Vertex Fitting at BESIII

Yupeng Pei (裴宇鹏)

(On behalf of BESIII Collaboration)

University of Science and Technology of China

2025-11-24

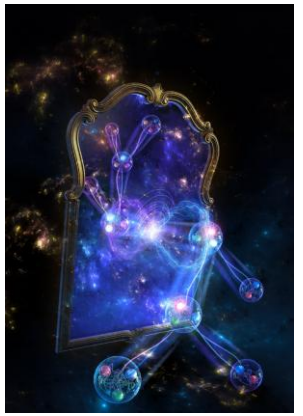
Outline

- Introduction
- 3 Vertex Fitting Algorithm in BESIII
  - VertexFitAlg (From Xu Min)
  - VertexFitRefineAlg (From Haokai Sun)
  - VertexFitUpgradeAlg (From Yupeng Pei)
- Secondary Vertex Fit
- Summary

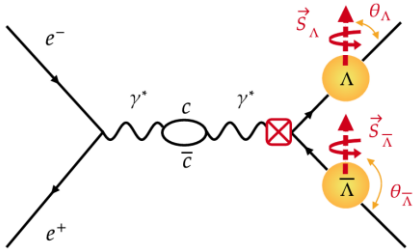
Introduction

- 10^{10} J/ψ data at BESIII ! ⇒ Hyperon factory ⇒ Many interesting physics
  - Hyperon CP violation test
  - Rare decay (Weak radiative decay and Semi-leptonic decay)
  - Hyperon EDM
- Hyperon: weak decay ⇒ long-lived particles
- More hyperon samples ⇒ High precision results ⇒ Need better performance of vertex reconstruction

Nature **606**, 64–69 (2022)



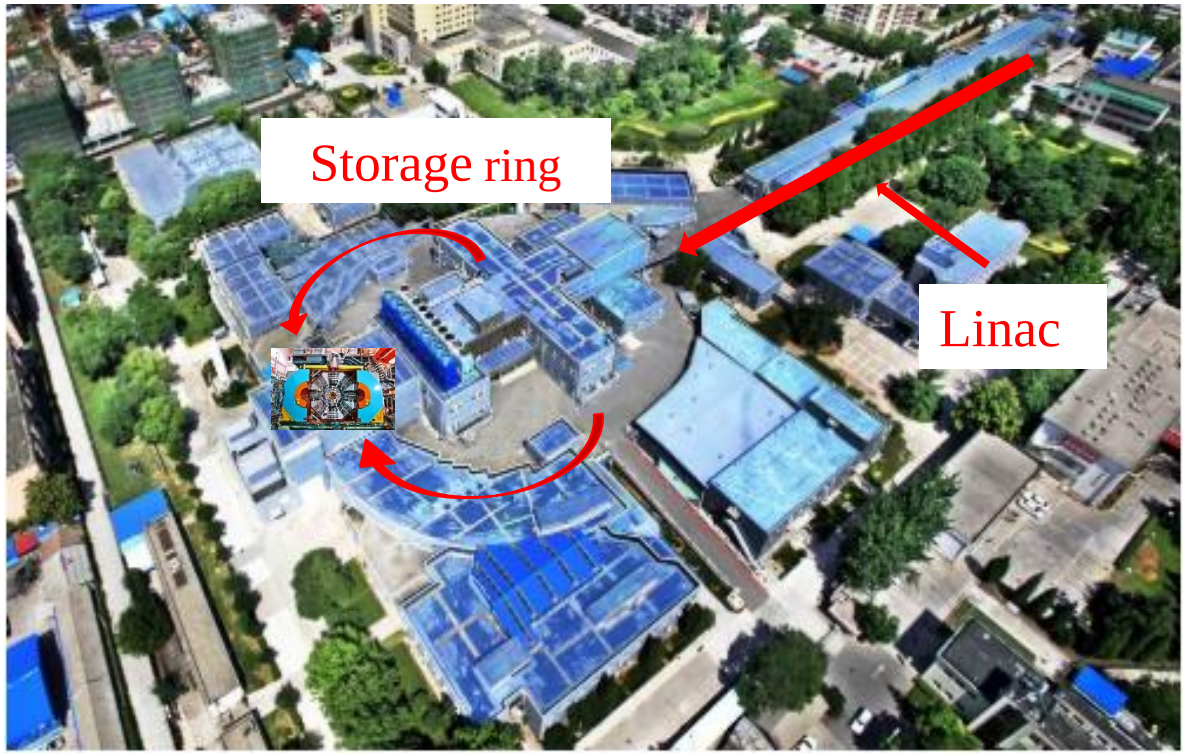
Λ EDM, arXiv:2506.19180



Hyperon	$Br(J/\psi \to Y\bar{Y})(\times 10^{-3})$	$N_B(\times 10^6)$	$c\tau(cm)$	Decay Mode
Λ	1.89 ± 0.09	18.9 ± 0.9	7.89	$\Lambda \to p\pi^-$ (63.9%)
Σ^+	1.07 ± 0.04	10.7 ± 0.4	2.40	$\Sigma^+ \to p\pi^0$ (51.6%)
Σ^0	1.17 ± 0.03	11.7 ± 0.3	~ 0	$\Sigma^0 \to \gamma\Lambda$ (~100%)
Ξ^-	0.97 ± 0.08	9.7 ± 0.8	4.91	$\Xi^- \to \Lambda\pi^-$ (~100%)
Ξ^0	1.17 ± 0.04	11.7 ± 0.4	8.71	$\Xi^0 \to \Lambda\pi^0$ (~100%)

Statistical uncertainty ~ 0.1% level!

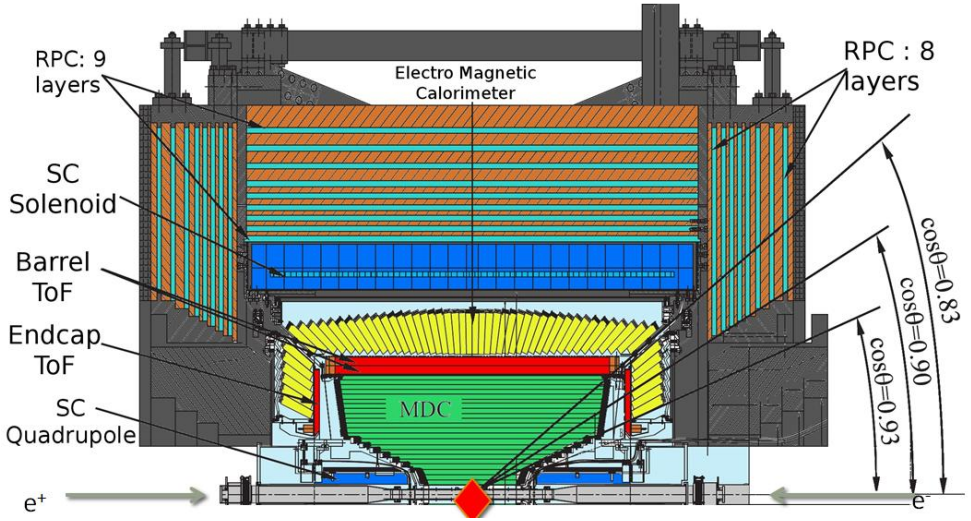
BEPCII and BESIII



Double ring: e^+ and e^-
Cross angle: 22 mrad
E_{cm}: $1.84 - 4.95$ GeV
Peak luminosity: 1.1×10^{33} cm^{-2}s^{-1} @ $\psi(3770)$

Electromagnetic Calorimeter
CsI(Tl): L = 28 cm
- Barrel $\sigma_E/E = 2.5\%$ @ 1 GeV
- Endcap $\sigma_E/E = 5.0\%$ @ 1 GeV

Muon Counter RPC
Barrel: 9 layers
Endcaps: 8 layers
- σ_{spatial} = 1.48 cm



Main Drift Chamber
Small cell, 43 layer
- σ_{xy} = 130 μm
- dE/dx ~ 6%
- σ_p/p = 0.5% @ 1 GeV/c

Time Of Flight
Plastic scintillator
- σ_T(barrel) = 68 ps
- σ_T(endcap) = 110 ps

(update to 60 ps with MRPC)

VertexFitAlg

- **Basic requirements**: Multi-track intersect at the same point
- **Method**: Lagrange Multiplier-Based Least Squares Method

Constraint function

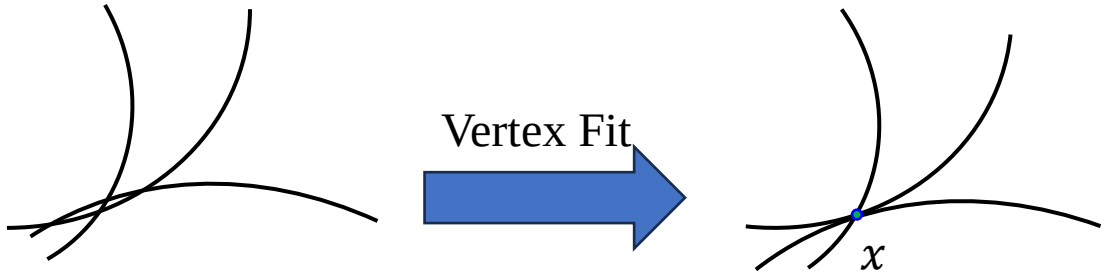
$$\chi^2 = (\alpha-\alpha_0)^T V_{\alpha 0}^{-1}(\alpha-\alpha_0) + (x-x_0)^T V_{x0}^{-1}(x-x_0) + 2\lambda^T(D\delta\alpha + E\delta x + d)$$

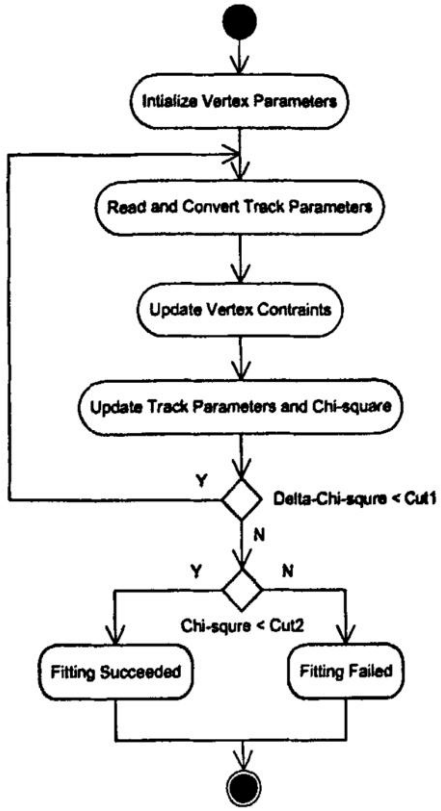
Intersect to 1 point

α: Track parameters ($7n$)

x: Vertex parameters (3)

- **Results**: vertex x and updated track pars

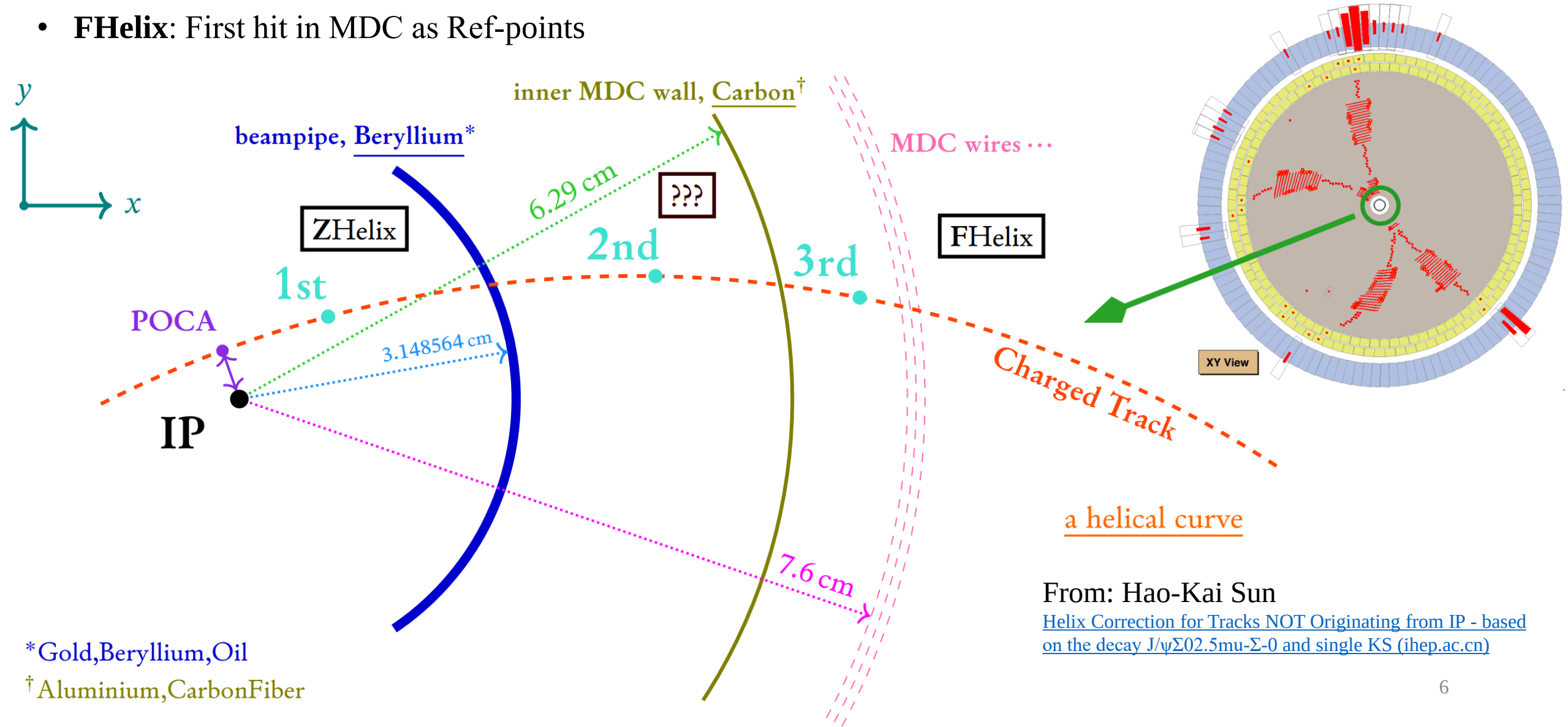




- Long-lived particles reconstructed by VertexFitAlg
  - $\Lambda \to p\pi^-$
  - $\Xi^- \to \Lambda\pi^-$

