Update on Secondary Vertexing

Shyam Kumar, Xin dong, Bishoy, Rongrong and others

Secondary Vertex Reconstruction

Primary Vertex Reconstruction:

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

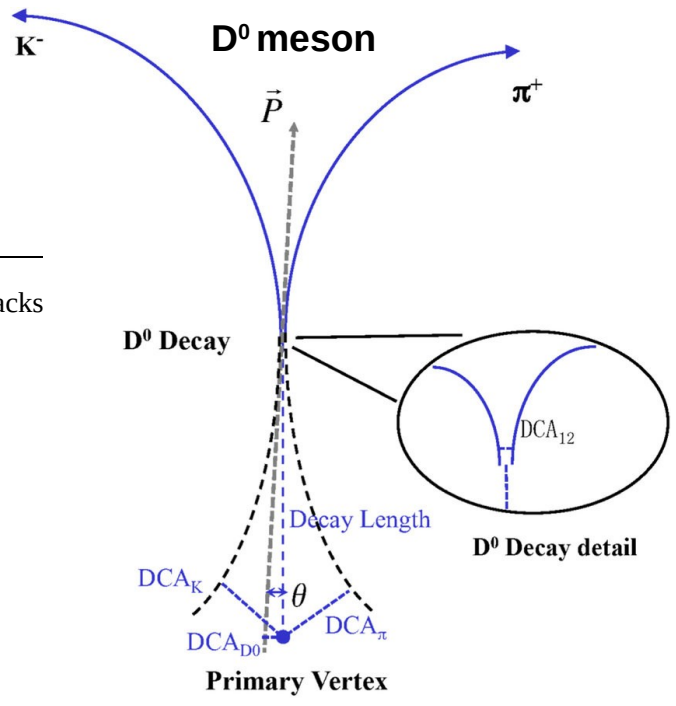
Primary vertex: allows access to the DCA of daughter tracks $\sigma_{PV} \propto \sigma_{dca} / \sqrt{N_{tracks}}$

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

$$\sigma_{SV} \propto \sigma_{dca} / \sqrt{N_{dau}}$$

N_{dau} = 2 for D^0

N_{dau} = 3 for Λ_c^+



Phys. Rev. C 102, 014905 (2020)

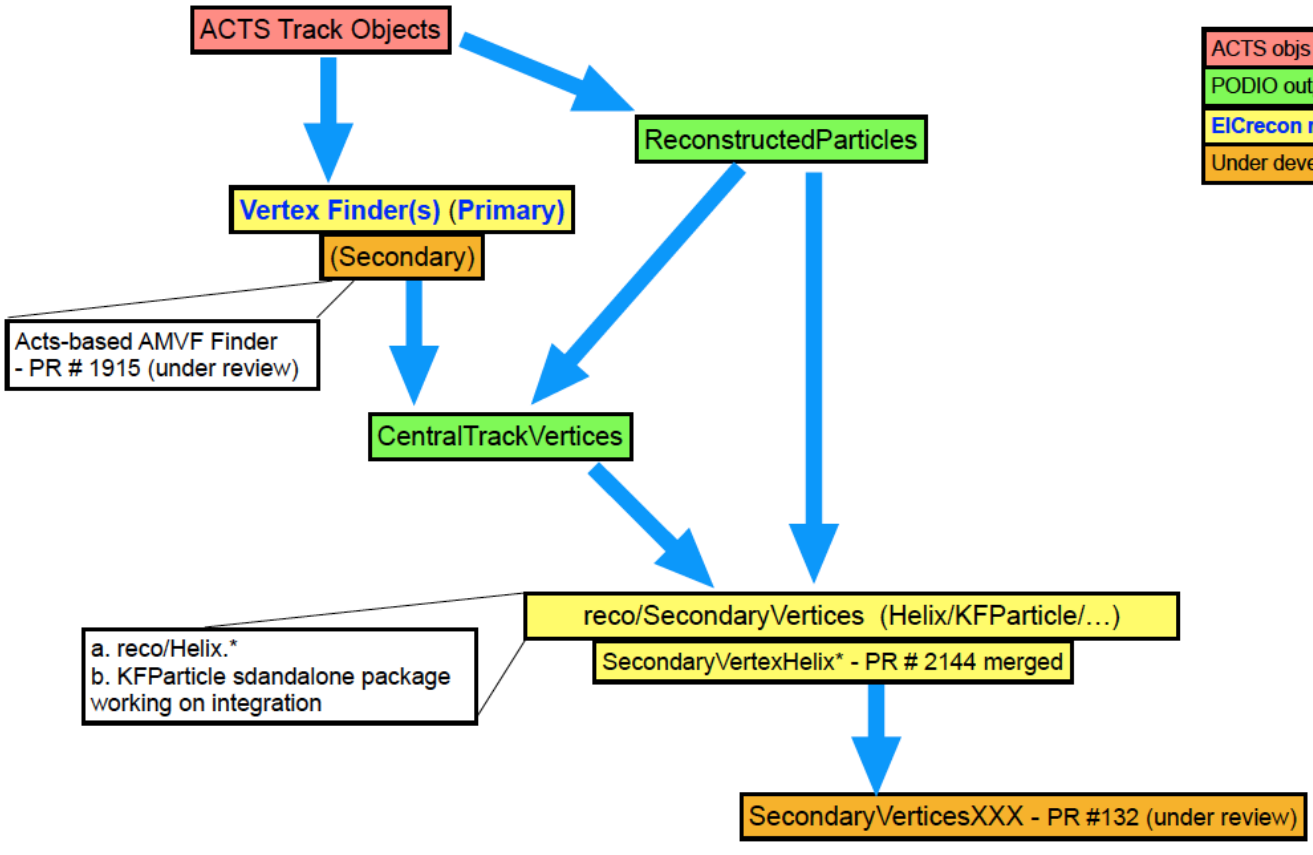
Secondary Vertex (SV) Reconstruction:

- **Helix Swimming**
- **Chi2 minimization (considers track errors): Shyam**
- AdaptiveMultiVertexFinder (considers track errors): Bishoy
- KFParticle (considers track errors): Xin and others

} Based on Kalman filter

Secondary Vertex

X. Dong (LBNL)
B. Dongwi (SBU)
S. Kumar



AMVF - under review (PR #1915)

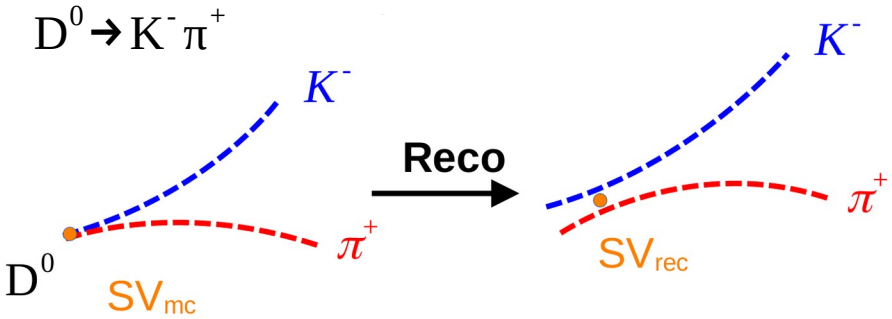
Helix - merged (PR #2144)

Chi2 fitting-need to implement in EICRecon

KFParticle - under development

Helix Swimming

Method based on STAR experiment which use track parameters to determine the vertex position

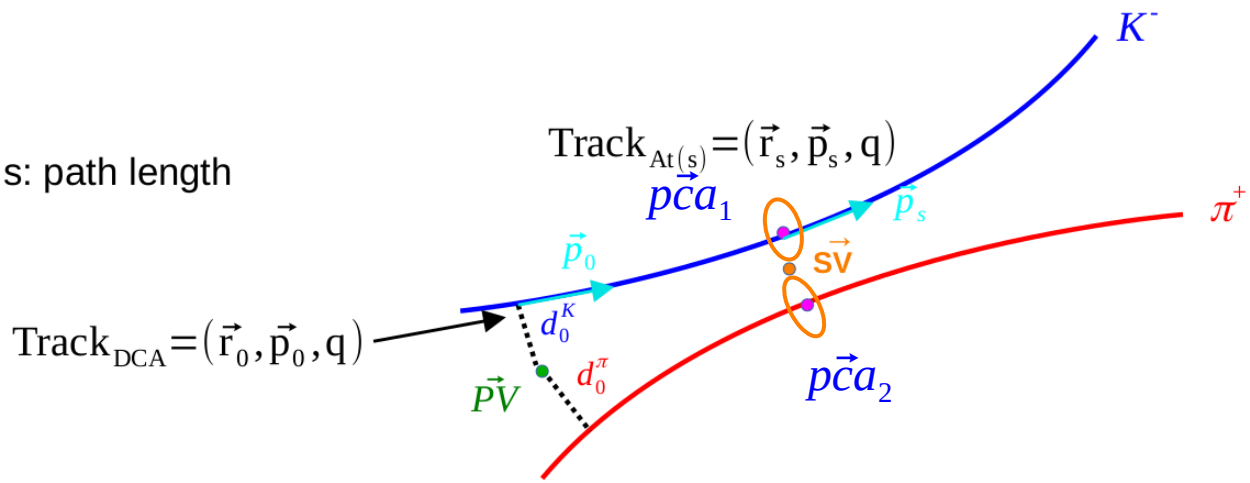


Track parameters (DCA):

$(l_0, l_1, \varphi, \theta, q/p)$ Local

(x, y, z, px, py, pz) Global

s: path length



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

$p_x = p \cos\varphi \sin\theta$, $p_y = p \sin\varphi \sin\theta$

$p_z = p \cos\theta$, $q = \text{sign}(q/p)$

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

Implemented in EICRecon (Xin dong, LBL)

Ignores the track **uncertainties**

Next: Compared the results from EICRecon to local code on the same simulation file, the two must be exactly same

