STCF vertex and Kinematic fitting algorithms

Mingyu Yu　*Teng Li*　*Xingtao Huang*

Shandong University

The 7th International Workshop on Future Tau Charm Facilities，Huangshan，11/25/2025

Contents

- Motivation
- Method
- Implementation in STCF
- Performance
- Summary

01 Motivation

part one

STCF(Super Tau-Charm Facility)

➢ Proposal of next-generation high-luminosity e^-e^+ collider—STCF

- **BEPCII/BESIII is expected to complete its mission in 10 years**

  Achievements in XYZ states, exotics, and charm physics have brought China to the forefront of particle physics

- **Further exploration of QCD tests,hadron spectroscopy and new physics beyond the SM**

BesIII

peak luminosity: $1 \times 10^{33} cm^{-2} s^{-1}$
center-of-mass energy: $2\sim4.6 Gev$

STCF

peak luminosity: $0.5 \times 10^{35} cm^{-2} s^{-1}$
center-of-mass energy:$2\sim7 Gev$

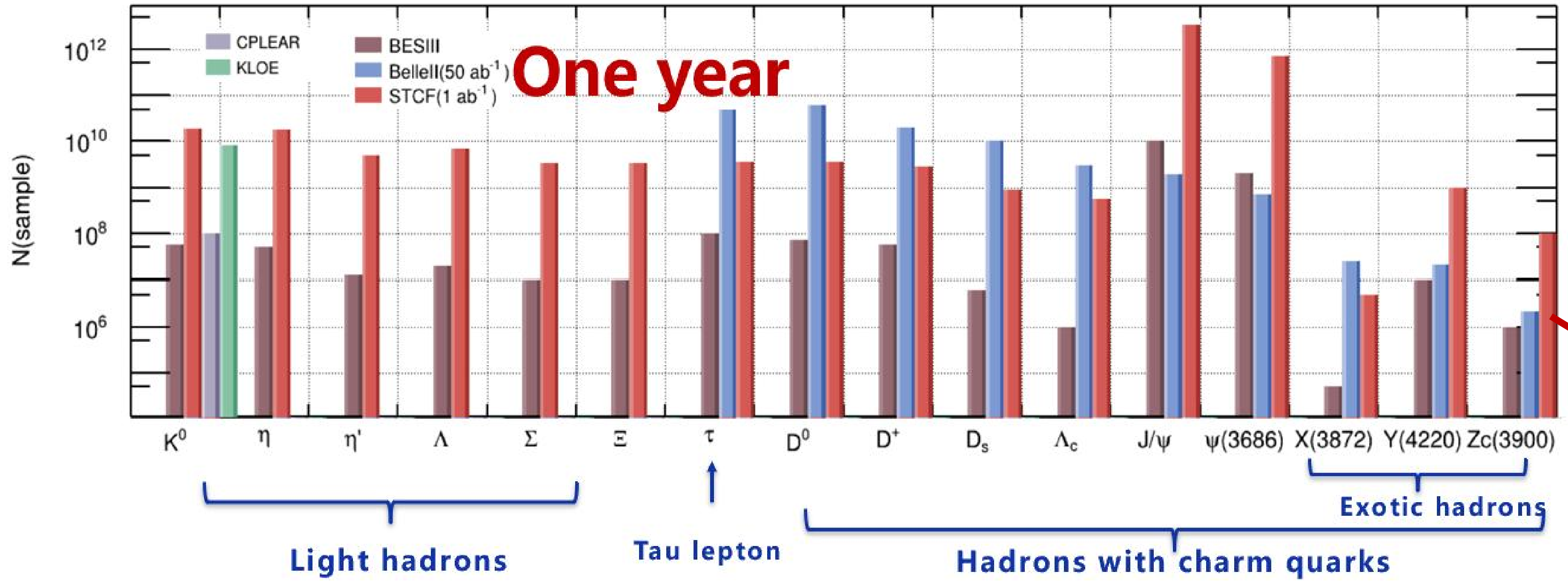


Figure.Expected Data Production at STCF

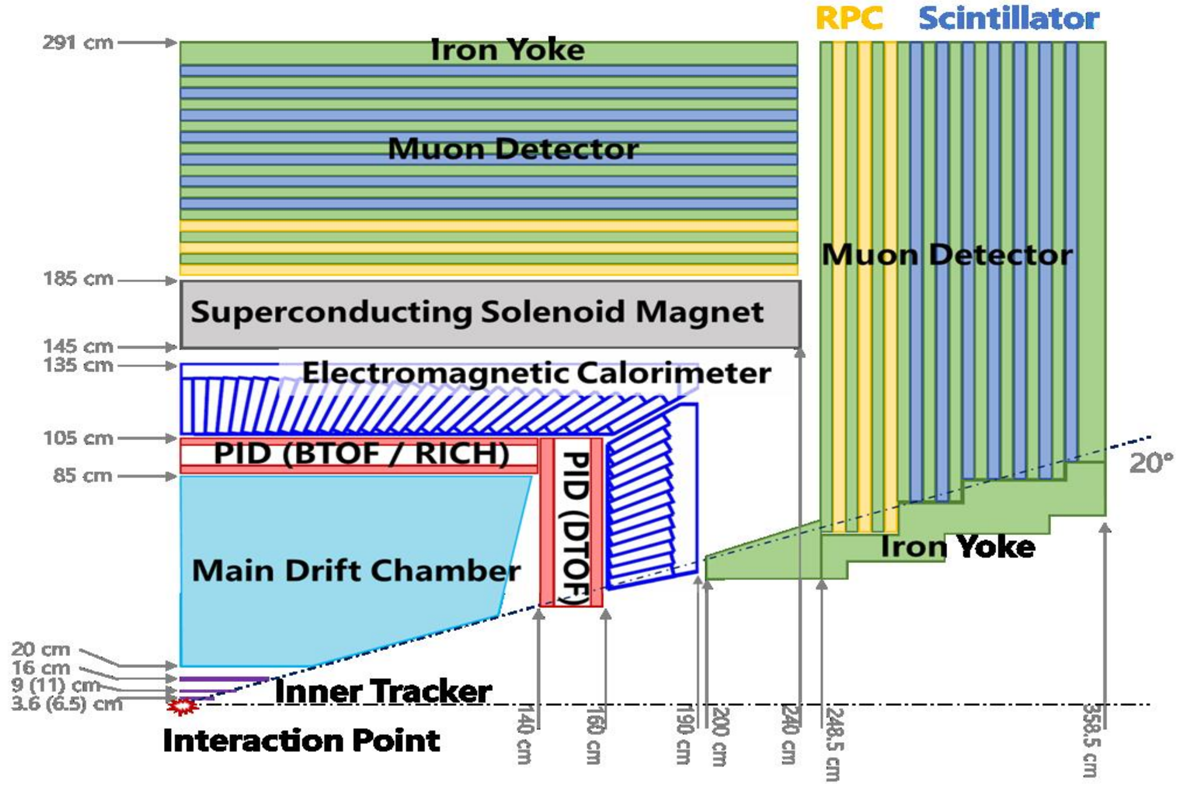


Figure.SuperTau-Charm Facility， STCF

Enable explore a broader range of physical topics and more precise tests

01 Leaf-by-leaf Vertex fit and Kinematic fit

➢ Key features of STCF

High-statistics + High-precision + High-background → **Challenge for data processing**

Accurate signal reconstruction and selection is essential for physical analysis, relying heavily on high performance analysis tools like vertexing and kinematic fitting

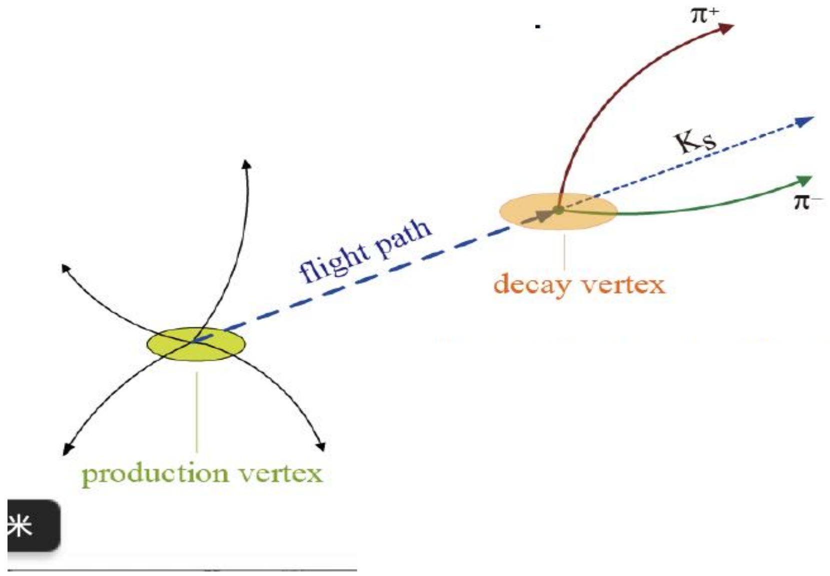
➢ Vertex fit —leaf-by-leaf fitting: plays an important role in event reconstruction

- **Vertex fit：** Charged tracks from the same parent are constrained to a common vertex

  ①Optimize track parameters
  ②Precise reconstruction of intermediate particle
  ③Suppress fake tracks in tracking

- **Second vertex fit:** Precision lifetime measurement of intermediate particles

  ①Optimizes intermediate particle parameter
  ②Suppresses backgrounds



01 Leaf-by-leaf Vertex fit and Kinematic fit

➢ Kinematic fit

Use physical laws to constraint the decay process

①Improve the resolution of particle parameters
②Suppresses background

Implemented in BESIII and demonstrate excellent performance !

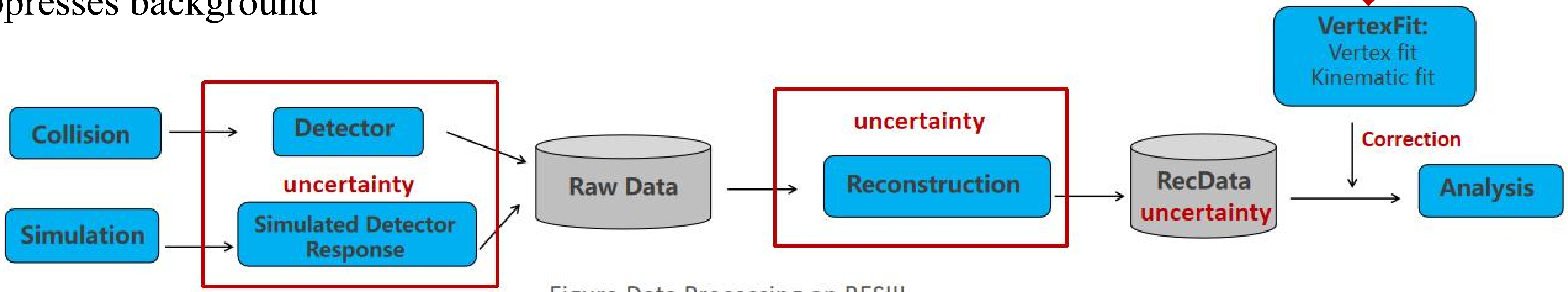


Figure.Data Processing on BESIII

➢ Limitations of VertexFit Package

- Leaf-by-leaf vertex fit is not applicable for neutrals
- Each decay vertex is fitted separately
- Separate steps for vertex fit and kinematic fit

