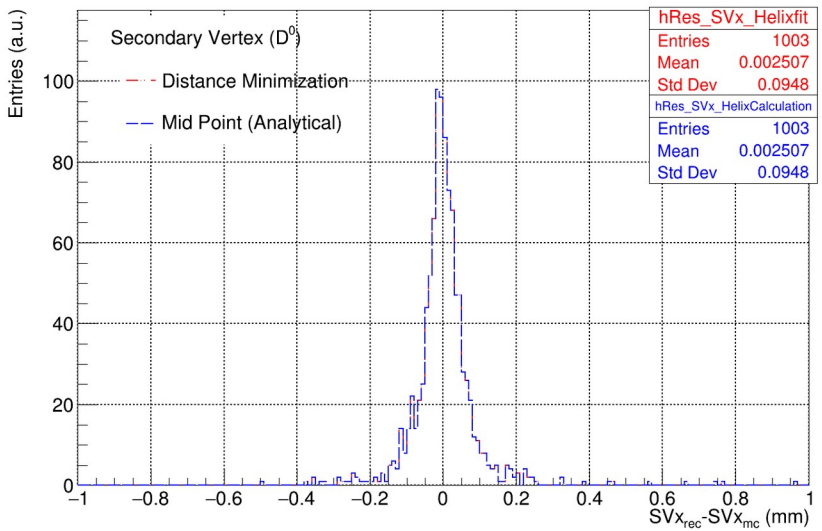
Code to Minimize the square of the distance

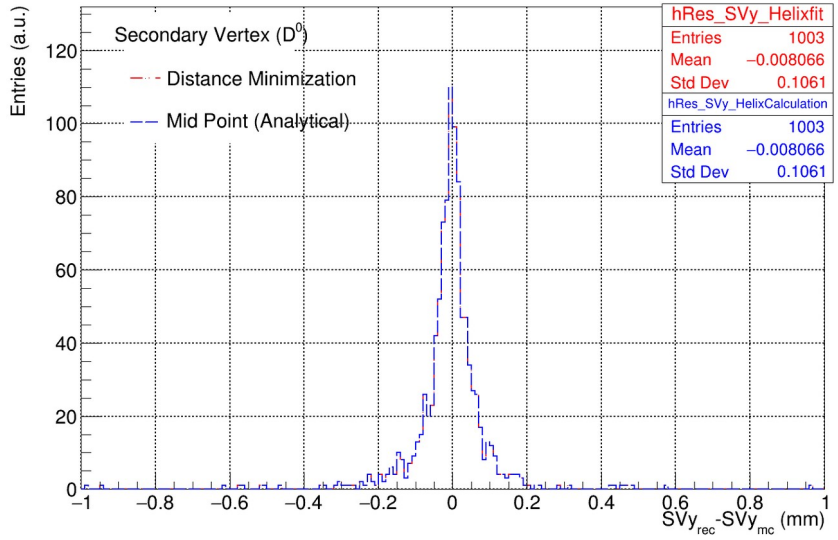
```
// Define function for Distance2 minimization between two helices
struct Distance2Minimization {
     StPhysicalHelix fhelix1, fhelix2;
     Distance2Minimization(StPhysicalHelix helix1, StPhysicalHelix helix2) : fhelix1(helix1),fhelix2(helix2) {}
    // Implementation of the function to be minimized
    double operator() (const double *par) {
    double x = par[0];
    double y = par[1];
    double z = par[2];
    double s1 = par[3];
    double s2 = par[4];
    double f = 0;
    TVector3 vertex(x, y, z);
    TVector3 p1 = fhelix1.at(s1);
    TVector3 p2 = fhelix2.at(s2);

    double d1 = (vertex - p1).Mag2();
    double d2 = (vertex - p2).Mag2();
    f = d1 + d2; // minimize total squared distance
    return f;
     }
     };
```

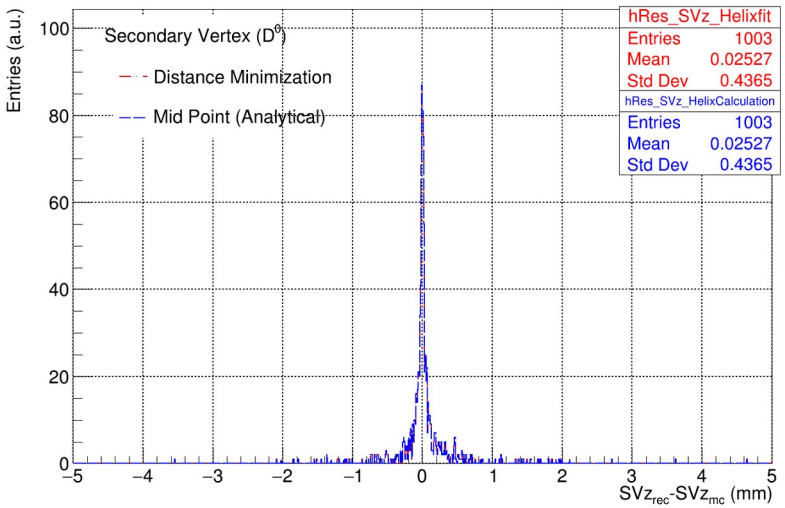
Initialization of parameters for fit function: can be improved in the future

```
StPhysicalHelix helix1(mom1, pos1, bField * tesla, charge1);
StPhysicalHelix helix2(mom2, pos2, bField * tesla, charge2);
pair<double, double> const ss = helix1.pathLengths(helix2);
TVector3 const p1_init = helix1.at(ss.first);
TVector3 const p2_init = helix2.at(ss.second);
// Perform Minimization
const Int_t nPar = 5;
Distance2Minimization d2Function(helix1,helix2);
ROOT::Math::Functor fcn(d2Function,nPar); // 5 parameters
ROOT::Fit::Fitter fitter;

double x0 =0.5*(p1_init.X() + p2_init.X()); double y0 =0.5*(p1_init.Y() + p2_init.Y()); double z0 =0.5*(p1_init.Z() + p2_init.Z());
double pStart[nPar] = {x0,y0,z0,ss.first,ss.second};
```