Information used: **Track parameters**

Chi2 Minimization

Considering the tracking errors
Minimization of chi-square

S. Kumar,INFN

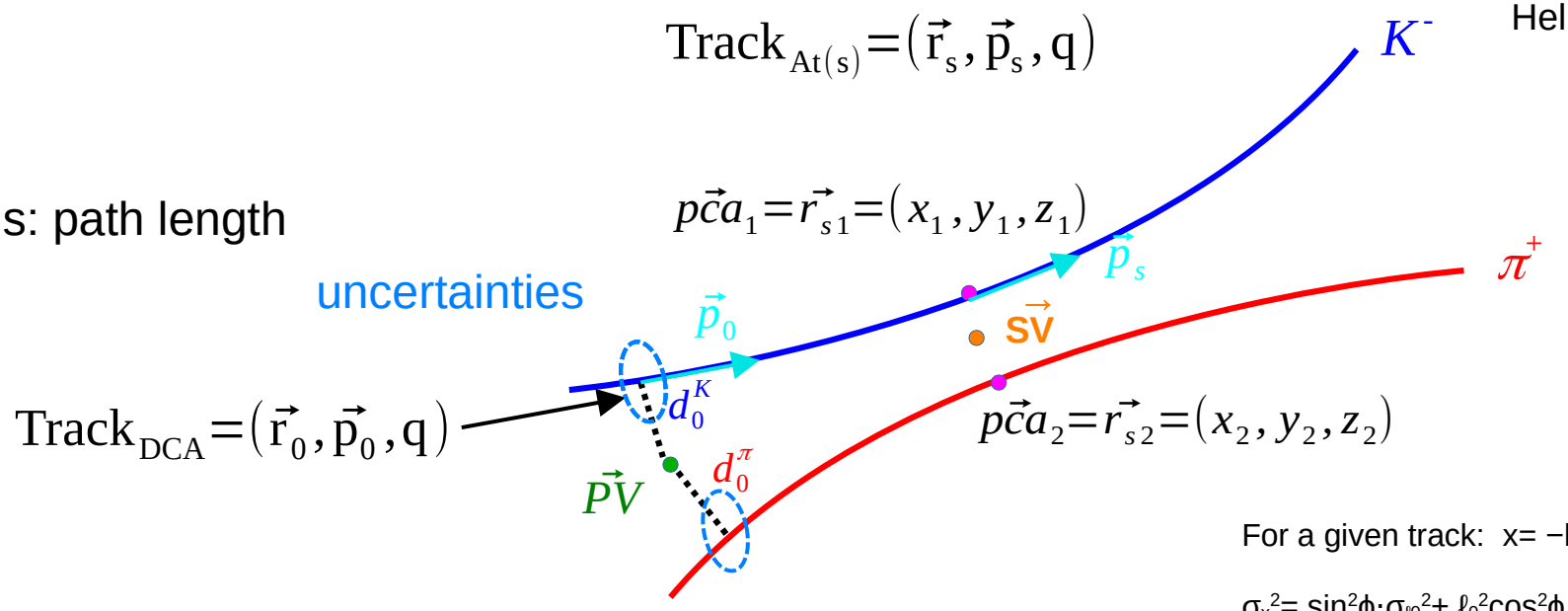
$D^0 \rightarrow K^- \pi^+$

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

Helix swimming is used for helix definition

s: path length

uncertainties



For a given track: $x = -l_0 \sin\phi$, $y = l_0 \cos\phi$, $z = l_1$

$\sigma_x^2 = \sin^2\phi \cdot \sigma_{l0}^2 + l_0^2 \cos^2\phi \cdot \sigma\phi^2 + 2 \cdot l_0 \sin\phi \cos\phi \cdot \text{Cov}(l_0, \phi)$
$\sigma_y^2 = \cos^2\phi \cdot \sigma_{l0}^2 + l_0^2 \sin^2\phi \cdot \sigma\phi^2 - 2 \cdot l_0 \sin\phi \cos\phi \cdot \text{Cov}(l_0, \phi)$
$\sigma_z^2 = \sigma_{l1}^2$

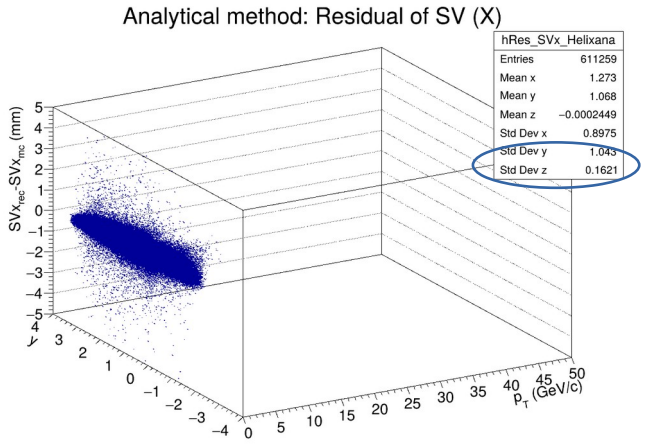
Parameters: (v_x, v_y, v_z, s_1, s_2)