(Long-Lived Particles)$\Sigma^+ \to P\gamma$

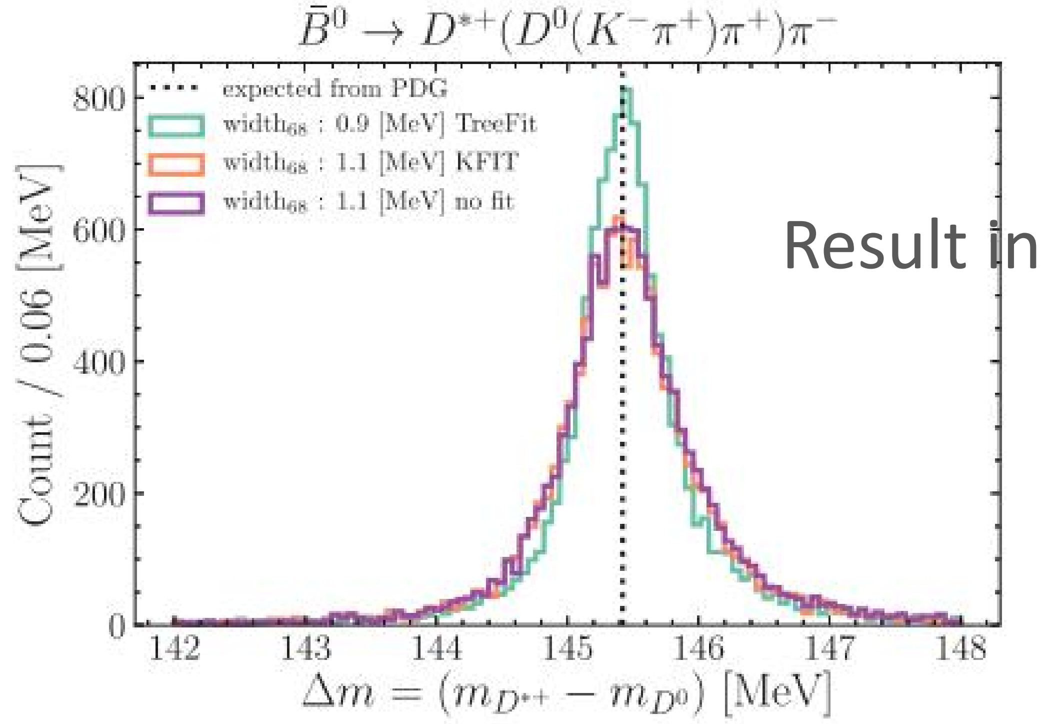
（assumed to originate from the collision point）

01 Global vertex fit

➢ Global Vertex Fit in Belle II

Developed based on Kalman filter

- Construct a global state vector
- Perform vertex fit and kinematic fit simultaneously



Result in BelleII

(KFit: Leaf-by-leaf vertex fit & kinematic fit;TreeFit:Performs global vertex fit)

➢ Vertexing and kinematic fitting tools in STCF

Both are indispensable component for physical analysis

Tool(implemened)		Method	Ref
offline/Analysis/VertexFit	Vertex fit	**Lagrange multipliers**	BESIII
	Kinematic fit	**Lagrange multipliers & Kalman Filter**	BESIII
offline/Analysis/GlobalVertexFit	Global vertex fit	**Kalman Filter**	BelleII

Do some modifications,but core methods remain unchanged !

02

Method

part two

02 Lagrange Multiplier Method

➢ **Main objectives of VertexFit & Global vertex fit is to find the optimal set of physical parameters**

- **Least Squares Method**

Estimate parameters by minimizing the sum of squared differences between measured values and theoretical predictions.

$$\chi^2 = (X - X_0)^T V^{-1} (X - X_0) = min$$

（X −measured values）

（X_0 − theoretical predictions)

The Method cannot handle constrained cases

- **The Constrained Least Squares Method**
  - Lagrange multipliers method
  - Kalman filter method

Incorporating constraints into the least squares method,both of them demonstrates excellent performance in constrained problems

Lagrange Multiplier Method

- **Use Lagrange multipliers (λ) to convert a constrained least squares problem (n variables, m constraints) into unconstrained one with (n + m) variables.**

$$\left.\begin{array}{l} \chi^2 = (X - X_0)^T V^{-1} (X - X_0) = min \\ h(X) = 0, \ \ where\ h = (h_1, h_2, \ldots . h_m) \end{array}\right] \qquad \chi^2 = (X - X_0)^T V_0^{-1} (X - X_0) \ + 2\lambda^T h(X) \ = min$$

(V^{-1}: Covariance Matrix of Parameters) ($h(X)$ – constraint equations)

- **State vector solution**
  - Minimizing the Chi-Square

Differentiate χ^2 with respect to $X\ and\ \lambda$

$$\frac{d\chi^2}{dX} = 0\ , \qquad \frac{d\chi^2}{d\lambda} = 0$$

X_0 –> X(optimal parameter)

- **Nonlinear constraints complicate the solution; Taylor expansion is applied to linearize them**

Iteration

$$h(X) = h(X_{\alpha-1}) + H(X - X_{\alpha-1})\ , \qquad H =- \frac{\partial h(X)}{\partial X}$$

$\downarrow$ insert

$$\frac{d\chi^2}{dX} = (X_\alpha - X_{\alpha-1})^T V_{\alpha-1}^{-1} (X_\alpha - X_{\alpha-1}) \ + 2\lambda^T h(X_\alpha) \ = 0$$

- **Solution of state vector**

$$X_\alpha = X_{\alpha-1} - V_0 H^T \lambda\ , \qquad \begin{array}{l} \lambda = V_D h(X_\alpha) \\ V_D = (HV_0H^T)^{-1} \end{array}$$

(Large state vector directly leads to high computational cost)

- **Recursive estimation of particle states under uncertainties and constraints, with optimal parameters and error matrix output.**

$$\chi^2 = (X_\alpha - X_{\alpha-1})^T C_{\alpha-1}^{-1} (X_\alpha - X_{\alpha-1}) + (m - h(X_\alpha))^T V_m^{-1} (m - h(X_\alpha)) = min$$

（$C_{\alpha-1}^{-1}$: Covariance Matrix of Parameters)　（V^{-1}: Measurement covariance matrix)　（$m - h(X_\alpha)$: Residual)

Matrix inversion depends on the constraint dimension， greatly reducing the computational cost

- **State vector solution**
  - Minimizing the Chi-Square

Differentiate χ^2 with respect to X

$$\frac{d\chi^2}{dX} = 0$$

$X_0 \rightarrow X$(optimal parameter)

- **Linear approximation：**

Iteration

$$h(X) = h(X_{\alpha-1}) + H(X - X_{\alpha-1})$$

$$\downarrow$$

$$(X_\alpha - X_{\alpha-1})^T C_{\alpha-1}^{-1} (X_\alpha - X_{\alpha-1}) + (m - h(X_\alpha))^T V_m^{-1} (m - h(X_\alpha)) = 0$$

- **Solution of the state vector**

$$X_\alpha = X_{\alpha-1} - K_\alpha r_\alpha,$$

（$K_\alpha = C_{\alpha-1} H R_\alpha^{-1}$ − Kalman gain matrix)

（$r_\alpha = m - h(X_\alpha)$ − Residual)

（$R_\alpha = V + H_\alpha^{\alpha-1} C_{\alpha-1} (H_\alpha^{\alpha-1})^T$ − residual covariance matrix)

02 Kalman filter in GlobalVertexFit

➢ Two feature of Kalman Filter in Global Vertex Fit

- **Global state vector:**

Incorporate all particles' information

- Final-state particle: (p_x, p_y, p_z)
- Intermediate particle
  - Resonance: (p_x, p_y, p_z, E)
  - Long-lived: $(\boxed{x, y, z, \theta,} p_x, p_y, p_z, E)$

improving stability and accuracy of the fit

- **Sequential Constraint Integration**

During each iteration, constraints are applied in a predefined sequence

$$\chi_\alpha^2 = \sum_k \chi_k^2 \text{ , } (\alpha\text{: the } \alpha\text{-th iteration, k: the k-th constraint})$$

$$\chi_k^2 = (X_k - X_{k-1})^T C_{k-1}^{-1} (X_k - X_{k-1}) + (m_k - h_k(X_k))^T V_k^{-1} (m_k - h_k(X_k))$$

Vertex constraint and kinematic constraints can be performed simultaneously within one iteration.

03

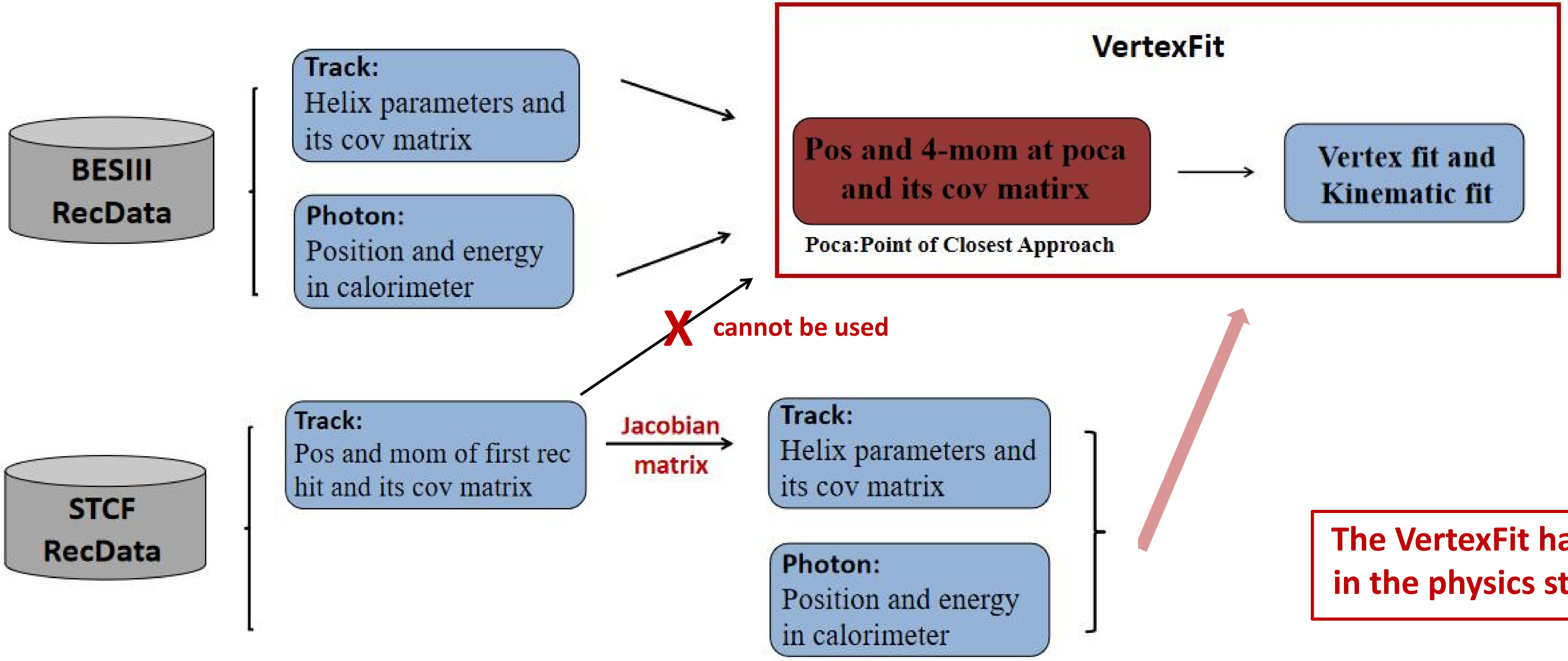
part three

Implementation in STCF

03 Implementation of VertexFit ---From BESIII

➢ Implementation of Vertx fit and Kinematic fit

Based on the BESIII VertexFit algorithm, we have implemented the corresponding algorithm for STCF after some adaption and modification according to the STCF DataModel

