

LHC-ATLAS実験における FTK Fast Simulationの開発

早稲田大学 植原靖裕

Outline

- ATLAS検出器 , 内部飛跡検出器
- ATLASトリガーシステム
- FastTracker (FTK) の飛跡再構成の流れ
- MC / FTK simulation
- FTK fast simulation(FastSim)の概要
- FastSim新手法の開発状況
- まとめ , 今後

ATLAS検出器

LHC加速器のビーム衝突点に設置

- 内部飛跡検出器
- カロリメータ
- ミューオン検出器

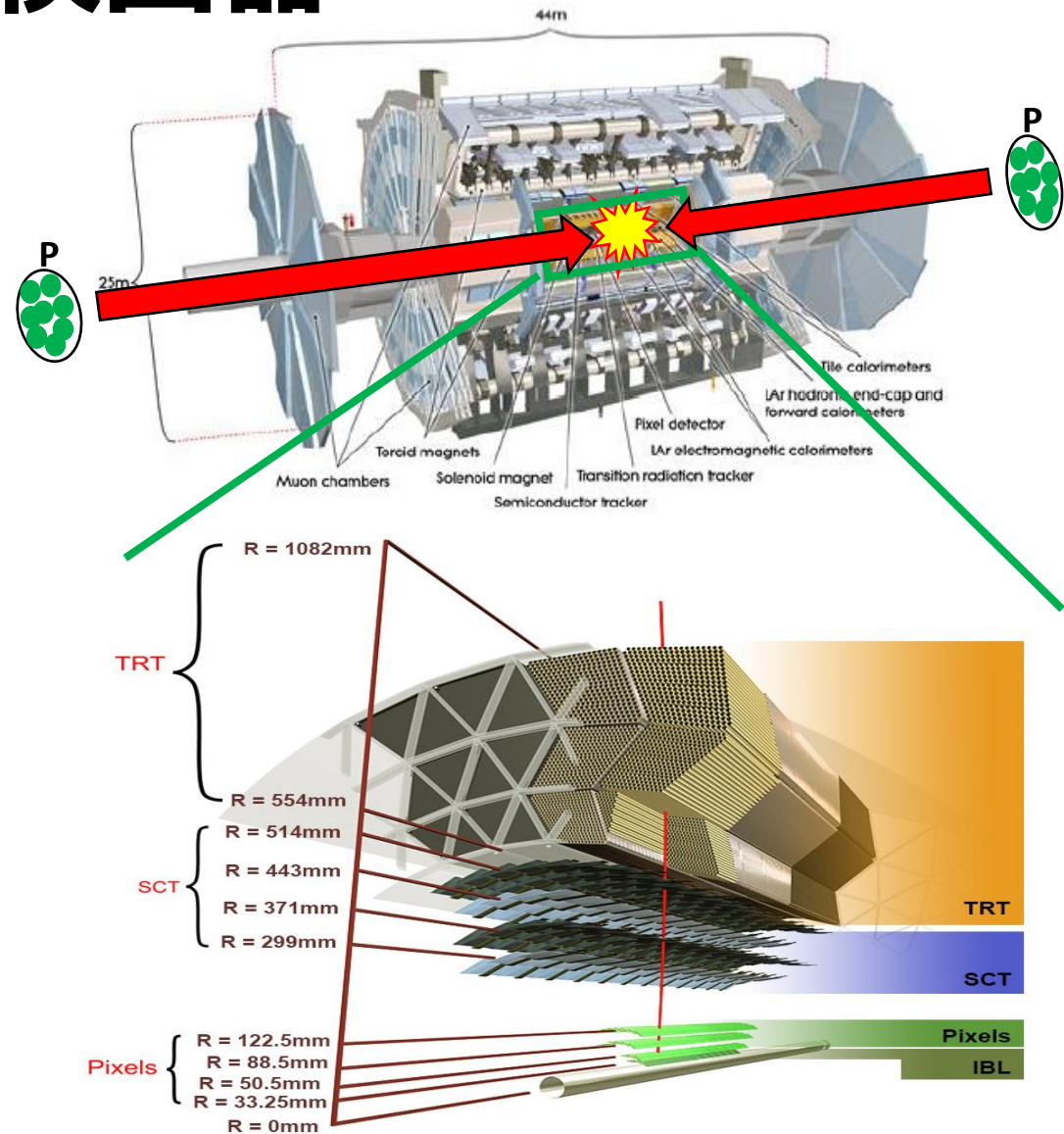
運動量、エネルギー等を精密に測定

○内部飛跡検出器

IBL : η - ϕ 2次元読み出し (1層)

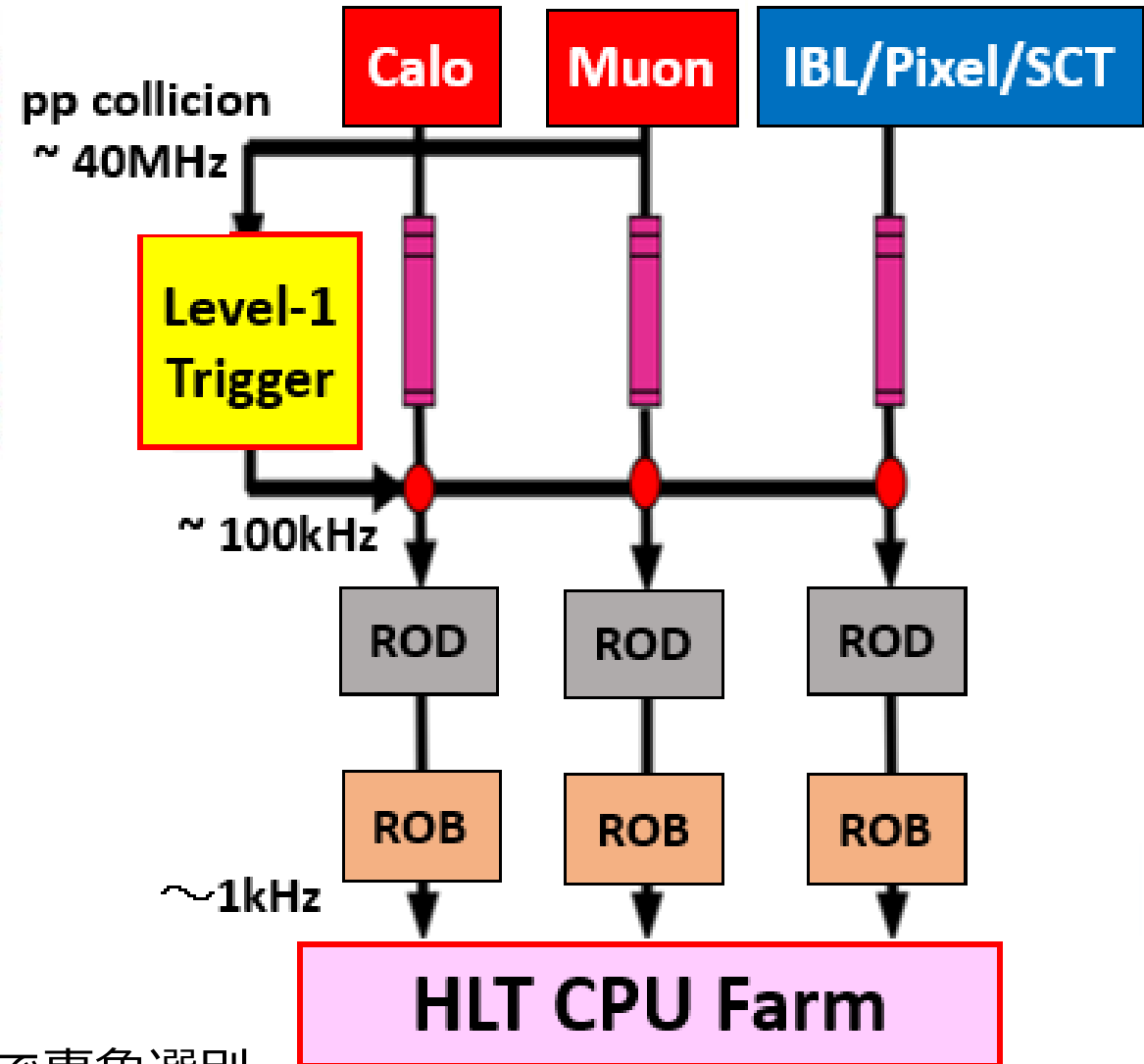
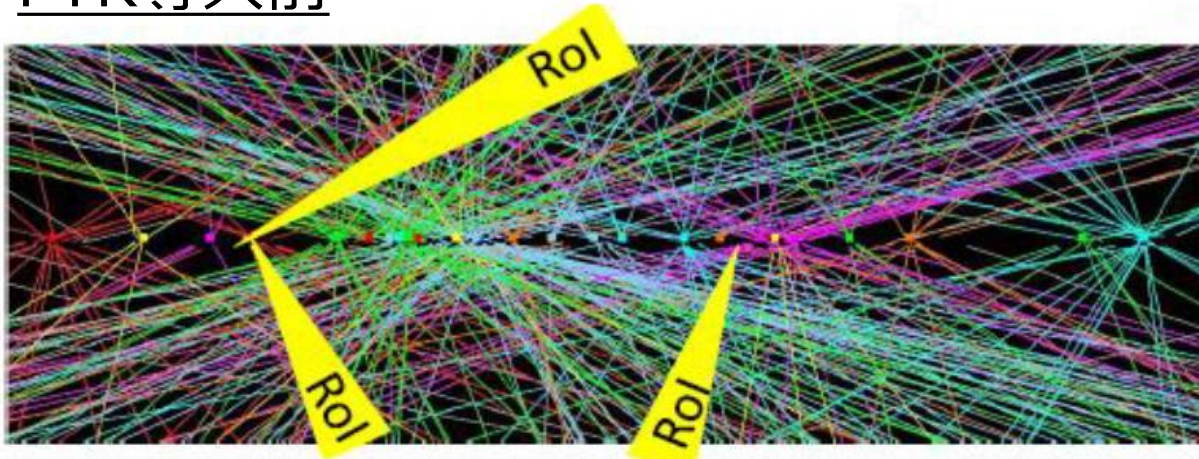
Pixel : η - ϕ 2次元読み出し (3層)
最内層をB-layer

SCT : η 方向 1次元の読み出し(8層)



ATLASトリガーシステム

FTK導入前



FTK導入前

再構成する事象 一部領域(RoI)

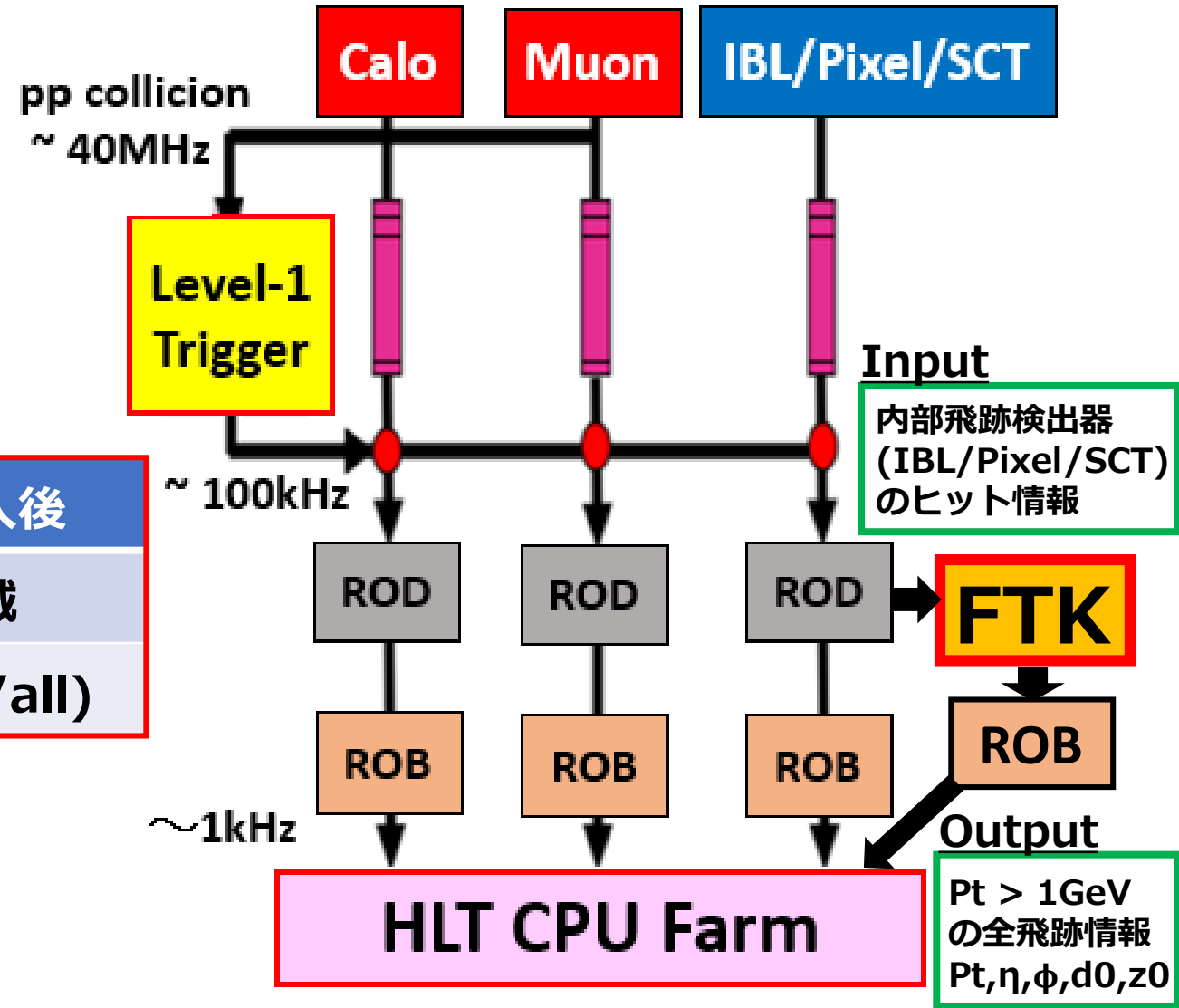
処理時間 100(ms/RoI)

✓ Level-1 : ハードウェア
事象選別 (RoI)

✓ HLT : ソフトウェア
RoIの飛跡と衝突点を再構成・Triggerで事象選別

ATLASトリガーシステム

FTK導入後



再構成する事象

FTK導入前

RoI

FTK導入後

全領域

処理時間

100(ms/RoI)

100(μs/all)

✓ FTK : ハードウェア
高速に全領域の飛跡・衝突点を再構成

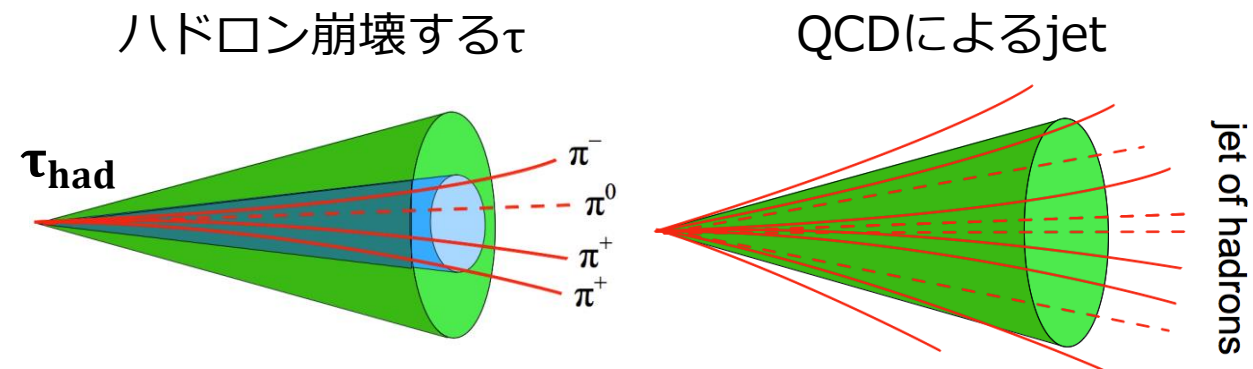
⇒ パイルアップによる影響を抑制
HLTでより精度の良い事象選別

FTKによる恩恵： τ triggerの改善

- ✓ ハドロン崩壊する τ の同定において
QCDによるjetとの分離が非常に重要

ハドロン崩壊する τ ：
狭い範囲に1or3本の荷電粒子の飛跡が存在

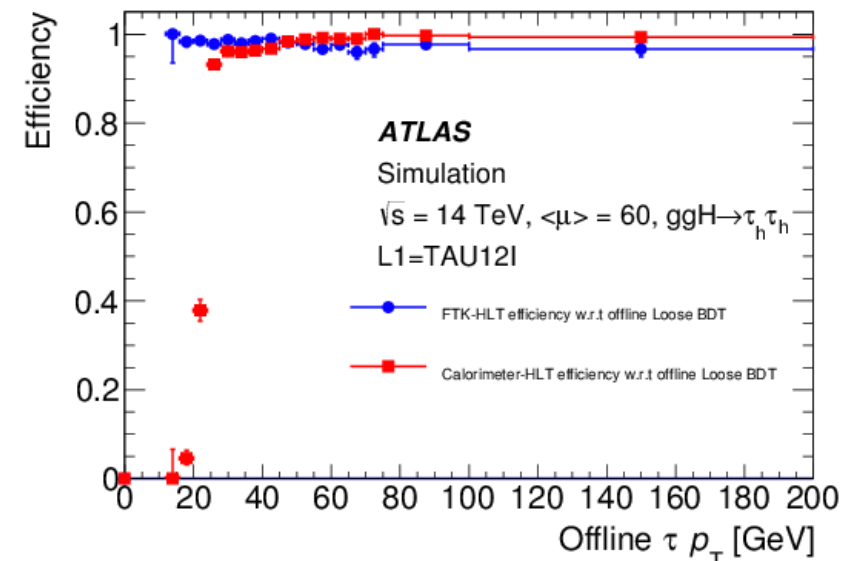
QCDによるjet：
広い範囲に荷電粒子が分布



○FTKによる恩恵

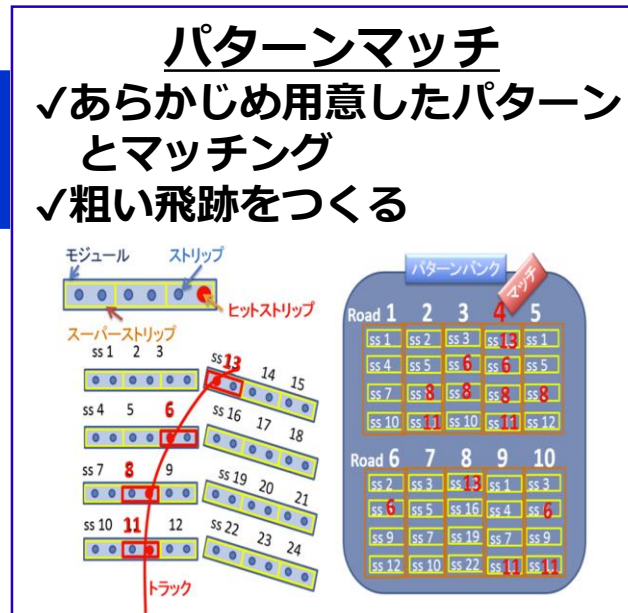
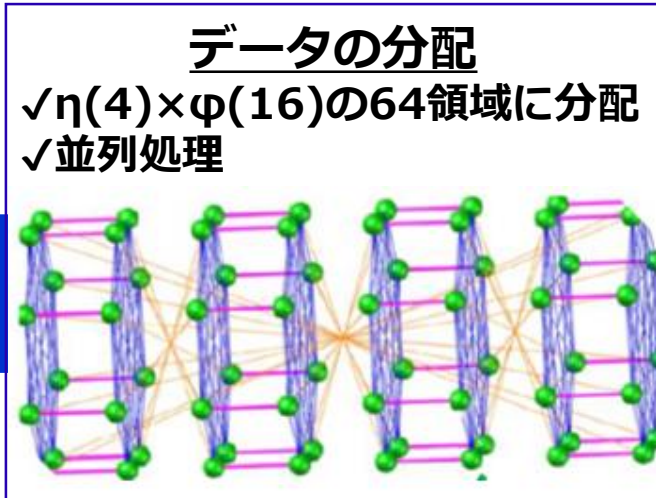
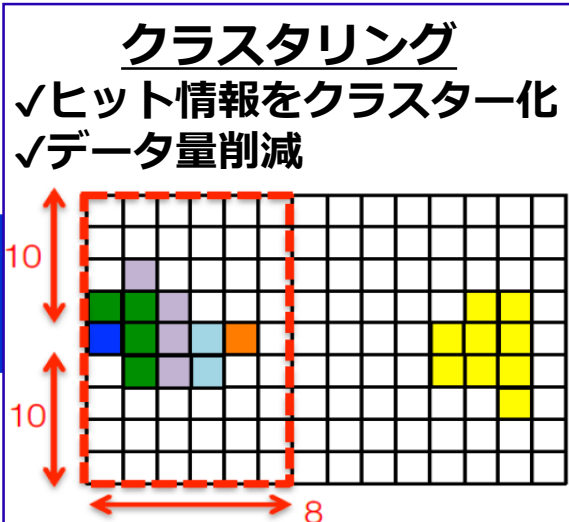
- 飛跡の再構成
 - 全領域における τ の同定
- 衝突点の再構成によるパイルアップの抑制
 - 分離変数の改善