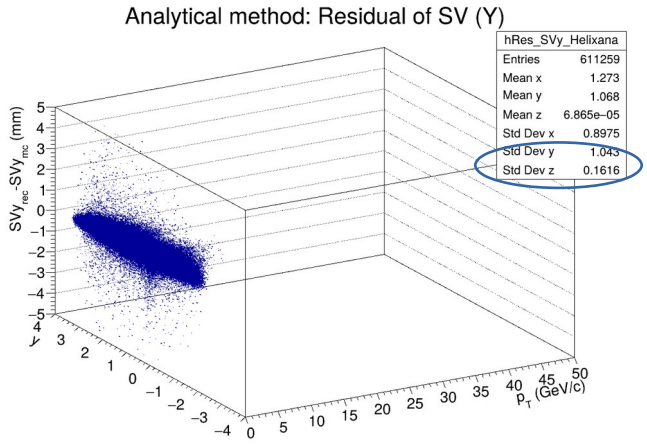
Minimize chi2: $\chi^2 = \frac{(x_1 - v_x)^2}{\sigma_{x1}^2} + \frac{(x_2 - v_x)^2}{\sigma_{x2}^2} + \frac{(y_1 - v_y)^2}{\sigma_{y1}^2} + \frac{(y_2 - v_y)^2}{\sigma_{y2}^2} + \frac{(z_1 - v_z)^2}{\sigma_{z1}^2} + \frac{(z_2 - v_z)^2}{\sigma_{z2}^2}$

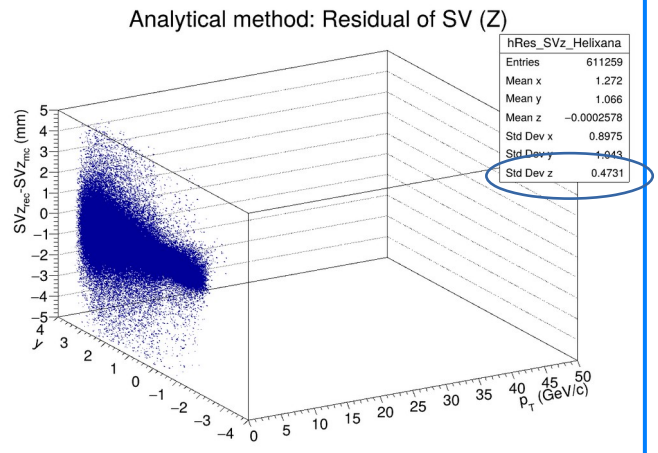
Information used: **Track parameters and Covariance**

Comparison of Results (Helix Swimming and Chi2)