Track Helix Type at BESIII

- **ZHelix**: Default way; POCA as Ref-points; Material effects including from BP to MIW
- **FHelix**: First hit in MDC as Ref-points

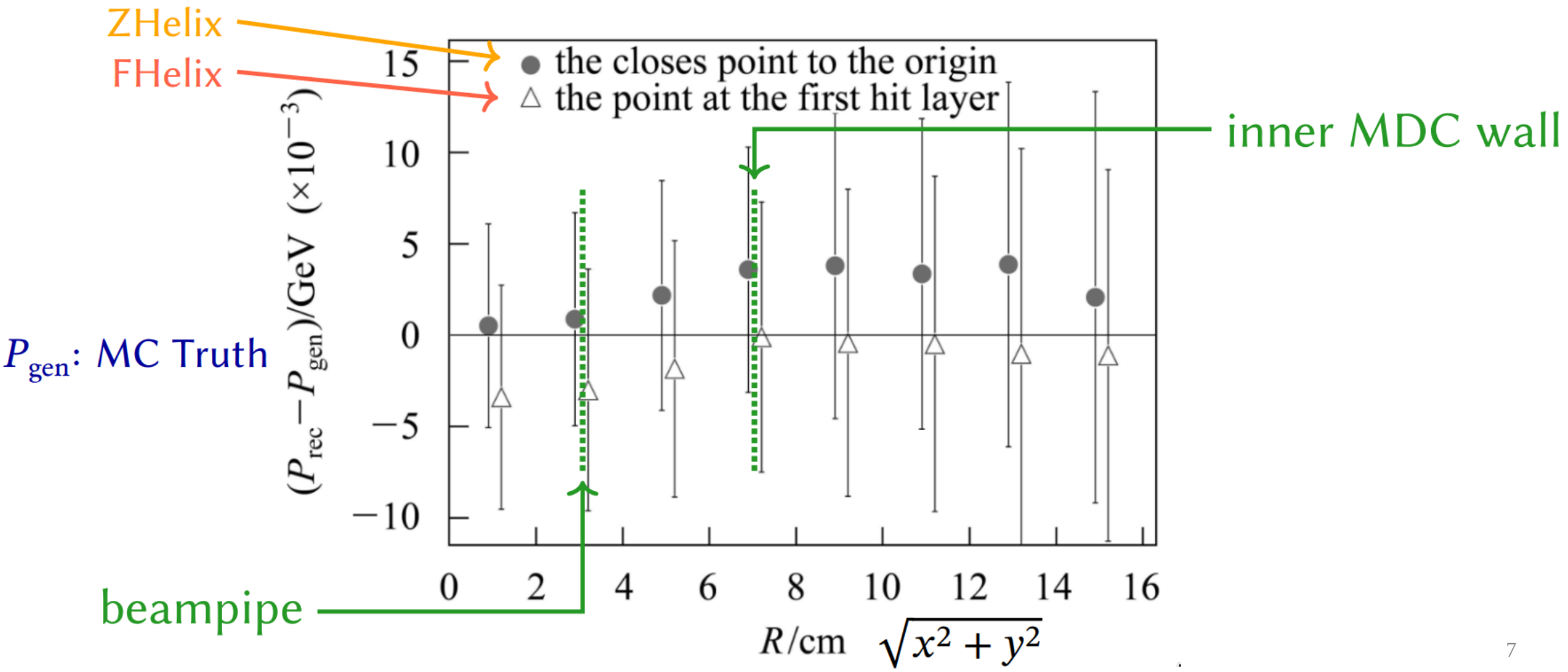


From: Hao-Kai Sun

Helix Correction for Tracks NOT Originating from IP - based on the decay J/ψΣ02.5mu-Σ-0 and single KS (ihep.ac.cn)

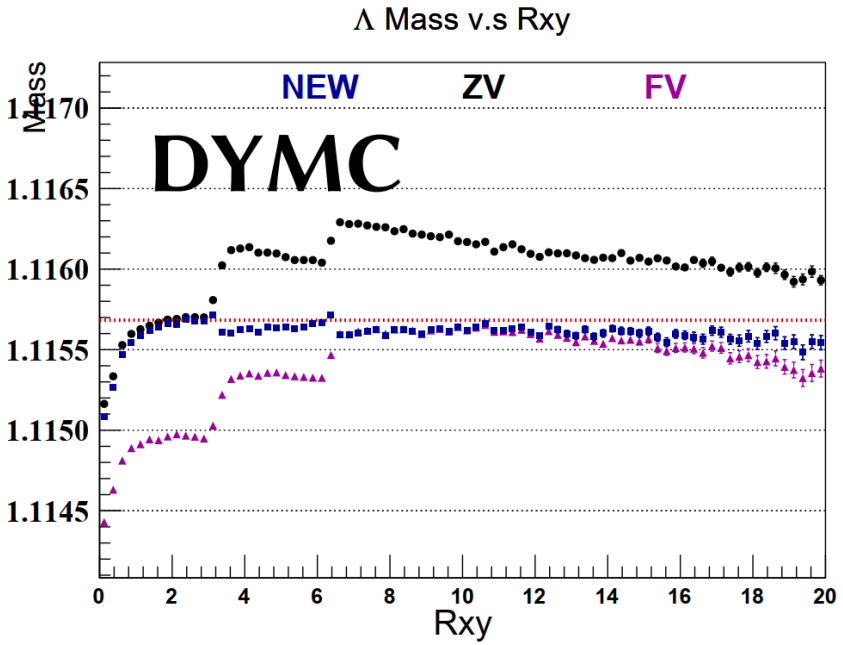
Issues of VertexFitAlg for Long-lived Particles

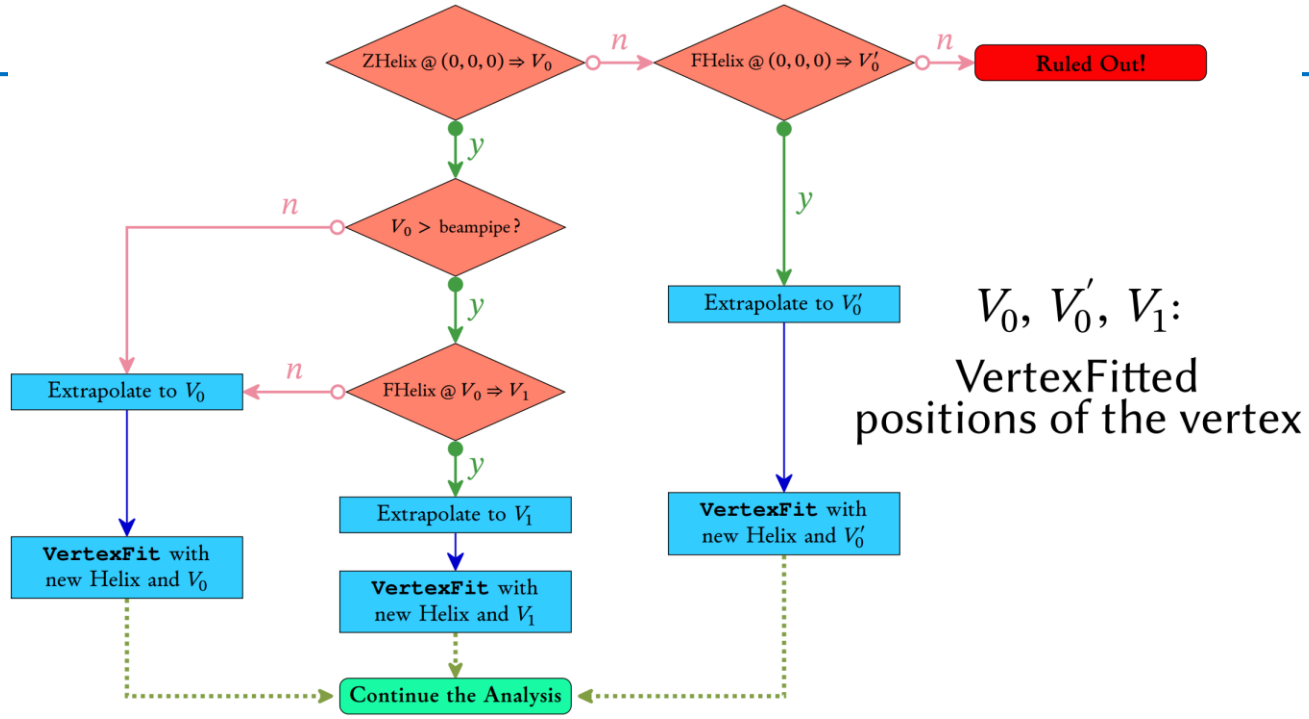
- No suitable helix type from BP to MIW Extrapolate Alg?
- Using helix type need hypothesis Decay point auto-finding ?

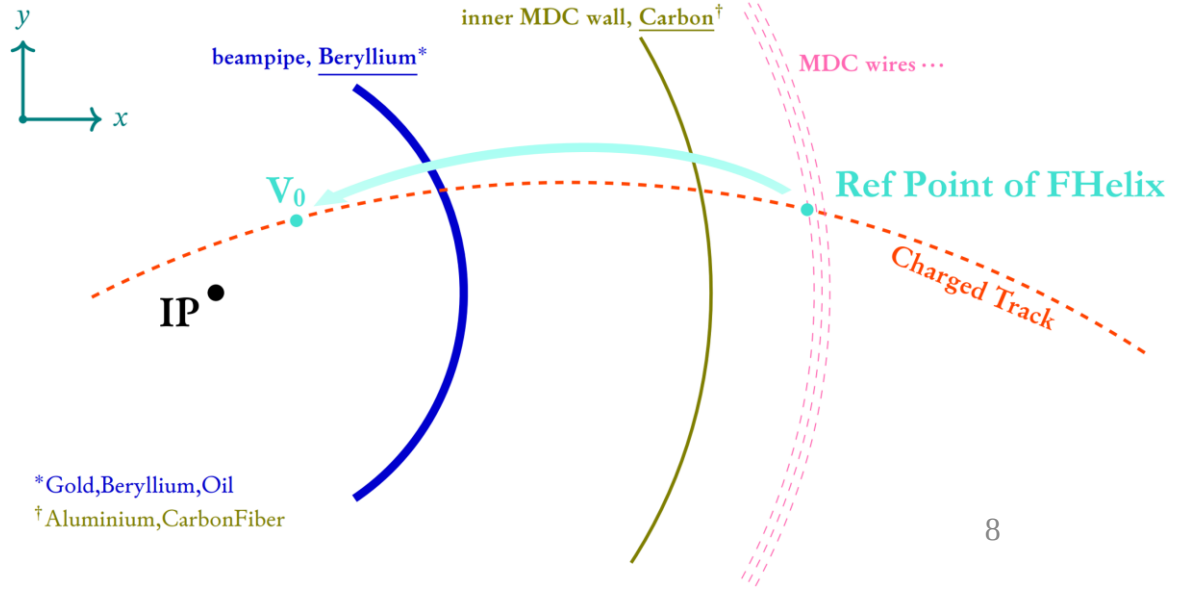


VertexFitRefineAlg

- A unified algorithm to schedule different helix types.
  - **Point Finding**: iteration by VertexFitAlg
  - **Extrapolation**: correct material effects
  - **Fit**: newhelix as input to do VertexFitAlg