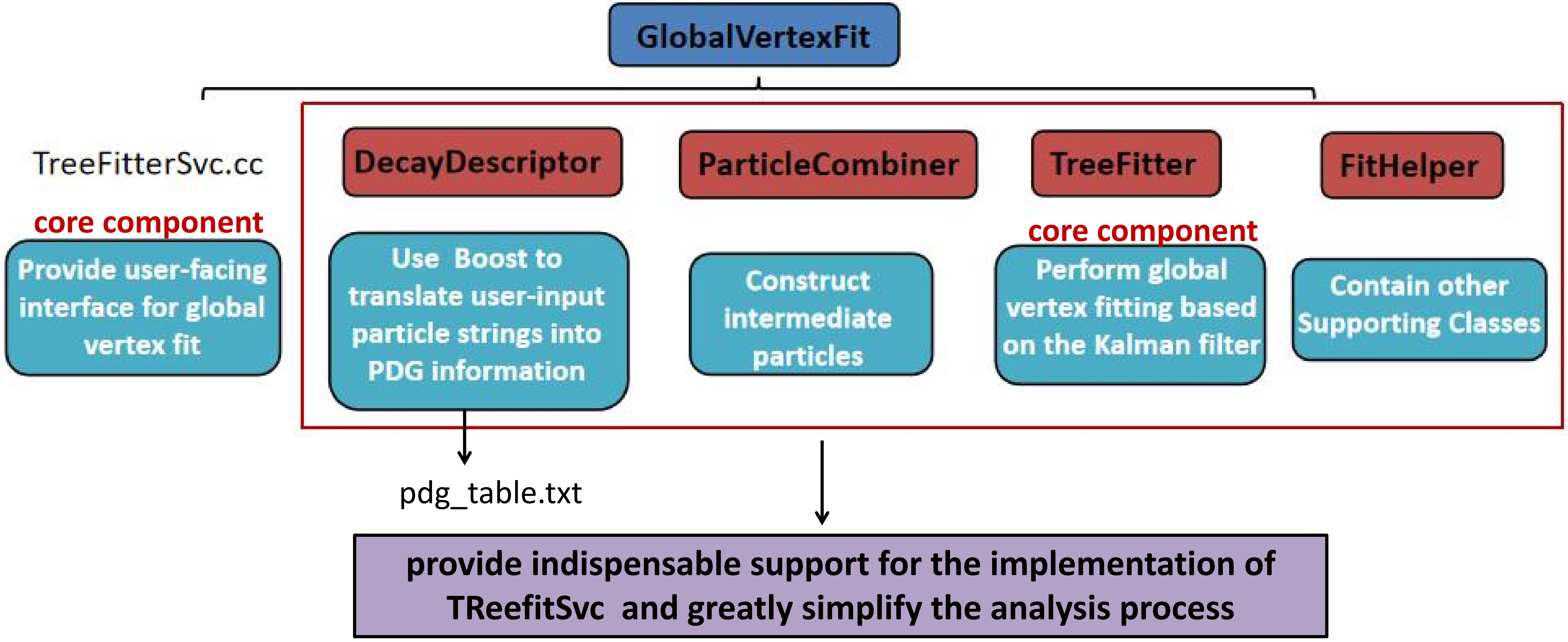


The VertexFit has been used in the physics study on STCF

Implementation of Global vertex fit --- From BelleII

➢ Implementation of Global vertx fit

Based on the BelleII - Global vertex fit algorithm, we have implemented a new algorithm for STCF after integrating related modules with different functionalities into a consolidated package



04 Performance

part four

04 Performance

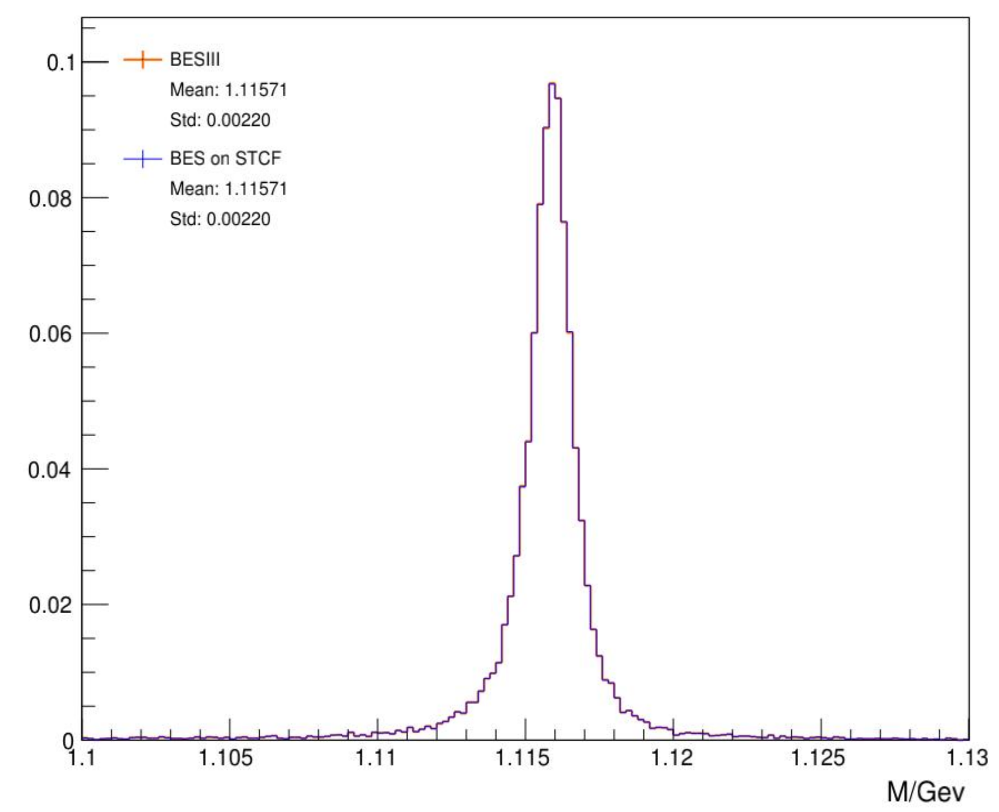
➢ VertexFit performance

$$J/\psi \to \Lambda\bar{\Lambda} \to (p\pi^-)(\bar{p}\pi^+)$$

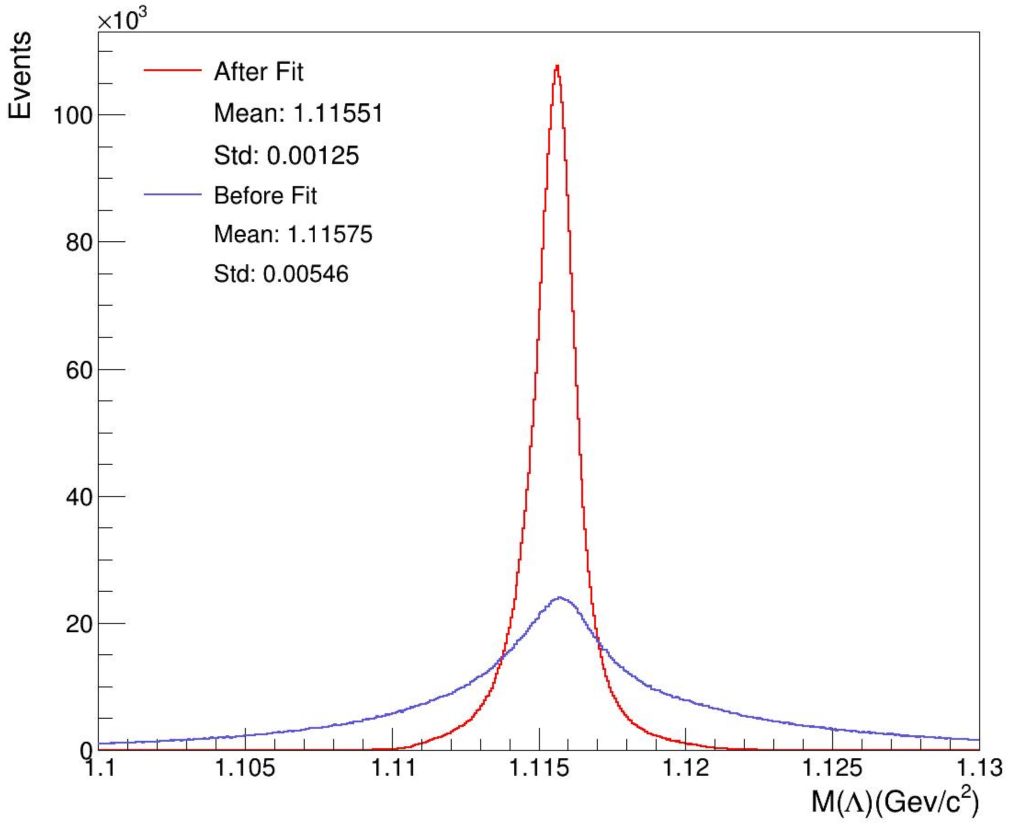
Proves a successful migration of our algorithm.

- **To confirm the accuracy of migration**:
  Data that successfully fitted in BESIII are input to STCF---consistent result
- **VertexFit algorithm shows good performance on STCF**

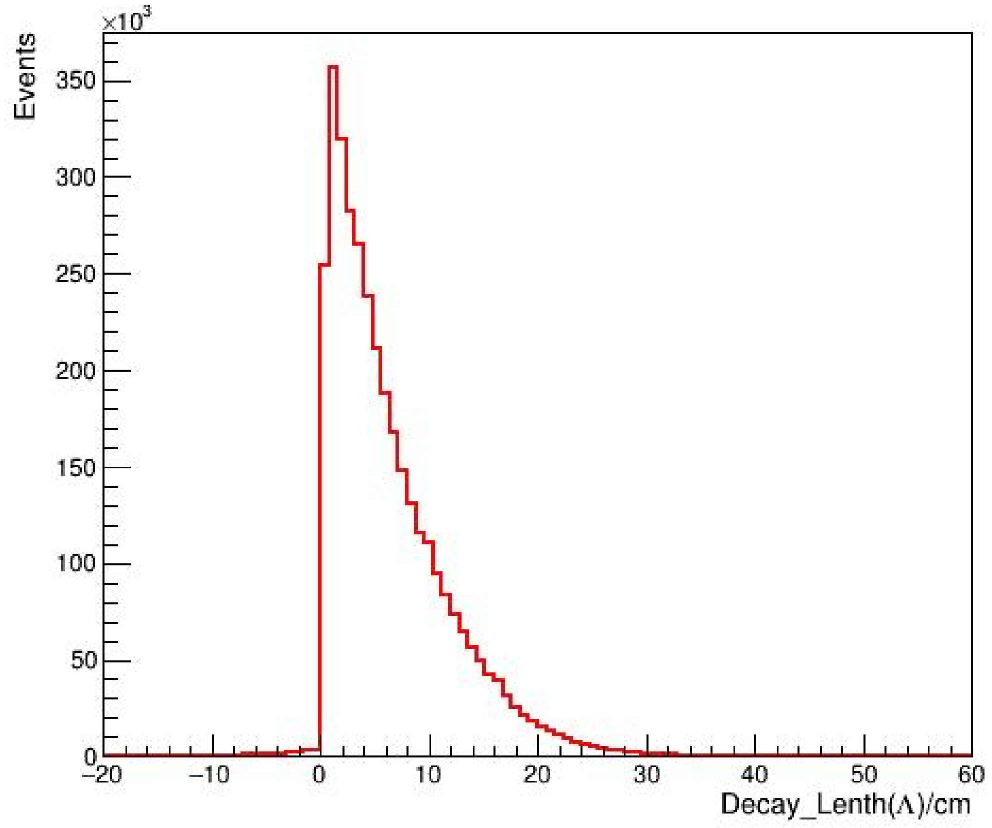
➢Invariant Mass of Λ(use BESIII data)