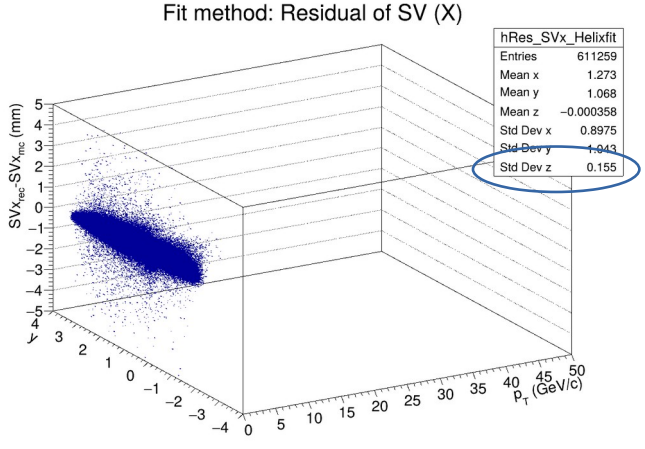
Helix Swimming

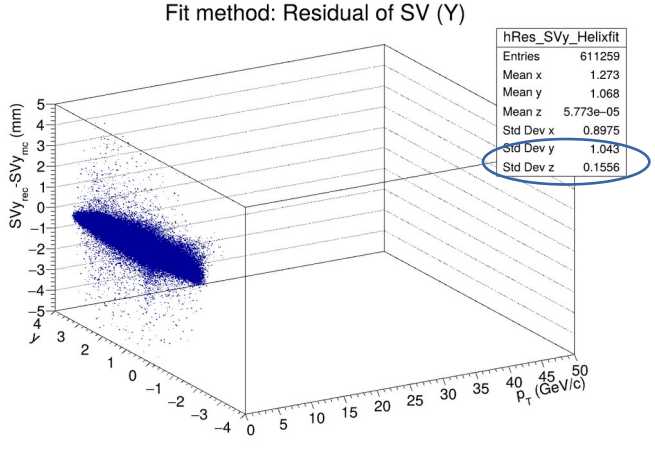


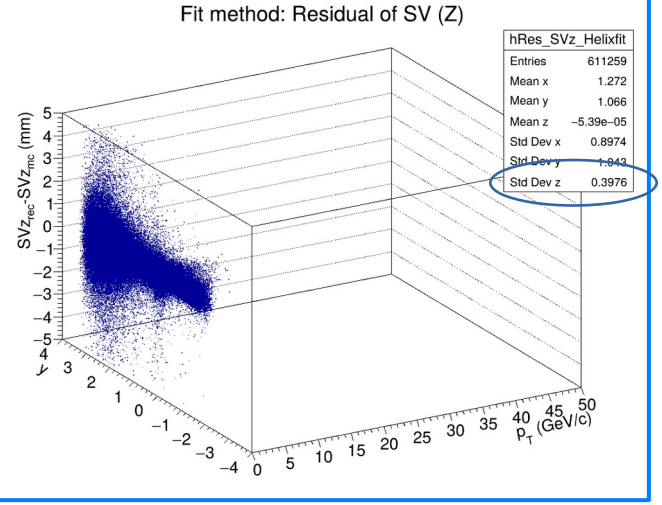




Chi2 fitting

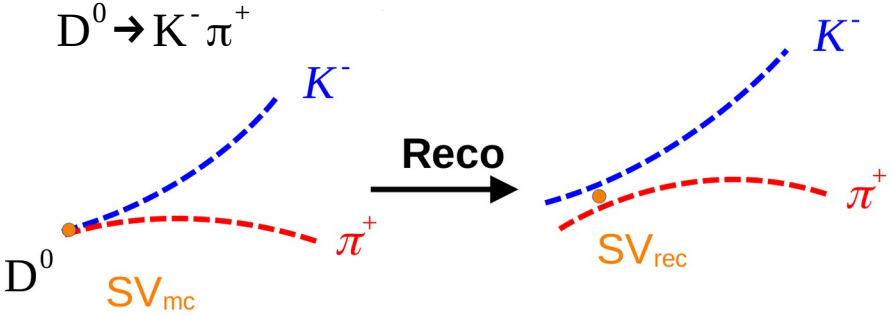






Secondary Vertex Resolution

Chi-square minimization using track parameters and covariances



D^0 Sample
ep (10x100, Q2> 1 GeV^2)

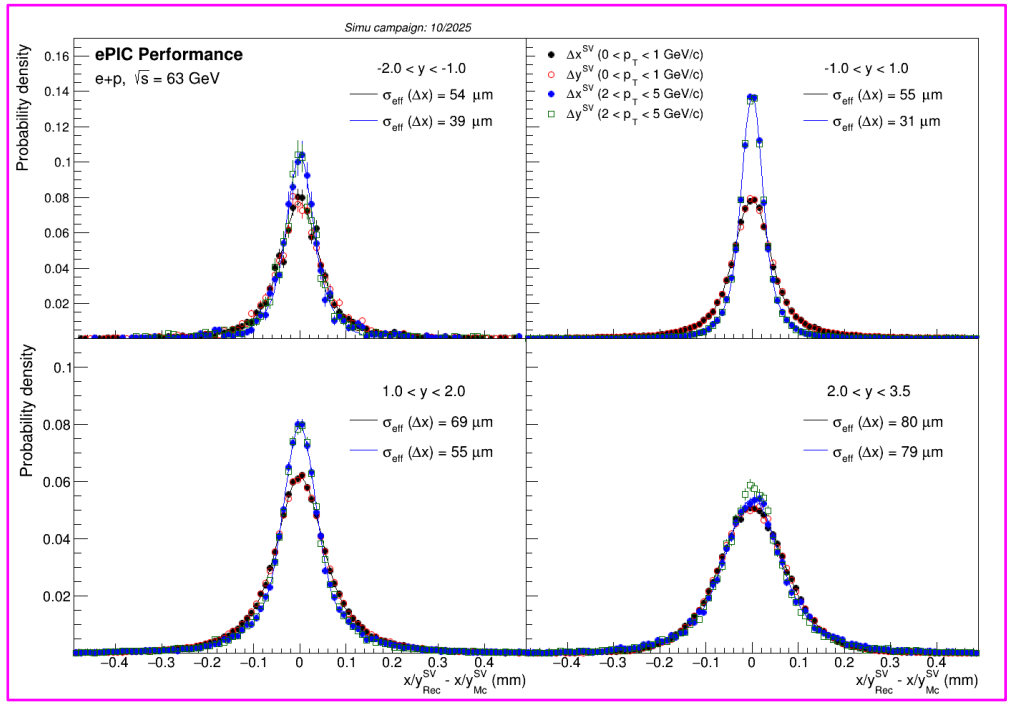
Fit Range: Mean+/- 2*StdDev

10.2025 preTDR

Two Gaussian (A_{core}, σ_{core}, A_{tail}, σ_{tail})

$$\sigma_{eff} = \sqrt{\frac{A_{core}}{(A_{core}+A_{tail})}\sigma^2_{core} + \frac{A_{tail}}{(A_{core}+A_{tail})}\sigma^2_{tail}}$$