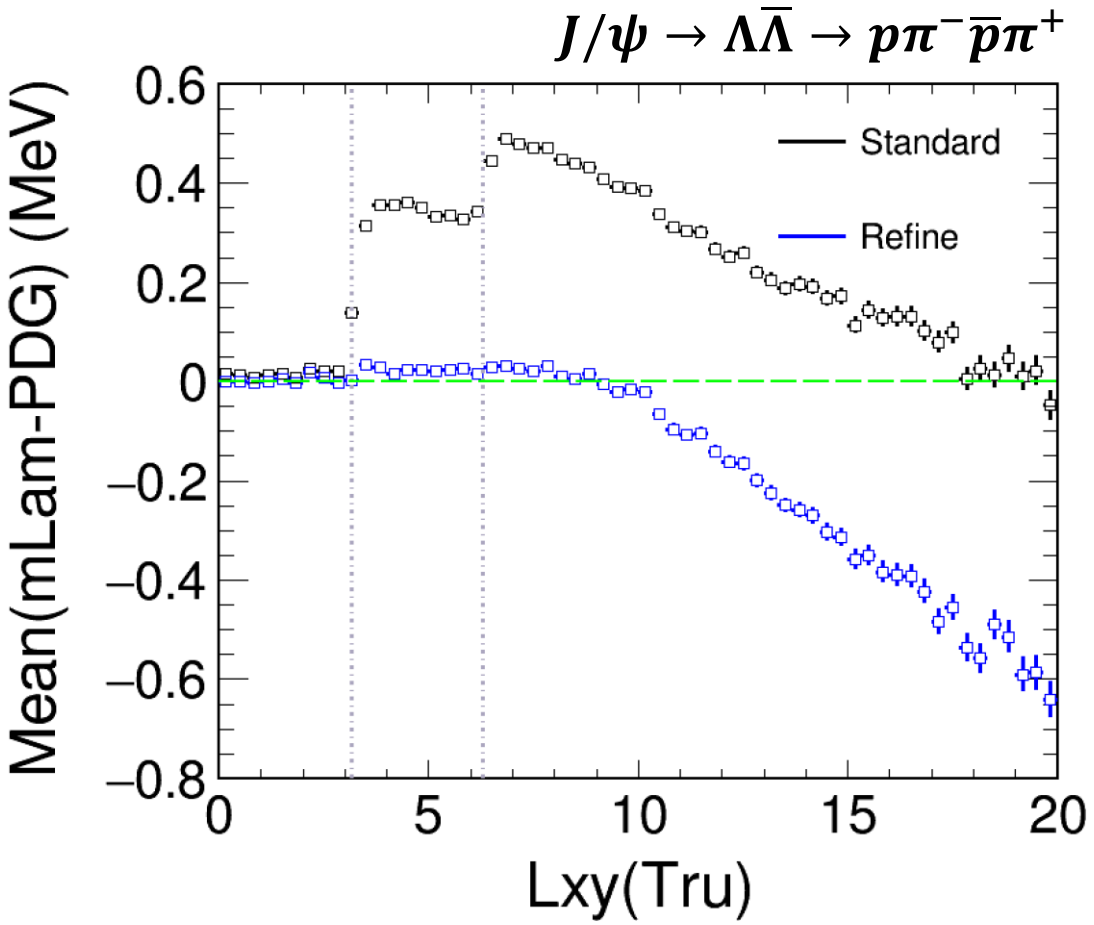


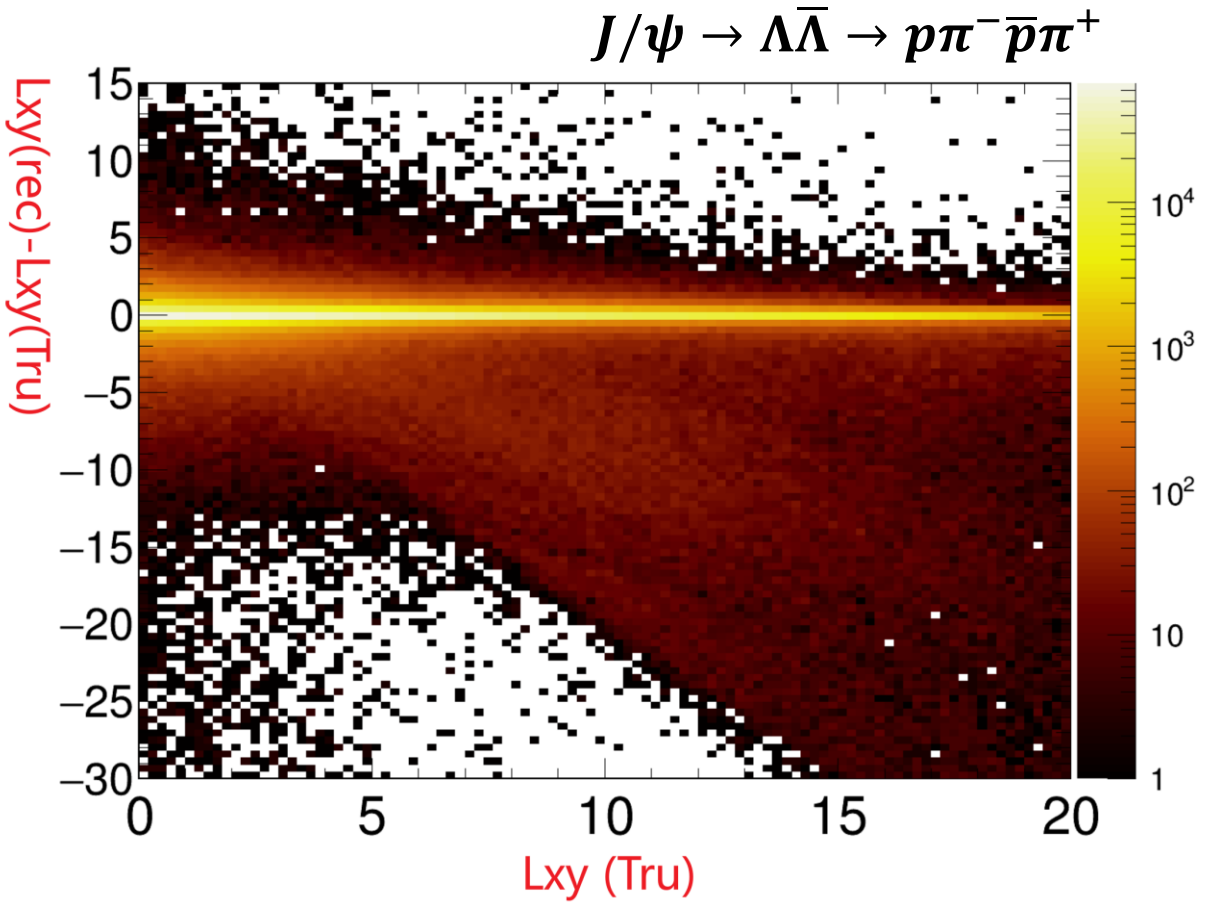




Issues with VertexFitRefineAlg

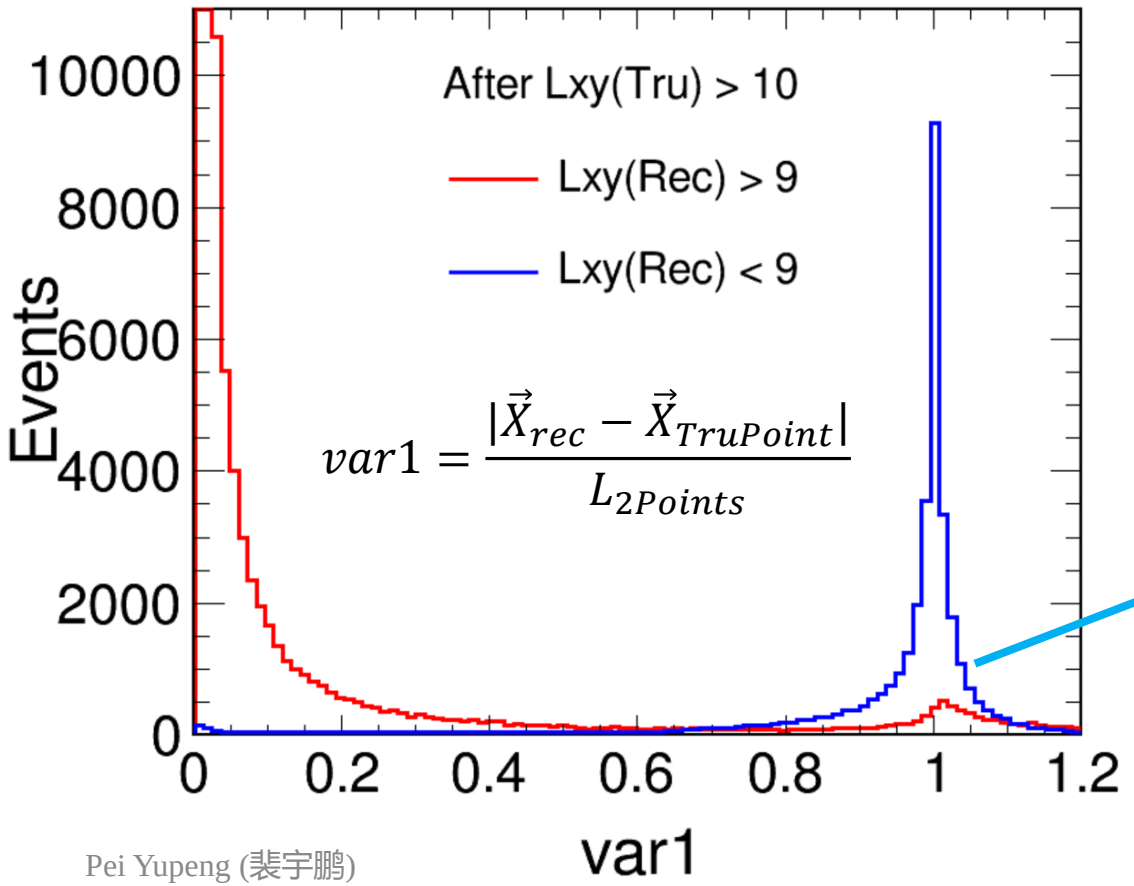
- Λ Mass systematic shift when $L_{xy} > 10$ cm
- From VertexFitRefineAlg, Λ far from IP decay ⇒ reconstruct smaller decayL ⇒ wrong mass

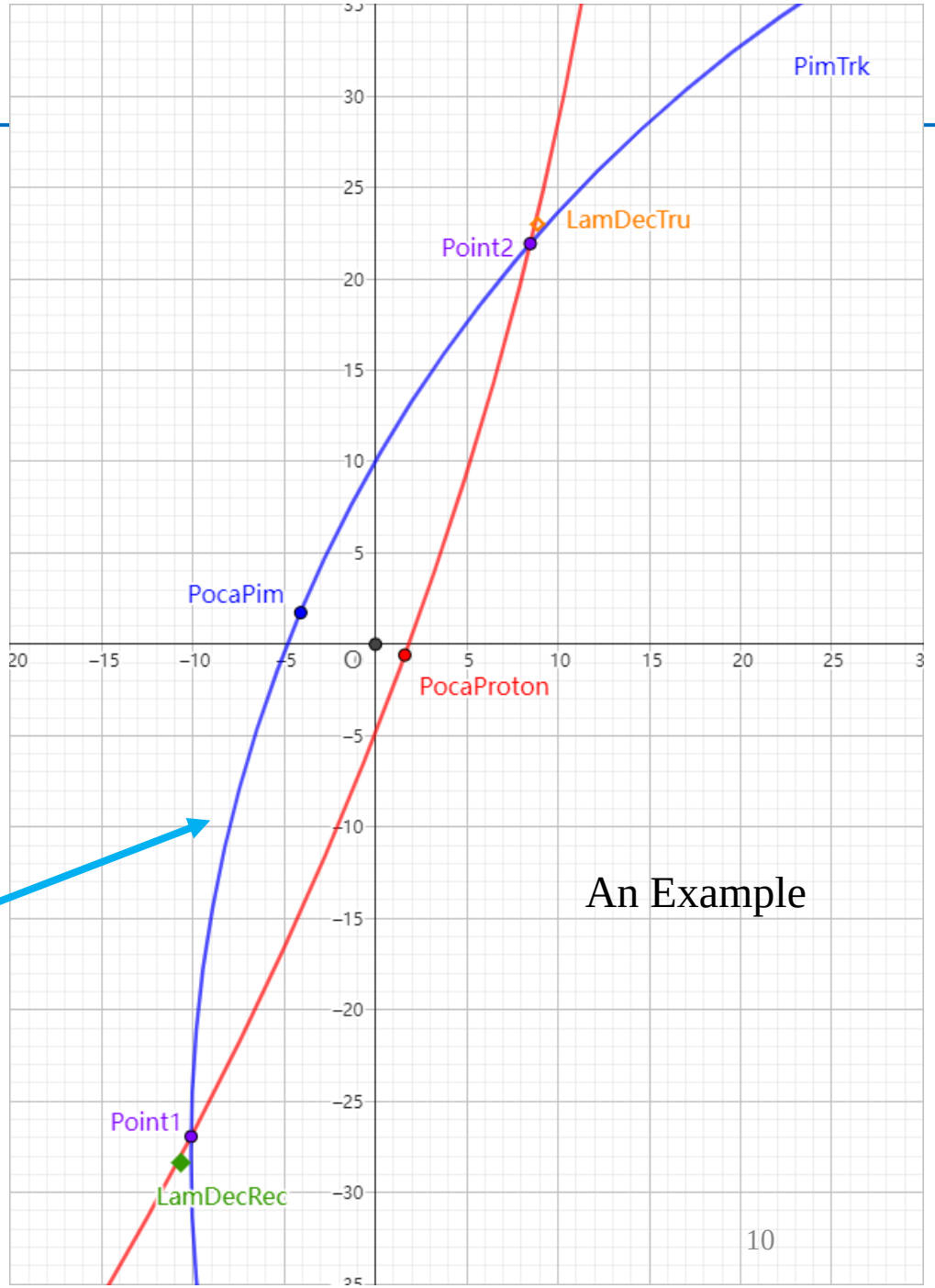




Wrong Vertex Finding

- 2 intersection points in xoy, vertex fit gives a wrong vtx
- Wrong vertex ⇒ wrong ext ⇒ wrong mass
- Unsuitable initial value ⇒ wrong points





What's Wrong with Input Track Parameter ?