FTKにおけるturn-onの改善



Fast Tracker(FTK) 飛跡再構成までの処理 7/19

IBL
Pixel
SCT
*Hit*情報



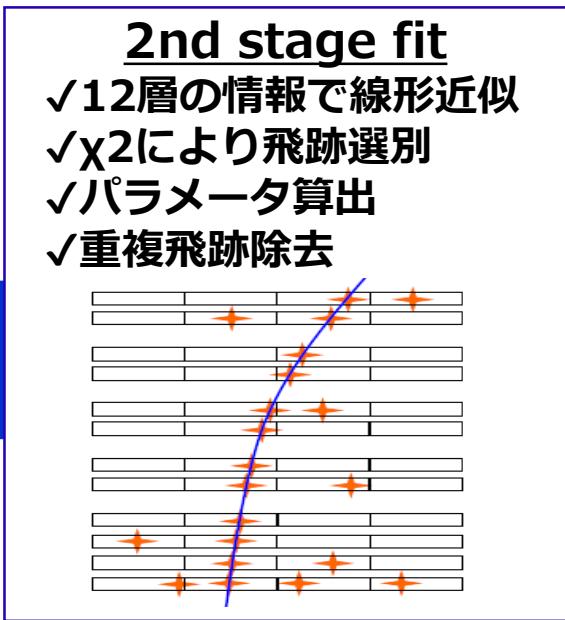
✓64領域をMerge

HLT

FTK track
 $P_t, \eta, \phi, d_0, z_0$

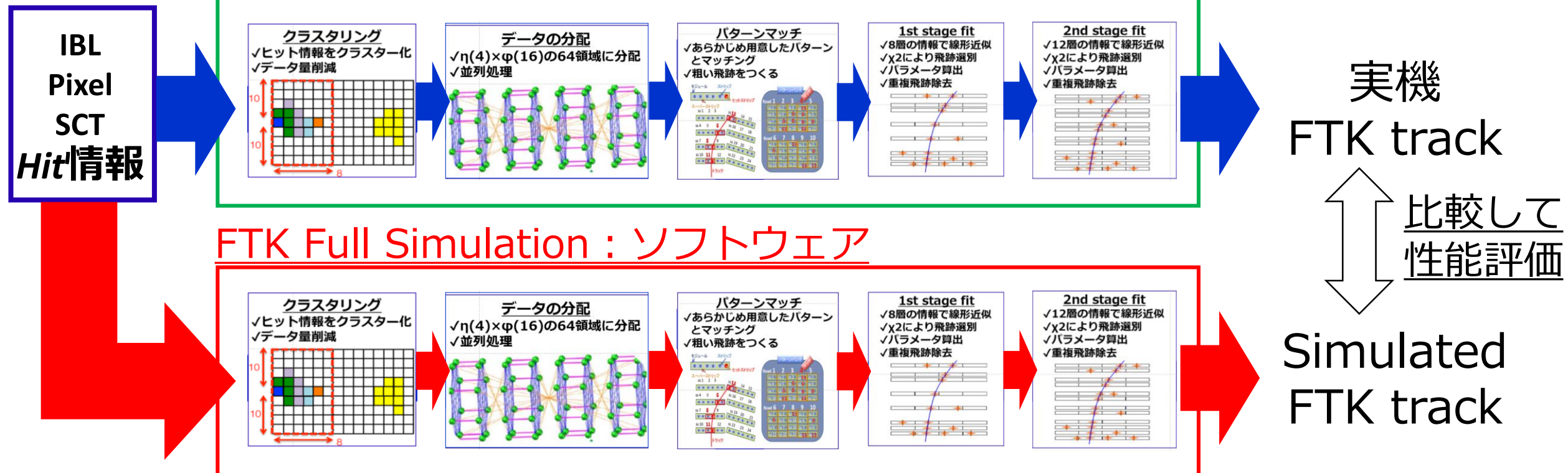
$$P_i = \sum_{k=1}^N C_{ik} X_k + q_i$$

($i = 1, 2, \dots, 5$)



FTK Simulation : 実機の性能評価

FTK System : ハードウェア

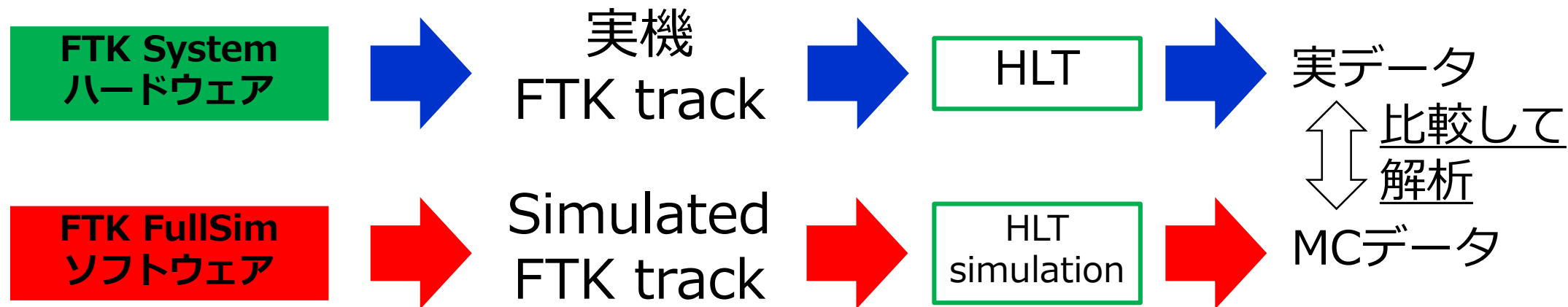


✓ FTK Systemを処理ごとにsimulate

⇒ 各処理をbit levelで動作確認可能

⇒ FTK Systemの理解・デバッグにおいて非常に重要

FTK Simulation : 解析における利用



○FTK Full Simulation(Full Sim)

✓ ハードウェアの処理をソフトウェアで再現

- クラスタリング
- データの分配
- パターンマッチ
- 1st, 2nd stage fit

処理時間大

リソース大

⇒ 全モンテカルロシミュレーションに
FTK Full Simulationを走らせるのは
不可能

***FTK Fast Simulation* の開発**

✓ 簡易なアルゴリズム

- 高速
- リソース小

✓ Simulated FTK track を再現
が必要不可欠

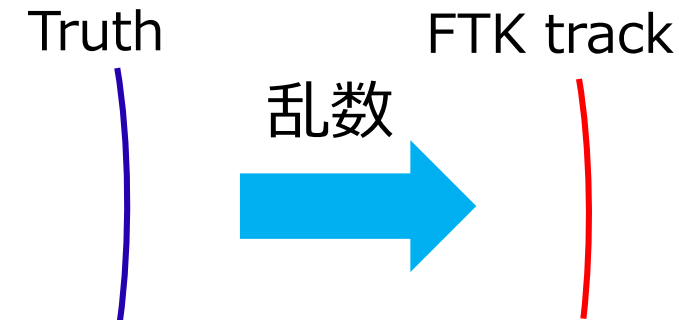
FTK Fast Simulation (FastSim)

- ✓ 高速処理
- ✓ FullSimによる再構成率・分解能を再現

○主なmethod

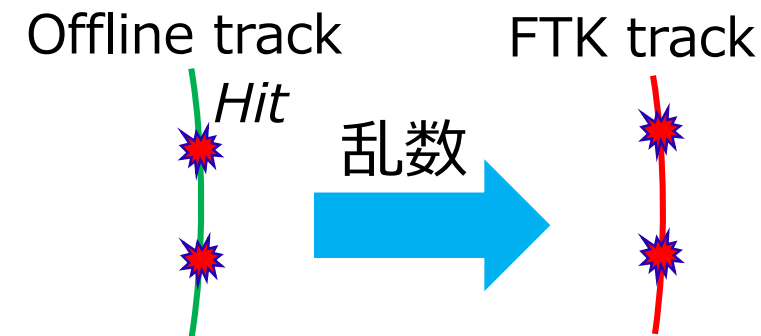
Truth-seeded approach

- ✓ FullSimの再構成率・分解能 (w.r.t Truth)を保存
- ✓ 乱数でFTK trackのパラメータを決定
- ✓ シンプルで高速



Offline-seeded approach

- ✓ FullSimの再構成率・分解能 (w.r.t Offline track)を保存
- ✓ 乱数でFTK trackのパラメータを決定
- ✓ シンプルで高速
- ✓ パイルアップを再現可能



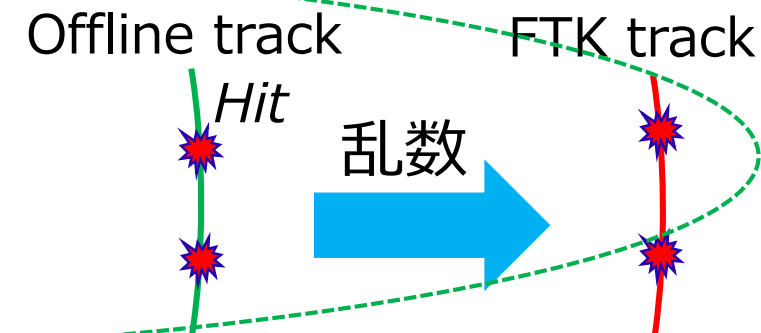
FTK Fast Simulation (FastSim)

- ✓ 高速処理
- ✓ FullSimによる再構成率・分解能を再現

シンプル・高速・パイルアップを再現可能な
*Offline-seeded method*を開発

Offline-seeded approach

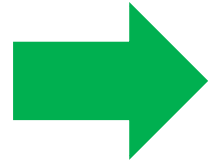
- ✓ FullSimの再構成率・分解能 (w.r.t Offline track)を保存
- ✓ 乱数でFTK trackのパラメータを決定
- ✓ シンプルで高速
- ✓ パイルアップを再現可能



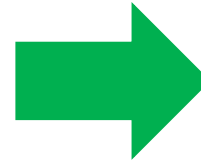
Offline-seeded Fast Simulation 概要

Input

Offline track parameters
 $q/P_t(P_t), \eta, \varphi, d_0, z_0$



Fast Simulation
 (参照)



Output

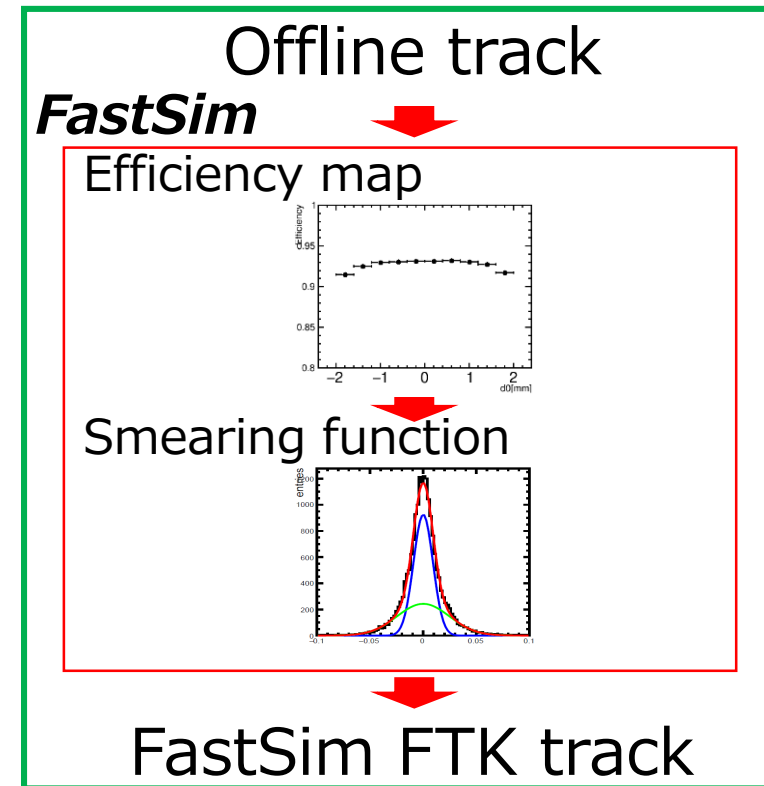
FastSim FTK track parameters
 $q/P_t(P_t), \eta, \varphi, d_0, z_0$

✓ Offline-seeded FastSimの構造

- Input : Offline track parameters
- Output : FastSim FTK track parameters
- 参照 : 各phase space(Region I)におけるEfficiency map
 各phase space(Region II)におけるSmearing function

✓ 理論上パイルアップも再現可能

シンプル, 高速, パイルアップ再現可能



使用サンプル

- ✓ 10muons/event を 500000events (5M muons) 使用
- ✓ 最もシンプルな飛跡でmethodの確立

Matchingの定義

- ✓ Offline tracksとFTK tracksを1:1に対応づけ
- ✓ minimum ΔR (min_ΔR) matchingを使用

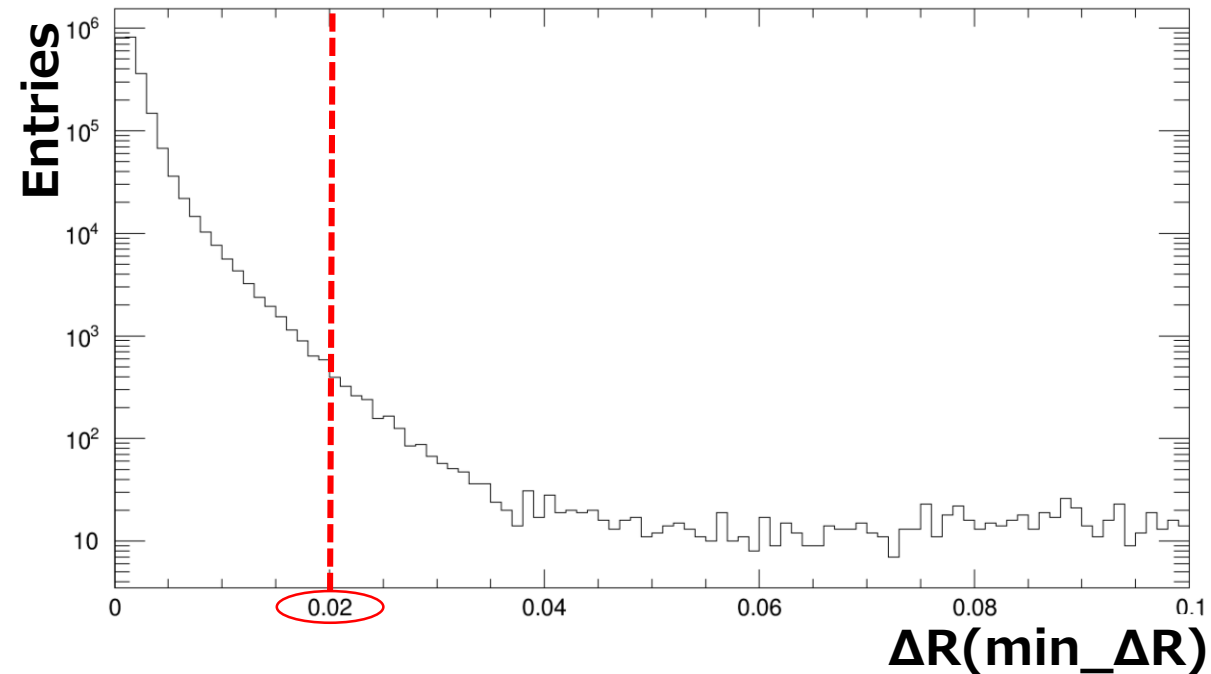
$$\Delta R = \sqrt{(\eta_{\text{offline}} - \eta_{\text{FTK}})^2 + (\phi_{\text{offline}} - \phi_{\text{FTK}})^2}$$

- ✓ ΔR matching threshold を0.02と定義

matching efficiency : 93%

$$\text{matching efficiency} = \frac{\text{matchした offline track の本数}}{\text{全offline trackの本数}}$$

“min_ΔR” 分布



Region I の定義 (For Efficiency)

各phase spaceにおけるEfficiency map と smearing function を用意するために領域の分け方を定義

- ✓ EfficiencyのOffline track parameter依存を調べた
- ✓ Efficiencyは Offline trackの q/Pt , η , $d0$, $z0$ に依存する