σ_{eff} decreases as we go at high p_T



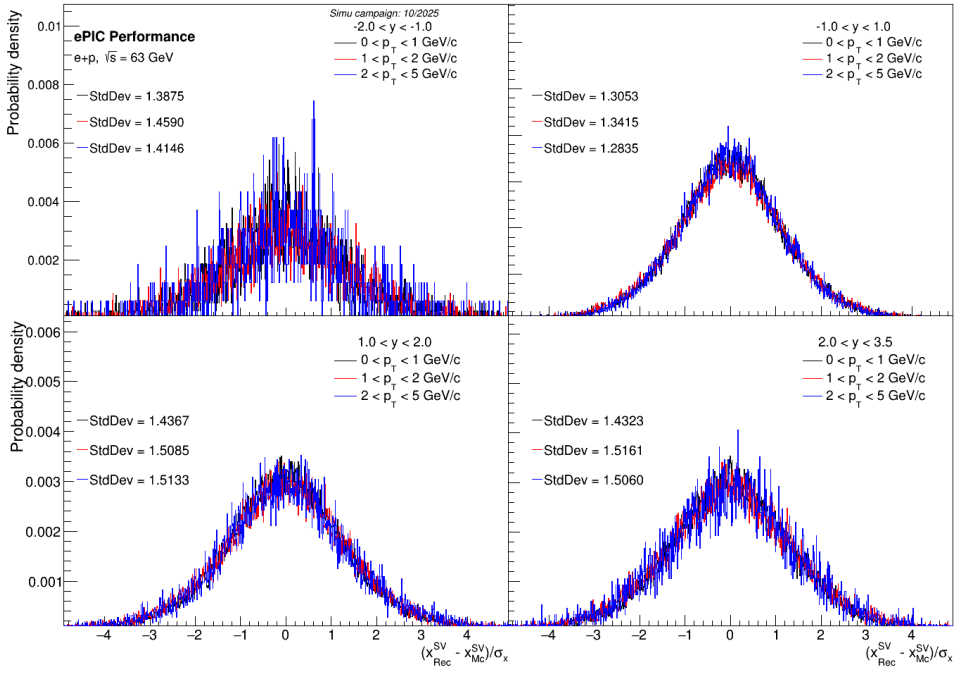
Pull distributions

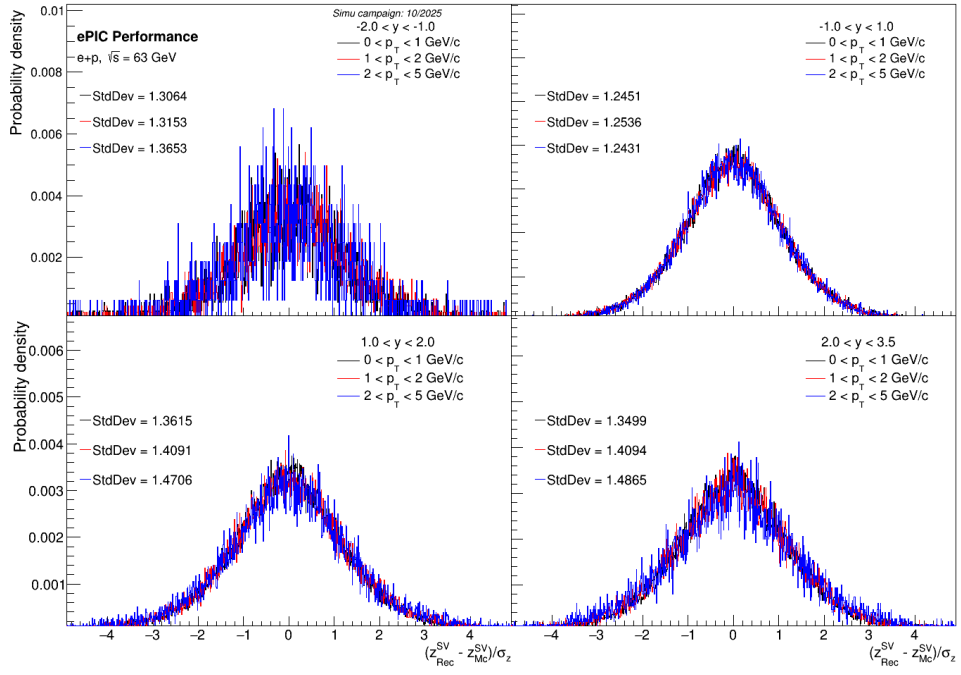
$D^0 \rightarrow K^- \pi^+$

D⁰ Sample **10.2025**
ep (10x100, Q2> 1 GeV²)