- Input of VertexFitAlg: **WTrackParameter** (7 pars: $E, p_x, p_y, p_z, x, y, z$)
- General WTrackParameter: $(\boldsymbol{P}, \vec{x})$ @ $Poca$

```
WTrackParameter StdTrkPar_proton = WTrackParameter(
                                   Mproton,
                                   KalTrk_proton->getZHelixP(),
                                   KalTrk_proton->getZErrorP() );
```

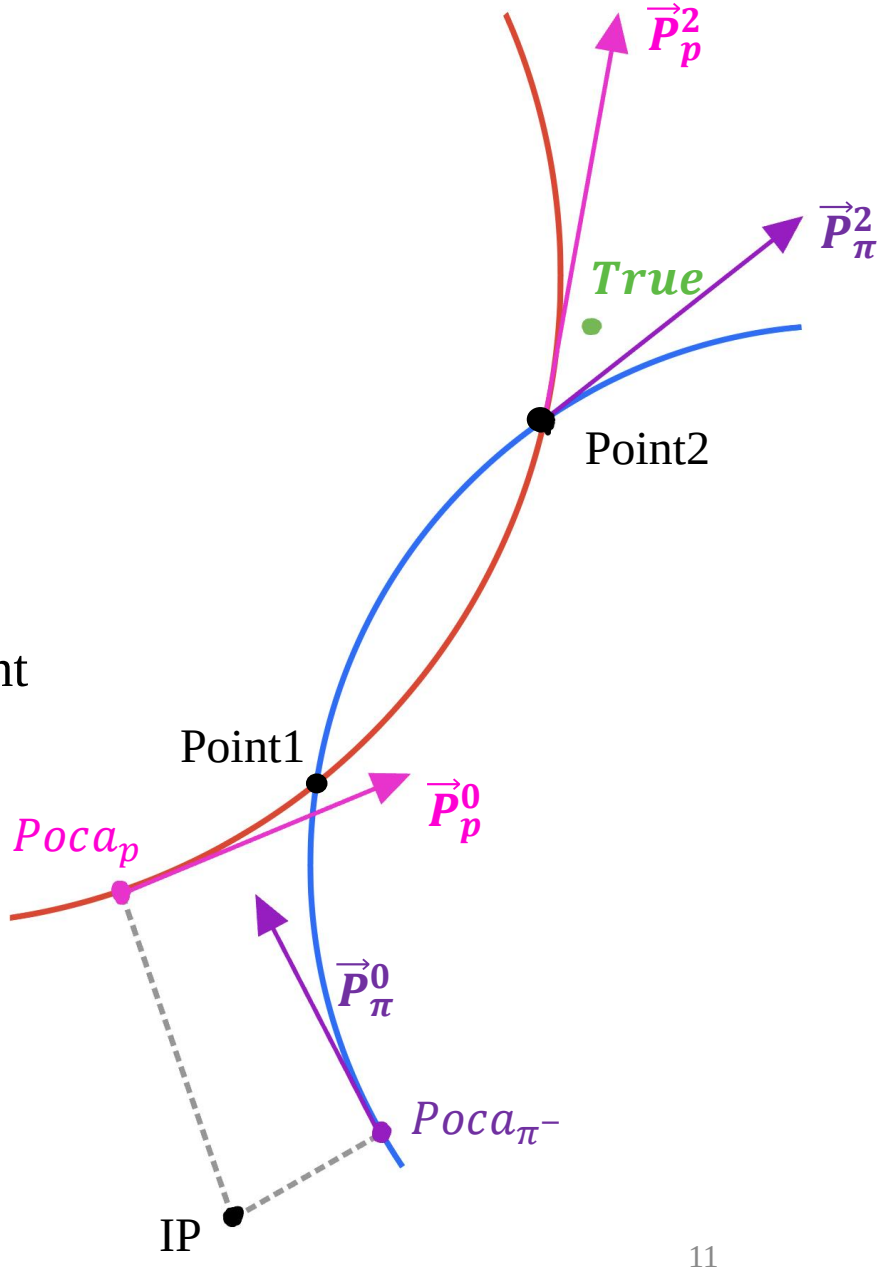
FHelix and NewHelix also @ $Poca$

- Poor z resolution @ BESIII ⇒ Two solutions of VertexFitAlg

  For far decay, initial value far away from true point ⇒ Wrong decay point

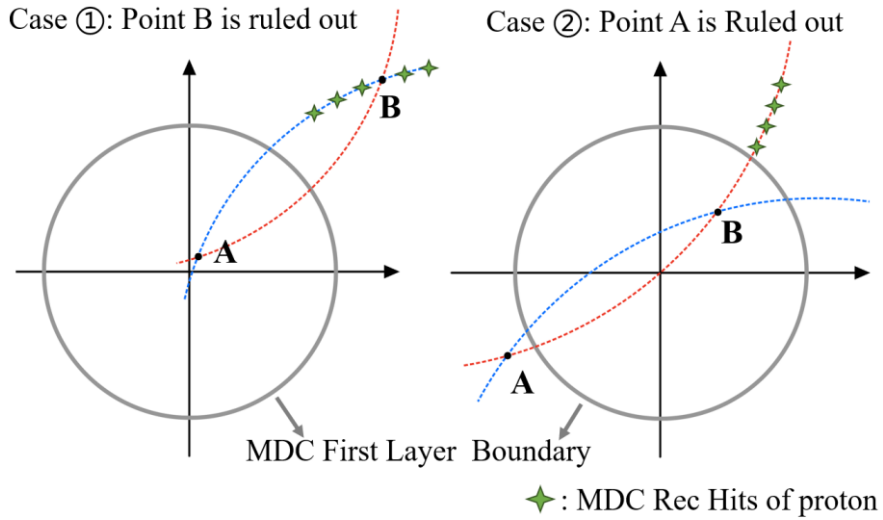
- VertexFitRefineAlg need upgrade:
  - **Point Finding**: ~~iteration by VertexFitAlg~~ Need new method
  - **Extrapolation**: correct material effects
  - **Fit**: newhelix as input to do VertexFitAlg

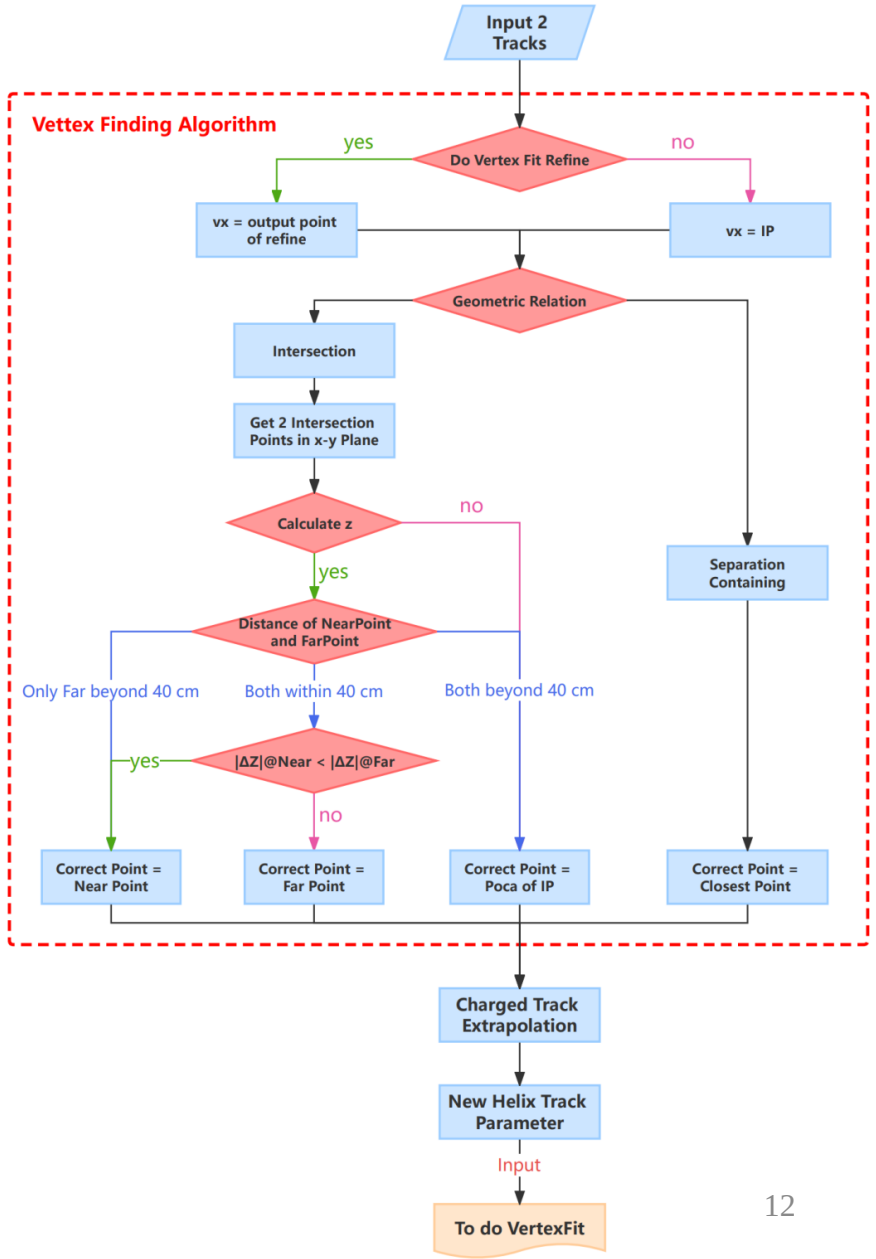
    WTrackParameter need Modify



VertexFitUpgradeAlg

- 2 charged tracks or 1 charge + 1 neutral virtual
- Similar with VertexFitRefineAlg, some upgrades:
  - **Point Finding**: compare $|\Delta z|$ @ 2 points
  - **Extrapolation**: correct material effects
  - **Construct New Helix**: helix pars @ decay point
  - **Fit**: newhelix as input to do VertexFitAlg
- About Point Finding Alg:
  - **Compare χ^2**: not working
  - **Using Hits Info**: not include from standard official output

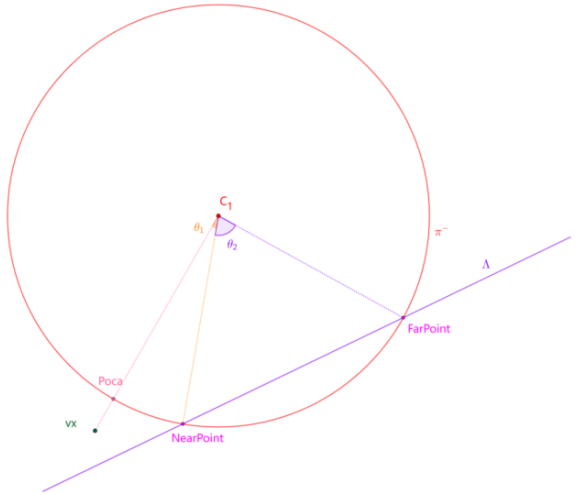


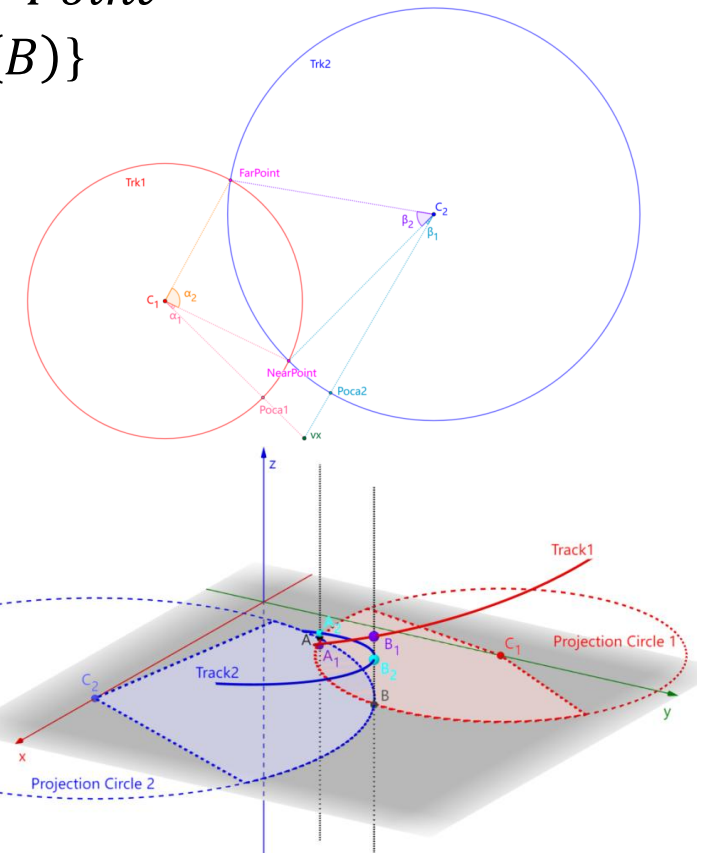


Point Finding

Intersection

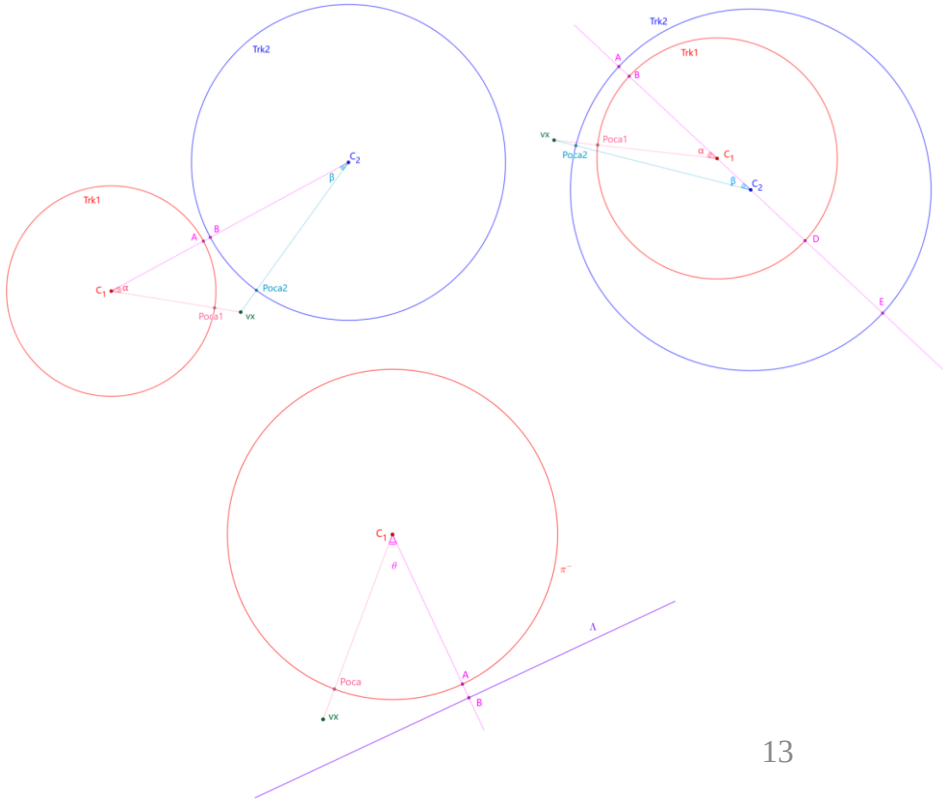
- 2 decay point solutions in xoy → How to select the correct one?
- 2 solutions in xoy; but 2 helixes only have 1 point in xyz.
  - $\Delta Z(Point) = \left|z_p - z_{\pi^-}\right|$ @ $Point$
  - Select the min$\{\Delta Z(A), \Delta Z(B)\}$