Comparison of Results







- **Distance minimization** and **mid point** estimation both works well for the true D^0

Comparison of MC secondary vertex to the reco vertex

86 files used

25.03.1/epic_craterlake/SIDIS/D0_ABCONV/pythia8.306-1.0/18x275

Comparison of Results

Analytical
Fit Parameters
Fit Parameters

```
-----------------
Mid Point (vx, vy, vz) from Helix Calculation = (-0.0453003,-0.016514,-0.064341)
Mid Point (vx, vy, vz) from fit = (-0.0453881,-0.016548,-0.0642932)
Secondary vertex (vx, vy, vz) from fit = (-0.0453003,-0.016514,-0.064341)
------------------
-----------------
Mid Point (vx, vy, vz) from Helix Calculation = (0.0131281,-0.0119958,-0.0965499)
Mid Point (vx, vy, vz) from fit = (0.0126326,-0.0121878,-0.0962801)
Secondary vertex (vx, vy, vz) from fit = (0.0131281,-0.0119958,-0.0965499)
------------------
-----------------
Mid Point (vx, vy, vz) from Helix Calculation = (-0.00270488,0.000410106,-0.0686854)
Mid Point (vx, vy, vz) from fit = (-0.00275871,0.000404454,-0.0686431)
Secondary vertex (vx, vy, vz) from fit = (-0.00270488,0.000410106,-0.0686854)
------------------
-----------------
Mid Point (vx, vy, vz) from Helix Calculation = (0.0135671,-0.00890401,-0.0799287)
Mid Point (vx, vy, vz) from fit = (0.0134079,-0.0089207,-0.0798036)
Secondary vertex (vx, vy, vz) from fit = (0.0135671,-0.00890401,-0.0799287)
------------------
-----------------
Mid Point (vx, vy, vz) from Helix Calculation = (-0.00679701,4.20239e-05,-0.0727941)
Mid Point (vx, vy, vz) from fit = (-0.00669404,0.000683956,-0.0726065)
Secondary vertex (vx, vy, vz) from fit = (-0.00679701,4.20239e-05,-0.0727941)
------------------
-----------------
Mid Point (vx, vy, vz) from Helix Calculation = (52.8759,17.5094,-896.419)
Mid Point (vx, vy, vz) from fit = (52.8759,17.5094,-896.418)
Secondary vertex (vx, vy, vz) from fit = (52.8759,17.5094,-896.419)
------------------
-----------------
Mid Point (vx, vy, vz) from Helix Calculation = (-0.470943,0.836423,9.68468)
Mid Point (vx, vy, vz) from fit = (-0.470943,0.836426,9.68465)
Secondary vertex (vx, vy, vz) from fit = (-0.470943,0.836423,9.68468)
------------------
-----------------
Mid Point (vx, vy, vz) from Helix Calculation = (-0.0203246,-0.0101872,-0.0647636)
Mid Point (vx, vy, vz) from fit = (-0.0203641,-0.0100814,-0.0659648)
Secondary vertex (vx, vy, vz) from fit = (-0.0203246,-0.0101872,-0.0647636)
------------------
```

Dimensions are in cm

Secondary Vertex Reconstruction (Λ_c)

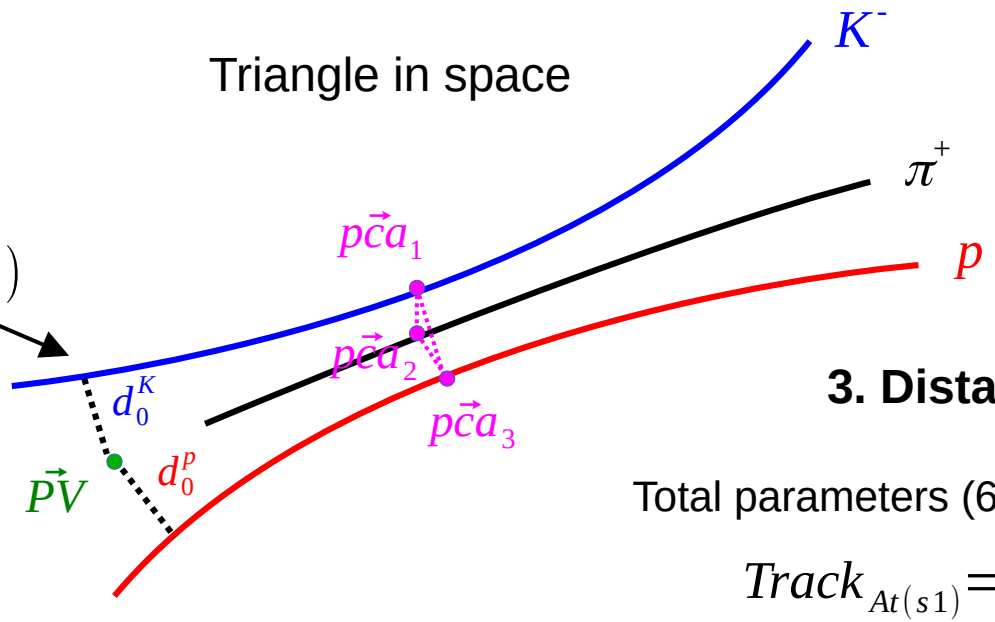
Secondary Vertex (Analytical approach)

$\Lambda_c^+ \rightarrow p K^- \pi^+$

SV for preliminary studies: Tracking errors are currently ignored

1. $\vec{SV} = \frac{\vec{pca}_1 + \vec{pca}_2 + \vec{pca}_3}{3}$

Triangle in space



$Track_{DCA} = (\vec{r}_0, \vec{p}_0, q)$

3. Distance2 Minimization

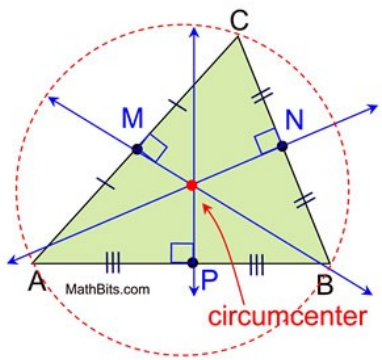
Total parameters (6) = (v_x, v_y, v_z, s_1, s_2, s_3)

$$Track_{At(s1)} = (\vec{r}_{s1}, \vec{p}1, q1)$$

$$Track_{At(s2)} = (\vec{r}_{s2}, \vec{p}2, q2)$$

$$Track_{At(s3)} = (\vec{r}_{s3}, \vec{p}3, q3)$$

2. CircumCircle Center



Minimize $d^2 = (\vec{r}_{s1} - \vec{v})^2 + (\vec{r}_{s2} - \vec{v})^2 + (\vec{r}_{s3} - \vec{v})^2$