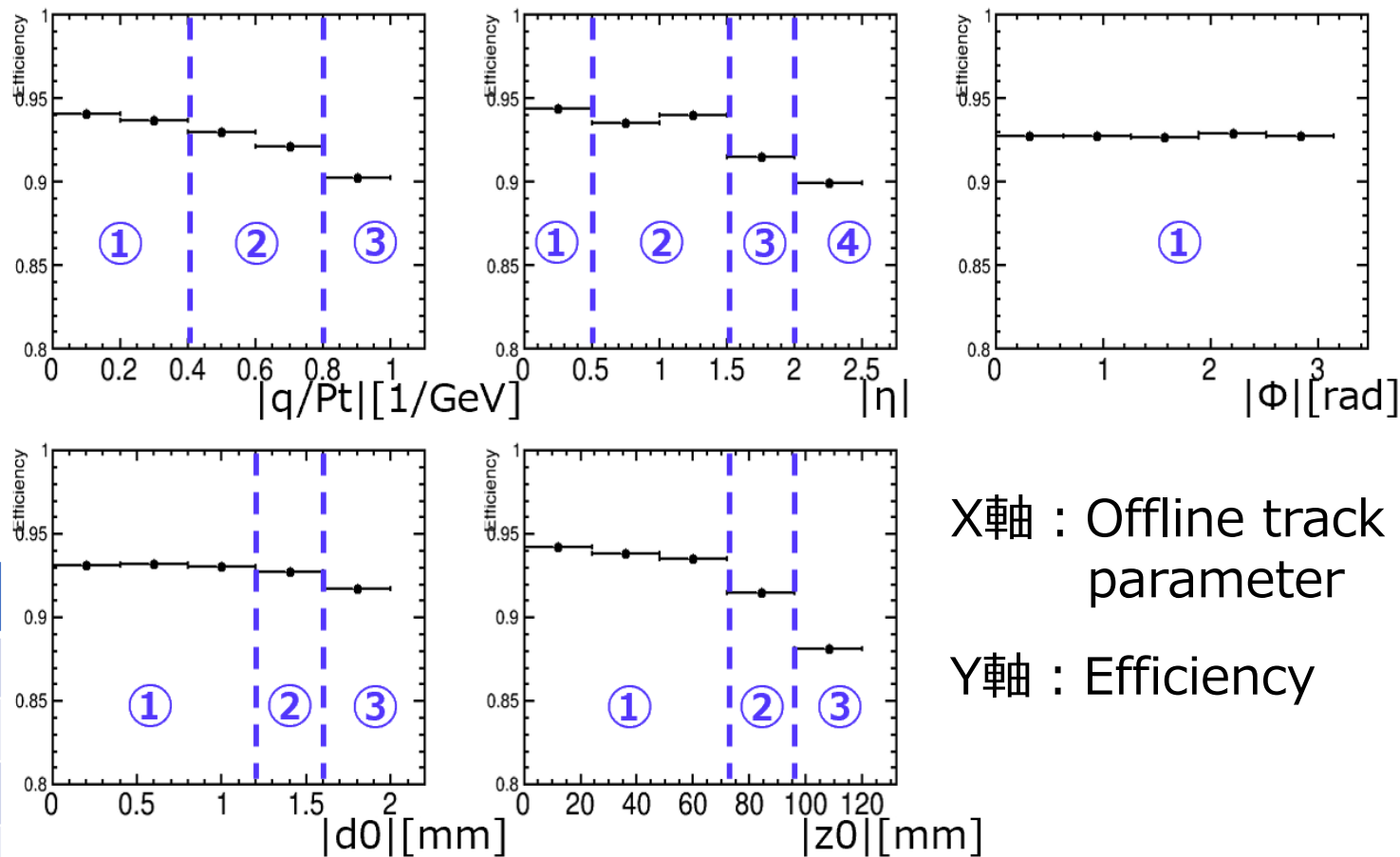
⇒ $|q/Pt|$, $|\eta|$, $|d0|$, $|z0|$

で計 108 領域を定義

Efficiency map のための領域分け

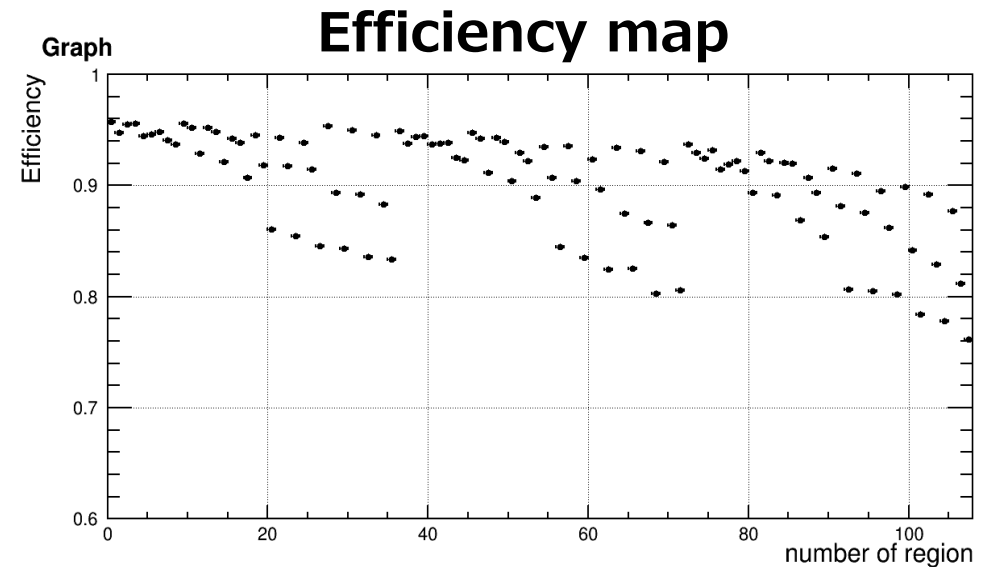
Region	①	②	③	④
$ q/Pt $	0 ~ 0.4	0.4 ~ 0.8	0.8 ~ 1	
$ \eta $	0 ~ 0.5	0.5 ~ 1.5	1.5 ~ 2	2 ~ 2.5
$ d0 $	0 ~ 1.2	1.2 ~ 1.6	1.6 ~ 2	
$ z0 $	0 ~ 72	72 ~ 96	96 ~ 120	

各parameterに関するmatching efficiency分布



Efficiency mapping

- ✓ Region I の各領域におけるefficiencyを調べた (Efficiency map)
- ✓ Efficiency mapをinputのOffline trackに適用
⇒ FullSim(w.r.t Offline)のEfficiencyを再現

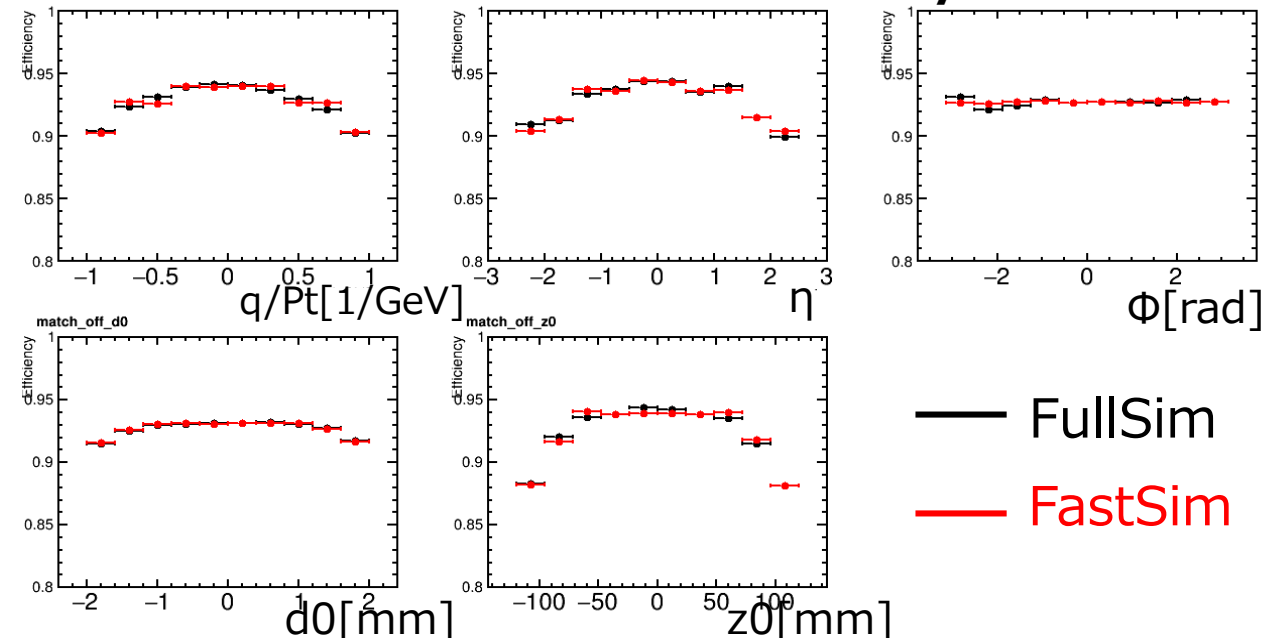


○ Performance

- ✓ FullSimとFastSimのefficiencyを比較した
 - FullSimとFastSimでよく一致

Efficiency map works well !

FullSim と FastSim のefficiency比較



Region II の定義 (For Smearing)

- ✓ Resolutionと Offline track パラメータの相関を調べた

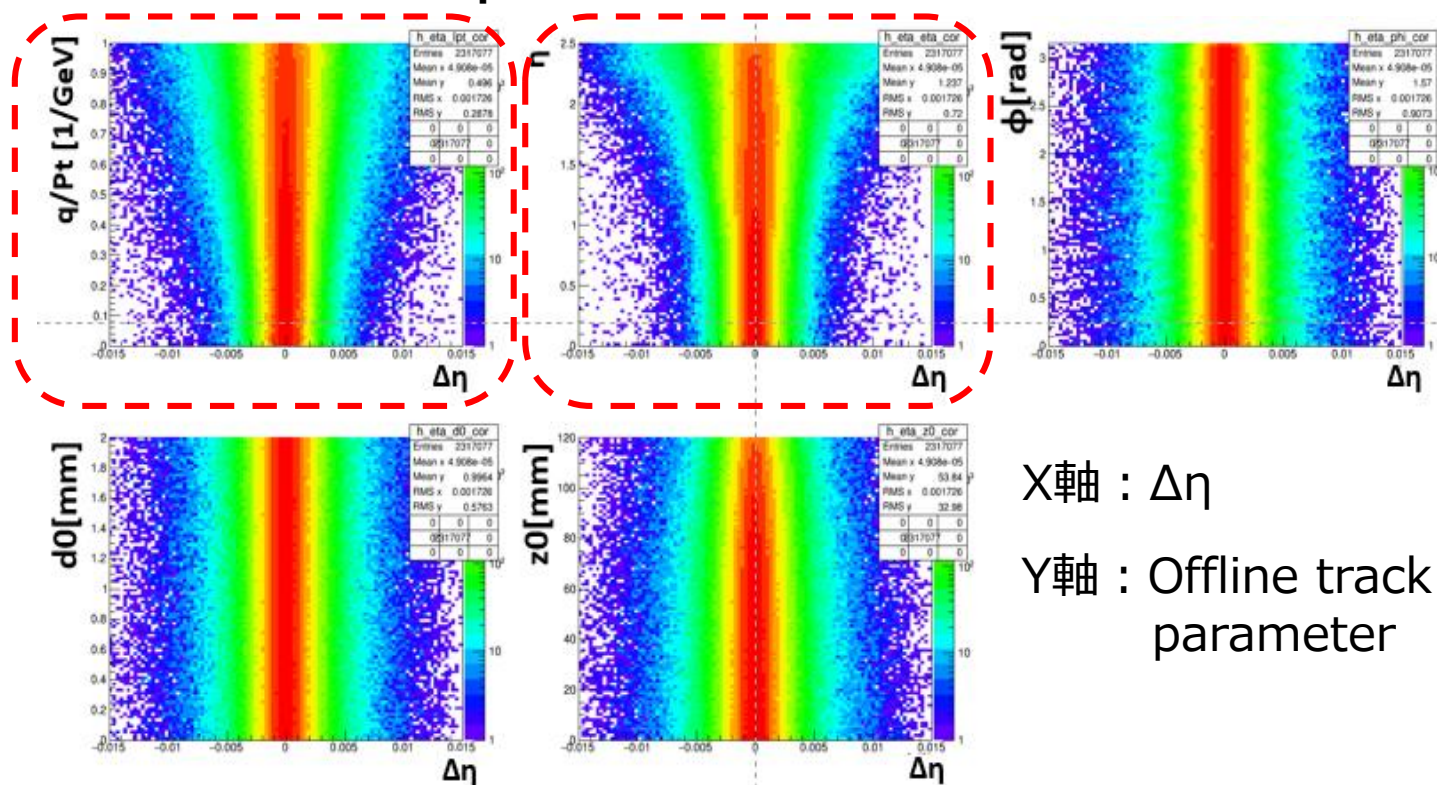
$$\text{Resolution} \equiv \text{Parameter}_{\text{offline}} - \text{Parameter}_{\text{FTK}}$$

- ✓ Resolutionは Offline trackの q/Pt と η に依存する

⇒ **|q/Pt| , | η |**で **108領域**を定義

$$108\text{領域} = 9 (q/Pt) \times 12 (\eta) \text{領域}$$

$\Delta\eta$ vs Offline track Parameter



X軸 : $\Delta\eta$

Y軸 : Offline track parameter

Smearing functionのための領域分け

Region	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫
q/pt	0~0.2	0.2~0.3	0.3~0.4	0.4~0.5	0.5~0.6	0.6~0.7	0.7~0.8	0.8~0.9	0.9~1.0			
η	0~0.3	0.3~0.5	0.5~0.7	0.7~0.9	0.9~1.1	1.1~1.3	1.3~1.5	1.5~1.7	1.7~1.9	1.9~2.1	2.1~2.3	2.3~2.5

Resolution Smearing

- ✓ Region II の各領域におけるresolutionを調べた
- ✓ 各resolution分布に対してdouble gaussianでfit

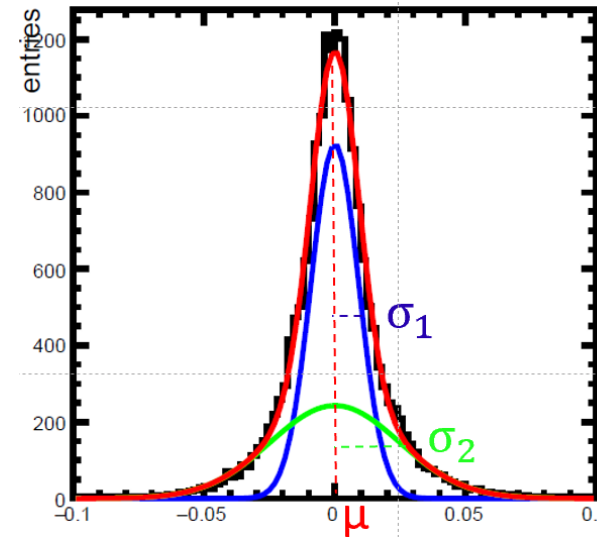
$$f(x) = A \left(\frac{\text{Frac}}{\sigma_1 \sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma_1^2}\right) + \frac{(1-\text{Frac})}{\sigma_2 \sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma_2^2}\right) \right)$$

- ✓ すべてのfit parameterをsmearing functionとして保存
- ✓ Smearing functionをOffline trackに適用

⇒ FullSim(w.r.t Offline)のresolutionを再現

○ Performance

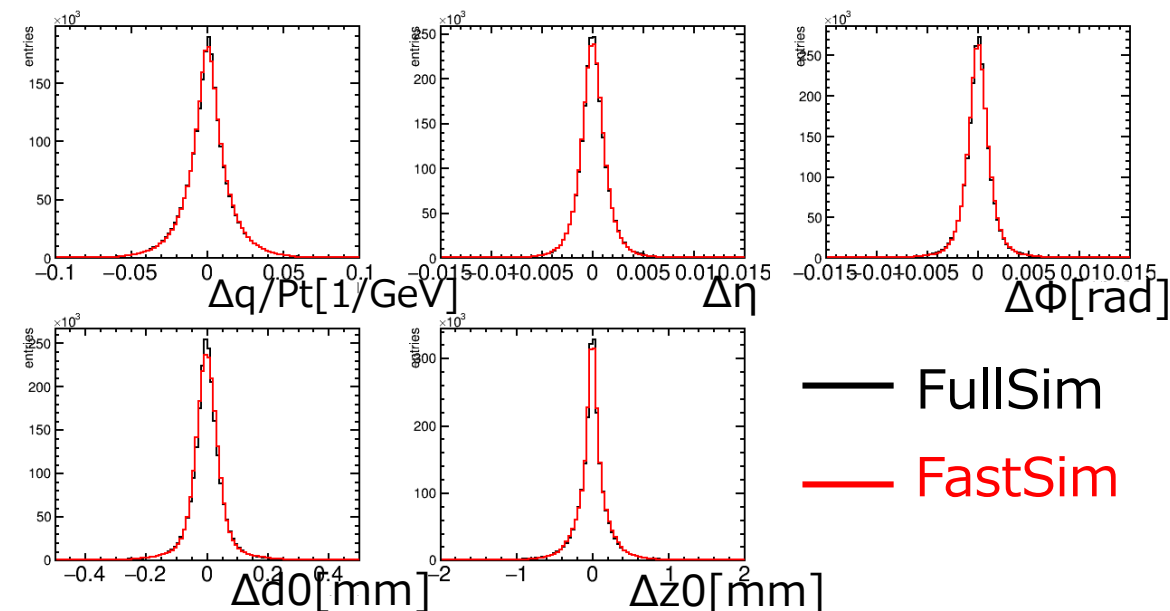
- ✓ FullSimとFastSimのresolutionを比較
 - Bulk partに顕著な違いはない
 - Tail partでずれ (next page)