Separation or Containing

- Only one solution
- Nearest point between 2 circles

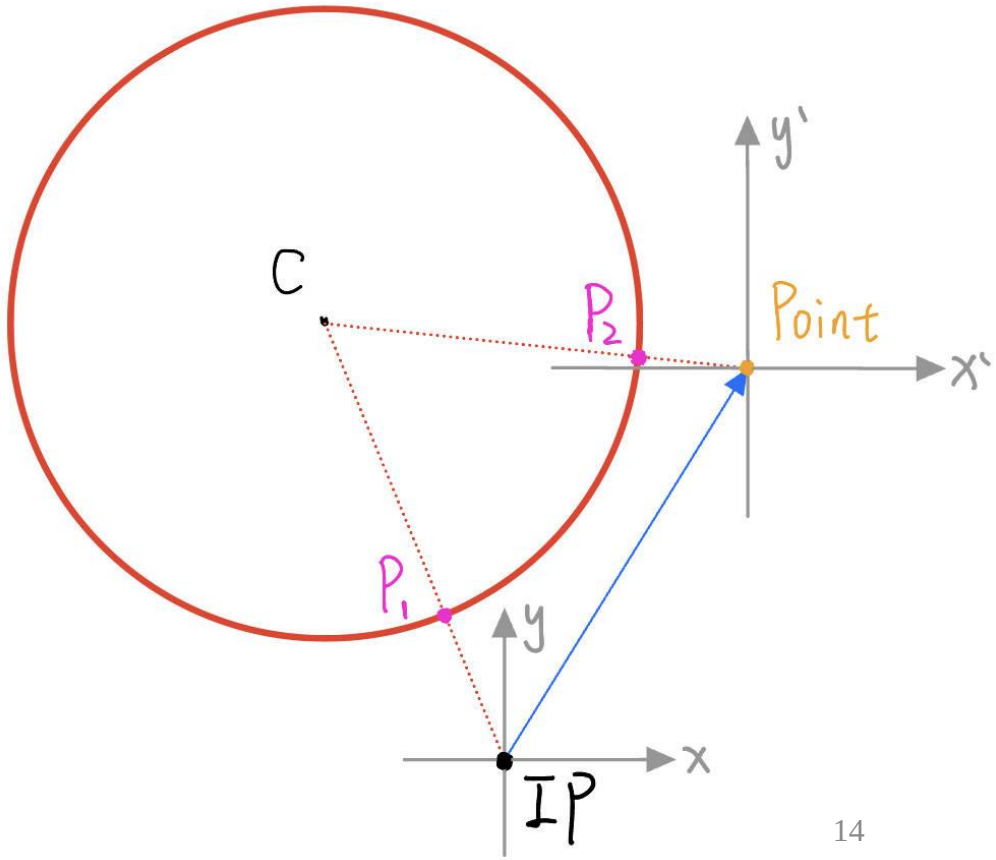


Construct New Helix

- Using **point** as a reference point, the 5-parameters and error of helix are obtained, coordinate frame as $x'y'z'$
- Set the WTrackParameter of track, 7-parameters (position and 4-momentum at point P_2)
- Coordinate transformation: Point($x'y'z'$) → IP(xyz)
- 7-parameters error matrix is invariant in the transformation

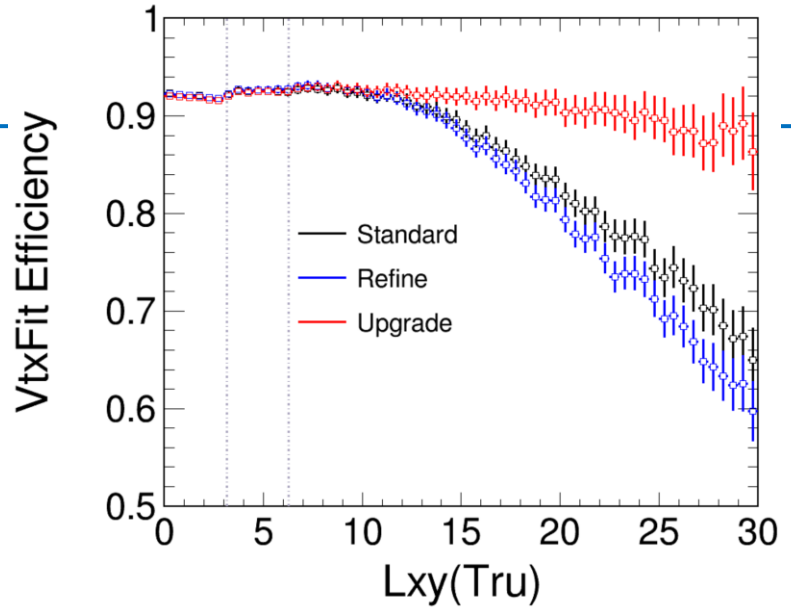
$$\vec{r_c} = Point - IP$$
$$\vec{P} = \vec{P}'$$
$$\vec{x} = \vec{x}' + \vec{r_c}$$
$$Ew = Ew'$$

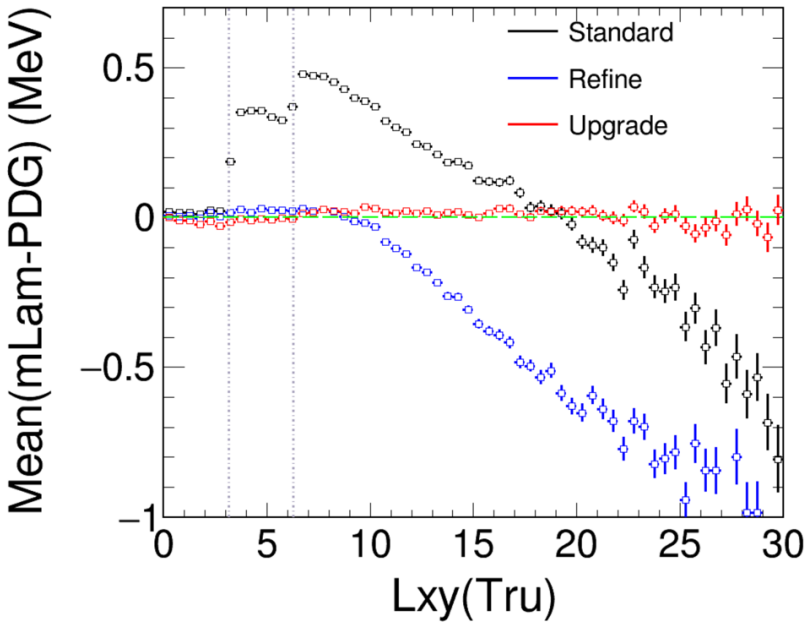
```
//pivot to point
fitTrack.pivot(point);

Hep3Vector PosVec = PointToVec(point);          Wpar of track when pivot as point

WTrackParameter wpar = WTrackParameter(xmassTmp[pid], fitTrack.a(), fitTrack.Ea());
HepVector mw = wpar.w();
HepSymMatrix mEw = wpar.Ew();

for ( int i = 0; i < 3; i++ )                   Coordinate Transformation
  mw[i+4] += PosVec[i];
wpar.setW(mw);

setTrackParameter( wpar );
```

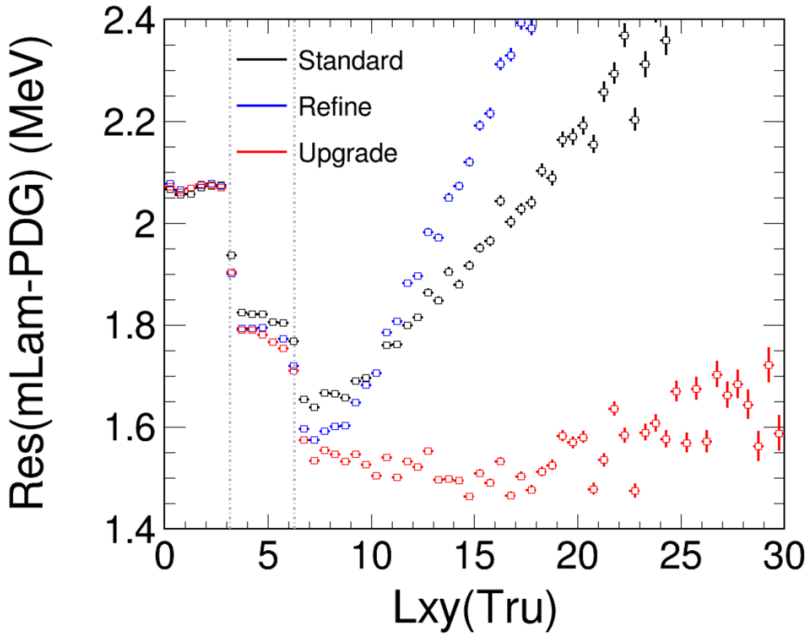


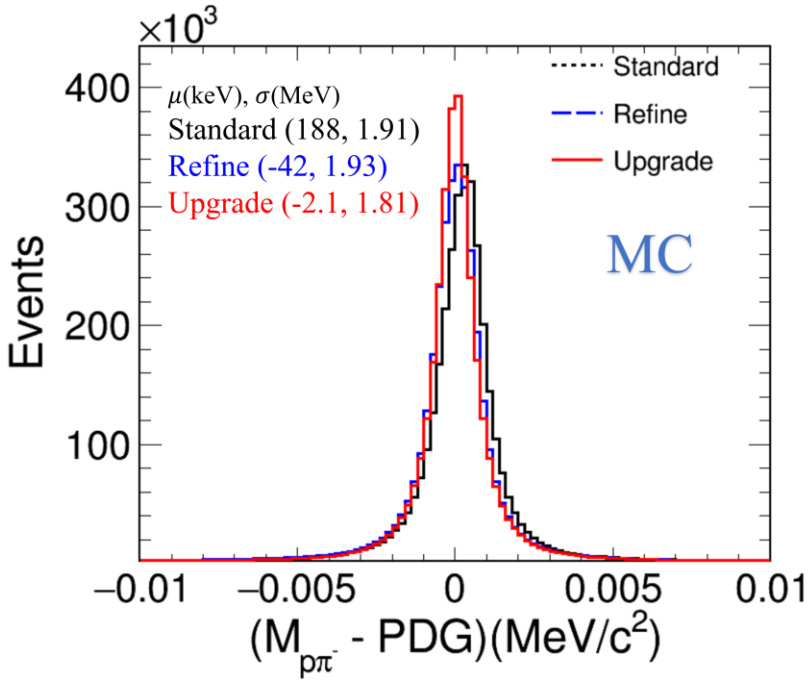
Validation by $J/\psi \to \Lambda\bar{\Lambda}$

- Solve the Λ mass shift issues when $L_{xy}(Tru) > 10$ cm.
- VertexFitUpgradeAlg gives best mass resolution