Distance2 Minimization

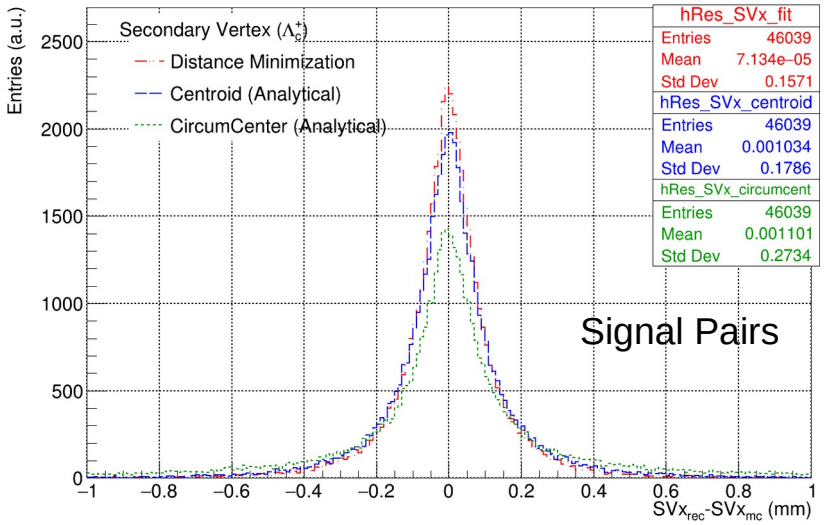
```
// Define function for Distance2 minimization between three helices
struct Distance2Minimization {
     StPhysicalHelix fhelix1, fhelix2, fhelix3;
     Distance2Minimization(StPhysicalHelix helix1, StPhysicalHelix helix2, StPhysicalHelix helix3) : fhelix1(helix1),fhelix2(helix2), fhelix3(helix3) {}
    // Implementation of the function to be minimized
    double operator() (const double *par) {
    double x = par[0];
    double y = par[1];
    double z = par[2];
    double s1 = par[3];
    double s2 = par[4];
    double s3 = par[5];
    double f = 0;

    TVector3 vertex(x, y, z);
    TVector3 p1 = fhelix1.at(s1);
    TVector3 p2 = fhelix2.at(s2);
    TVector3 p3 = fhelix3.at(s3);

    double d1 = (vertex - p1).Mag2();
    double d2 = (vertex - p2).Mag2();
    double d3 = (vertex - p3).Mag2();
    f = d1 + d2 + d3; // minimize total squared distance
    return f;
     }
     };
```

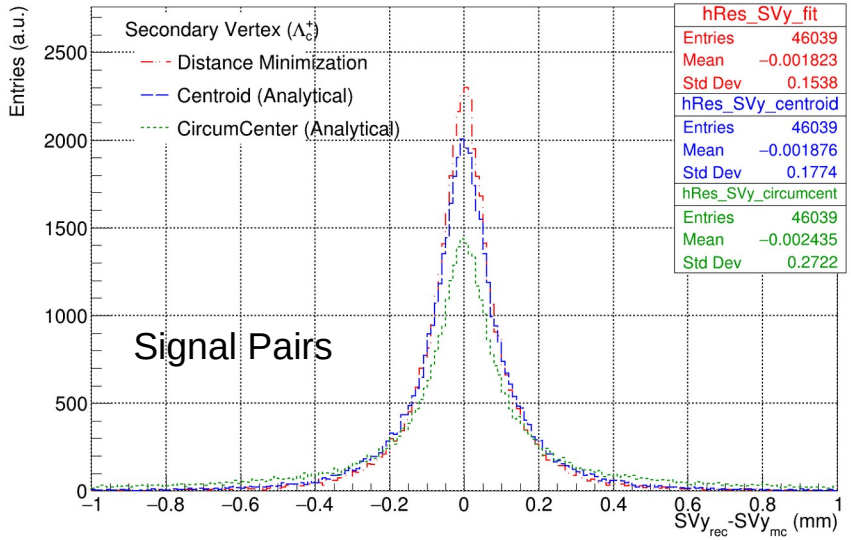
Function to minimize

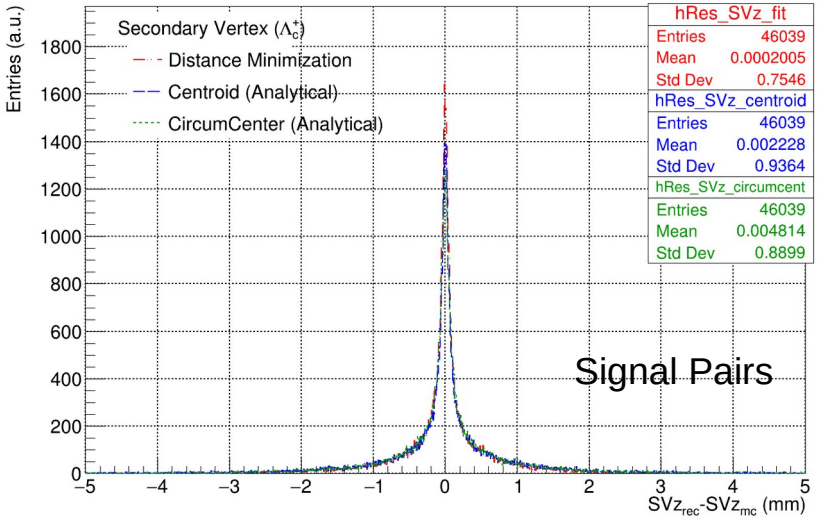
```
pair<double, double> const ss12 = helix1.pathLengths(helix2);
pair<double, double> const ss13 = helix1.pathLengths(helix3);
TVector3 const p1_init = helix1.at(ss12.first);
TVector3 const p2_init = helix2.at(ss12.second);
TVector3 const p3_init = helix3.at(ss13.second);

  // Perform Minimization
const Int_t nPar = 6;
Distance2Minimization d2Function(helix1,helix2,helix3);
ROOT::Math::Functor fcn(d2Function,nPar); // 5 parameters
ROOT::Fit::Fitter fitter;

double x0 =1./3*(p1_init.X() + p2_init.X()+ p3_init.X()); double y0 =1./3*(p1_init.Y() + p2_init.Y() + p3_init.Y()); double z0 =1./3*(p1_init.Z() + p2_init.Z() + p3_init.Z());
double pStart[nPar] = {x0,y0,z0,ss12.first,ss12.second,ss13.second};
fitter.SetFCN(fcn, pStart,nPar,1);
```