$$\text{Pulls} = \frac{SV^{x}_{rec} - SV^{x}_{mc}}{\sigma_x}$$

$$\text{Pulls} = \frac{SV^{z}_{rec} - SV^{z}_{mc}}{\sigma_z}$$

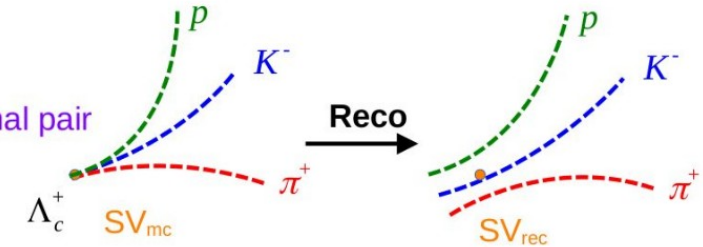




Topological features and ML

Pull distributions

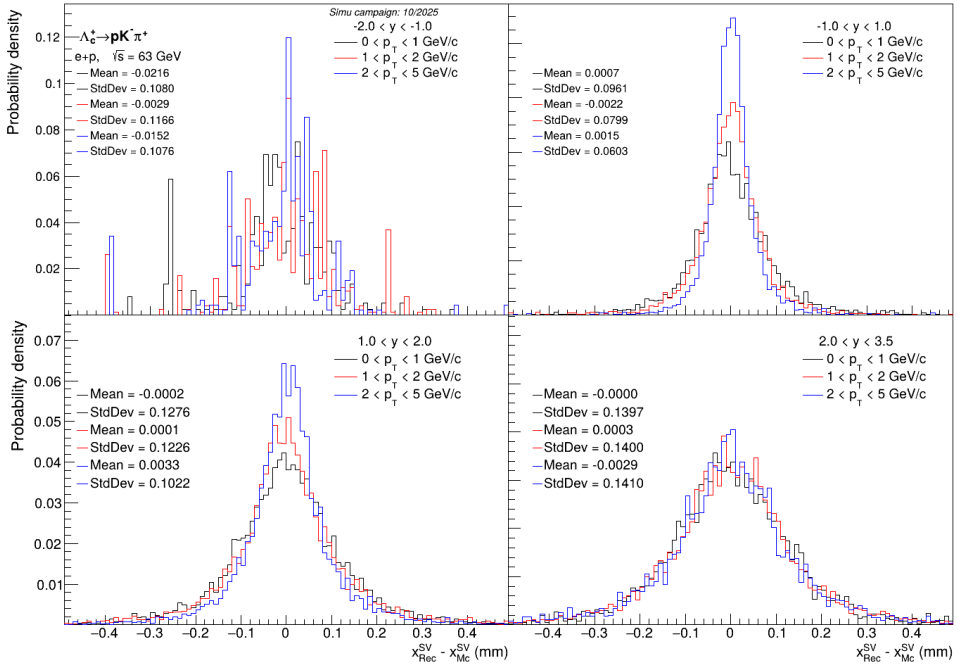
$\Lambda_c^+ \rightarrow p K^- \pi^+$ Signal pair

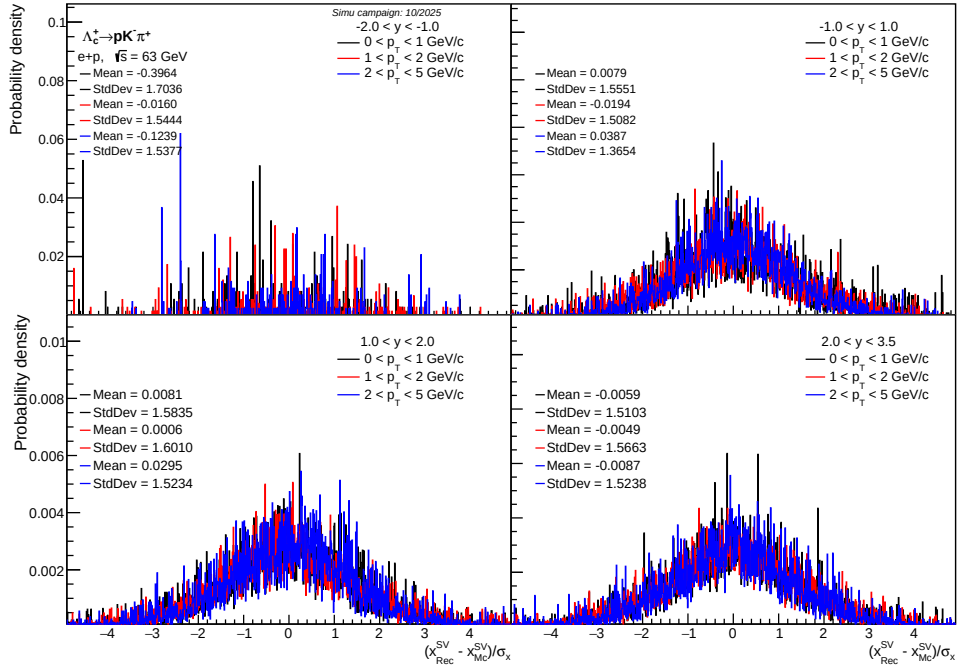


Λ_c Sample **10.2025**
ep (10x100, Q2> 1 GeV²)

$$\left(SV^x_{rec} - SV^x_{mc}\right) \quad \text{mm}$$

$$\text{Pulls} = \frac{SV^x_{rec} - SV^x_{mc}}{\sigma_x}$$





AdaptiveMultiVertexFinder (AMVF)

Considers track parameters and covariance (fitting approach: Kalman)