— Double gaussian
— Bulk part
— Tail part

$$\text{Frac} = \frac{\text{Bulk part}}{\text{Double gauss}}$$

FullSim と FastSim のresolutionを比較



— FullSim
— FastSim

Tail partの要因

- ✓ Δd_0 に関してtail partの要因を調べる
ためにBulk partとtail partのOffline track
のparameterを比較

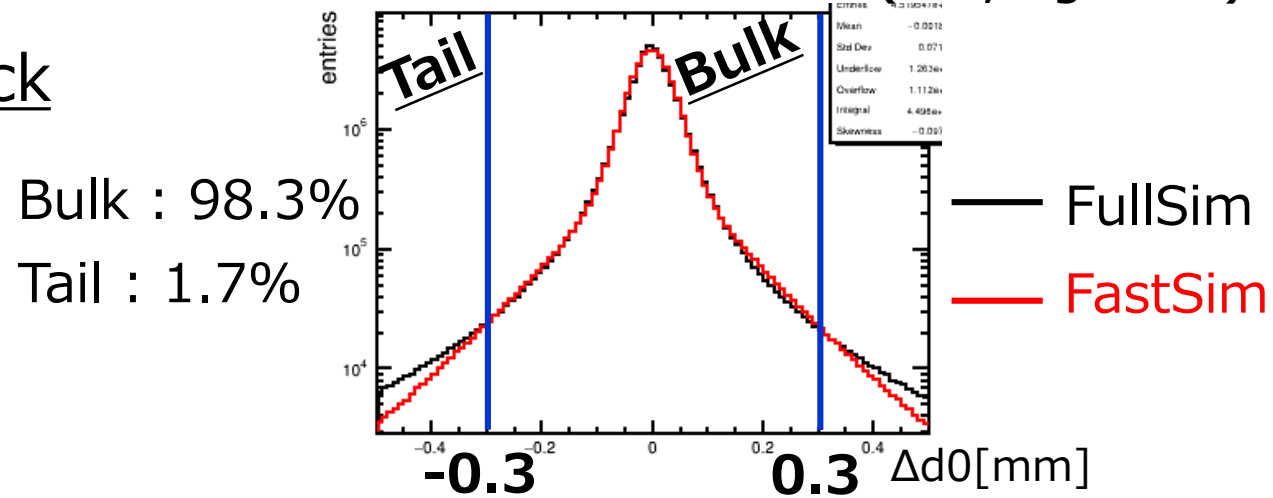
○結果

- ✓ η と Φ の分布に顕著な違いがみられた
 - η : High $|\eta|$ の領域
 - Φ : 22個のピークが見られた
- ⇒ 検出器の構造が要因として考えられる
("22" がB-layerの数と一致)

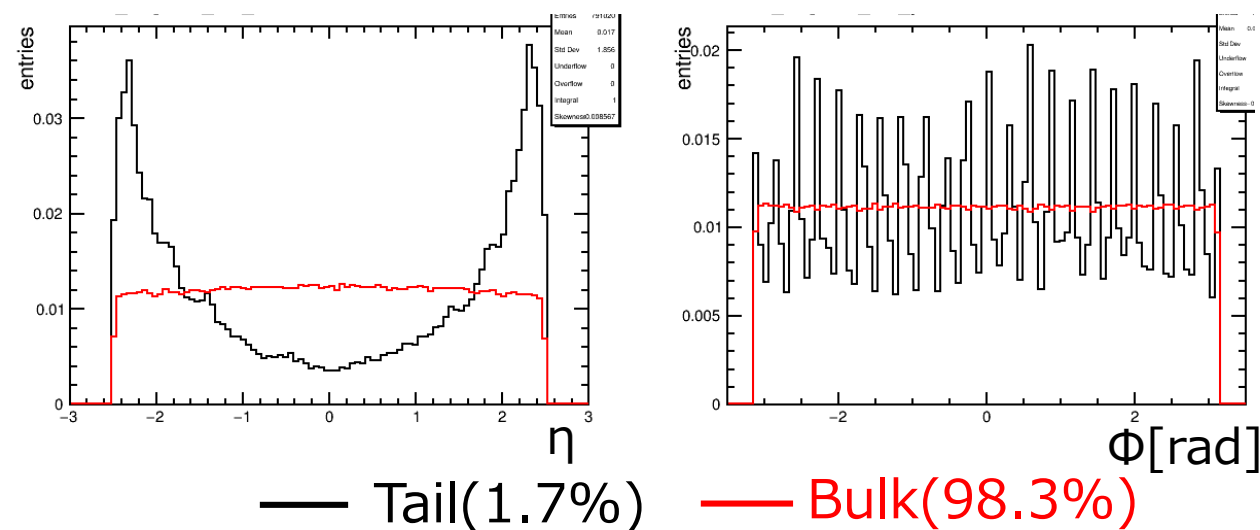
○今後

- ✓ Tail partにおけるHit情報を詳細に調べ
FullSimとFastSimの一致を試みる

FullSim と FastSim のresolution比較 (d_0 , log scale)



Bulk part と tail part におけるOffline track分布の比較



Summary & Future

OSummary

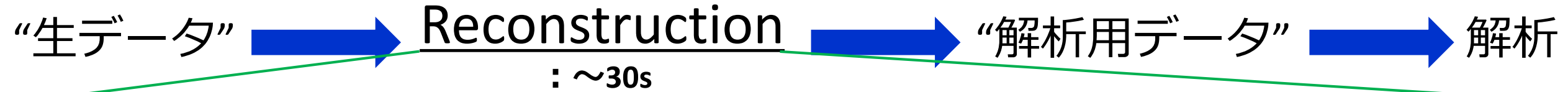
- ✓ FTKを用いた解析に高速なFTK Simulation(FastSim)の開発が必要である
- ✓ シンプルかつ高速なOffline-seeded FastSimの開発を行った
 - FullSimによる再構成率を再現
 - FullSimによる分解能を再現

OFuture

- ✓ ATLASの解析frameworkへの導入を完了させる
- ✓ 導入後、trigger simulationによるFullSimとFastSimの比較を行う
- ✓ Trigger groupからfeedbackをもらいupdateを行う
 - パイルアップ の考慮
 - Track Parameterの相関の考慮

Backup

ATLAS解析frameworkへ導入



Online trackを再構成
RoIから全領域で : +1~2s

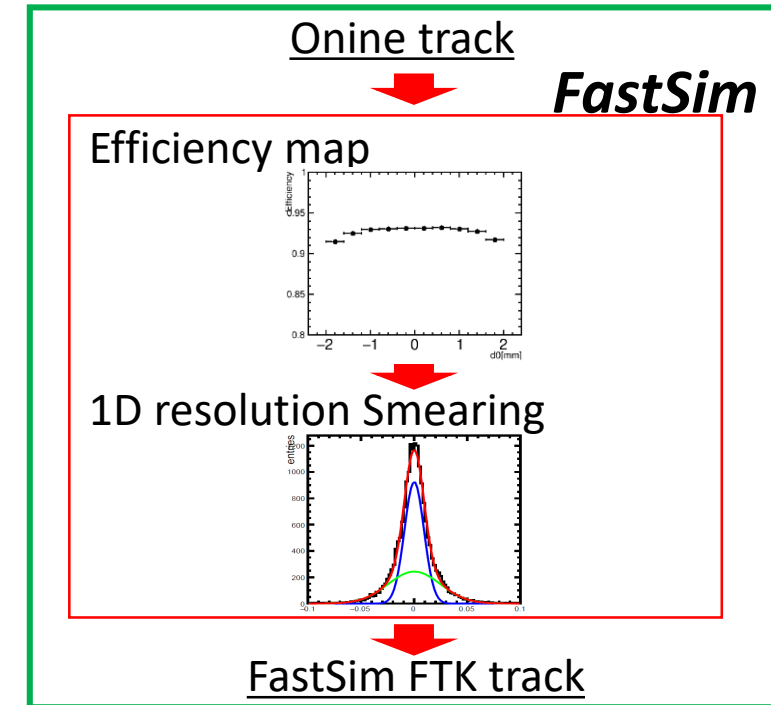
Online-seeded FastSim
“MC FTK track”を再構成 : ~10ms

Trigger Simulation → Offline trackを再構成

- ✓ Offline-seeded Fast Simulation の1st versionを提供
 - Efficiency map ✓ *Done*
 - Resolution smearing ✓ *Almost Done* (tail study : *ongoing*)
- ✓ Reconstruction内でFTK trackを再構成
- ✓ SeedをOnline Trackに移行 (RoI ⇒ 全領域)

○今後

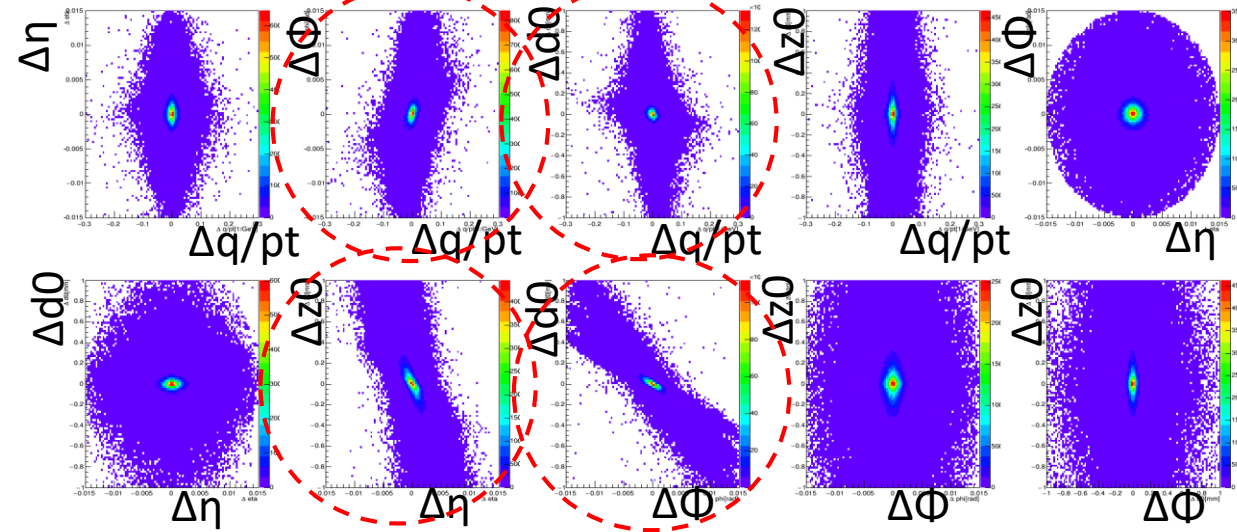
- ✓ Trigger simulationによるFullSimとFastSimの比較
- ✓ 各trigger groupからfeedbackをもらいアップデート
 - パイルアップ の考慮
 - Track parameter の相関の考慮



2D resolution smearing

- ✓ OfflineとFTK track parameter の相関を見た
 - $\Delta q/Pt$ vs $\Delta\Phi$, $\Delta q/Pt$ vs $\Delta d0$, $\Delta\Phi$ vs $\Delta d0$, $\Delta\eta$ vs $\Delta z0$ に強い相関
- ✓ 相関を考慮し2次元でresolution smearing (2D resolution smearing)を試みた
 - ⇒ First stepとして $\Delta\eta$ vs $\Delta z0$ に着目

Resolution(offline – full sim FTK)の相関



2D resolution smearingの手順

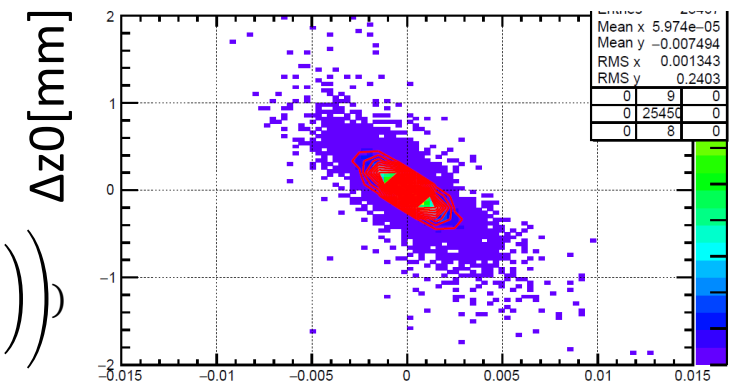
- ✓ Region II の各領域における $\Delta\eta$ vs $\Delta z0$ 分布を調べた
- ✓ 各 $\Delta\eta$ vs $\Delta z0$ 分布に対して2D double gaussian でfit

$$f(x,y) = A \left(\frac{B}{2\pi\sigma_{x1}\sigma_{y1}\sqrt{1-\rho_{xy}^2}} \exp\left(\frac{-1}{2(1-\rho_{xy}^2)}\left(\left(\frac{x-\mu_x}{\sigma_{x1}}\right)^2 - 2\rho_{xy}\left(\frac{x-\mu_x}{\sigma_{x1}}\right)\left(\frac{y-\mu_y}{\sigma_{y1}}\right) + \left(\frac{y-\mu_y}{\sigma_{y1}}\right)^2\right)\right) + \right.$$

$$\left. \rho_{xy} : \text{correlation factor} \quad \frac{(1-B)}{2\pi\sigma_{x2}\sigma_{y2}\sqrt{1-\rho_{xy}^2}} \exp\left(\frac{-1}{2(1-\rho_{xy}^2)}\left(\left(\frac{x-\mu_x}{\sigma_{x2}}\right)^2 - 2\rho_{xy}\left(\frac{x-\mu_x}{\sigma_{x2}}\right)\left(\frac{y-\mu_y}{\sigma_{y2}}\right) + \left(\frac{y-\mu_y}{\sigma_{y2}}\right)^2\right)\right)\right)$$

- ✓ すべてのfit parameterをsmearing functionとして保存
- ✓ Smearing functionをOffline trackに適用

2D double gaussianでのfitした様子



— 2D double gaussian $\Delta\eta$

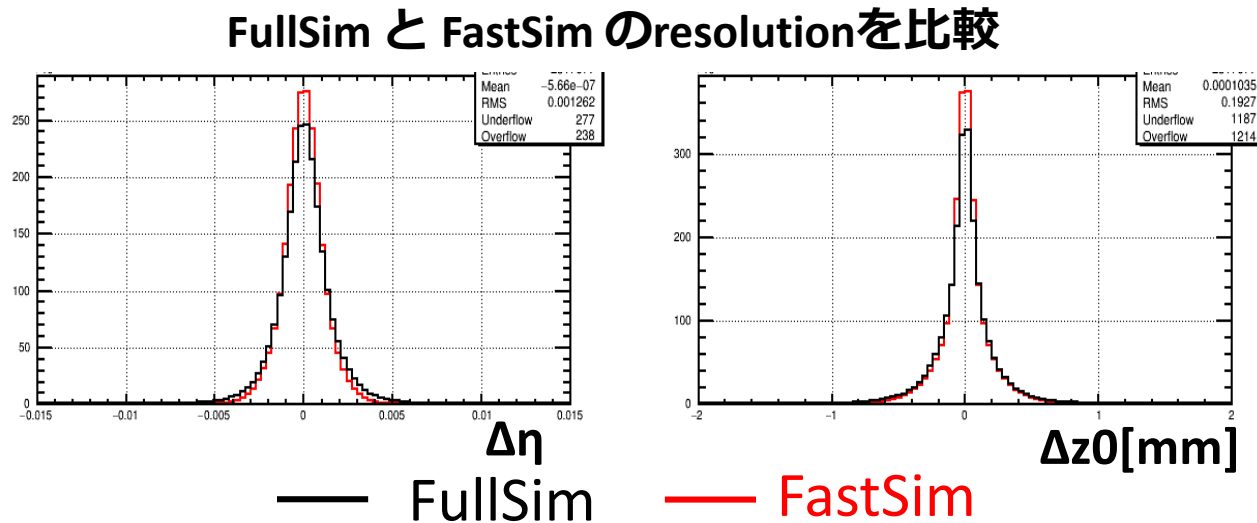
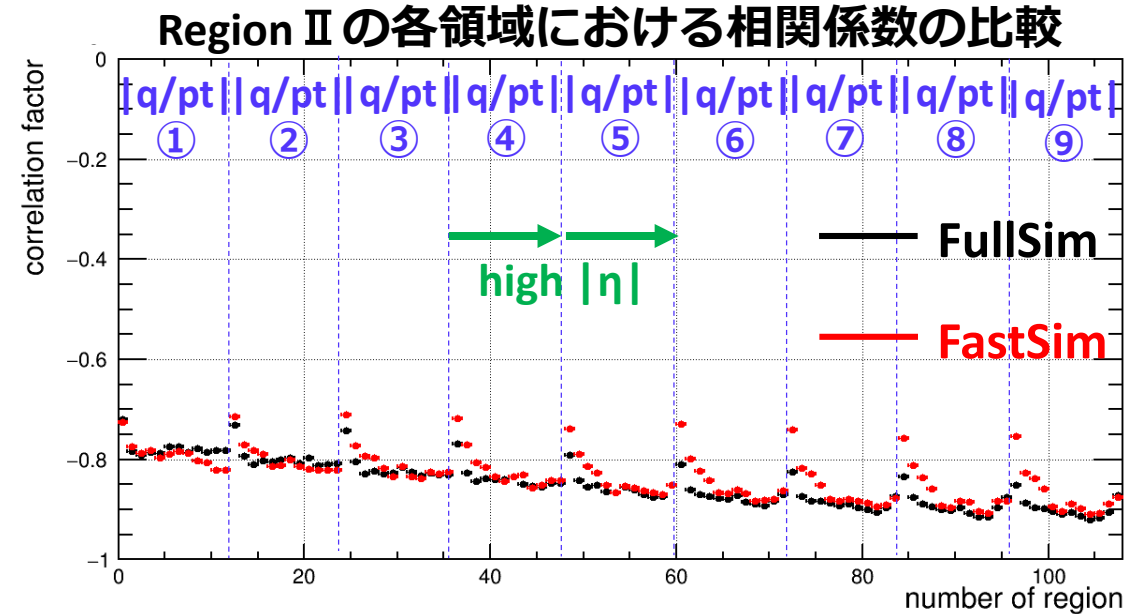
2D resolution smearing

○Performance

- ✓ FullSimとFastSimの相関係数を比較
 - high $|\eta|$ で一致
 - low $|\eta|$ では一致していない
- ✓ FullSimとFastSimのresolutionを比較
 - FullSimとFastSimでずれ

○今後

- ✓ 適切なsmearing functionを得るために fit algorithmの改善を行う (特に low $|\eta|$ におけるfitを改善する)



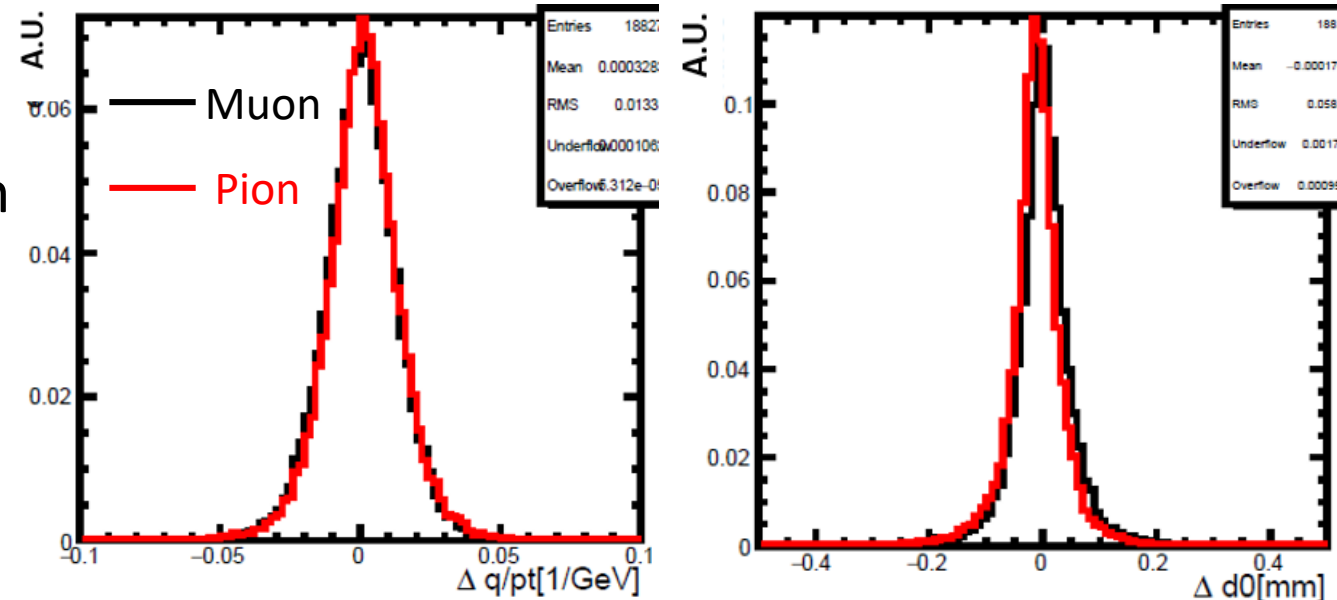
Pion FTK Tracks

- ✓ We checked the FTK tracks produced by Pion.
 - Since Pion interaction to detector is higher than Muon.