➢Invariant Mass of Λ



➢DecayLength of Λ



04 Performance

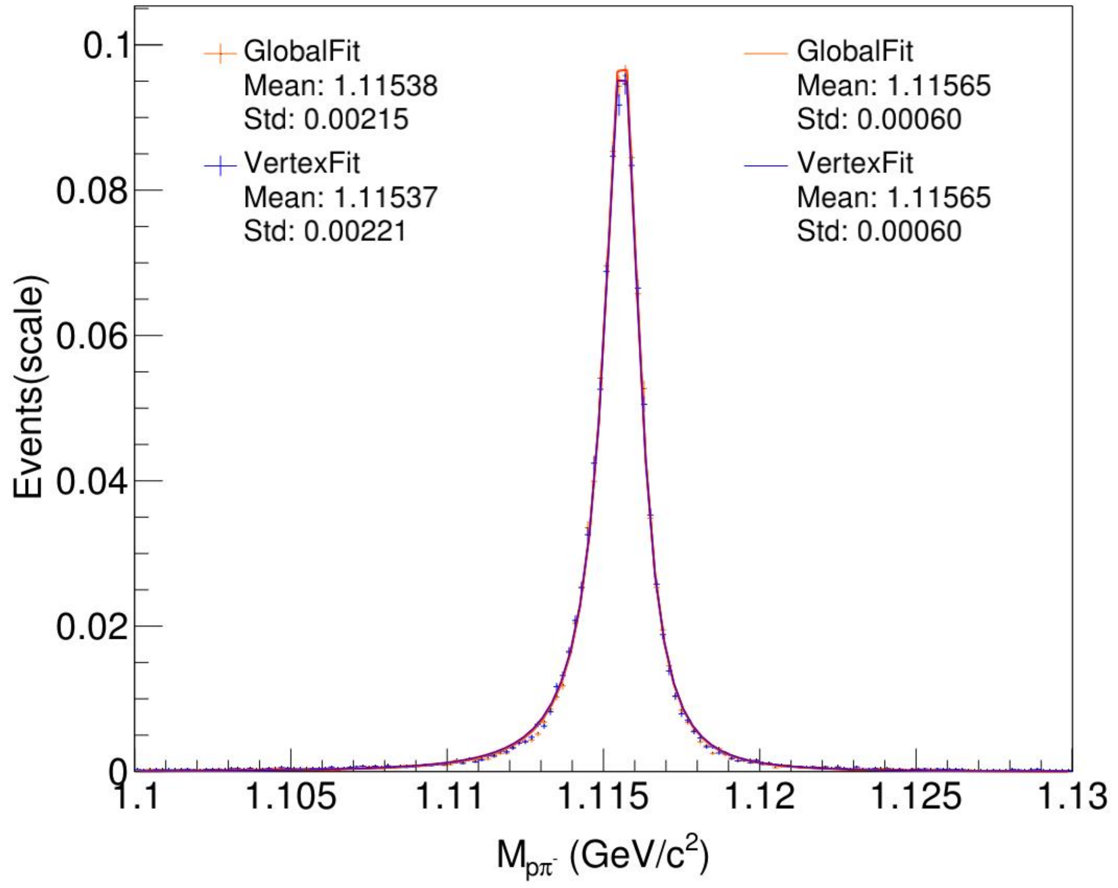
➢ **GlobalVertexFit performance based on different physics processes**

$$J/\psi \to \Lambda\bar{\Lambda} \to (p\pi^-)(\bar{p}\pi^+)$$

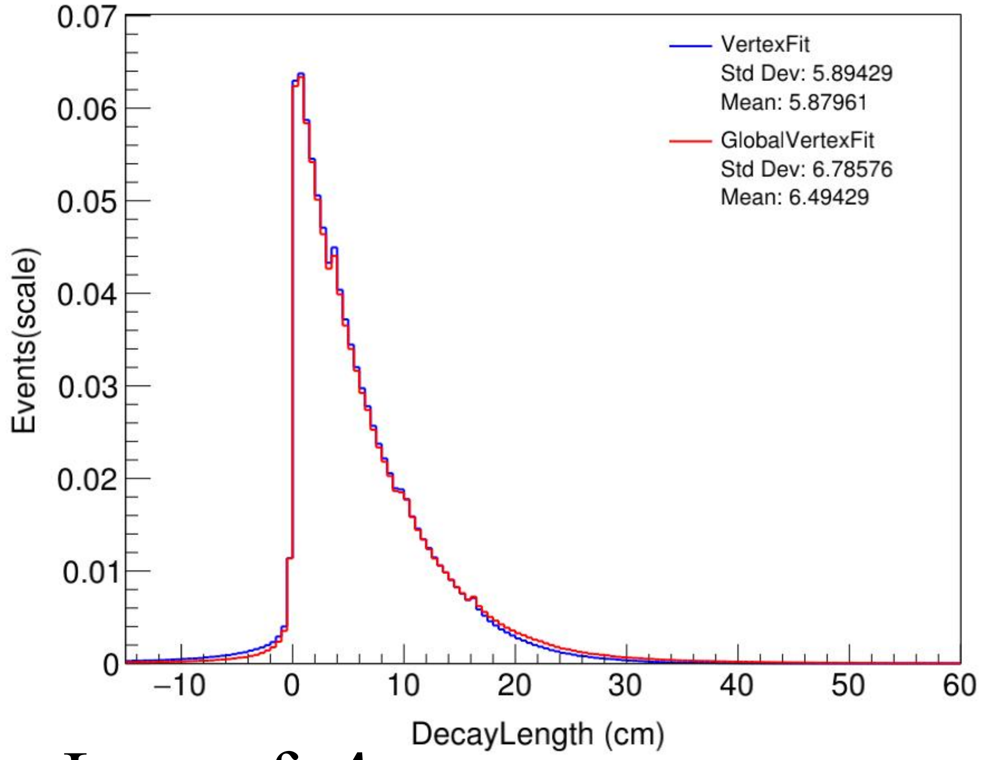
VertexFit：42.922% vs GlobalVertexFit 44.079%

slight improvement in the mass resolution and 1% increase in efficiency

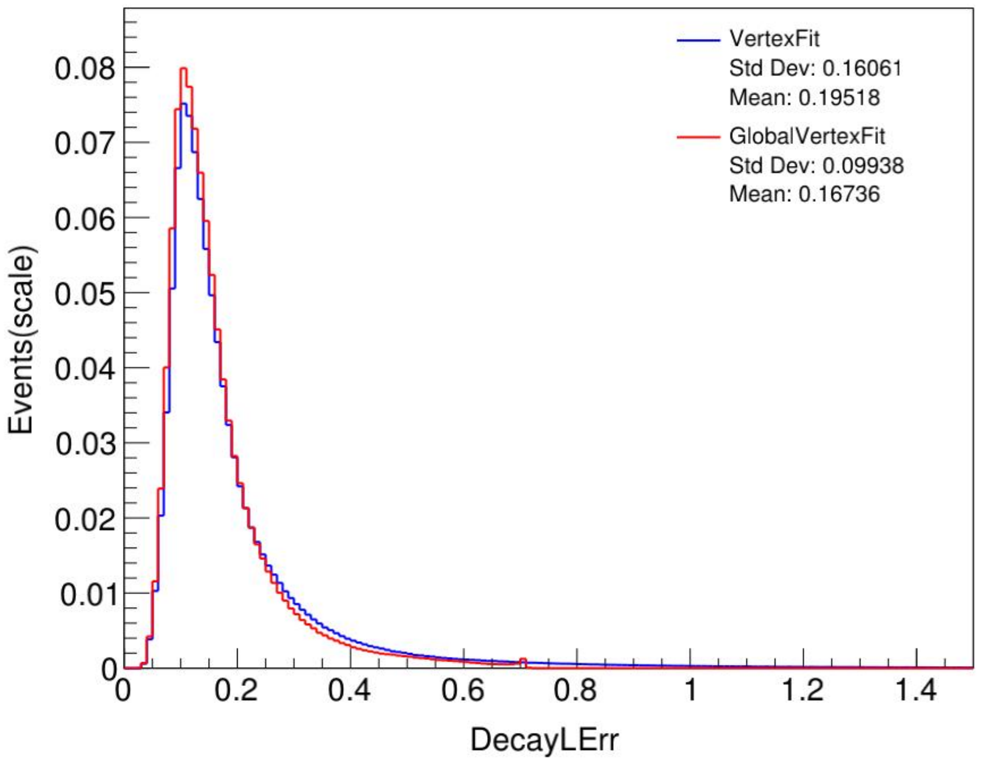
➢Invariant Mass of Λ



➢DecayLength of Λ



➢DecayLerr of Λ



04 Performance

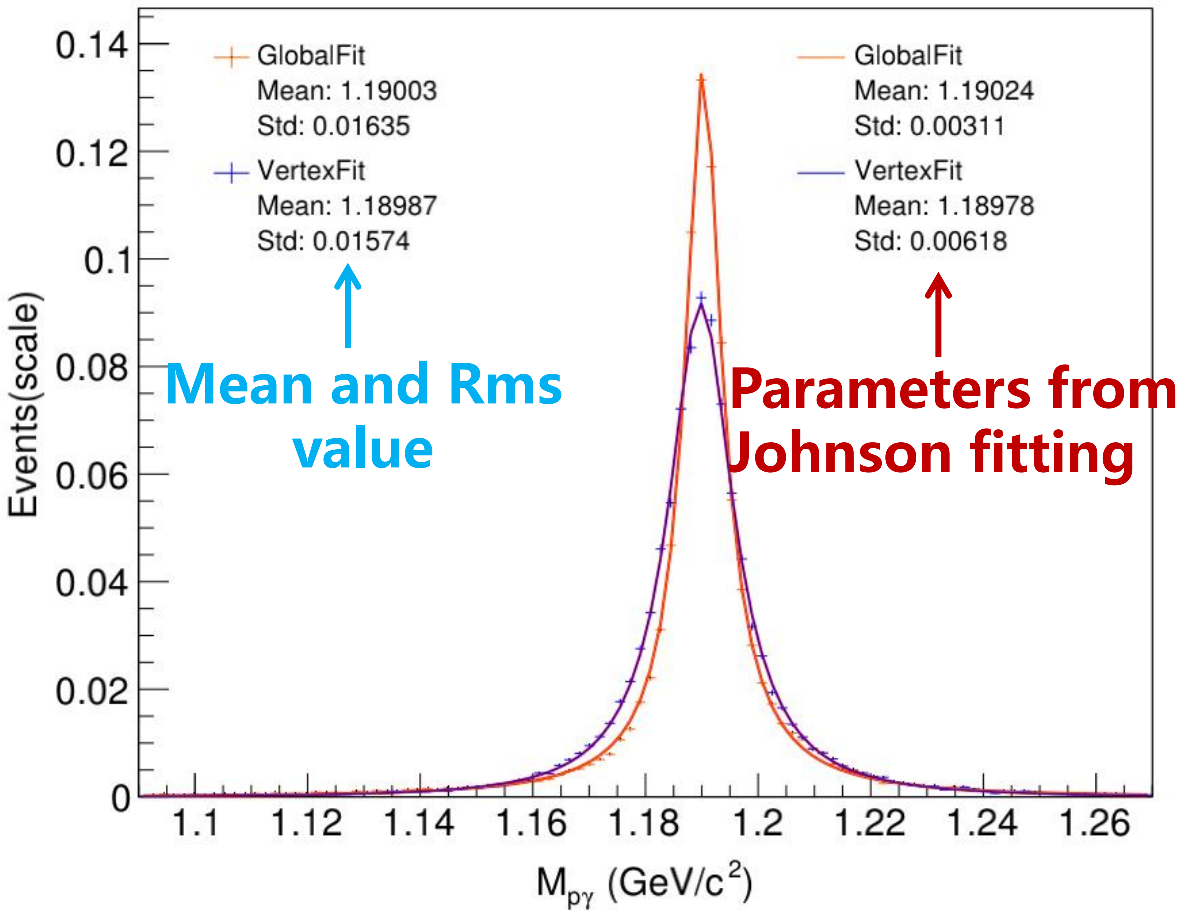
➢ Global Vertex Fitting performance based on different physics processes