Initialization of parameters for fit function: can be improved in the future

Comparison of Results (Λ_c)

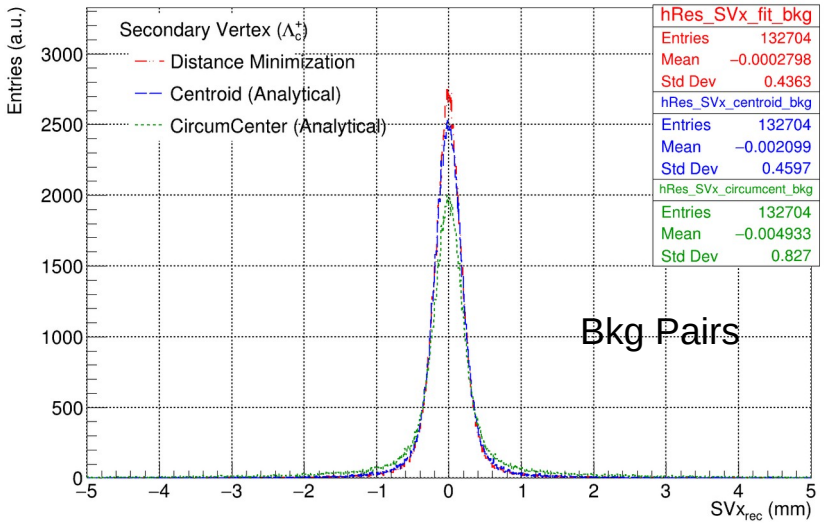


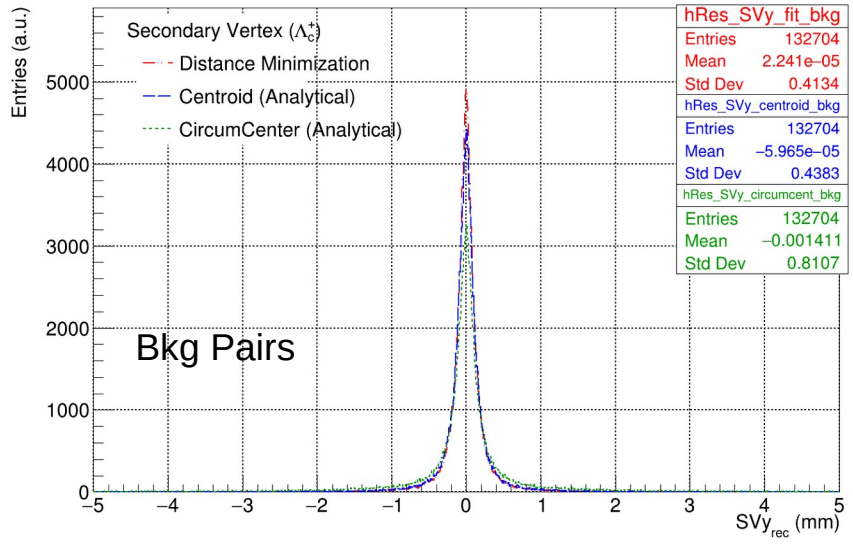


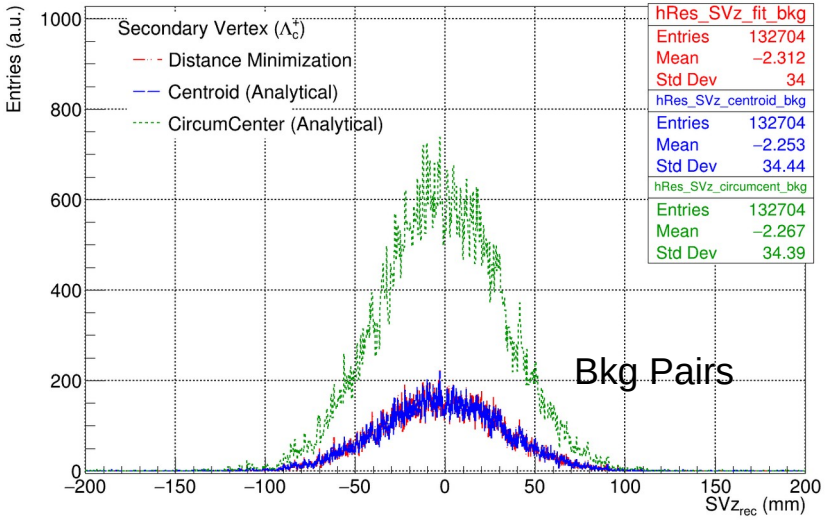


- **Distance minimization** is giving better results
- Of Course, chi2 minimization will be the best option but requires covariance information at DCA

Comparison of Results (Λ_c)







- **Distance minimization** is giving better results
- Of Course chi2 minimization will be best but requires covariance information at DCA