○ Compare Pion with Muon

- ✓ We checked resolution of pion in 108 regions.
 - ✓ Right plots show the resolution comparison between Muon and Pion in one of the 108 regions.
 - No difference in q/pt , η , Φ and z_0
 - Slight shift in d_0
- ⇒ Currently under investigation.

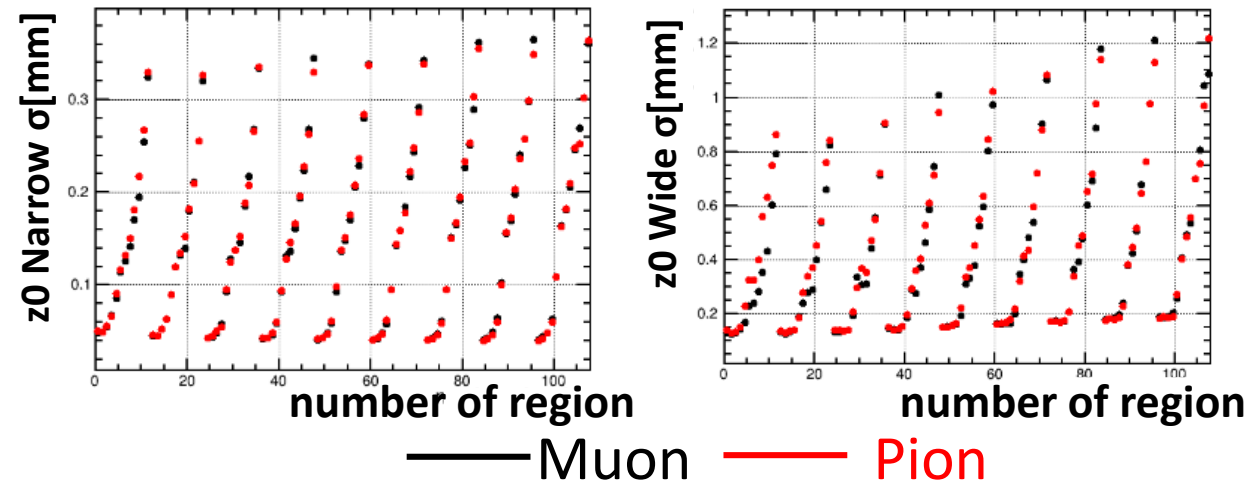
Resolution comparison between Muon and Pion



Applying Muon function to Pion

- ✓ We compared narrow σ and wide σ between Pion and Muon.
- ✓ Right plots show narrow σ and wide σ of z_0 . (other parameters are in backup p.24,25)
- ✓ Overall, there is no big difference.
⇒ We tried applying smearing function of Muon to Pion Offline tracks.

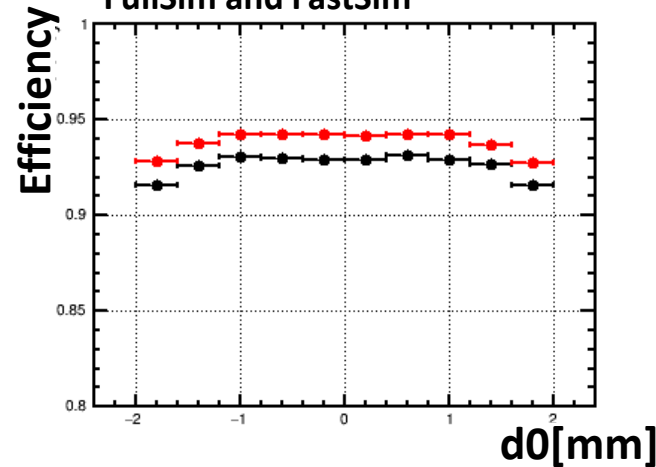
Narrow σ & Wide σ for each of 108 regions Muon and Pion



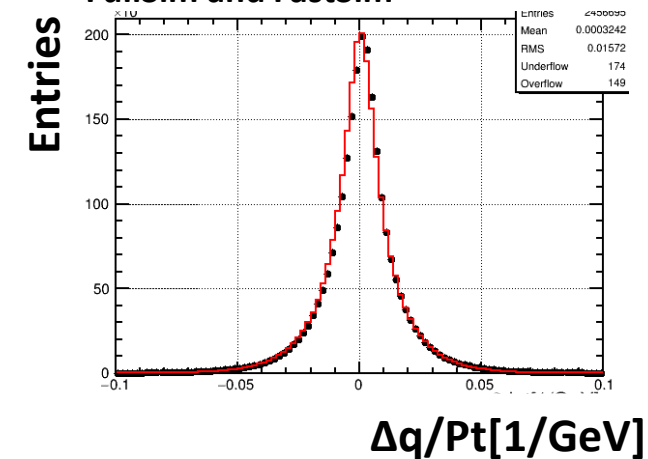
Performance

- ✓ Resolution smearing works well.
- except d_0
- ✓ Efficiency of FullSim is about 1~2% lower than that of **FastSim**.

Efficiency Comparison between FullSim and FastSim



Resolution comparison between FullSim and FastSim



Applying method to Pion

- ✓ We prepared efficiency map and smearing function for Pion and applied them to Pion Offline tracks.

Performance

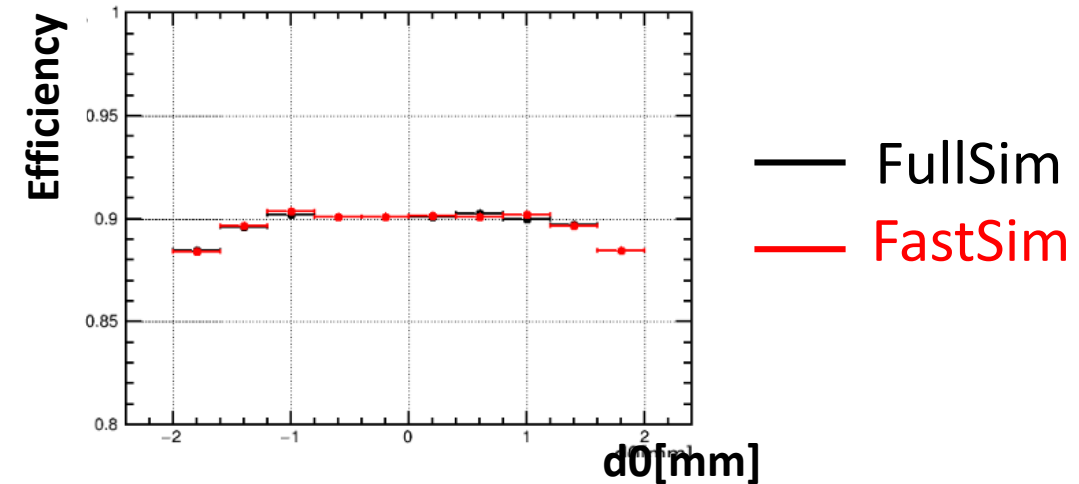
- ✓ Top right plot shows the efficiency comparison between FullSim and FastSim.
- ✓ Bottom right plot shows the resolution comparison between FullSim and FastSim.

Both efficiency map and smearing function work well.

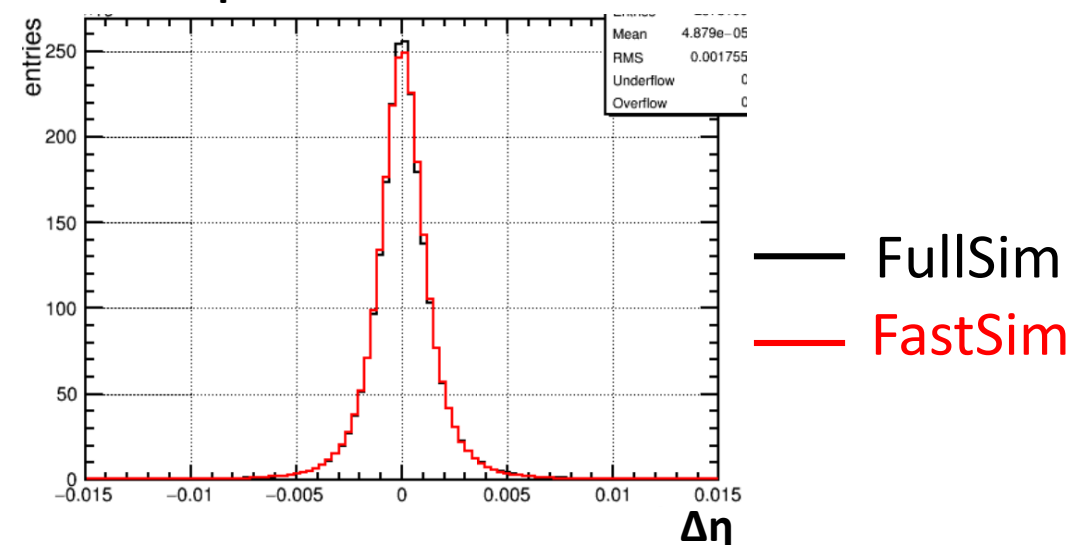
Next

- ✓ Consider how to treat the difference between Pion and Muon

Efficiency Comparison between FullSim and FastSim



Resolution comparison between FullSim and FastSim



Resolution smearing with hit information

✓ Tailの改善を試みるためにOffline trackのhit情報を考慮

✓ Offline trackを2つのカテゴリに分類

- Tracks with IBL and B-layer hit
- Tracks missing IBL or B-layer hit

✓ カテゴリごとにsmearing functionを用意し適用

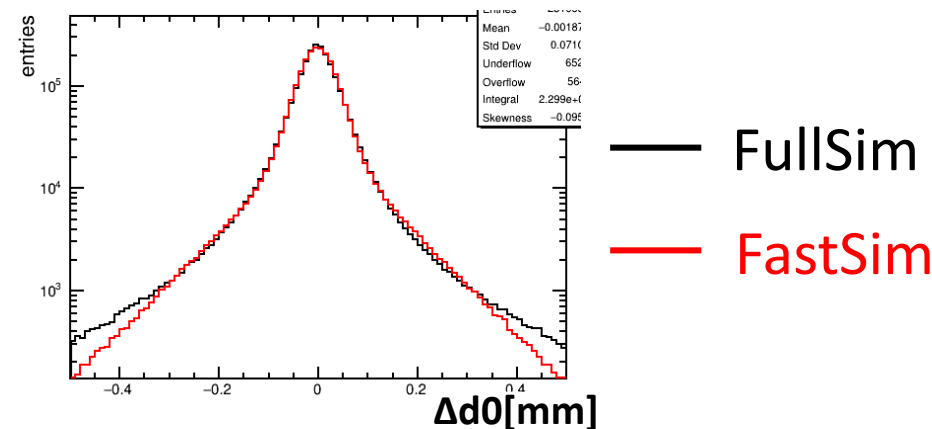
Performance

✓ FullSimとFastSimのresolutionを比較

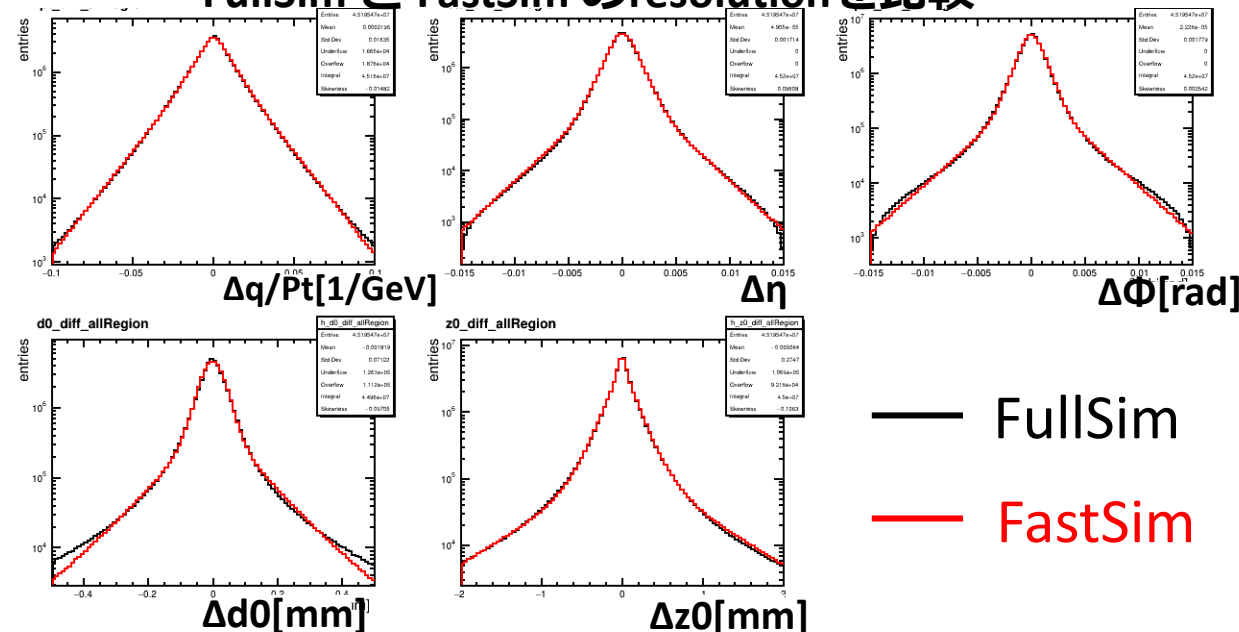
- q/P_t , η , Φ , z_0 に関して良く一致
- d_0 に関してtail partでずれが残る

✓ Hit情報を使ったが大きな改善はなかった

FullSim と FastSim のresolution比較 (d_0 , log scale)



FullSim と FastSim のresolutionを比較



修士論文研究発表会

2015年2月7日(金)

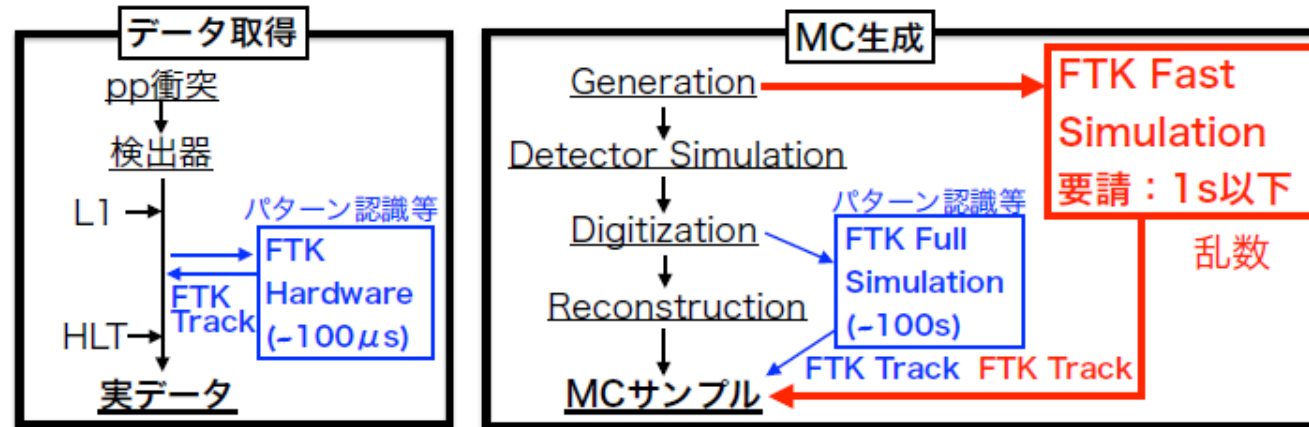
寄田研究室 修士2年 仲松 弥

スライド

Process time

FTK Fast Simulation

Run2以降、MCシミュレーションにもFTKを組み込む



2016年中に必要なMCサンプル数 : O(1G)以上

実行可能なFTK Full Simulation数 : 10M/月

→100倍以上速いシミュレーションが必要(=Fast Simulation)

方法: "Truth-seeded"

Truth(ジェネレータの情報)を元にFTK飛跡のパラメータを乱数で決定

※Fullとのトリガー使用時の差を把握・補正する必要あり

<本研究の目的> Truth-seededによるFast Simulationを作り

Full Simulationの再構成率・分解能を乱数で再現する

2017年2月8日(火)

修士論文発表会

早稲田大学先進理工学研究科

物理学及応用物理学専攻寄田研究室

修士2年亘龍太郎

スライド

Process time

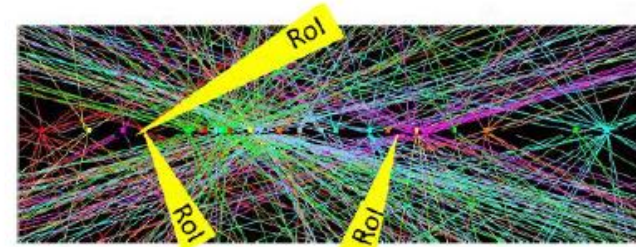
研究背景

2/13

- ATLAS実験ではLHCの高輝度環境化への対応としてFTKの挿入が進められている

FTK :

- 高速で飛跡を再構成するハードウェアシステム
- 入力：内部飛跡検出器のSi検出器12層からのヒット情報
- 出力： $p_T > 1\text{GeV}/c$ 以上の飛跡のトラックパラメータ
($p_T, \eta, \phi, d_0, z_0$)



	FTK挿入前	FTK挿入後
再構成対象	RoIのみ	全領域
処理時間	100ms/RoI	100μs/all

- FTKを用いてデータを解析するためには、比較するための大量のシミュレーションサンプルが必要($O(10^9)$)
- 現状のシミュレーションサンプルの再構成時間：~30秒 /event (FTKなし)
- 現状のFTKシミュレーションによる再構成：~60秒 /event