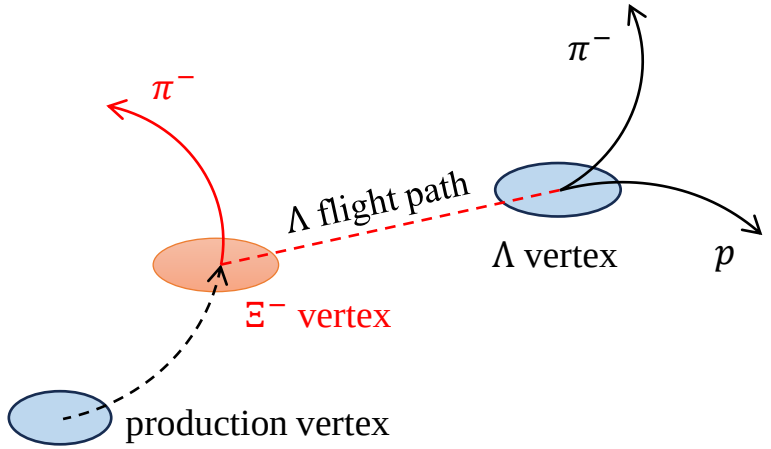




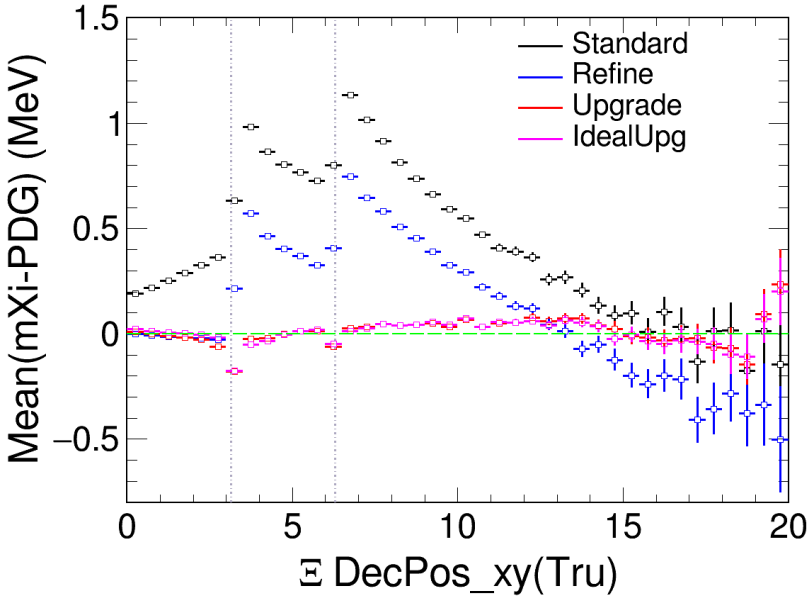


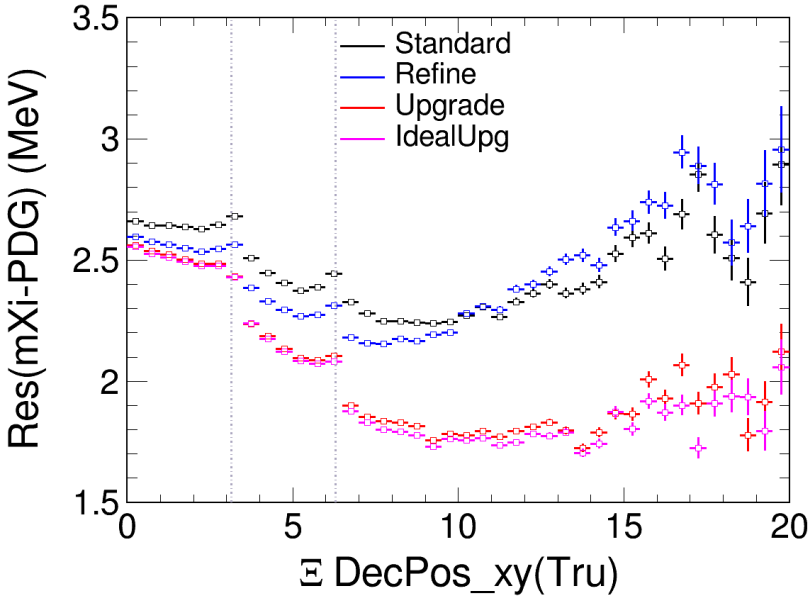
Validation by $\Xi^- \to \Lambda\pi^-$

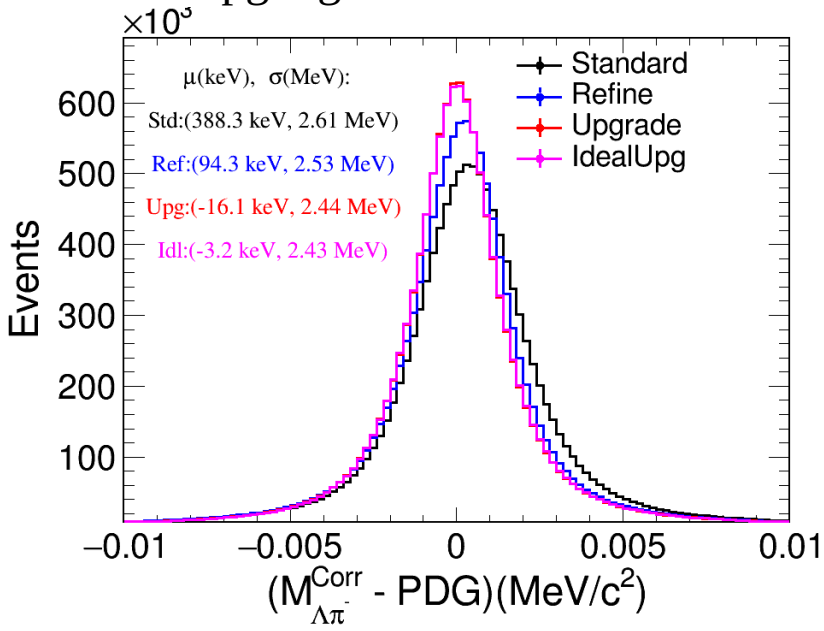
- Works well for 1 charged track and 1 neutral virtual track



IdealUpg: the decay point replaced to the truth, representing the optimal performance of VtxFitUpgAlg





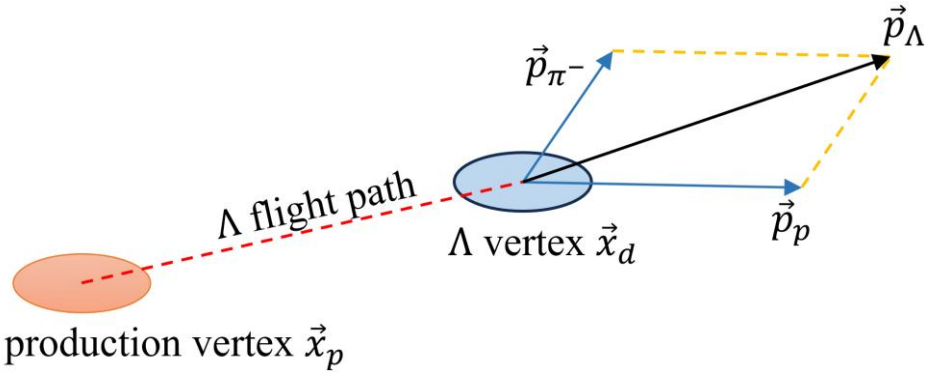


Secondary Vertex Fit

- **Basic requirements**: calculate the flight path of long-lived particles
- **Method**: Lagrange Multiplier-Based Least Squares Method

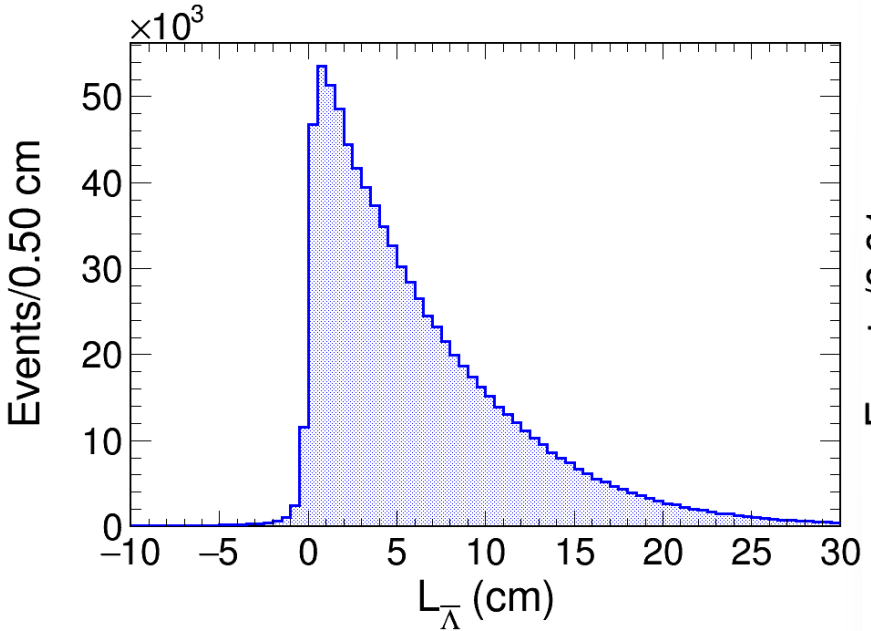
$$\chi^2 = (\alpha - \alpha_0)^T V_{\alpha 0}^{-1} (\alpha - \alpha_0) + 2\lambda^T (D\delta\alpha + E\delta c\tau + d)$$

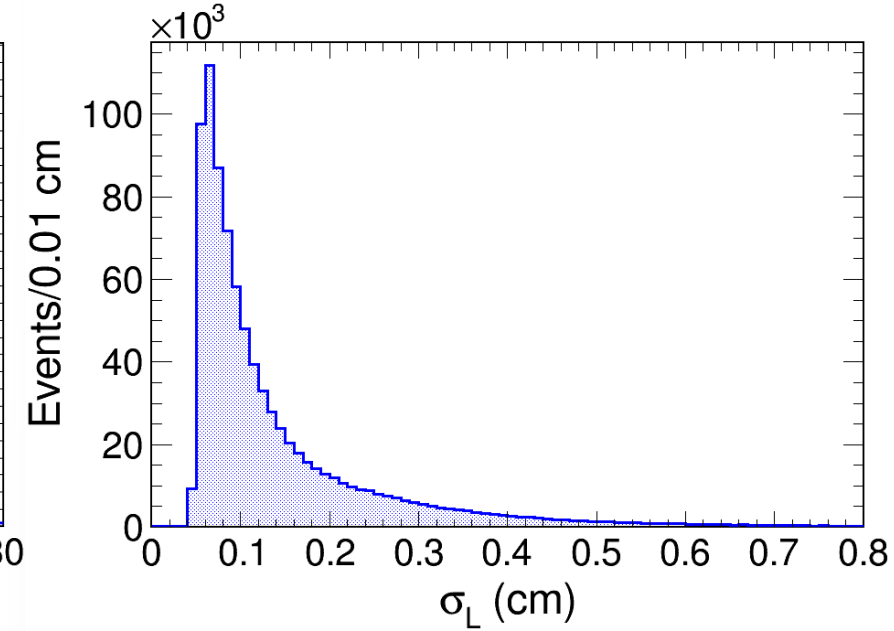
$$\alpha = (p_x, p_y, p_z, E, x_d, y_d, z_d, x_p, y_p, z_p) \qquad \vec{x}_d - \vec{x}_p = \frac{c\tau}{m}\vec{p}$$



p, π^- track pars don't update !
Need combinatorial Alg, see Mingyu's talk.

- **Results**: decay length $c\tau$ and updated Λ track pars





For $\Xi^- \to \Lambda(\to p\pi^-)\pi^-$:
4 vertex fits performed, 2 initial + 2 secondary, no unified parameters update

Summary

- Vertex fit algorithm after three iterations significantly improves the hyperons reconstruction.
- Still some issues remaining:
  - Can't process π^0/γ included fit. ($\Sigma^+ \to p\pi^0, \Xi^0 \to \Lambda\pi^0$)
  - Fit independently, no unified parameters update. ($\Xi^- \to \Lambda\pi^- \to p\pi^-\pi^-$)
  - Data/MC discrepancy handling
- More to say … … IP hypothesis of track reconstruction
  - Track helix type
  - WTrackParameter value setup

Vertex finding algorithm at event reconstruction level ?

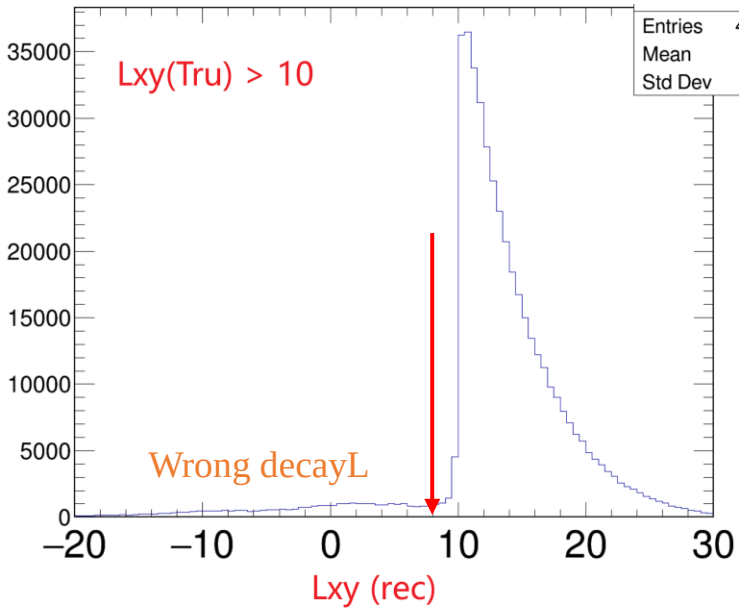
- Many valuable insights learned will benefit future collider experiments (STCF …)

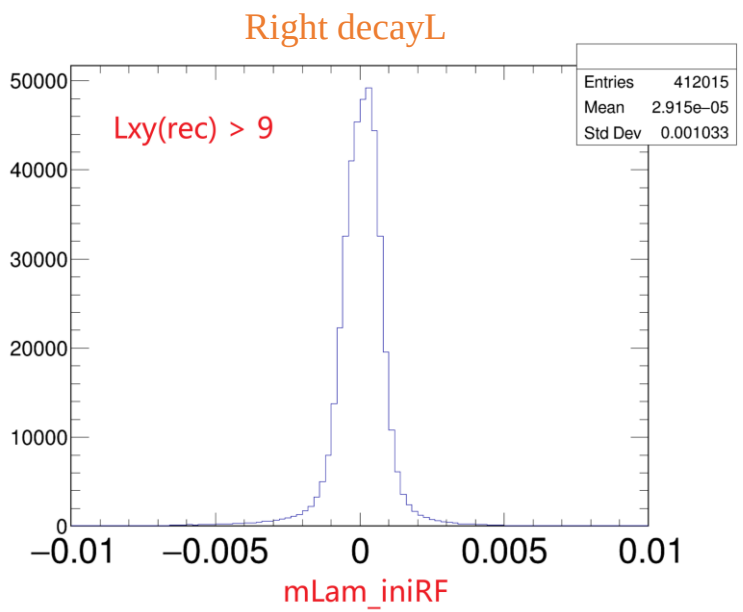
Thank you!