$$J/\psi \rightarrow \Sigma^{+}(Pr)\overline{\Sigma}(\pi_0(rr)\overline{P})$$

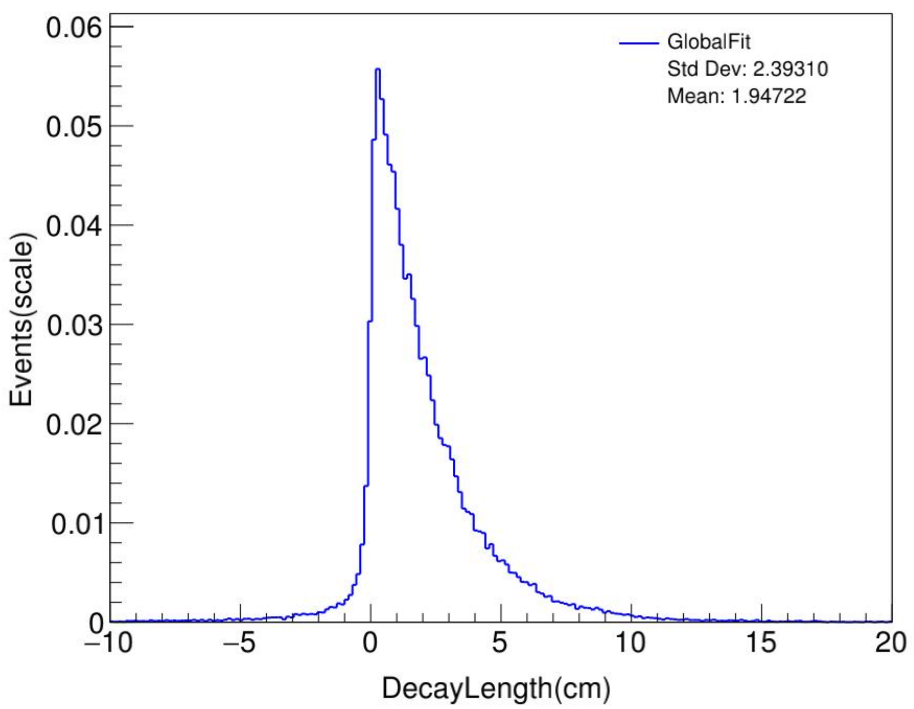
VertexFit：66.64% vs GlobalVertexFit 69.44%

significant improvement in both mass resolution and efficiency(3%)

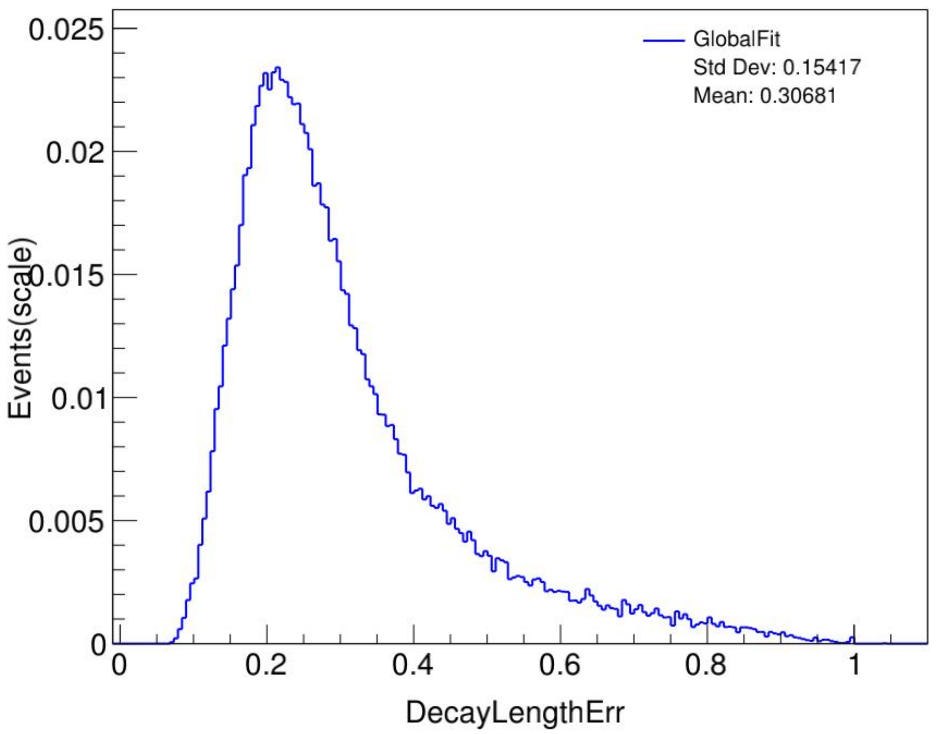
➢ Invariant Mass of Σ^{+}



➢DecayLength of Σ^{+}



➢DecayLerr of Σ^{+}



VertexFit cannot achieve this

04 Performance

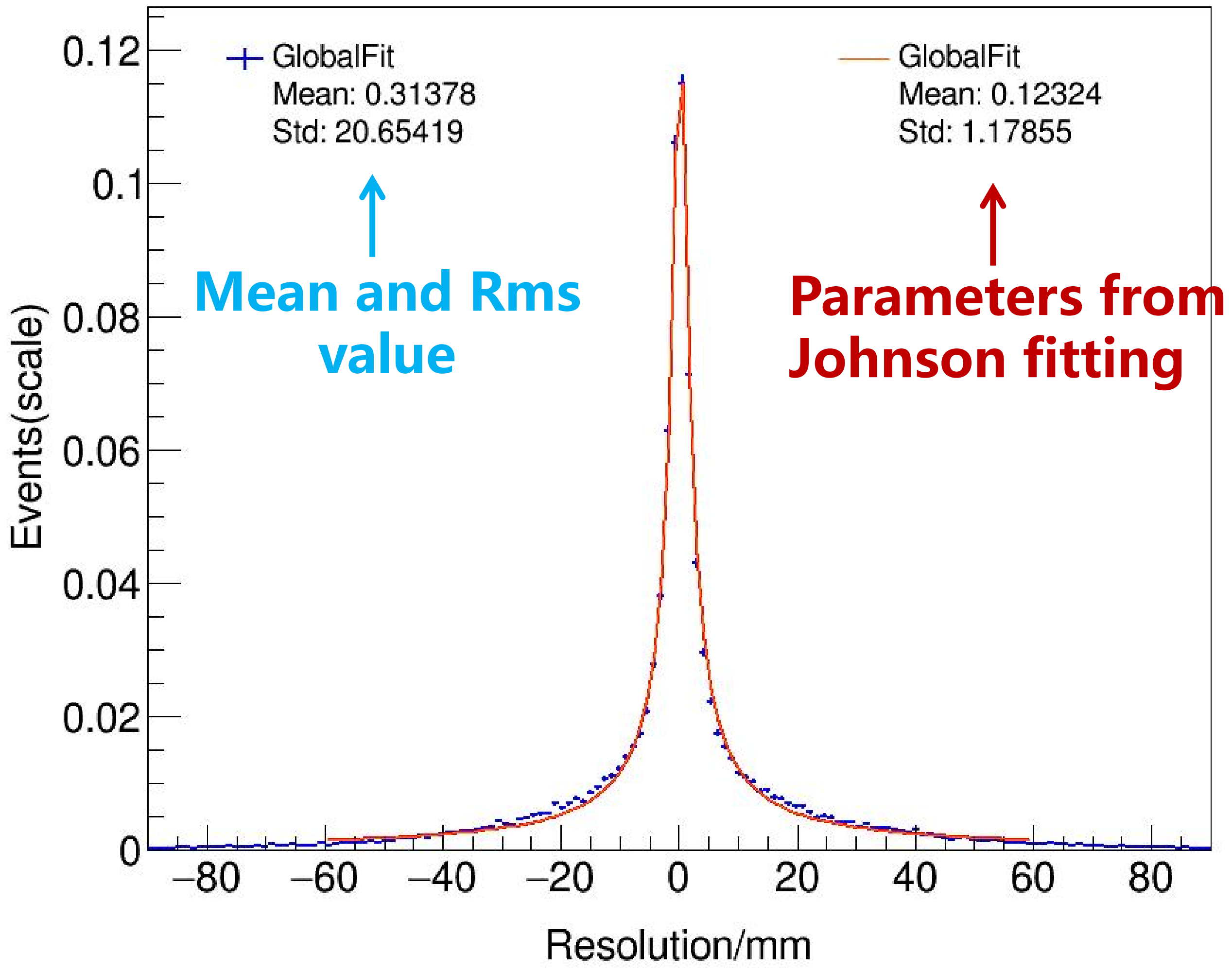
➢ **Global Vertex Fitting performance based on different physics processes**