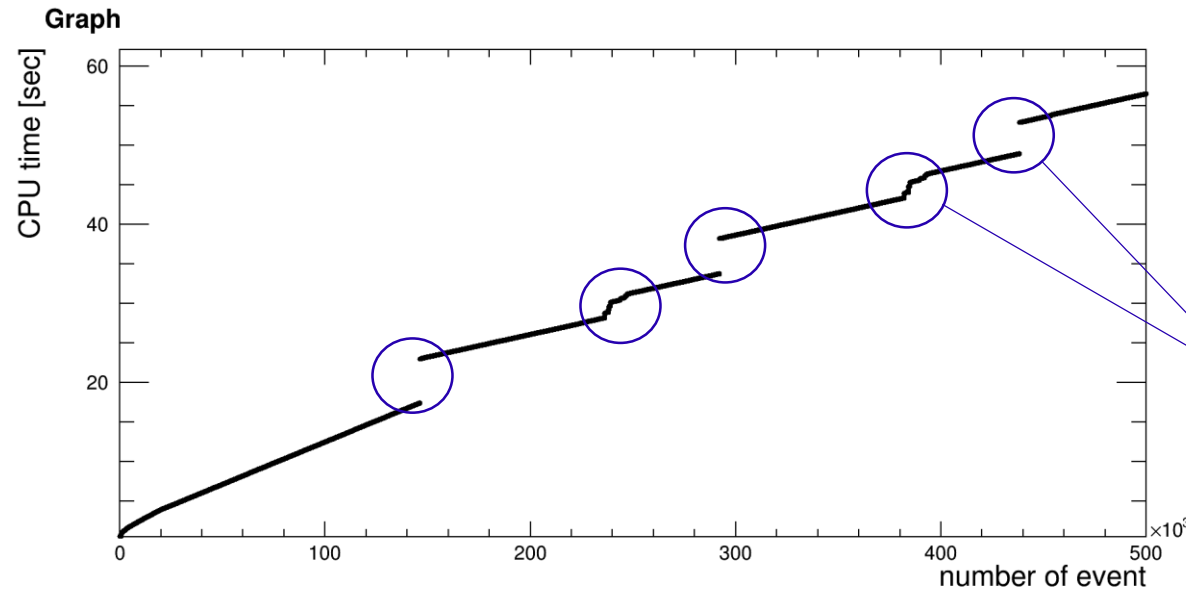


本研究の目的：
高速なFTKのシミュレーションを構築すること

以降、従来のシミュレーションをfull sim, 高速なシミュレーションをfast simと呼称する。

Process Time

- ✓ 10muon 500k eventsを使用 (Input : offline track 2,499,308 tracks)
- ✓ コード開始から測定、1Eventの処理が終わるタイミングで計測、プロット
- ✓ 同コードを10回回した平均時間を測定



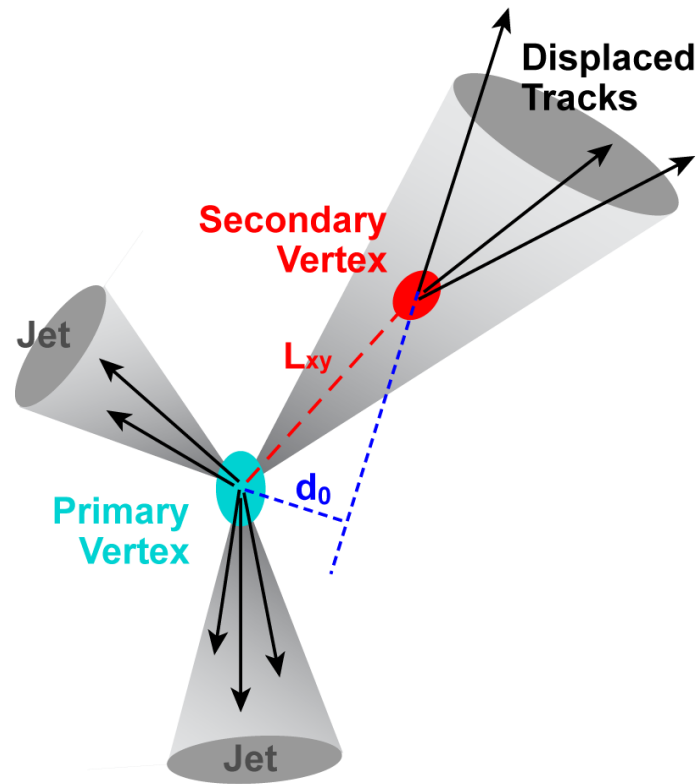
横軸 : Event数
縦軸 : CPU time[sec]

- ✓ 各所の非線形、時間のboundaryは現調査中。

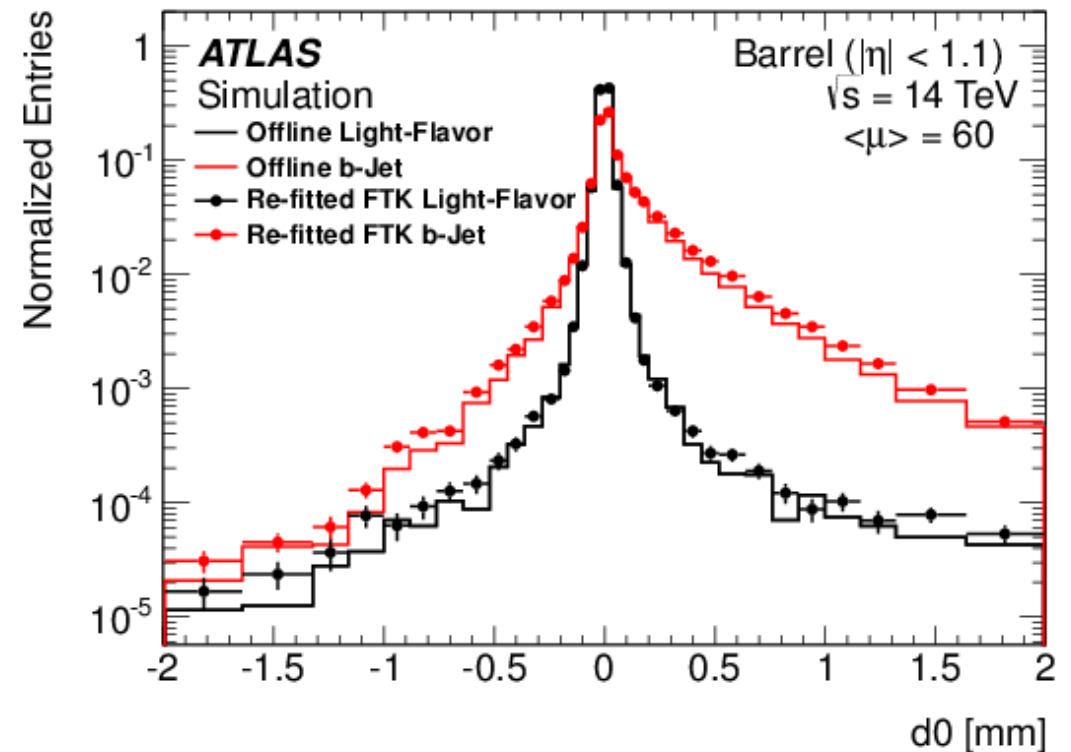
CPU time : 平均 73 [sec]
⇒ 29 μ s/track

先行研究 (仲松さん truth-seeded)
single muon : 27 μ s/event

FTKによる恩恵 : b-jet tagging



Bハドロン：寿命長い
 ⇒ 二次崩壊点
 インパクトパラメータ(d_0)大



FTKによる全領域での衝突点・飛跡再構成
 ⇒ オフラインと同等のクオリティで
 d_0 を再構成

5つのパラメータについて

FTK の扱う飛跡パラメータは以下の5つ (helix parameter という)

P_t ... xy平面での運動量 (横運動量)

η ... 天頂角を表す擬ラピディティ

ϕ ... x軸方向からの方位角

d_0 ... xy平面上における、粒子の飛跡と衝突点の間の最近接距離

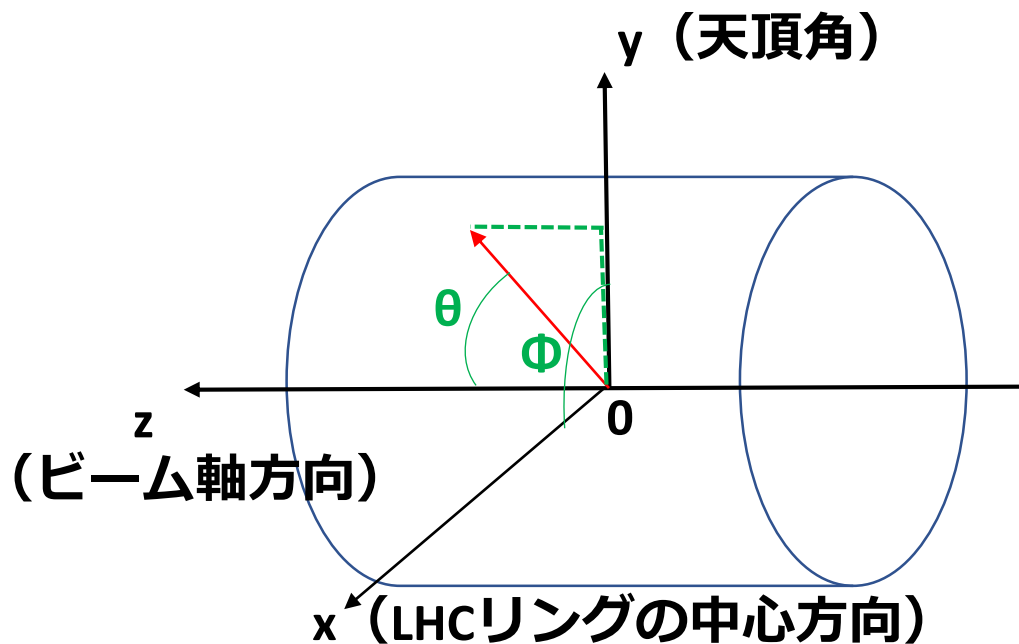
z_0 ... 最近接点のz座標

$$P_t = \sqrt{P_x^2 + P_y^2}$$

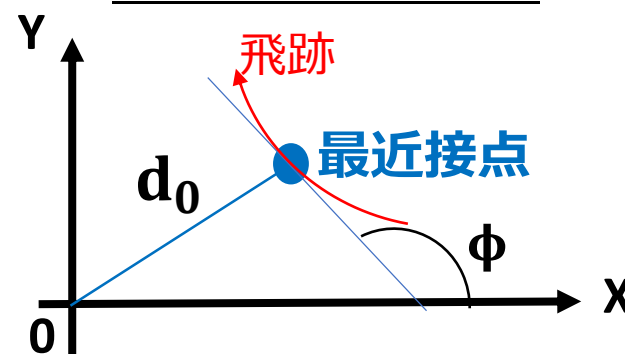
P_x : 荷電粒子のx方向の運動量

$$\eta = -\ln \left(\tan \left(\frac{\theta}{2} \right) \right)$$

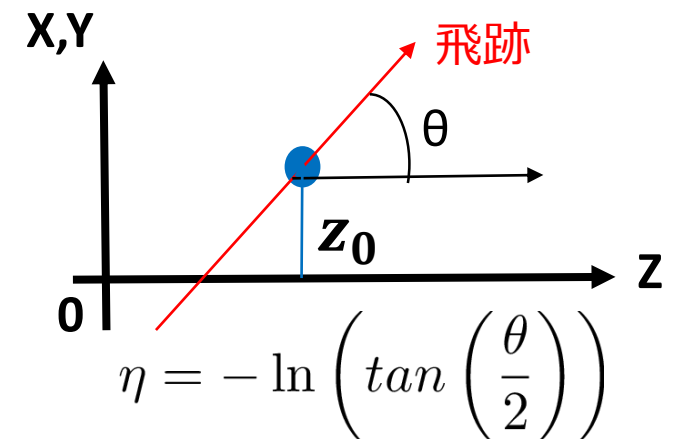
ATLAS座標系



ビーム軸垂直平面

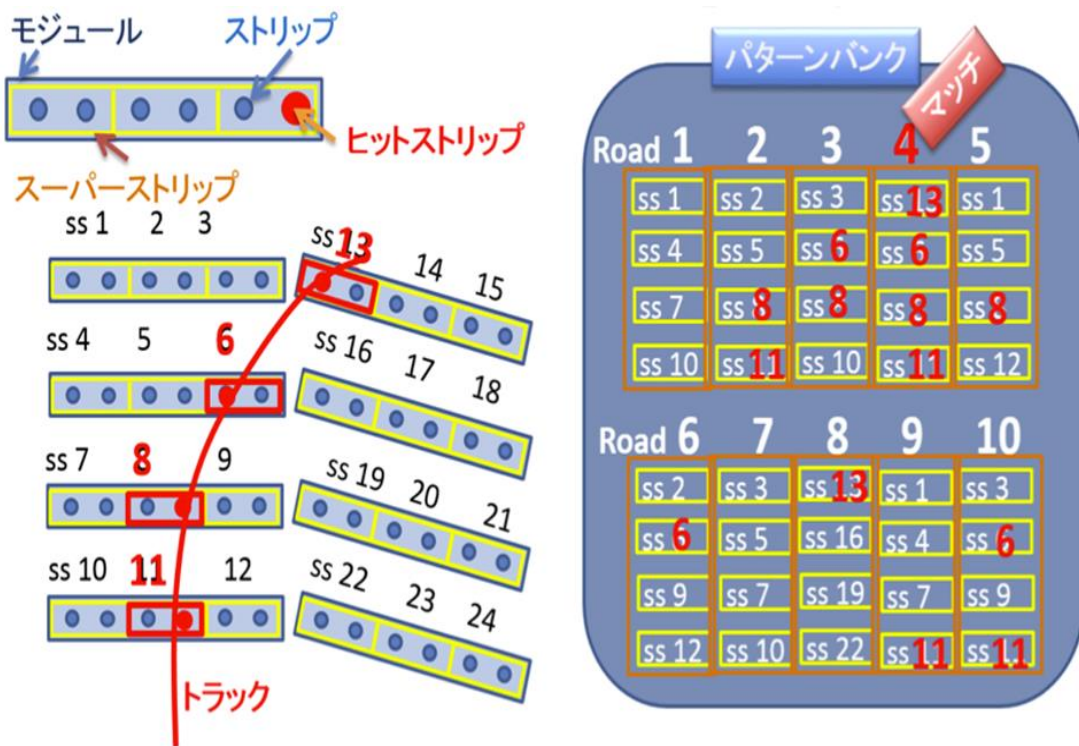


ビーム軸を含む平面



内部飛跡検出器とFTKの飛跡再構成の原理

①パターンマッチ



ヒットの情報を荒い位置情報に変換
あらかじめ用意したパターンと比較し一致
したものを飛跡の情報として取得

②線形近似

1st stage fit ... 質の悪い飛跡を排除

$$\chi^2 = \sum_{i=1}^6 \left(\sum_{j=1}^{11} S_{ij} x_j + h_i \right)^2$$

S_{ij}, h_i : 定数項
 x_j : ヒットの座標

2nd stage fit ...

1st stage fitで排除されなかった飛跡の内
さらに質の良い飛跡を選別

$$\chi^2 = \sum_{i=1}^{11} \left(\sum_{j=1}^{16} S_{ij} x_j + h_i \right)^2$$

S_{ij}, h_i : 定数項
 x_j : ヒットの座標

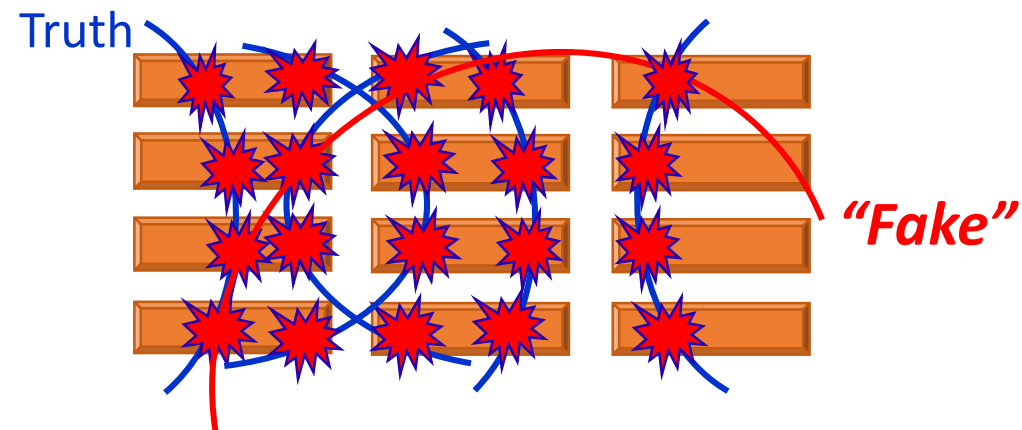
各パラメータの算出

$$\tilde{p}_i = \sum_{l=1}^N C_{il} x_l + q_i$$

\tilde{p}_i : パラメータ
 C_{il}, q_i : 定数項
($i=1,2,\dots,5$)
 x_l : ヒットの座標

FTK Fake track (“Fake”)

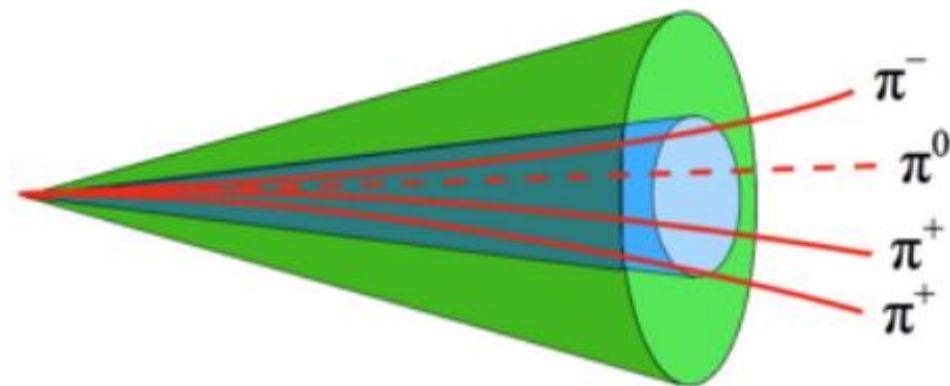
- ✓ 本来飛跡がないところに誤って飛跡を再構成
 - ✓ Hit情報から飛跡を再構成することにより生成
 - Truthとは無関係
 - Offline trackにも共通の“Fake”が再構成される
- ⇒ Offline trackからであれば再現できる