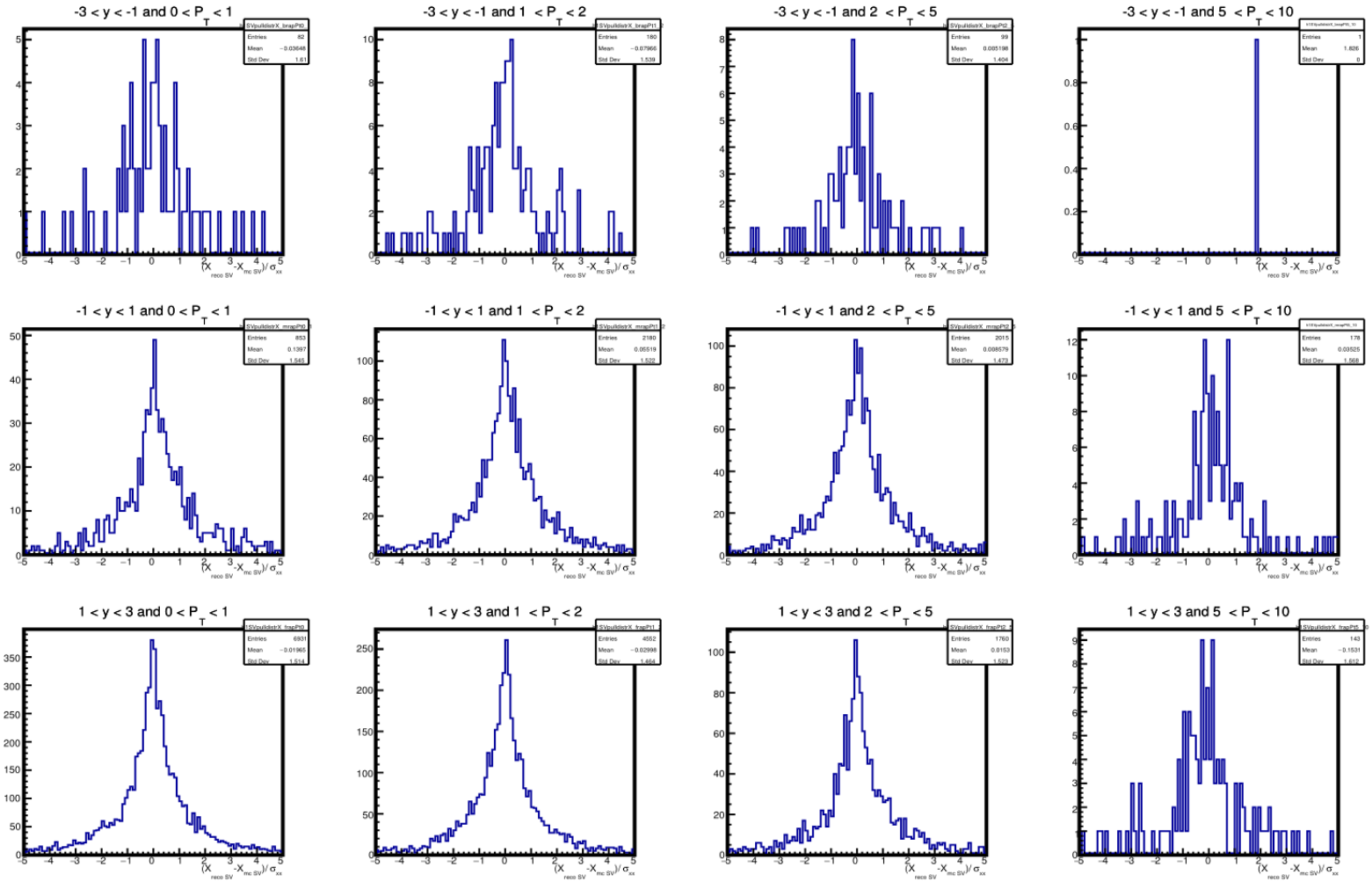
Additional time information can help

B. Dongwi (SBU)

$\Lambda^0 \rightarrow p\,\pi^-$

Slide 16

Pulls are ~1.5



Apply on D^0 sample

Information used: **Track parameters, time, and Covariance**

KFParticle

Considers track parameters and covariance (fitting approach: Kalman)

Additional kinematic constraint e.g. mass

Implementation Requires:

Xin dong, LBL

- Track Parameters in Global Coordinates
- Covariance in Global Coordinates

$(l_0, l_1, \varphi, \theta, q/p)$ Local

(x, y, z, px, py, pz) Global

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

$p_x = p \cos\varphi \sin\theta$, $p_y = p \sin\varphi \sin\theta$

$p_z = p \cos\theta$, $q = \text{sign}(q/p)$

Under development

```
#include <KFParticle/KFPTrack.h>

  KFPTrack& t;

  t.SetParameters(xyzp);
  t.SetCovarianceMatrix(CovXyzp);
  t.SetCharge(charge);
```

$C_{(l_0, l_1, \varphi, \theta, q/p)}$

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

Jacobian Matrix (J)

Global

$$C_{(x,y,z,px,py,pz)} = J\, C_{(l_0,l_1,\varphi,\theta,q/p)}\, J^T$$

Summary

- Helix swimming is already available in EICRecon but ignores the track covariances.
- In the meantime, a Chi2-fitting method is used in the analysis for D^0 and Λc^+, which uses track parameters and covariances and also allows access to the chi2 of the fit
- AMVF can be very useful with timing information (pull request not merged yet)
- KFParticle can be further helpful and is widely used for heavy-flavor analyses in the community (still development)

Thank You !

Secondary Vertex Reconstruction (D^0)

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

For a given track: $x = -l_0 \sin\phi$, $y = l_0 \cos\phi$, $z = l_1$

$\sigma_x^2 = \sin^2\phi\cdot\sigma_{\ell_0}^2 + \ell_0^2\cos^2\phi\cdot\sigma\phi^2 + 2\cdot\ell_0\sin\phi\cos\phi\cdot\mathrm{Cov}(\ell_0,\phi)$
$\sigma_y^2 = \cos^2\phi\cdot\sigma_{\ell_0}^2 + \ell_0^2\sin^2\phi\cdot\sigma\phi^2 - 2\cdot\ell_0\sin\phi\cos\phi\cdot\mathrm{Cov}(\ell_0,\phi)$
$\sigma_z^2 = \sigma_{\ell_1}^2$

Chi2 Minimization Code

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

```
// 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

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

Covariance Matrix in ACTS

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] = 3.89044e-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)

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

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

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