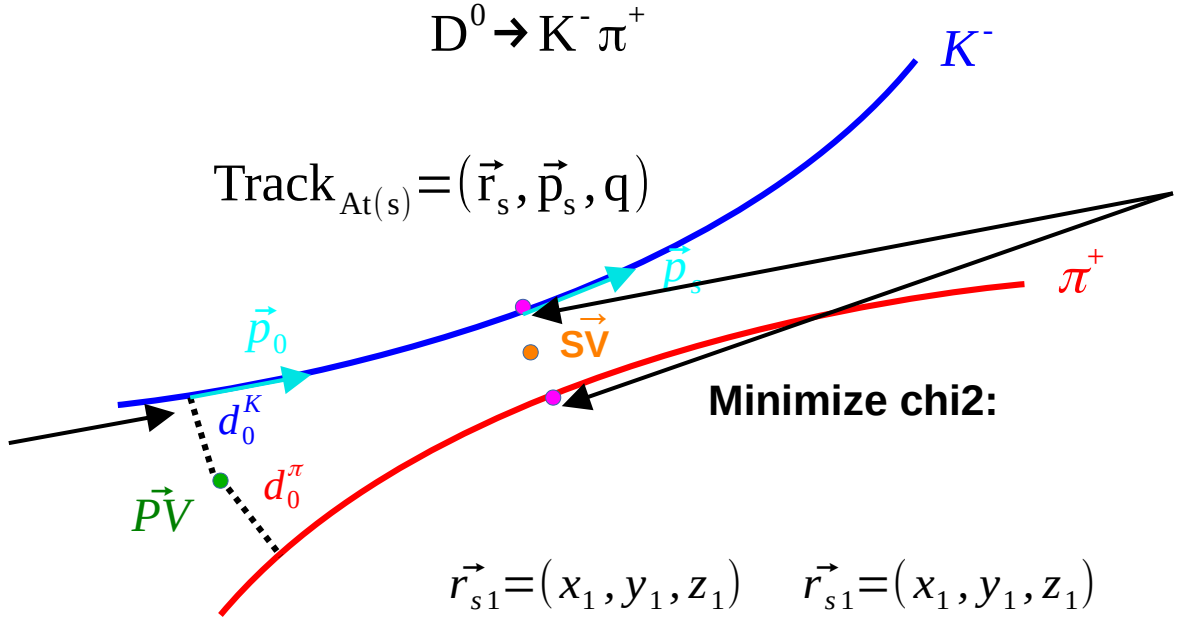
STAR experiment

The $\Lambda_c^{\pm}$ decay vertex is reconstructed as the mid-point of the distance of closest approach (DCA) between the three daughter tracks. To improve separation of signal

Approach 3 (considering the tracking errors)
Minimization of chi-square

Vertex position SV = (v_x, v_y, v_z) parameters

$D^0 \rightarrow K^- \pi^+$



If we know the covariance at this point

s: path length

$\text{Track}_{\text{At(s)}} = (\vec{r}_s, \vec{p}_s, q)$

$\text{Track}_{\text{DCA}} = (\vec{r}_0, \vec{p}_0, q)$

Minimize chi2:

$\vec{r_{s1}} = (x_1, y_1, z_1) \quad \vec{r_{s1}} = (x_1, y_1, z_1)$

$$\chi^2 = \frac{(x_1 - v_x)^2}{\sigma_{x1}^2} + \frac{(x_2 - v_x)^2}{\sigma_{x2}^2} + \frac{(y_1 - v_x)^2}{\sigma_{y1}^2} + \frac{(y_2 - v_x)^2}{\sigma_{y2}^2} + \frac{(z_1 - v_x)^2}{\sigma_{z1}^2} + \frac{(z_2 - v_x)^2}{\sigma_{z2}^2}$$

$$\frac{\partial \chi^2}{\partial v_x} = 0 \qquad \frac{\partial \chi^2}{\partial v_y} = 0 \qquad \frac{\partial \chi^2}{\partial v_z} = 0$$

Secondary Vertex Reconstruction (D^0)

$$v_x = \frac{\left(\frac{x_1}{\sigma_{x1}^2} + \frac{x_2}{\sigma_{x2}^2}\right)}{\left(\frac{1}{\sigma_{x1}^2} + \frac{1}{\sigma_{x2}^2}\right)} \qquad v_y = \frac{\left(\frac{y_1}{\sigma_{y1}^2} + \frac{y_2}{\sigma_{y2}^2}\right)}{\left(\frac{1}{\sigma_{y1}^2} + \frac{1}{\sigma_{y2}^2}\right)} \qquad v_z = \frac{\left(\frac{z_1}{\sigma_{z1}^2} + \frac{z_2}{\sigma_{z2}^2}\right)}{\left(\frac{1}{\sigma_{z1}^2} + \frac{1}{\sigma_{z2}^2}\right)}$$

$x = -l_0 \sin\phi$, $y = l_0 \cos\phi$, $z = l_1$

For a given track:

$\sigma x2 = \sin^2\phi \cdot \sigma\ell0^2 + \ell0^2 \cos^2\phi \cdot \sigma\phi^2 + 2 \cdot \ell0 \sin\phi \cos\phi \cdot \mathrm{Cov}(\ell0, \phi)$
$\sigma y2 = \cos^2\phi \cdot \sigma\ell0^2 + \ell0^2 \sin^2\phi \cdot \sigma\phi^2 - 2 \cdot \ell0 \sin\phi \cos\phi \cdot \mathrm{Cov}(\ell0, \phi)$
$\sigma z^2 = \sigma\ell1^2$