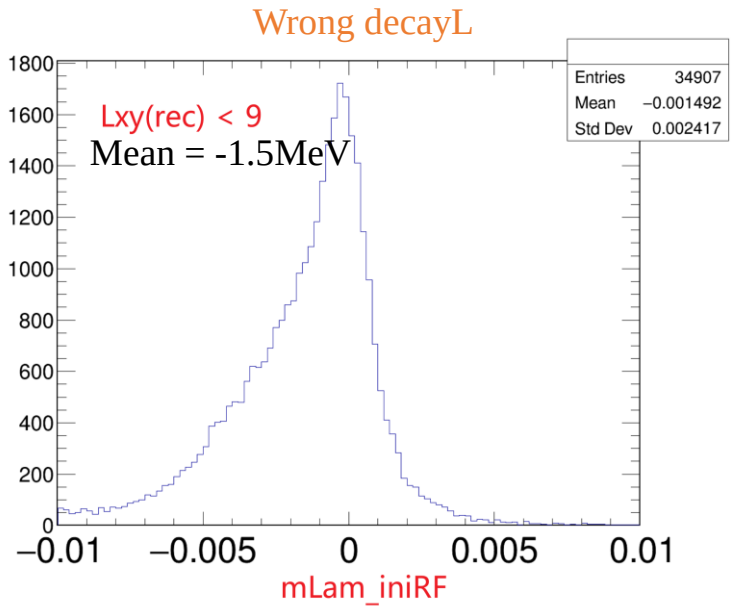
Back up

Wrong Decay Length

- For $L_{xy}(\mathrm{Tru}) > 10$ cm, 8% events reconstructed wrong decay length ⇒ Mass shift





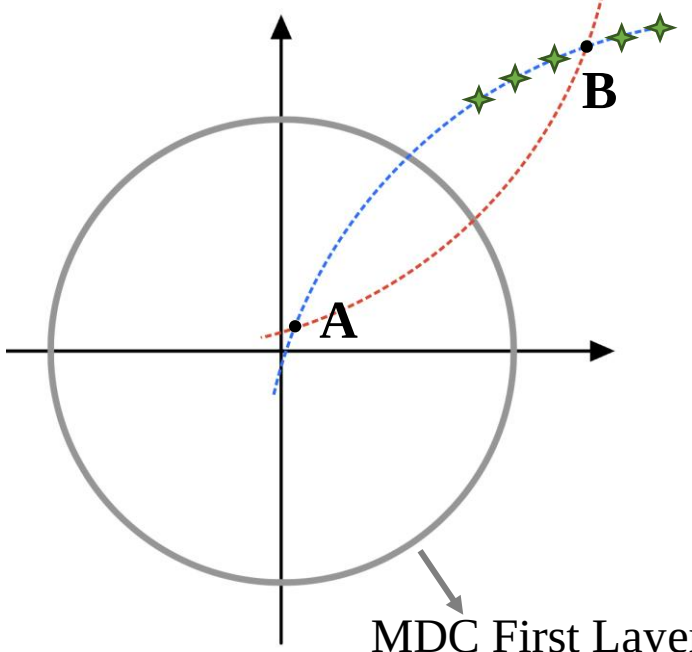


Use the MDC Hits Info

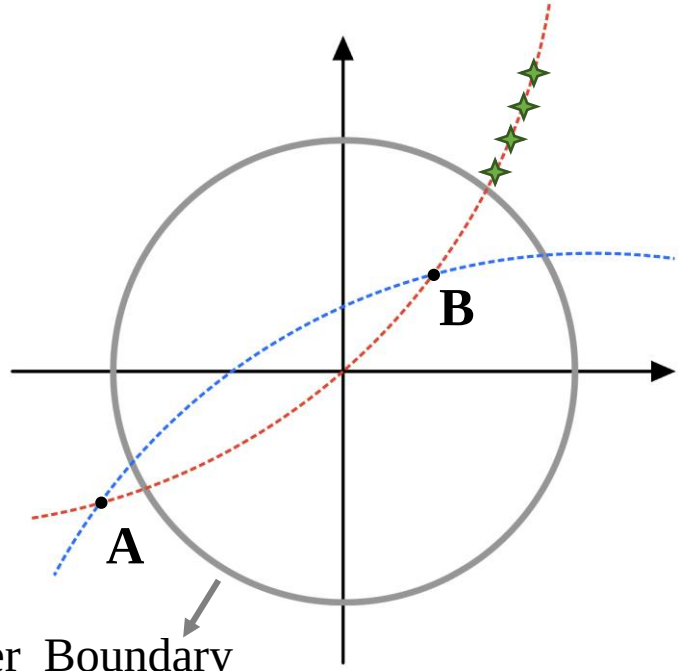
Except this slide, the results at other slides don't use the hit info

- Preliminary method: when the point is in the MDC ($L_{xy} > 7.885cm$ region):
  - ① There are hits around this point on both sides.
  - ② There are no hits around this point on both sides.
- The **MdcRecHits positions** of proton track are used, which can improve the point correct rate.

Case ①: Point B is ruled out

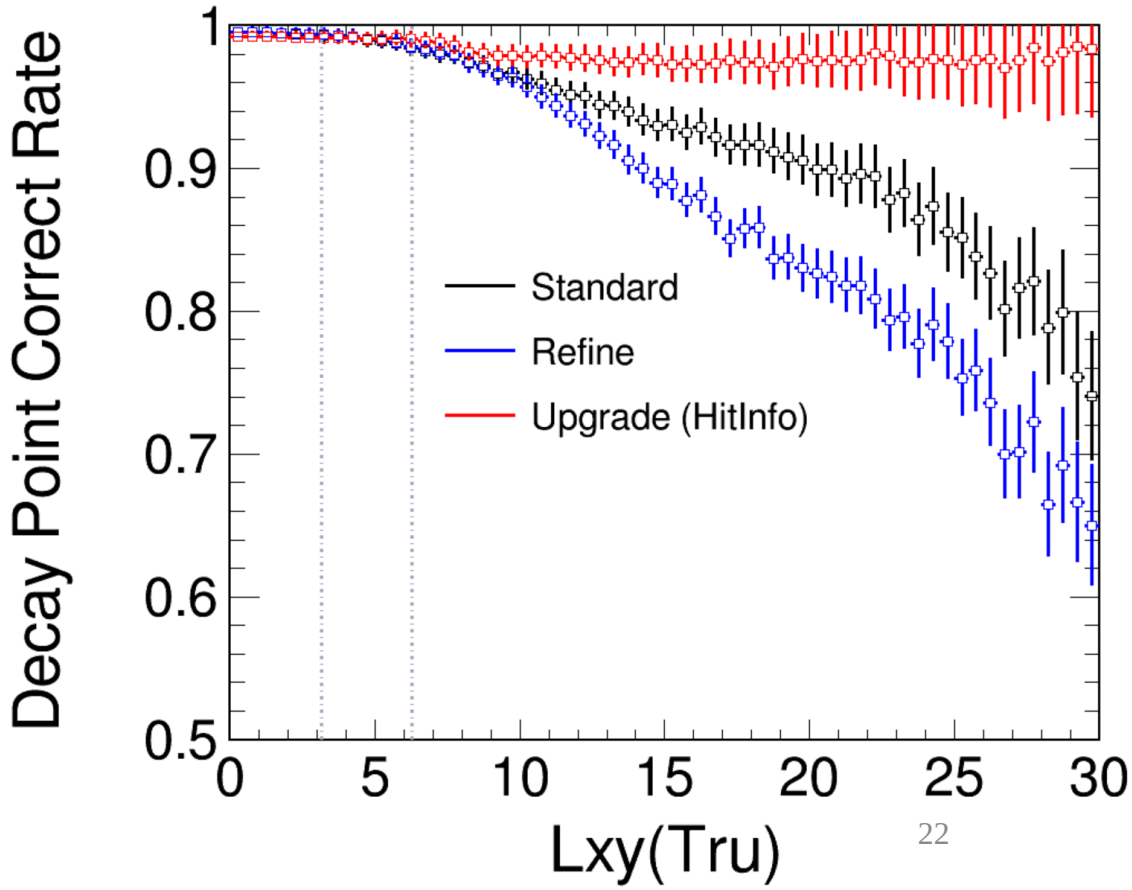


Case ②: Point A is Ruled out

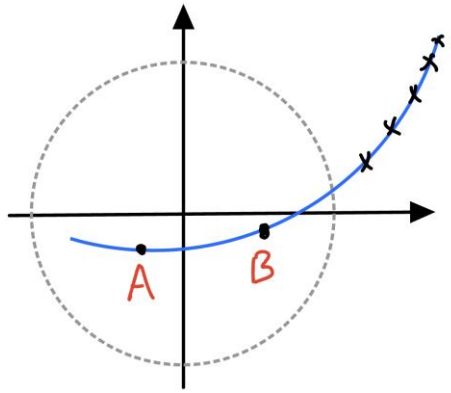


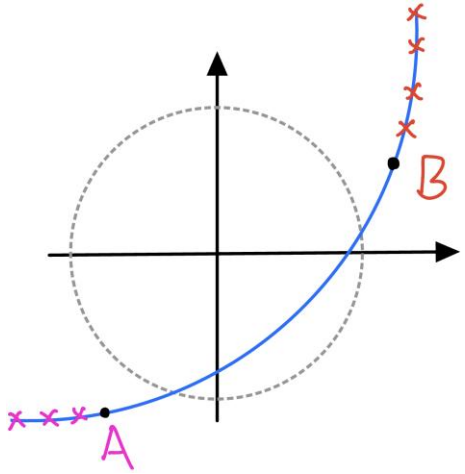
MDC First Layer Boundary

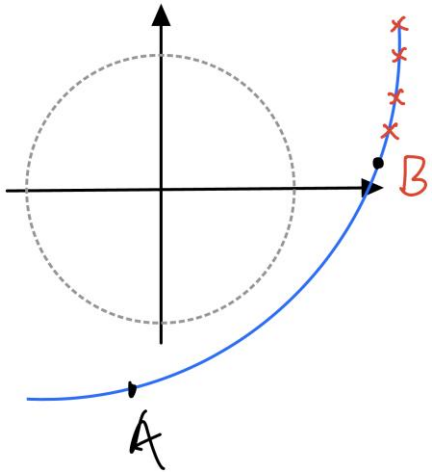
✦: MDC Rec Hits of proton

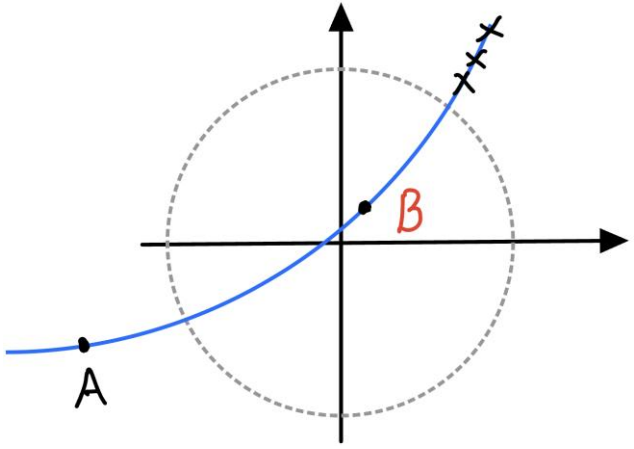


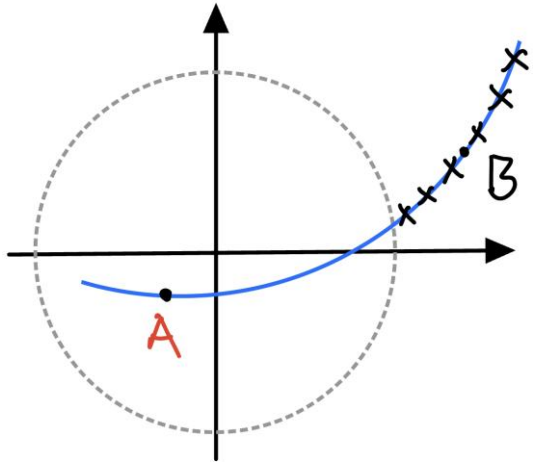
Preliminary Method of MdcHits Process

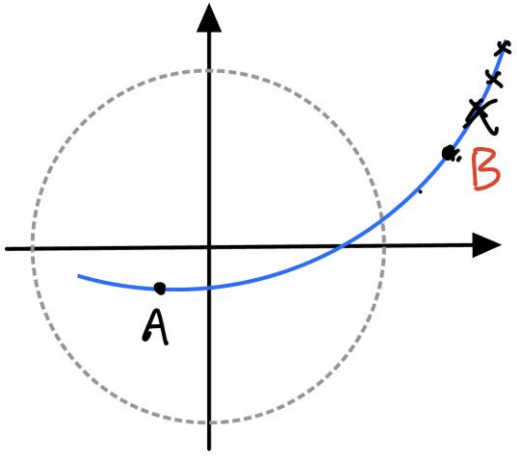






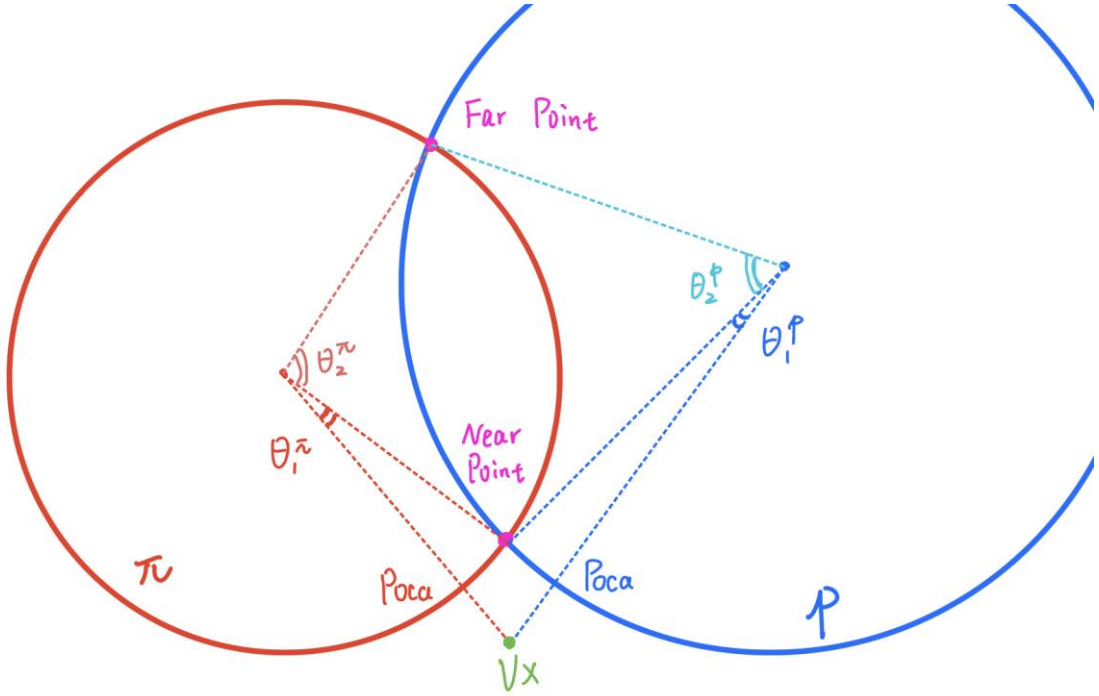






Find Correct Point | Intersection

- VtxFit of p and π ⇒ vx
- {Point1, Point2} ⇒ {Near Point, Far Point}
- Ensure z of Near Point:
  - VFHelix of p and π pivot to vx
  - θ_1: angle between POCA and Near Point
  - VFHelix rotate θ_1 from POCA to Near Point ⇒ z_1
- Ensure z of Far Point:
  - VFHelix rotate θ_2 from Near to Far ⇒ z_2 of p and π
  - Calculate the pitch of 2 VFHelix, obtain all the candidates for p and π within the MDC range (-129.1, 129.1 cm) :
    - ✓ **zVec_p:** $z_2^p, z_2^p \pm h^p, z_2^p \pm 2h^p$ ... ...
    - ✓ **zVec_pi:** $z_2^\pi, z_2^\pi \pm h^\pi, z_2^\pi \pm 2h^\pi$ ... ...
  - Loop the two vectors, the combination with the least $|\Delta z|$ will be considered as the z-coordinate of p and π at the far point.