$$J/\psi \to \Sigma^{+}(Pr)\overline{\Sigma}(\pi_0(rr)\overline{P})$$

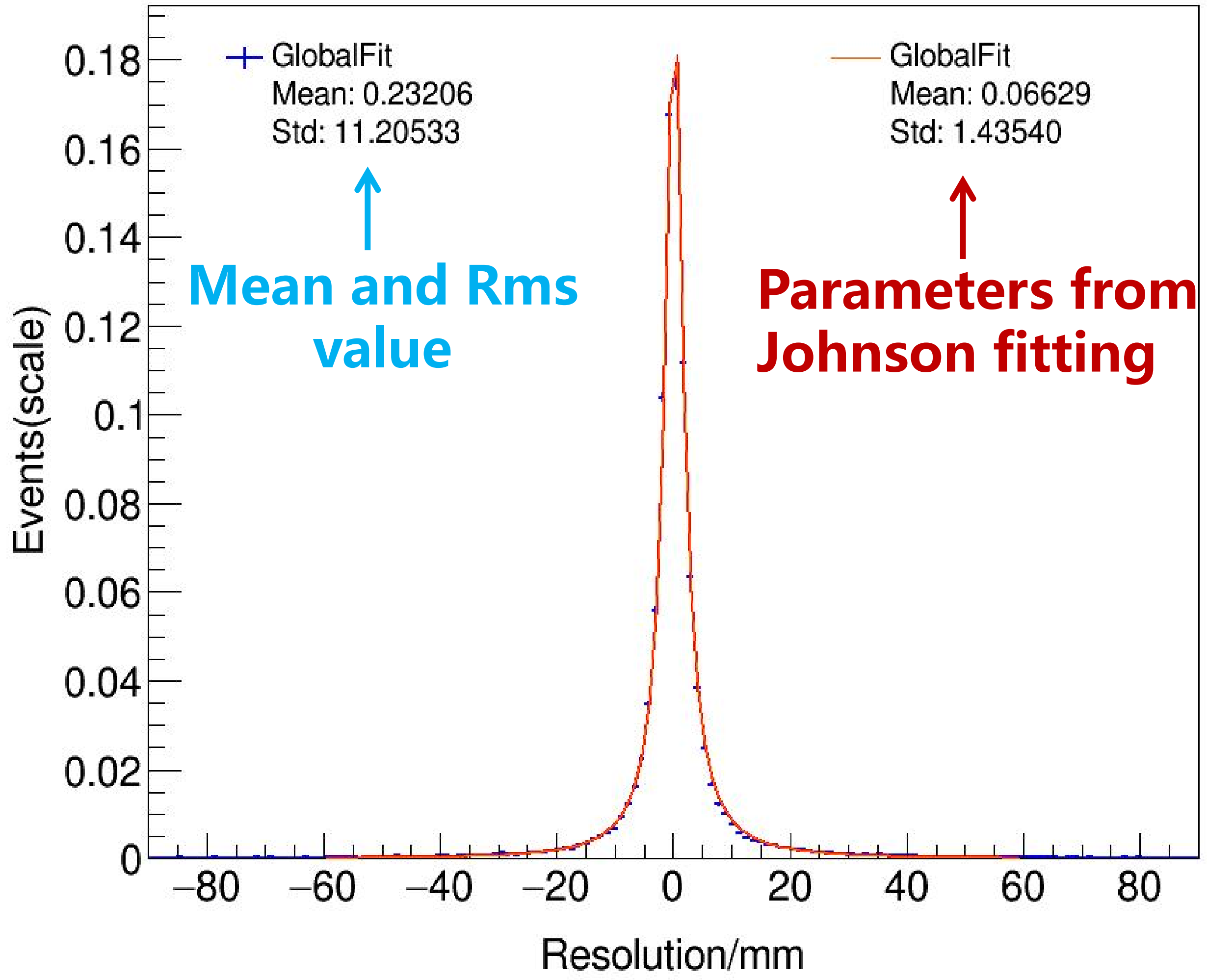
- **To test the accuracy of fitted DecayL:**
  Compare it with MCTruth (MC - Fitted)

GlobalVertexFit shows good performace on STCF and demonstrate its unique advantages in integrating particle informations

➢DecayLength Resolution of Σ^{+}



➢DecayLength Resolution of$\overline{\Sigma}$



05 Summary

part five

05 Summary

- Vertexing and kinematic fit are very important tools for physics analysis
- We have implemented both vertexing and kinmatitic fitting algorithms for STCF by learning from BESIII and BelleII
  - VertexFit---BESIII:
    Lagrange multiplier/Kalman filter-based Vertex fit and Kinematic fit
  - GlobalVertexFit---BelleII:
    Kalman filter-based GlobalVertexFit

Dedicated performance study have been done, the results show they are in good shape

- Further study and optimization is ongoing

Thank you!

山东大学
SHANDONG UNIVERSITY

backup

➢ Global Vertex Fit Strategy for Neutral Particles

$$(\underbrace{p_x, p_y, p_z}_{p}, \underbrace{p_x, p_y, p_z}_{\gamma}, \underbrace{x, y, z, \theta, p_x, p_y, p_z, E}_{\Sigma^+})$$

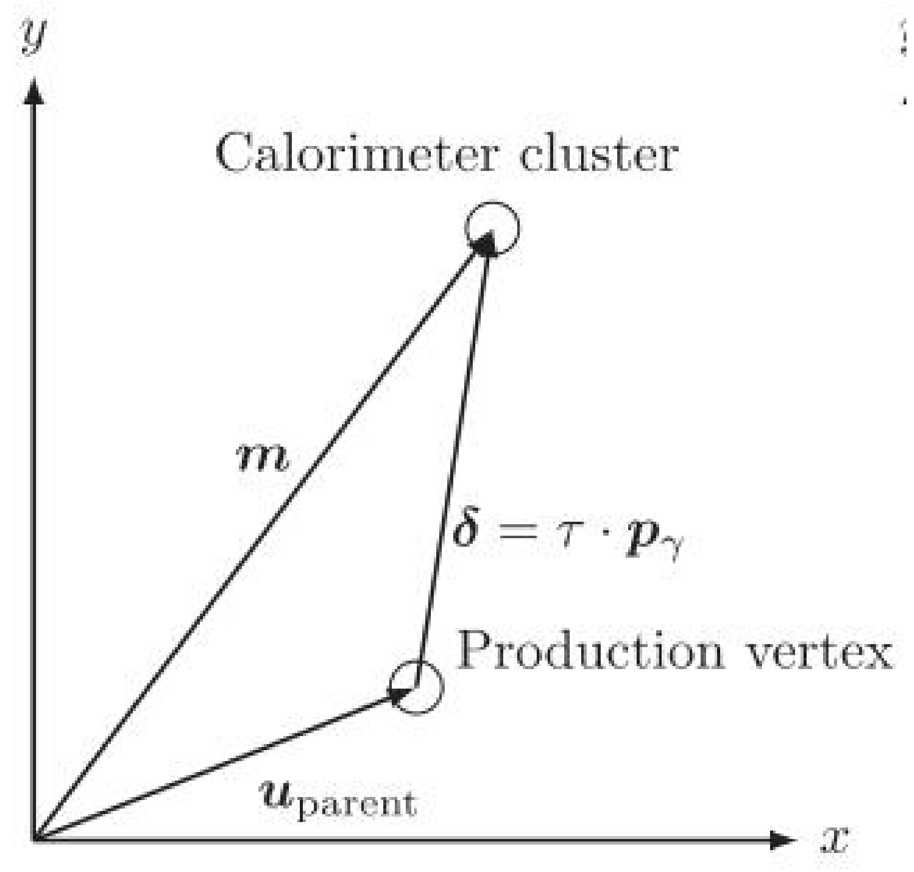
- Constraint Application Order

①track

Update the momentum and vertex position of charged particles based on reconstructed track information

②photo

Update the photon momentum using the vertex information refined by the track update.。



backup

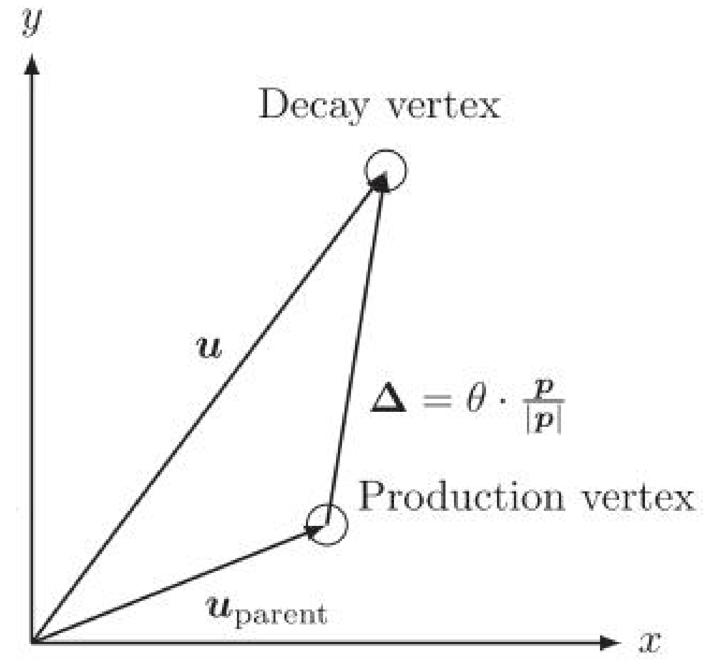
- **Decay Vertex:**

  Global vertex fitting uses a geometric method to compute the initial decay vertex:

  - If two or more charged tracks exist, the two with highest momentum are selected, and their point of closest approach in 3D space is calculated as the initial decay vertex.
  - This vertex is further refined during fitting with additional constraints.

- **decay length$\boldsymbol{\theta}$**

  The decay length is extracted using a vector triangle based on the decay geometry.



backup

➢Equations of Helical Motion for charged particles

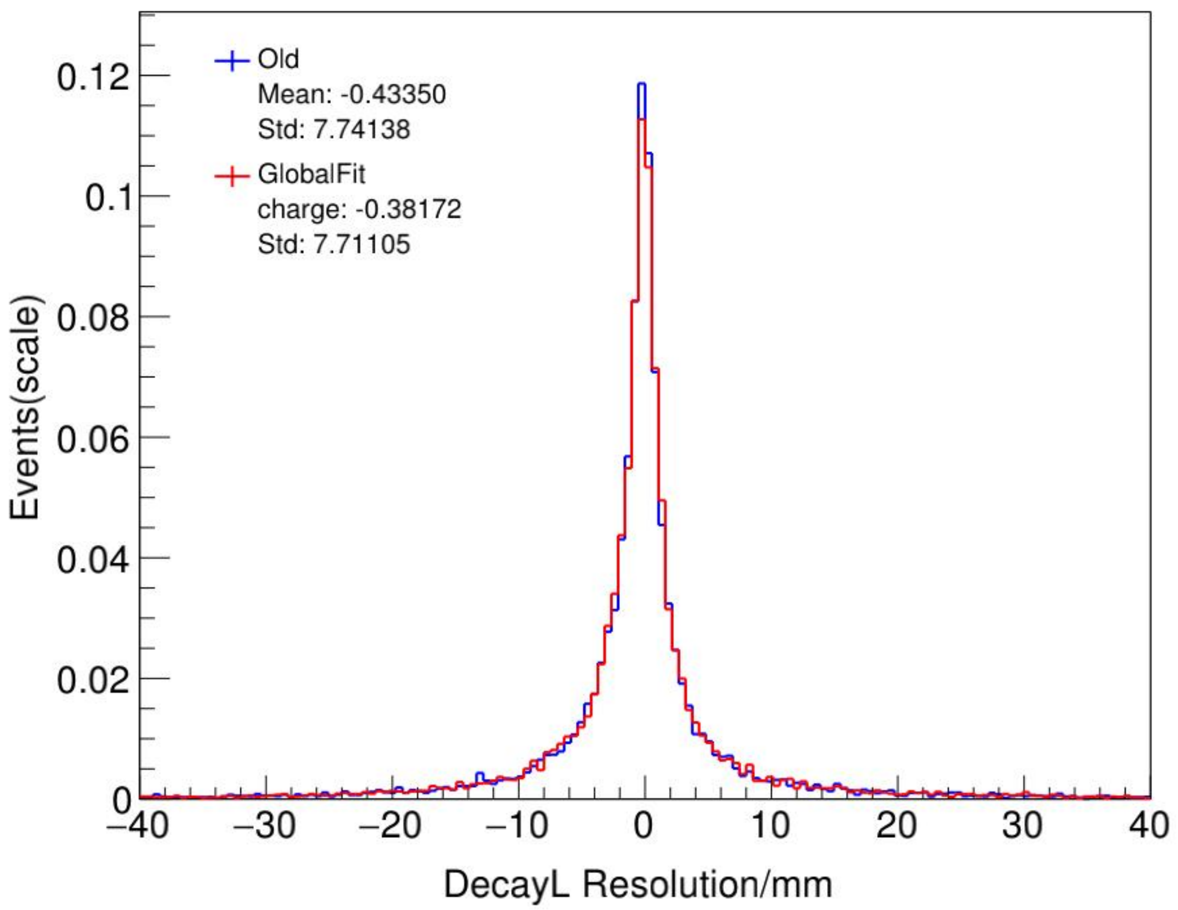
$J/\psi \rightarrow \Xi^-(\Lambda(p\pi^-)\pi^-)\bar{\Xi}^+(\bar{\Lambda}(\bar{p}\pi^+)\pi^+)$

带电粒子是螺旋线运动方程：

$$x_p - x_d + \frac{p_x}{a}\sin(ac\tau/m) + \frac{p_y}{a}(1-\cos(ac\tau/m)) = 0,$$
$$y_p - y_d + \frac{p_y}{a}\sin(ac\tau/m) - \frac{p_x}{a}(1-\cos(ac\tau/m)) = 0,$$
$$z_p - z_d + \frac{p_z}{m}c\tau = 0,$$

→ GlobalVertexFit algorithm

➢DecayLength Resolution of Ξ^-



backup

1.2.1 Include necessary headerfile

```
#include "GlobalVertexFit/TreeFitterSvc.h"
```

1.2.2 Create TreeFitterSvc and Initialize

```
SniperPtr<TreeFitterSvc> _svc(getParent(), "TreeFitterSvc");
m_TreeFitterSvc = _svc.data();
```

//create final state particlelist

The input string is passed in the form of "name:label",the name must match the definitions in the ROOT pdg_table.txt, and the label can be named arbitrarily

```
m_TreeFitterSvc->fillParticleList("pi+:loose");
```

//create intermediate-state particlelist

The second string describes the decay process you want to construct, and the daughter particles' particlelist must be constructed before.