We need covariances at pca_1 and pca_2

Further chi2 minimization can be performed in the code

Chi2 Minimization Code

These covariances must be at pca1 and pca2 points

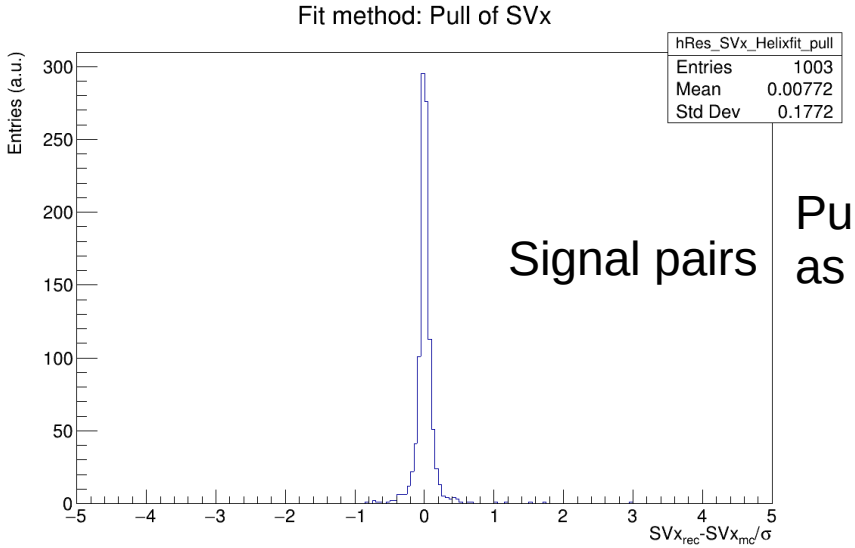
```
// Define function for Chi2 minimization between two helices
struct Chi2Minimization {
     StPhysicalHelix fhelix1, fhelix2;
     std::array<float, 21> fcov1, fcov2; // full covariance matrix

     Chi2Minimization(StPhysicalHelix helix1, StPhysicalHelix helix2, std::array<float, 21> cov1, std::array<float, 21> cov2) : fhelix1(helix1),fhelix2(helix2), fcov1(cov1), fcov2(cov2) {}
    // Implementation of the function to be minimized
    double operator() (const double *par) {
    double x = par[0];
    double y = par[1];
    double z = par[2];
    double s1 = par[3];
    double s2 = par[4];
    double f = 0;
    TVector3 vertex(x, y, z);
    TVector3 p1 = fhelix1.at(s1);
    TVector3 p2 = fhelix2.at(s2);
    TVector3 mom1 = fhelix1.momentumAt(s1,  bField * tesla);
    TVector3 mom2 = fhelix2.momentumAt(s2,  bField * tesla);

   // x= -l0 sinϕ , y=l0 cosϕ , z=l
   // Recalculate l0 at PCA for error propagation
     float l0_track1 = p1.Pt(); float l1_track1 = p1.Z(); double phi_track1 = mom1.Phi();
     float l0_track2 = p2.Pt(); float l1_track2 = p2.Z(); double phi_track2 = mom2.Phi();
     // Track1: σx^2=sin^2ϕ·σℓ0^2+ℓ0^2cos^2ϕ·σϕ^2+2·ℓ0sinϕcosϕ·Cov(ℓ0,ϕ)
     float sigx1_2 = sin(phi_track1)*sin(phi_track1)*fcov1[0] + l0_track1*l0_track1*cos(phi_track1)*cos(phi_track1)*fcov1[5]+ 2.0*l0_track1*sin(phi_track1)*cos(phi_track1)*fcov1[3];
     float sigx2_2 = sin(phi_track2)*sin(phi_track2)*fcov2[0] + l0_track2*l0_track2*cos(phi_track2)*cos(phi_track2)*fcov2[5]+ 2.0*l0_track2*sin(phi_track2)*cos(phi_track2)*fcov2[3];

     // σy^2=cos^2ϕ·σℓ0^2+ℓ0^2sin^2ϕ·σϕ^2−2·ℓ0sinϕcosϕ·Cov(ℓ0,ϕ)
     float sigy1_2 = cos(phi_track1)*cos(phi_track1)*fcov1[0] + l0_track1*l0_track1*sin(phi_track1)*sin(phi_track1)*fcov1[5]-2.0*l0_track1*sin(phi_track1)*cos(phi_track1)*fcov1[3];
     float sigy2_2 = cos(phi_track2)*cos(phi_track2)*fcov2[0] + l0_track2*l0_track2*sin(phi_track2)*sin(phi_track2)*fcov2[5]-2.0*l0_track2*sin(phi_track2)*cos(phi_track2)*fcov2[3];

     // σz^2
     float sigz1_2 = fcov1[2];
     float sigz2_2 = fcov2[2];
    double d1_x = 10.*(vertex - p1).X(); double d2_x = 10.*(vertex - p2).X();
    double d1_y = 10.*(vertex - p1).Y(); double d2_y = 10.*(vertex - p2).Y();
    double d1_z = 10.*(vertex - p1).Z(); double d2_z = 10.*(vertex - p2).Z();

     f = d1_x*d1_x/sigx1_2 + d2_x*d2_x/sigx2_2 + d1_y*d1_y/sigy1_2 + d2_y*d2_y/sigy2_2 + d1_z*d1_z/sigz1_2+ d2_z*d2_z/sigz2_2; // chi2
      return f;
     }
```

Root files store the covariance at the DCA, I did a quick try with that

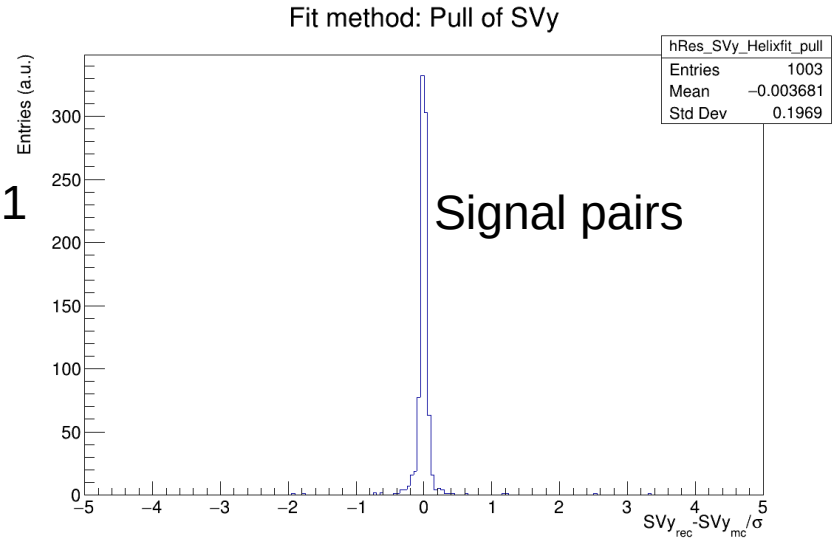
Note: Covariances are taken from DCA to primary vertex
Need to estimate at pca1 and pca2 can be done in ACTs