○ “Fake”による問題

Ex.) ハドロン崩壊する τ の誤同定

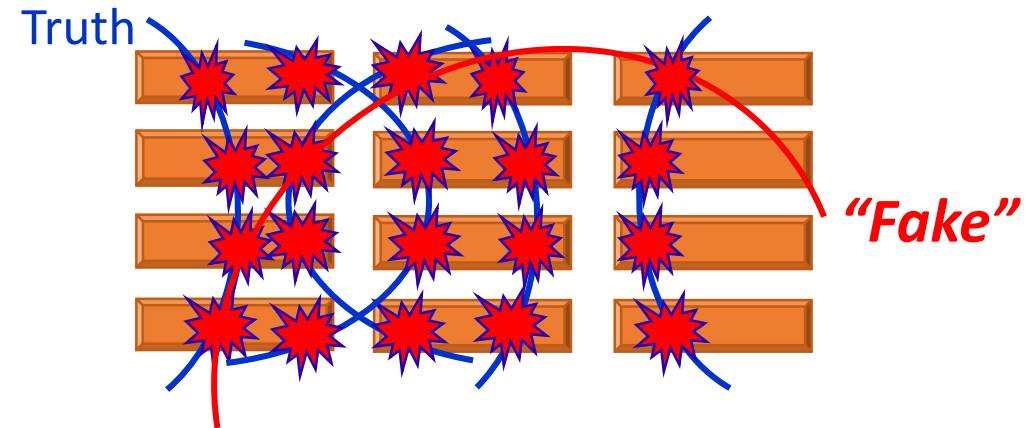
ハドロン崩壊する τ



✓ π^\pm が 1 or 3 本で同定

FTK Fake track (“Fake”)

- ✓ 本来飛跡がないところに誤って飛跡を再構成
 - ✓ Hit情報から飛跡を再構成することにより生成
 - Truthとは無関係
 - Offline trackにも共通の“Fake”が再構成される
- ⇒ Offline trackからであれば再現できる



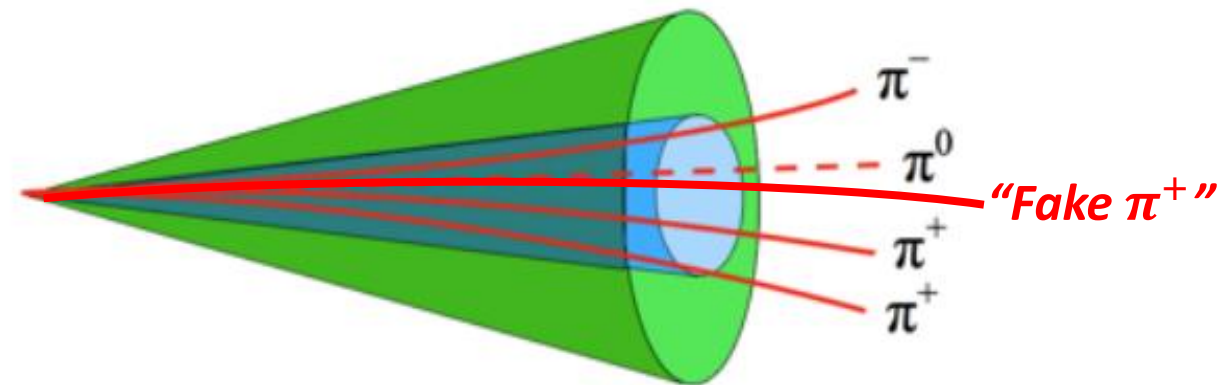
○ “Fake”による問題

Ex.) ハドロン崩壊する τ の誤同定

- 本来3本なのに4本と勘違い
⇒ τ として同定されない
- 本来2本なのに3本と勘違い
⇒ τ として同定

- ✓ FastSimで再現しなければ
MCとデータにずれ

ハドロン崩壊する τ



- ✓ π^\pm が1 or 3本で同定

再構成率の再現

✓ Full simの再構成率分布を用意

○特徴

✓ 正負で対称

✓ $q/Pt, \eta, d0, z0$ に依存

⇒ $q/Pt, \eta, d0, z0$ の正領域を分割

領域分けの定義

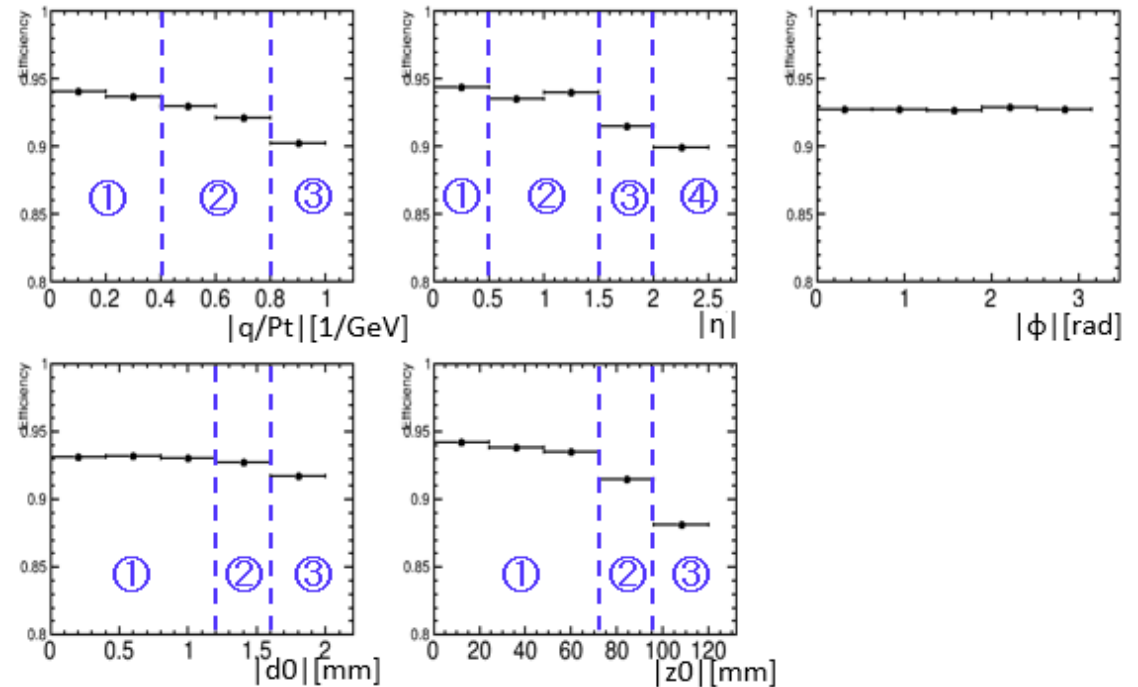
region	①	②	③	④
$ q/Pt $	0 ~ 0.4	0.4 ~ 0.8	0.8 ~ 1	
$ \eta $	0 ~ 0.5	0.5 ~ 1.5	1.5 ~ 2	2 ~ 2.5
$ d0 $	0 ~ 1.2	1.2 ~ 1.6	1.6 ~ 2	
$ z0 $	0 ~ 72	72 ~ 96	96 ~ 120	

✓ $|q/Pt|, |\eta|, |d0|, |z0|$ で108領域に分割

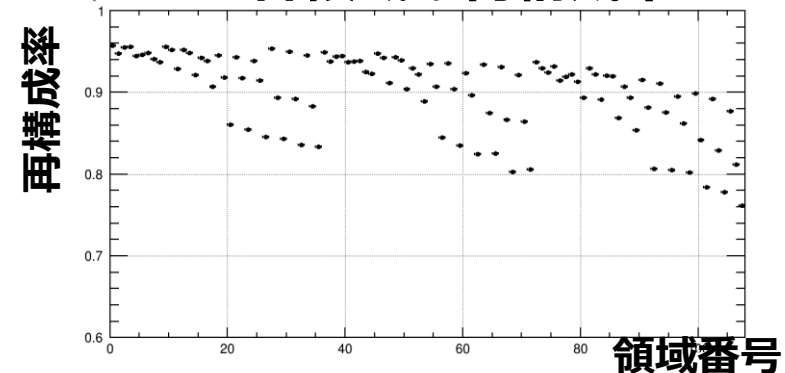
⇒ 各領域の再構成率を調べる

$$\text{再構成率} = \frac{\text{matched offline track の本数}}{\text{offline track の本数}}$$

再構成率とParameterの相関



108各領域の再構成率



再構成率の再現

Input

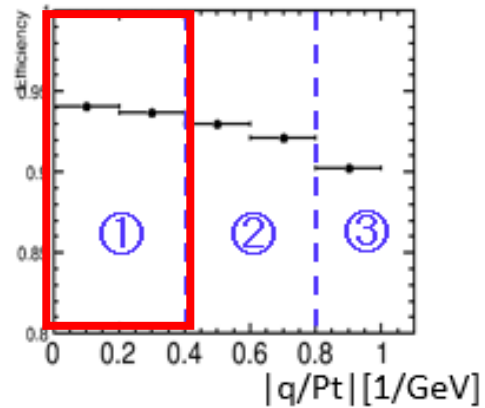
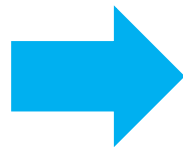
Offline track parameter
 $q/P_t(P_t), \eta, \phi, d_0, z_0$

再構成率の再現

✓ $|q/Pt|$ がどの領域か調べる

Input

Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$



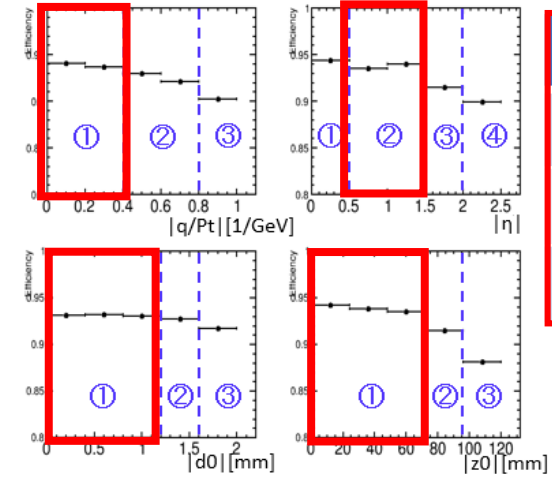
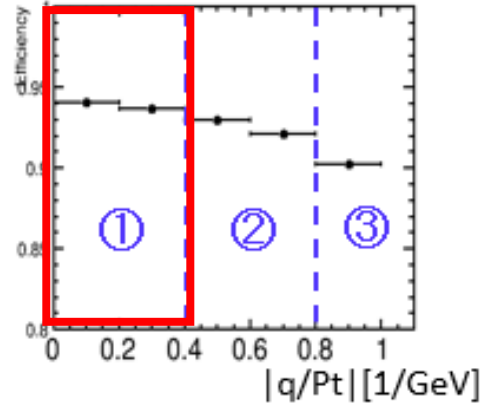
再構成率の再現

✓ $|q/Pt|$ がどの領域か調べる

✓ 同様に $|\eta|, |d0|, |z0|$ が
どの領域か調べる

Input

Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$



$ q/Pt $	①
$ \eta $	②
$ d0 $	①
$ z0 $	①

/ 108

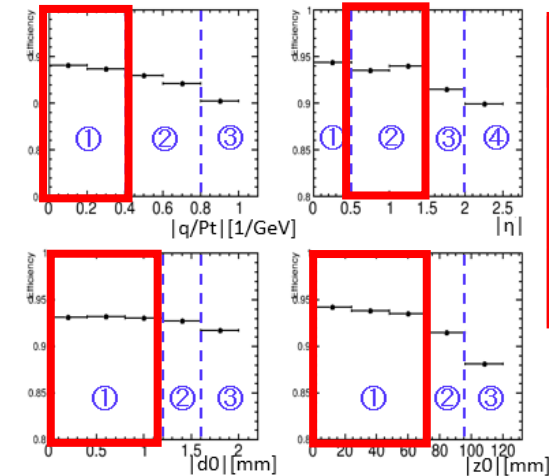
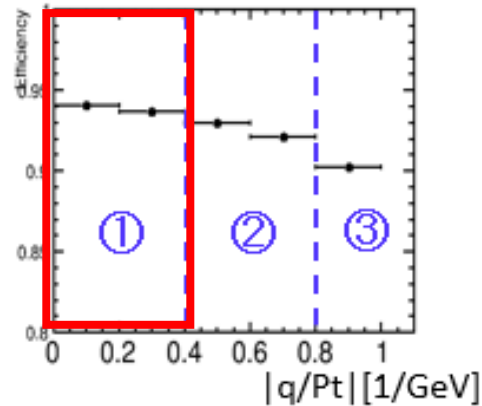
再構成率の再現

✓ $|q/Pt|$ がどの領域か調べる

✓ 同様に $|\eta|, |d0|, |z0|$ が
どの領域か調べる

Input

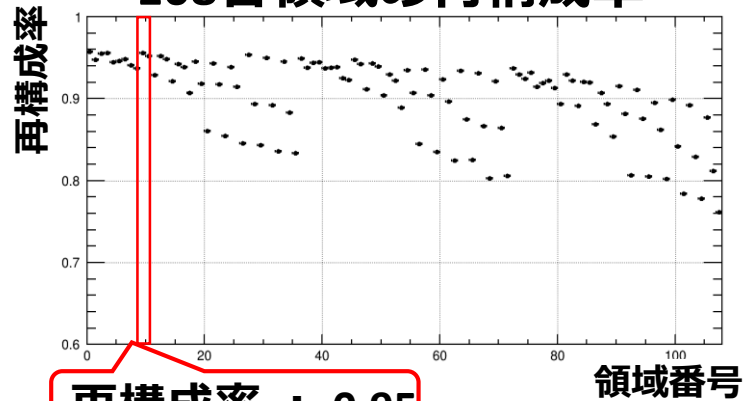
Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$



$ q/Pt $	①
$ \eta $	②
$ d0 $	①
$ z0 $	①

/ 108

✓ Offline trackが属する領域
の再構成率を調べる
108各領域の再構成率



再構成率 : 0.95

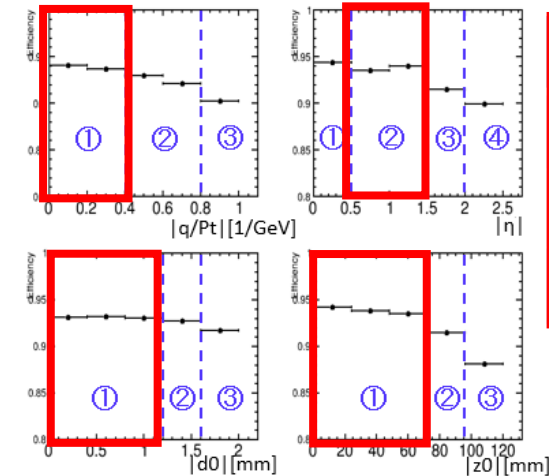
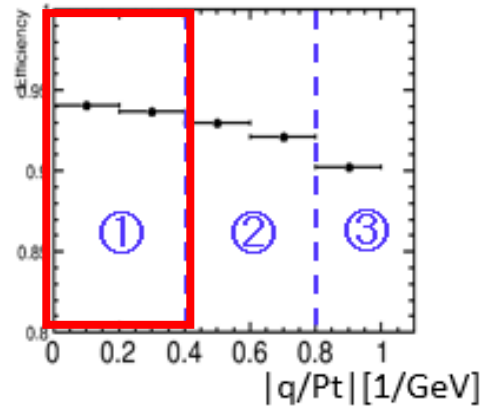
再構成率の再現

✓ $|q/Pt|$ がどの領域か調べる

✓ 同様に $|\eta|, |d0|, |z0|$ が
どの領域か調べる

Input

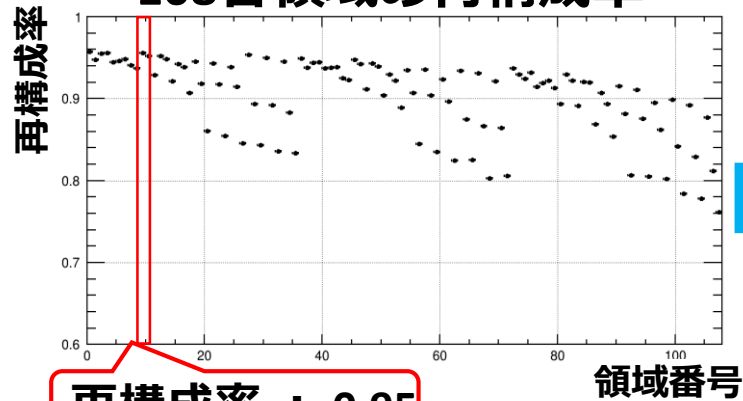
Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$



$ q/Pt $	①
$ \eta $	②
$ d0 $	①
$ z0 $	①

/ 108

✓ Offline trackが属する領域
の再構成率を調べる
108各領域の再構成率



✓ 0 ~ 1 の乱数を1つ生成

乱数 : 0.91

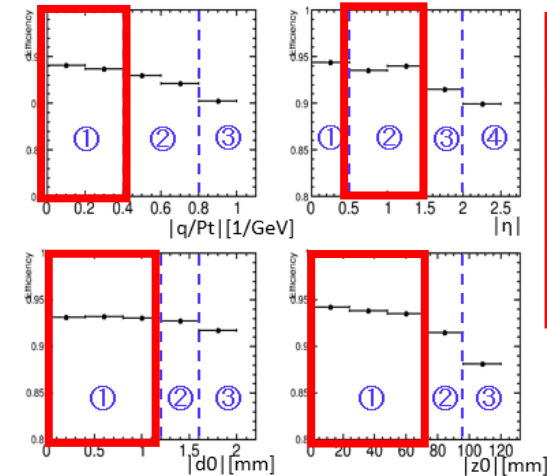
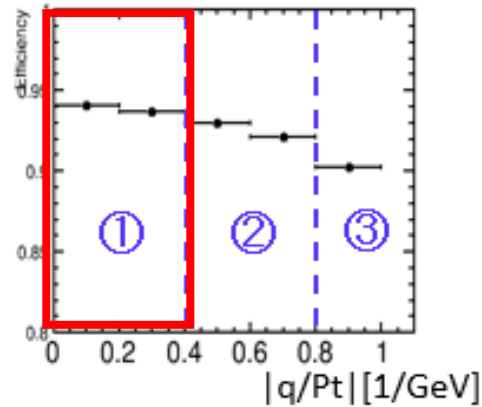
再構成率の再現

✓ $|q/Pt|$ がどの領域か調べる

✓ 同様に $|\eta|, |d0|, |z0|$ が
どの領域か調べる

Input

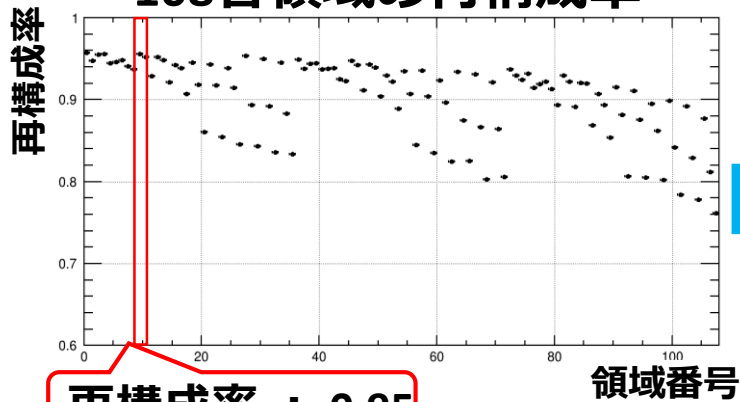
Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$