The third parameter specifies the mass cut to be applied to the intermediate particles before fitting, and it can be an empty string.

The last parameter indicates whether to construct its antiparticle.

```
m_TreeFitterSvc->reconstructDecay(0,"Lambda0:ppi->proton:loose pi-:loose","0.9<M<1.32", true)
```

1.2.3 Start GlobalVertexFit

//The reconstruction information is stored in the form of (ReconstructedParticle, PDGCode)

```
m_RPar.clear();
m_RPar.emplace_back(ReconstructedParticle,PDGCode);
```

//Start GlobalVertexFit

```
m_TreeFitterSvc->init();
m_TreeFitterSvc->setCustomOrigin(0,0,0);//set vertex and err
m_TreeFitterSvc->setCustomOriginCovariance(a,b,c);
m_TreeFitterSvc->transtoParticle(m_RPar);

m_TreeFitterSvc->constructParticle(i);//construct intermediate particle

bool okfit = m_TreeFitterSvc->treeFit();//GlobalVertexFit
if(!okfit) return true;
```

1.2.4 Other method in GlobalVertexFit

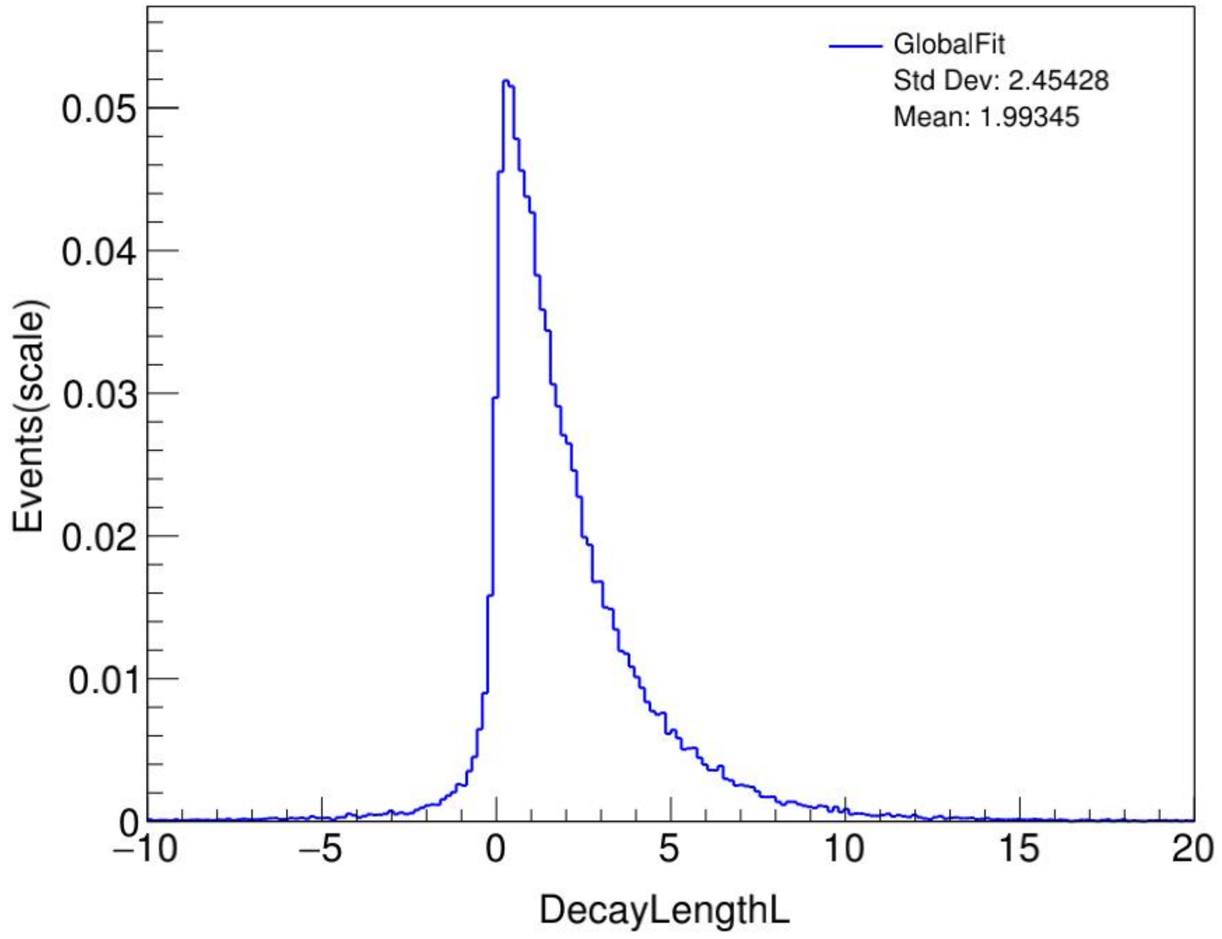
```
//Get all particles in decaytree
const std::vector<Particle> Particles =    m_TreeFitterSvc->getParticle();

//Get Headparticle List(like jpsi),it store the index of head particle in Particles
const ParticleList* Headlist = m_TreeFitterSvc->getHeadList();

/*headpar is the head of decay chain like jpsi*/
/*The headpar and particle mentioned below are both objects of the Particle class*/

//Get chisq after fit
double chi = headpar.getExtraInfo("chiSquared");

//Get daughter i or get daughter index
const Particle* daughteri = particle.getDaughter(i);
const std::vector<int>& daughterindice = particle.getDaughterIndices();

//Get 4mom
ROOT::Math::PxPyPzEVector mom4 = particle.get4Vector();

//Get decaylength and err
double decayl = particle.getExtraInfo("decayLength");
double decaylerr = particle.getExtraInfo("decayLengthErr");
```

backup

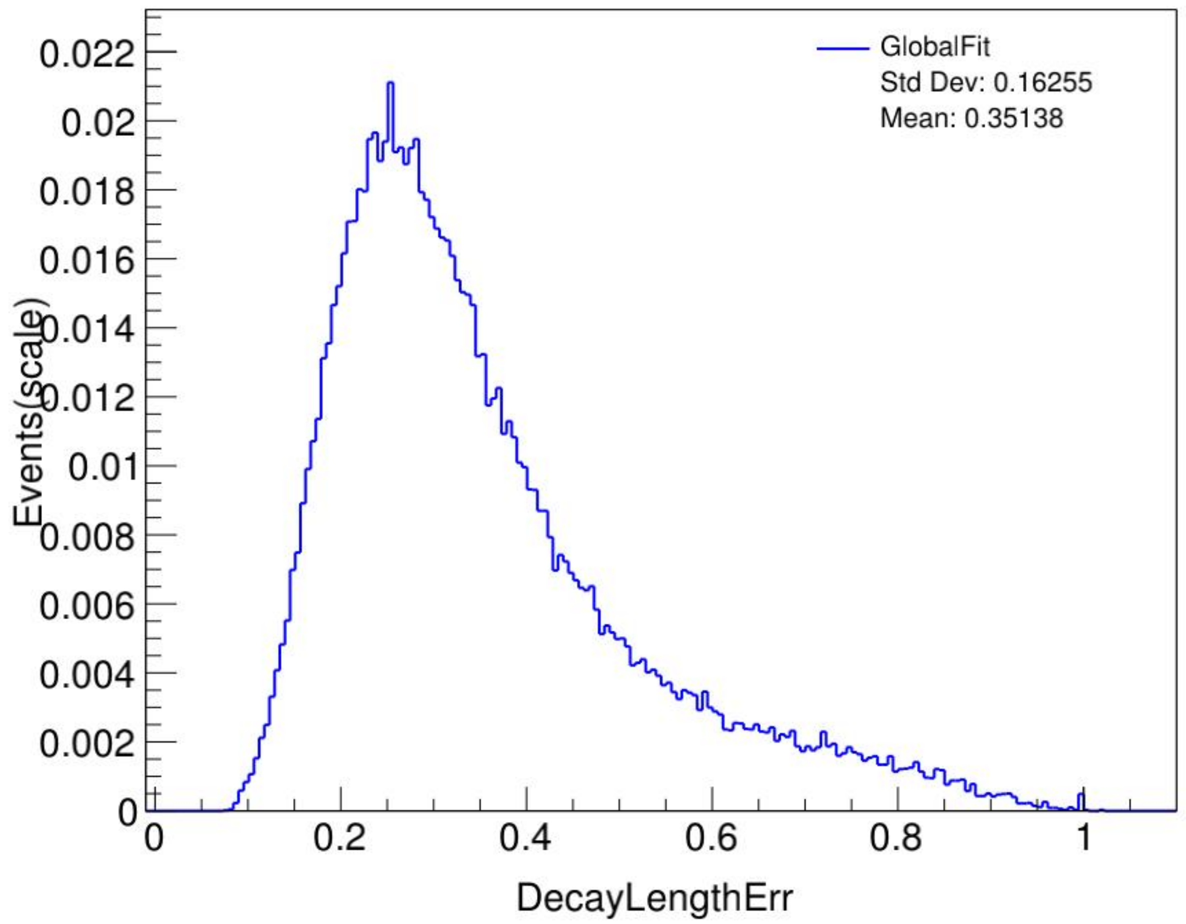
- performance of GlobalvertexFit after 4c