Why consider the multiple solutions of z:

- Unable to determine the time order of the near point and far point

$\Delta Z(Point) = |z_p - z_{\pi^-}|$ @ $Point$

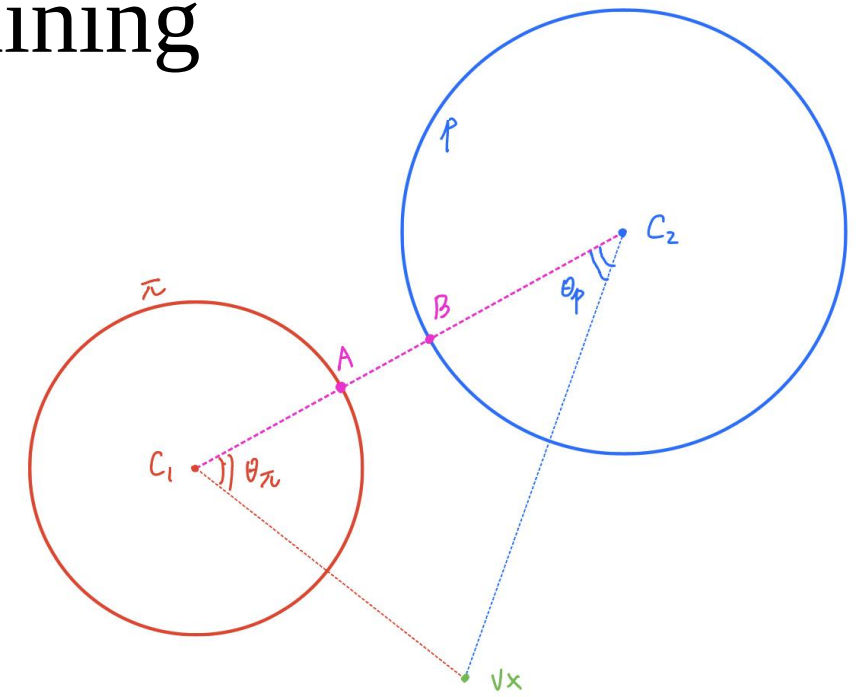
$\Delta Z(Near) < \Delta(Far)$ ⇒ NearPoint is correct

Successful rate: 95.4%

Find Correct Point | Separation or Containing

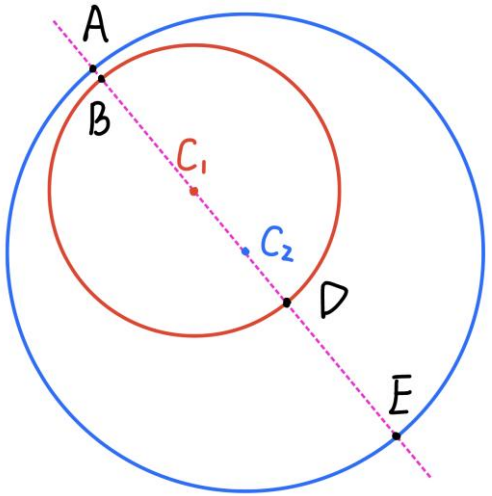
Separation

- VtxFit of p and $\pi \Rightarrow$ vx
- Calculate the closest point of the two circles **{A, B}**
- Ensure z
  - VFHelix of p and π pivot to vx
  - θ: angle between POCA and Point
  - VFHelix rotate θ from POCA to Near Point $\Rightarrow z$

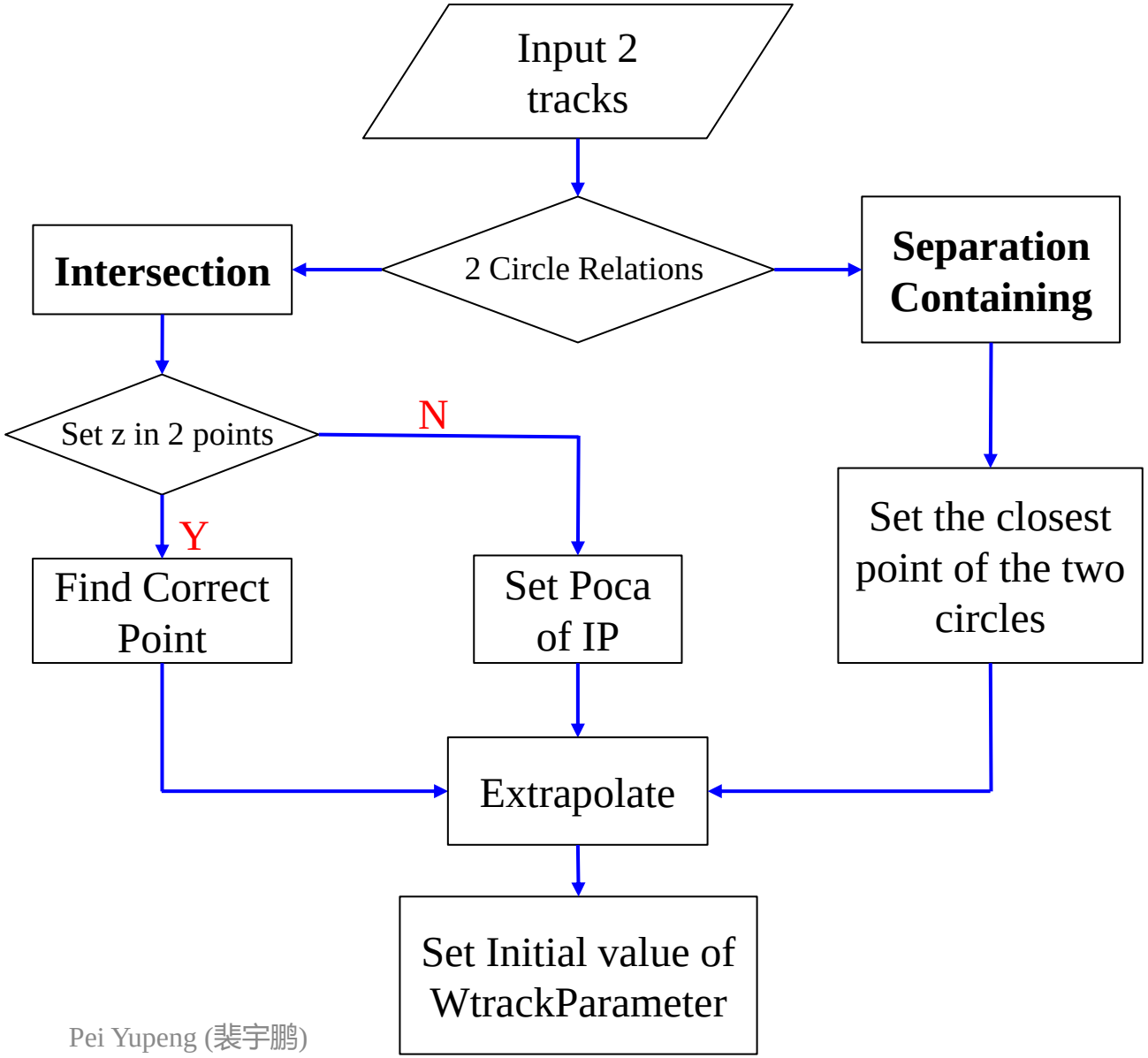


Containing

- Nearest two points (AB or DE)
- Same as separation

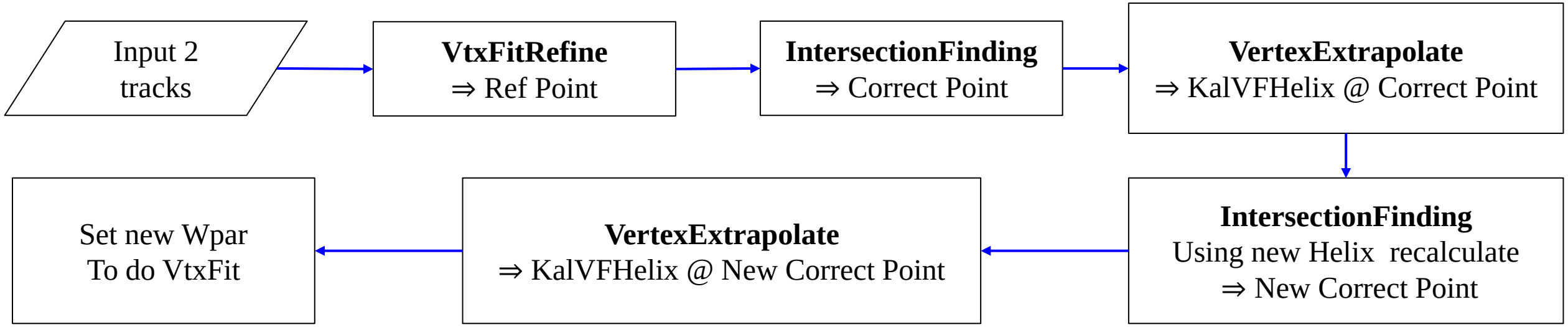


A class **IntersectionFinding** to find the correct point




```
class IntersectionFinding
{
  public:
    static IntersectionFinding* instance();
    ~IntersectionFinding() {}

    void init();

    void setTracksInfo( vector<RecMdcKalTrack*> KalTrkVec, vector<RecMdcKal

    // Calculate the IntSecing points for intersect or closest point for se
    int CalCirIntSecPoint();

    void setPointIDMethod( int _PointIDMethod ) { PointIDMethod = _PointIDM
    void setRefPoint( HepPoint3D _RefPoint ) { RefPoint = _RefPoint; }
    void setHelixType( string _HelixType ) { HelixType = _HelixType; }
    void IsDebug( bool _debug = false ) { debug = _debug; }

    HepPoint3D getRefPoint() {return RefPoint;}
    string getHelixType() {return HelixType;}

    vector<HepPoint3D> getCorrectPointVec() {return CorrectPointVec;}
    vector<HepPoint3D> getWrongPointVec() {return WrongPointVec;}
    vector<HepPoint3D> getDefaultPointVec() {return DefaultPointVec;}
    vector<HepPoint3D> getNearPointVec() { return NearPointVec; }
    vector<HepPoint3D> getFarPointVec() { return FarPointVec; }
    vector<vector<double> > getFarZPosVecVec() { return ZposVecFarVec; }

    HepPoint3D getCorrectPoint( int n ) {return CorrectPointVec[n];}
    HepPoint3D getWrongPoint( int n ) {return WrongPointVec[n];}
    HepPoint3D getDefaultPoint( int n ) {return DefaultPointVec[n];}
    HepPoint3D getNearPoint(int n) { return NearPointVec[n]; }
    HepPoint3D getFarPoint(int n) { return FarPointVec[n]; }
    vector<double> getFarZPosVec(int n) { return ZposVecFarVec[n]; }

    int getPointFlag() { return PointFlag; }
    int getTrksStat() { return TrksStat; }
    int getBestZposStat() { return BestZposStat; }
    VFHelix getTrkHelixVecOrigin(int n) { return TrkHelixVecOrigin[n]; }
    VFHelix getTrkHelixVecInPivot(int n) { return TrkHelixVecInPivot[n]; }
```

A New Vertex Fit Algorithm




```
32 class VertexFitUpgrade
33 {
34     friend class VertexFit;
35
36   public:
37     static VertexFitUpgrade* instance();
38     ~VertexFitUpgrade();
39
40     // initialization, cleanup.
41     void init();
42
43     void AddTrack(const int index, RecMdcKalTrack* p,
44                   const RecMdcKalTrack::PidType pid);
45
46     // add methods
47     void AddVertex(int number, VertexParameter vpar, std::vector<int> lis);
48     void AddVertex(int number, VertexParameter vpar, int n1, int n2);
49     void AddBeamFit(int number, VertexParameter vpar, int n);
50
51     // set iteration number and chisq cut
52     void setIterNumber(const int niter = 10);
53     void setChisqCut(const double chicut = 1000, const double chiter = 1.0e-3);
54     void setMagCorrFactor(const double factor = 1.000);
55
56     bool Fit();
57     bool Fit(int n);
58
```

```
 96 bool VertexFitUpgrade::Fit()
 97 {
 98     string FuncName = "VertexFitUpgrade::Fit()";
 99     if ( !CheckTracks() )
100     {
101       std::cerr << "VertexFitUpgrade: Ntracks = " << m_tracksOrigin.size() << "
102       return false;
103     }
104
105     //// setup the initial vertex
106     HepPoint3D vWideVertex(0., 0., 0.);
107
108     HepSymMatrix evWideVertex(3, 0);
109     evWideVertex[0][0] = 1.0E12;
110     evWideVertex[1][1] = 1.0E12;
111     evWideVertex[2][2] = 1.0E12;
112
113     VertexParameter wideVertex;
114     wideVertex.setVx(vWideVertex);
115     wideVertex.setEvx(evWideVertex);
116
117     //// using the idea of Refine to get the initial ref point;
118     HepPoint3D RefPoint = vtxfit->vx(0);
119     m_HelixType = "Z";
120     bool ZFit = false;
121     bool fitstat = true;
```

Momentum of Λ and Ξ^-