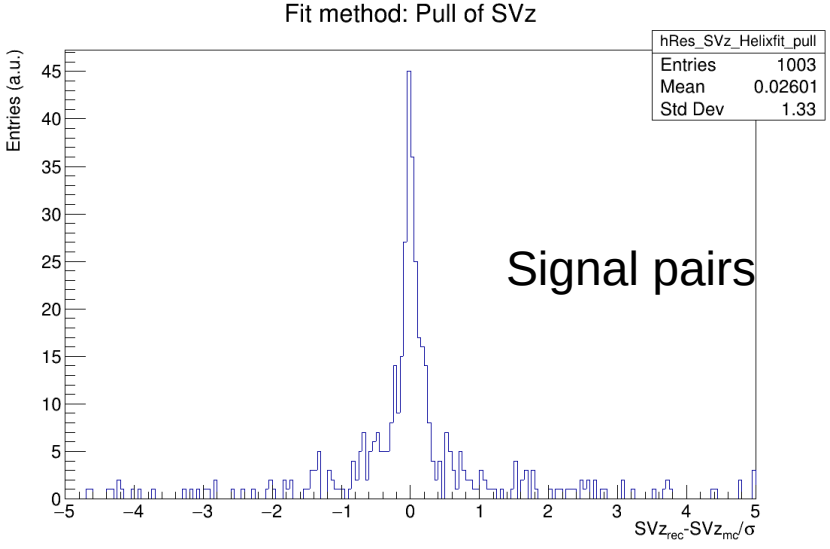
Chi2 Minimization Results

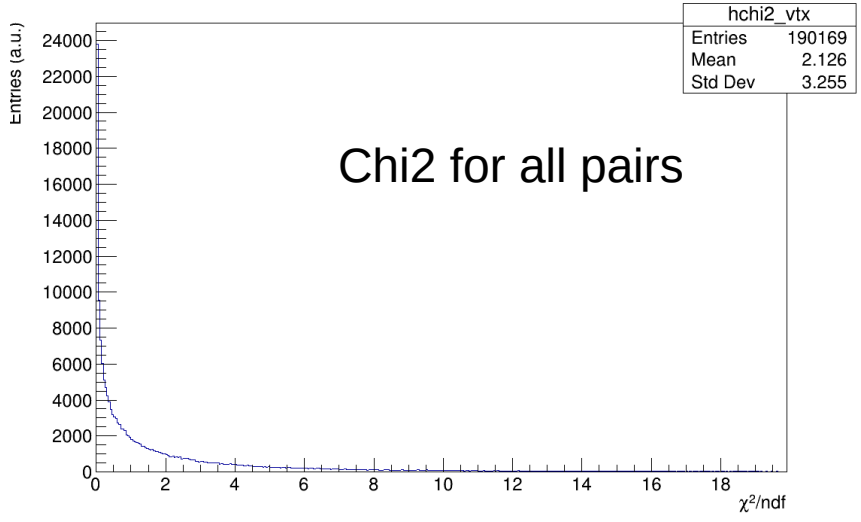
Covariance can be improved (required at the DCA between two tracks)