$J/\psi \rightarrow \Sigma^{+}(Pr)\overline{\Sigma}(\pi_{0}(rr)\overline{P})$

- $\overline{\Sigma}$ decayLength distribution



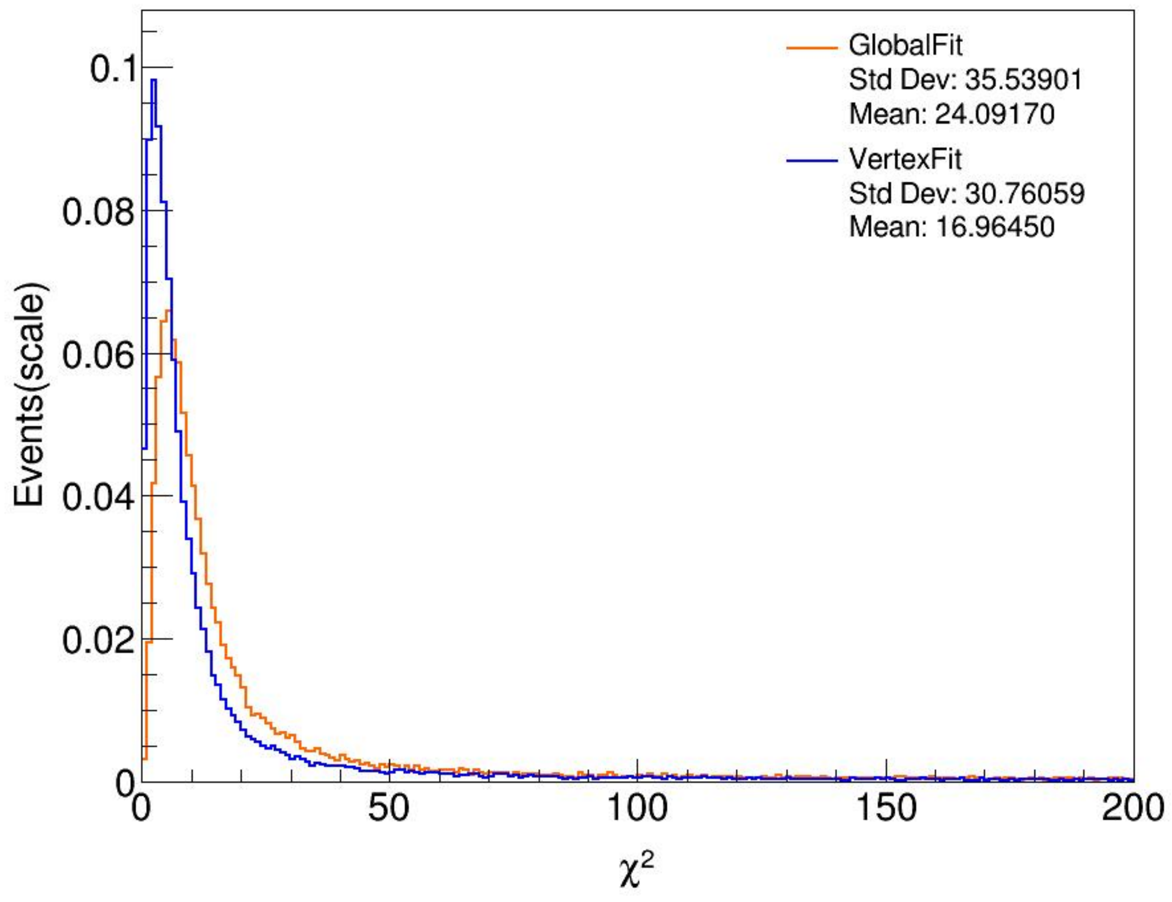
- $\overline{\Sigma}$ distribution of decayL err



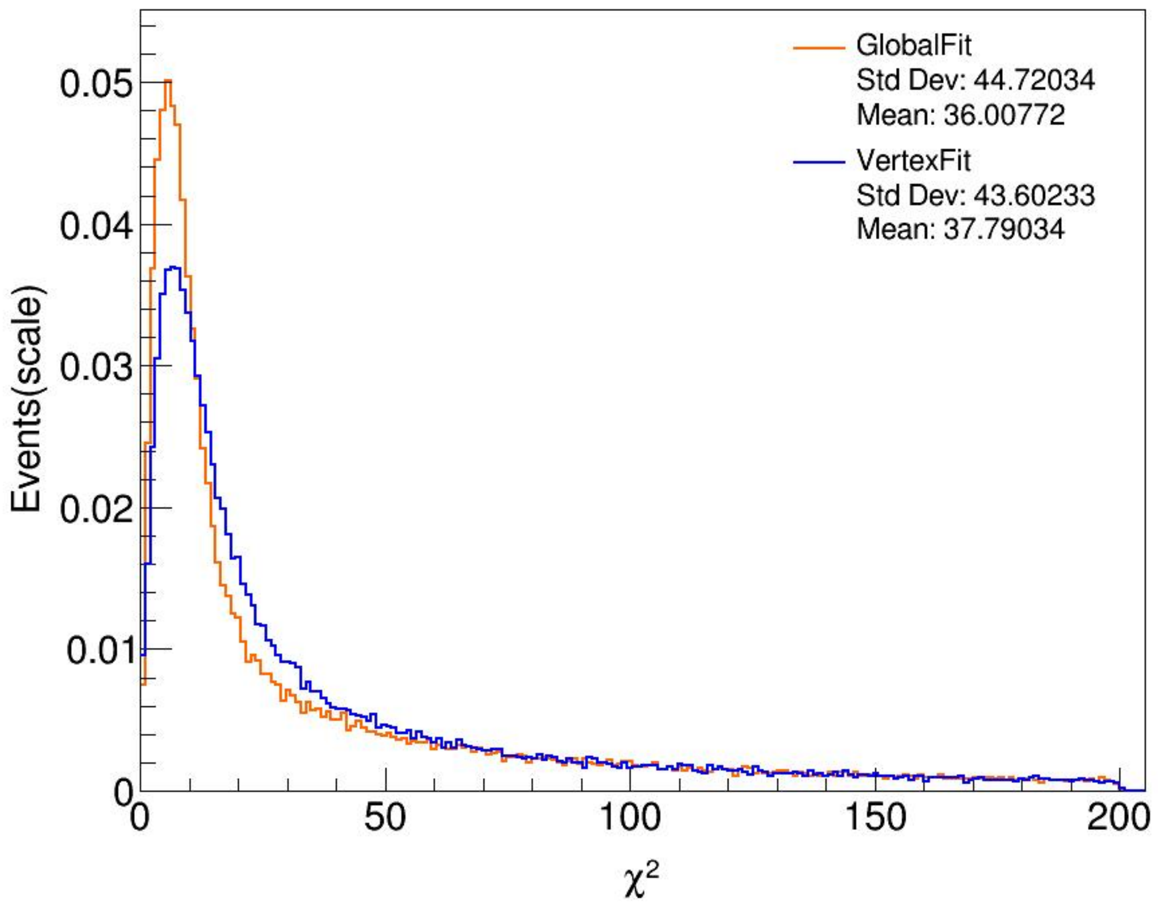
backup

➢Chisq distribution

$J/\psi \to \Lambda\overline{\Lambda} \to (p\pi^-)(\bar{p}\pi^+)$



$J/\psi \to \Sigma^+(Pr)\overline{\Sigma}(\pi_0(rr)\overline{P})$



backup

- ➢U ser guide

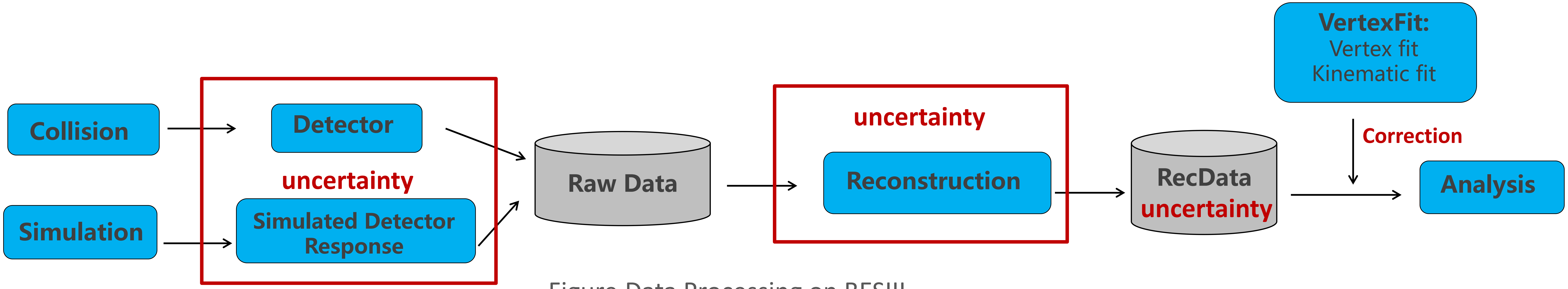
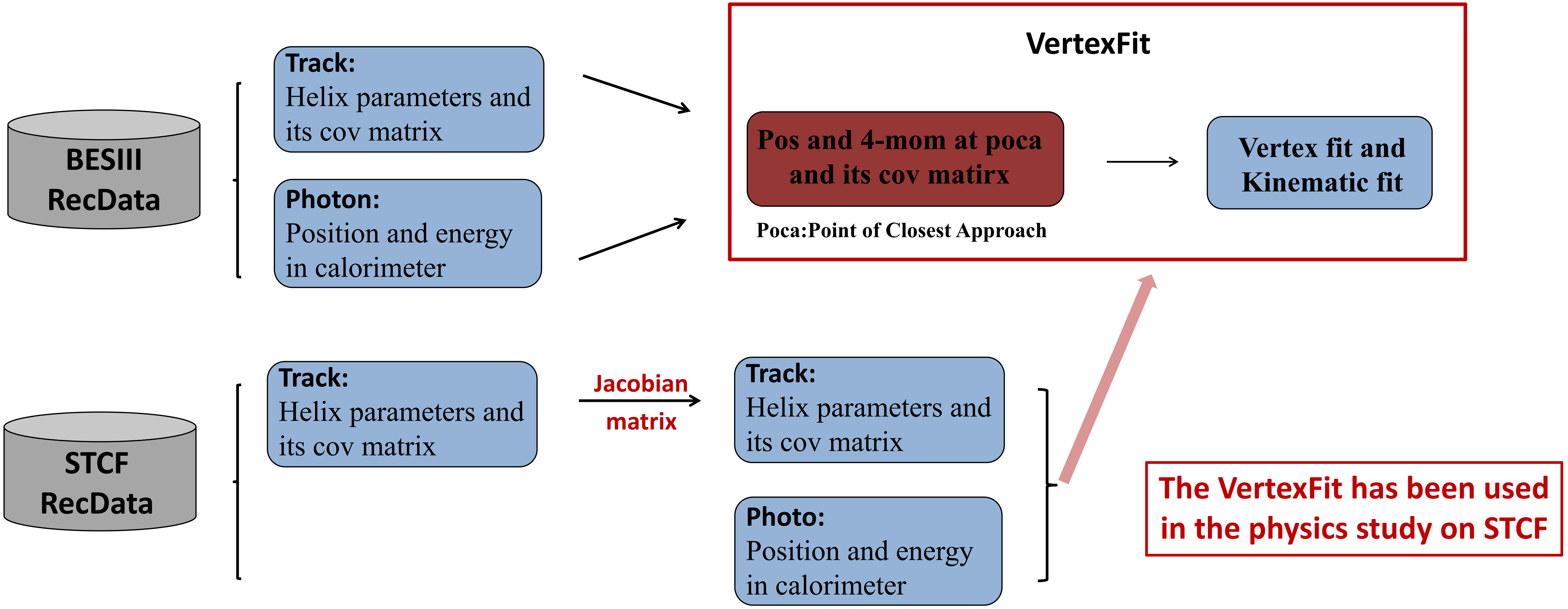


Figure.Data Processing on BESIII

03 Implementation of VertexFit ---From BESIII

➢ Implementation of Vertx fit and Kinematic fit

Based on the BESIII VertexFit algorithm, we have implemented the new algorithm for STCF after some daption and modification according to the STCF DataModel



03 Implementation of Global vertex fit --- From BelleII

➢ Implementation of Global vertx fit

Based on the BelleII - Global vertex fit algorithm, we have implemented GlobalVertexFit on STCF