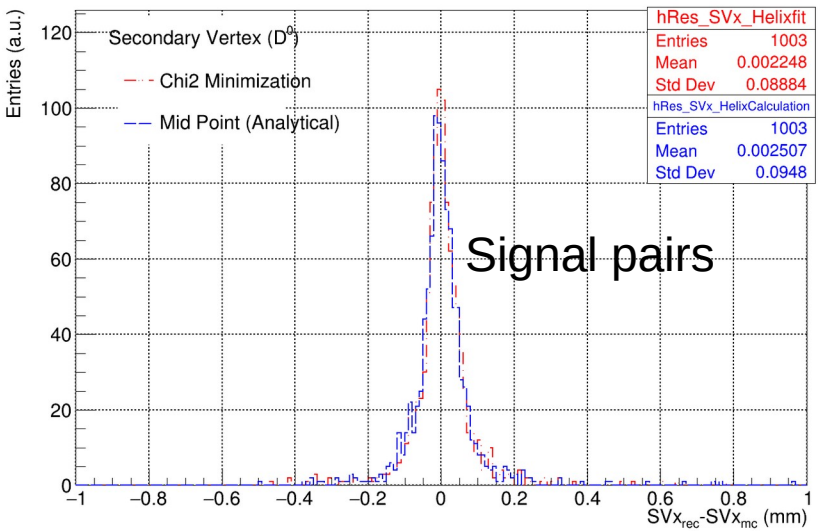
Pulls are not ~1 as expected

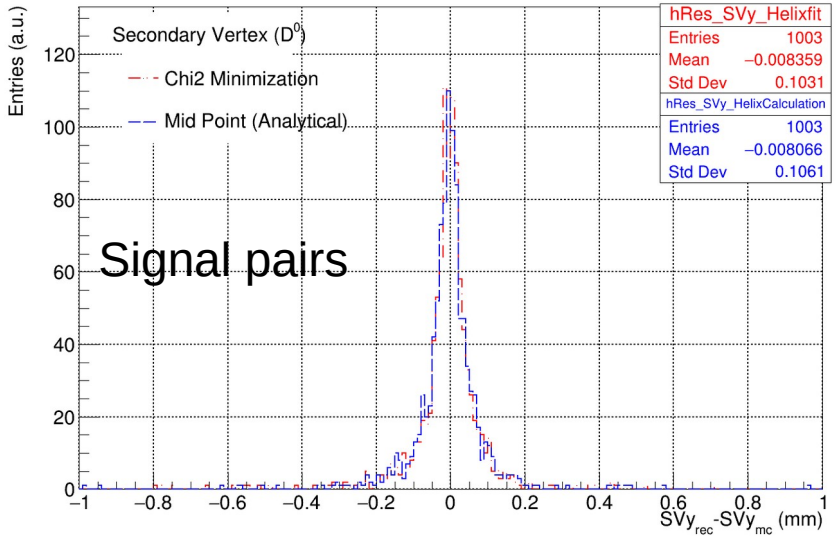


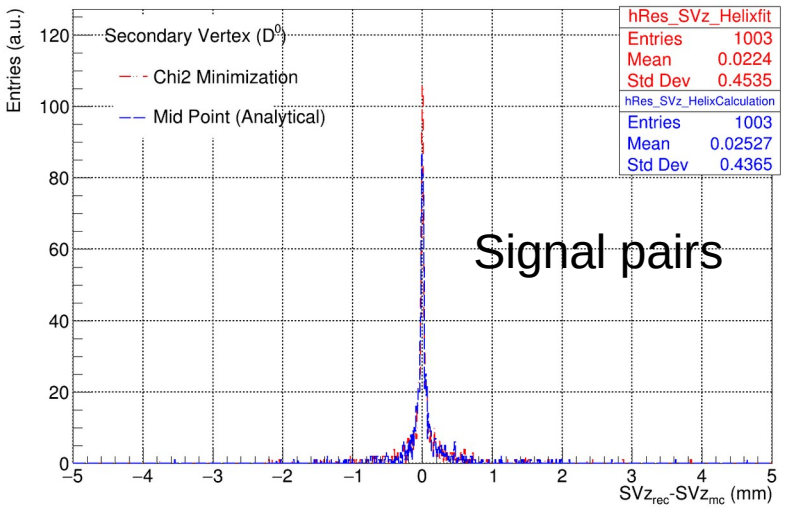




Comparison of Results







- **Chi2 minimization** and **mid point** estimation both works well for the true D^0

Comparison of MC secondary vertex to the reco vertex

86 files used

25.03.1/epic_craterlake/SIDIS/D0_ABCONV/pythia8.306-1.0/18x275

Summary

- I investigated different approaches to improve secondary vertexing
- Distance minimization can be a good solution can be a good solution until KFParticle or the AdaptiveMultiVertexFinder becomes available
- Chi2 minimization can also be implemented, but it requires knowledge of the covariances at the DCA points between two tracks
- The expectation is to have pulls consistent with unity after using the proper covariances
- For three-daughter decays, distance minimization can be a good solution for the moment

Thank you for your attention!

Covariance Matrix

Track Parameters $(l_0, l_1, \phi, \theta, q/p, time)$

For each fitted track we get track parameters and covariance matrix

$$
\begin{array}{c} \\ l_0 \\ l_1 \\ \phi \\ \theta \\ q/p \\ time \end{array}
\begin{array}{c}
\begin{array}{cccccc} l_0 & l_1 & \phi & \theta & q/p & time \end{array} \\
\begin{bmatrix}
\sigma^2(l_0) & \mathrm{cov}(l_0,l_1) & \mathrm{cov}(l_0,\phi) & \mathrm{cov}(l_0,\theta) & \mathrm{cov}(l_0,q/p) & \mathrm{cov}(l_0,t) \\
. & \sigma^2(l_1) & \mathrm{cov}(l_1,\phi) & \mathrm{cov}(l_1,\theta) & \mathrm{cov}(l_1,q/p) & \mathrm{cov}(l_1,t) \\
. & . & \sigma^2(\phi) & \mathrm{cov}(\phi,\theta) & \mathrm{cov}(\phi,q/p) & \mathrm{cov}(\phi,t) \\
. & . & . & \sigma^2(\theta) & \mathrm{cov}(\theta,q/p) & \mathrm{cov}(\theta,t) \\
. & . & . & . & \sigma^2(q/p) & \mathrm{cov}(q/p,t) \\
. & . & . & . & . & \sigma^2(t)
\end{bmatrix}
\end{array}
$$

Symmetric matrix: Independent entries = n(n+1)/2 = 6*7/2 = 21

```
Processing ReadCovarianceArray_new.C...
Event 0, number of tracks: 8
  Track 0 covariance:
    cov[0] = 0.0104456
    cov[1] = 2.366e-06
    cov[2] = 0.0103324
    cov[3] = -0.000289634
    cov[4] = 7.52669e-07
    cov[5] = 8.04972e-06
    cov[6] = -8.12589e-07
    cov[7] = 0.000284166
    cov[8] = 4.50984e-08
    cov[9] = 7.82997e-06
    cov[10] = 0.000153944
    cov[11] = 7.02629e-07
    cov[12] = -4.94218e-06
    cov[13] = 5.14138e-09
    cov[14] = 0.00011838
    cov[15] = -1.41347e-06
    cov[16] = -3.20263e-06
    cov[17] = 3.89944e-08
    cov[18] = -8.82041e-08
    cov[19] = 2.242e-09
    cov[20] = 0.000333566
```

Cov[0] = cov(l0, l0)
Cov[1] = cov(l0, l1)
Cov[2] = cov(l1, l1)
Cov[3] = cov(l0, φ)
cov[4] = cov(l1, φ)
cov[5] = cov(φ, φ)
cov[6] = cov(l0, θ)
cov[7] = cov(l1, θ)
cov[8] = cov(φ, θ)
cov[9] = cov(θ, θ)
cov[10] = cov(l0, q/p)

Cov[11] = cov(l1, q/p)
Cov[12] = cov(φ, q/p)
Cov[13] = cov(θ, q/p)
Cov[14] = cov(q/p, q/p)
Cov[15] = cov(l0, time)
Cov[16] = cov(l1, time)
Cov[17] = cov(φ, time)
Cov[18] = cov(θ, time)
Cov[19] = cov(q/p, time)
Cov[20] = cov(time, time)

$\sigma_{l0} = \sqrt{(Cov[0])}$ $\sigma_{l1} = \sqrt{(Cov[2])}$

$\sigma_{\phi} = \sqrt{(Cov[5])}$ $\sigma_{\theta} = \sqrt{(Cov[9])}$

$\sigma_q/p = \sqrt{(Cov[14])}$