$ q/Pt $	①
$ \eta $	②
$ d0 $	①
$ z0 $	①

/ 108

✓ Offline trackが属する領域
の再構成率を調べる
108各領域の再構成率



再構成率 : 0.95

✓ 0 ~ 1 の乱数を 1 つ生成

乱数 : 0.91

✓ 再構成率と乱数の大小比較

• **再構成率** \geq **乱数**
 \Rightarrow Offline trackを保存

• **再構成率** $<$ **乱数**
 \Rightarrow 保存しない

Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$

分解能の再現

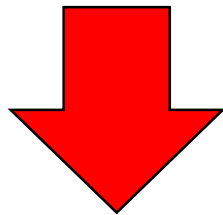
分解能 = matched Offline track parameter
- matched FTK track parameter

- ✓各パラメータ,108各領域の分解能分布を用意
- ✓それぞれをDouble gaussianでfit

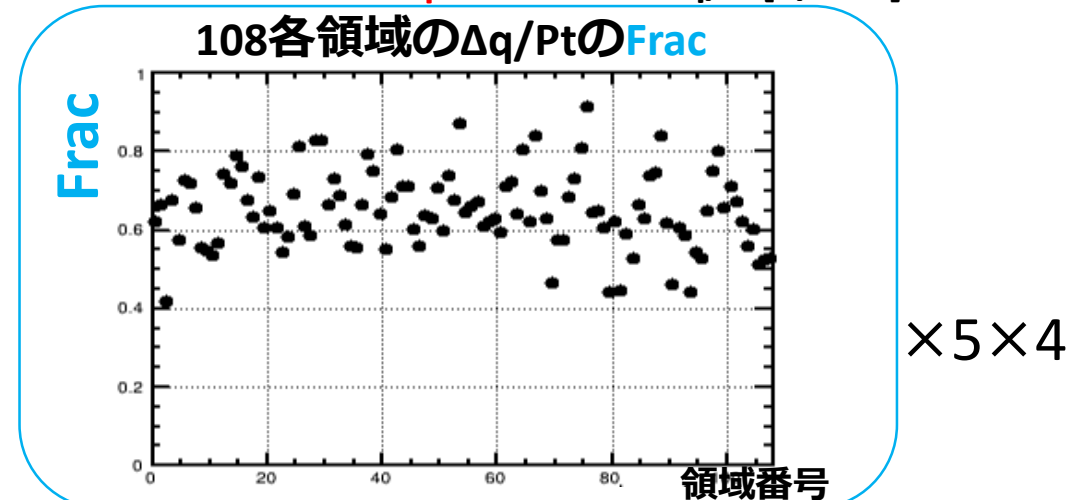
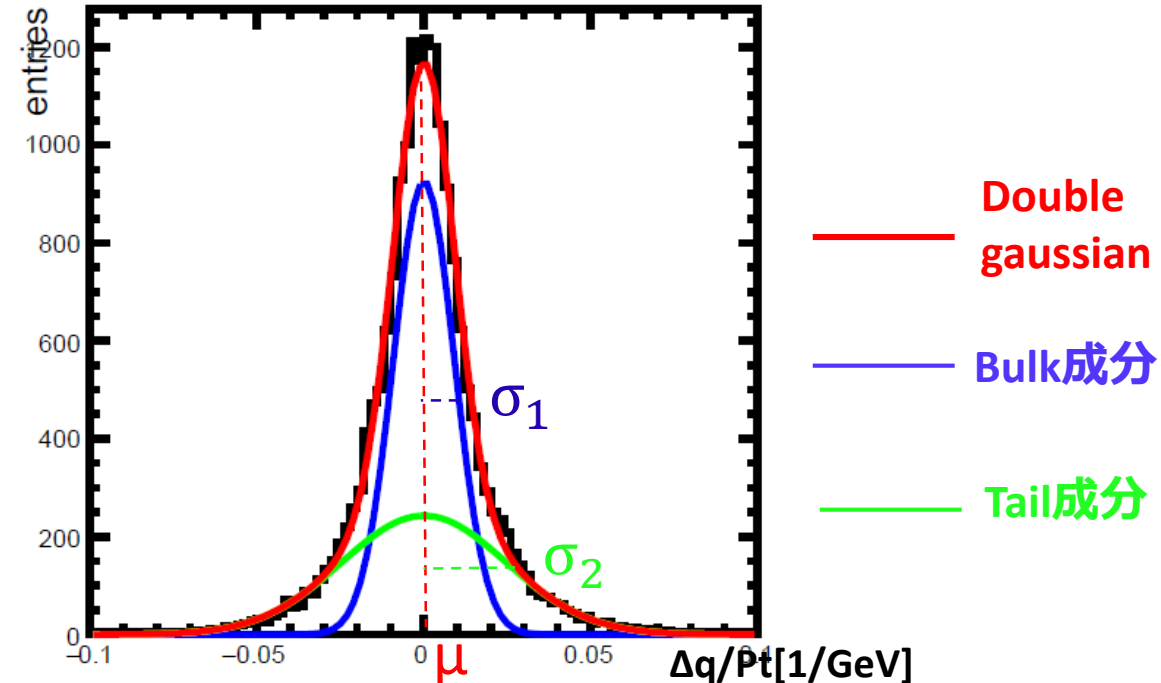
○Double gaussian

$$f(x) = A \left(\frac{\text{Frac}}{\sigma_1 \sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma_1^2}\right) + \frac{(1-\text{Frac})}{\sigma_2 \sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma_2^2}\right) \right)$$

$\text{Frac} = \frac{\text{Bulk成分の面積}}{\text{Double gaussの面積}}$



- ✓各パラメータ108各領域の Frac , μ , σ_1 , σ_2 を保存
(5 Parameter \times 108領域 \times 4 FitParameter = 2160 Parameters を保存)



分解能の再現

再構成率の再現で保存した
Offline trackを使用

Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$

$ q/Pt $	①
$ \eta $	②
$ d0 $	①
$ z0 $	①

/ 108

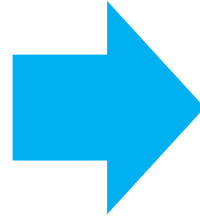
分解能の再現

再構成率の再現で保存した
Offline trackを使用

Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$

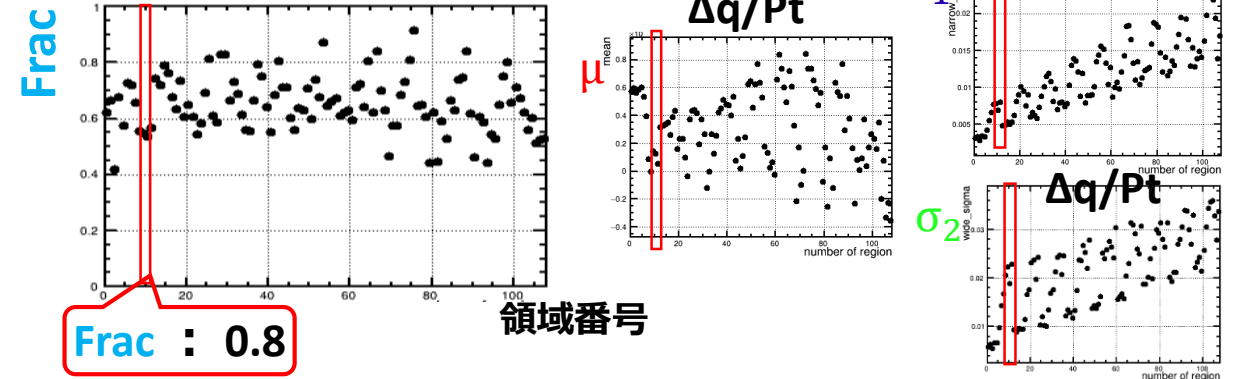
$ q/Pt $	①
$ \eta $	②
$ d0 $	①
$ z0 $	①

/ 108



✓ Offline trackが属する領域の
各パラメータのFrac, μ , σ_1 , σ_2 を調べる

108各領域の $\Delta q/Pt$ のFrac



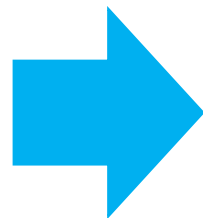
分解能の再現

再構成率の再現で保存した
Offline trackを使用

Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$

$ q/Pt $	①
$ \eta $	②
$ d0 $	①
$ z0 $	①

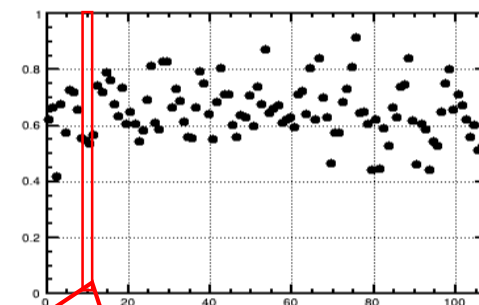
/ 108



✓ Offline trackが属する領域の
各パラメータのFrac, μ, σ_1, σ_2 を調べる

108各領域の $\Delta q/Pt$ のFrac

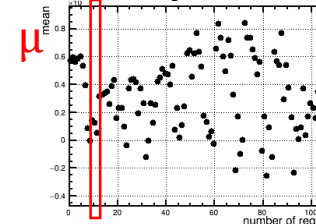
Frac



Frac : 0.8

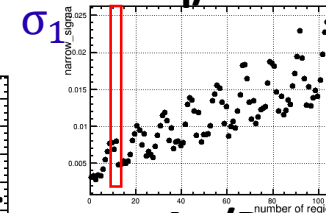
領域番号

$\Delta q/Pt$

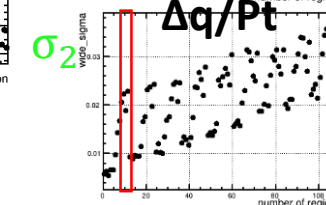


μ

$\Delta q/Pt$



σ_1



σ_2



✓ 0 ~ 1 の乱数を 1 つ生成

乱数 : 0.68

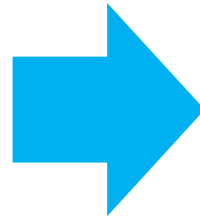
分解能の再現

再構成率の再現で保存した
Offline trackを使用

Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$

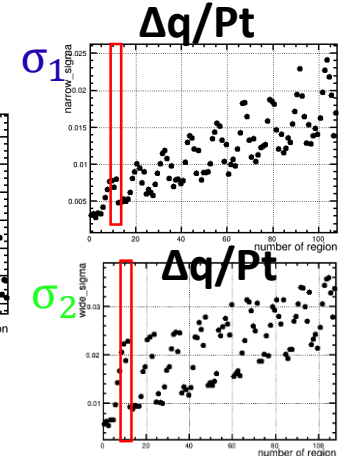
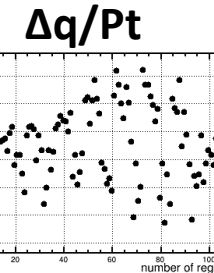
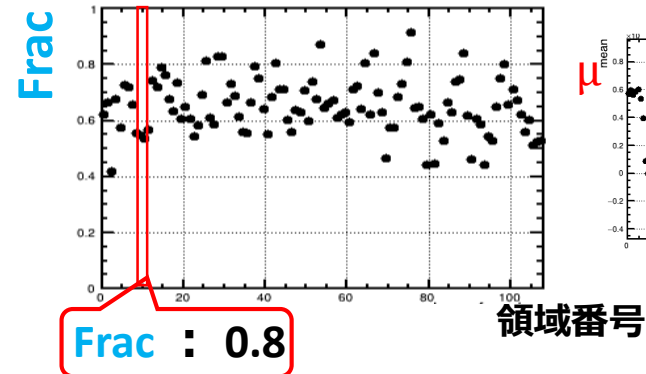
$ q/Pt $	①
$ \eta $	②
$ d0 $	①
$ z0 $	①

/ 108



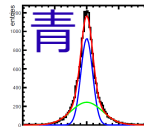
✓ Offline trackが属する領域の
各パラメータのFrac, μ, σ_1, σ_2 を調べる

108各領域の $\Delta q/Pt$ のFrac

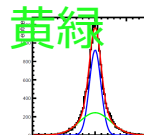


✓ Fracと乱数の大小比較

- $Frac \geq$ 乱数
⇒ gRandom -> Gaus(μ, σ_1)



- $Frac <$ 乱数
⇒ gRandom -> Gaus(μ, σ_2)



✓ 0 ~ 1の乱数を1つ生成

乱数 : 0.68



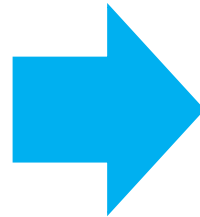
分解能の再現

再構成率の再現で保存した
Offline trackを使用

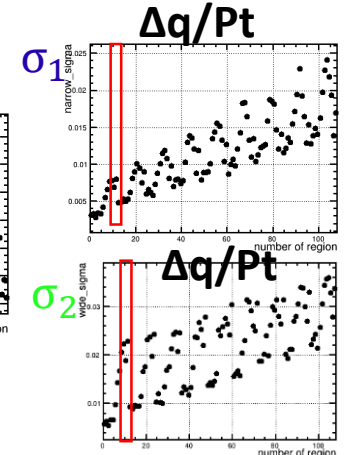
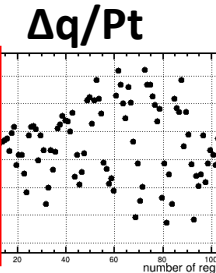
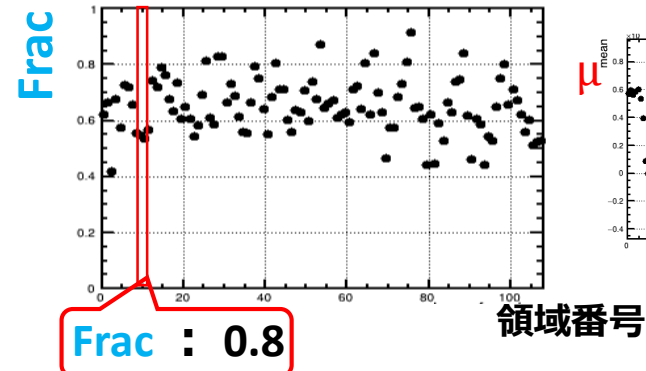
Offline track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$

$ q/Pt $	①
$ \eta $	②
$ d0 $	①
$ z0 $	①

/ 108

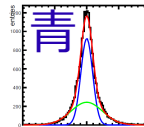


108各領域の $\Delta q/Pt$ のFrac

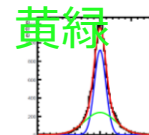


✓ Fracと乱数の大小比較

- $Frac \geq$ 乱数
⇒ gRandom -> Gaus(μ, σ_1)



- $Frac <$ 乱数
⇒ gRandom -> Gaus(μ, σ_2)



Output

FastSim FTK track parameter
 $q/Pt(Pt), \eta, \phi, d0, z0$

FastSim FTK track parameter
= Offline track parameter
- Gaus

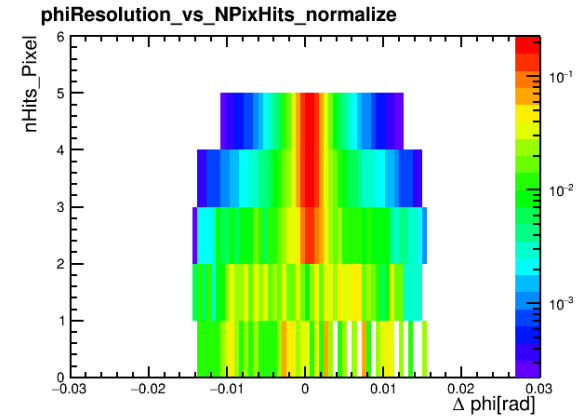
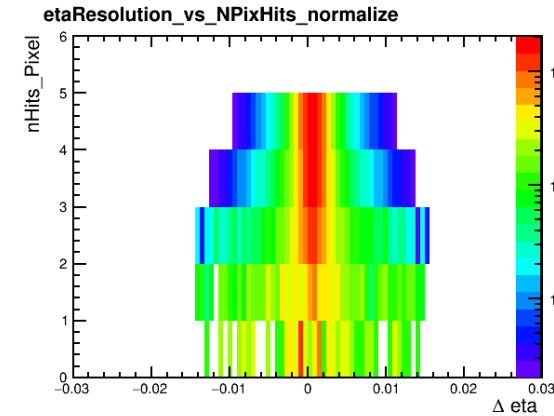
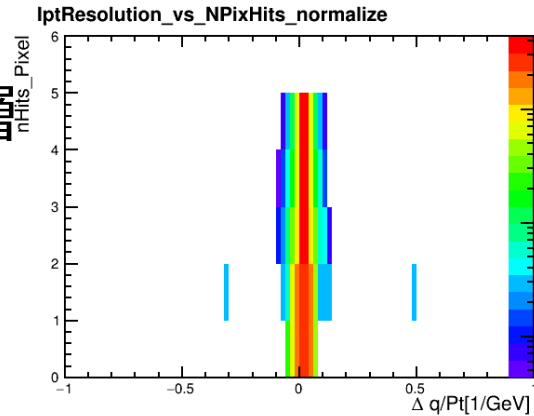
✓ 0 ~ 1 の乱数を1つ生成

乱数 : 0.68

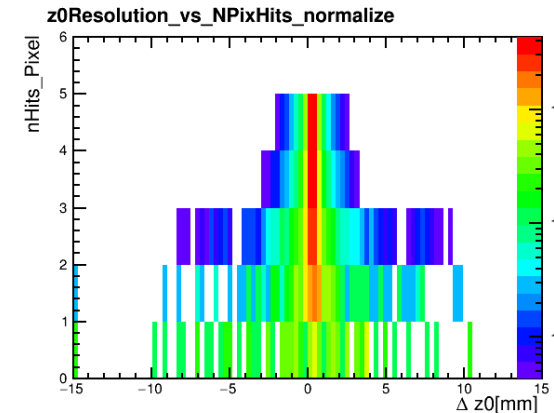
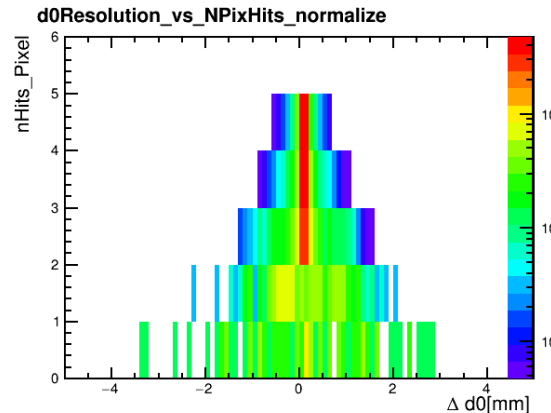


Resolution vs nPixelHits_offlinetrack(muon)

✓ offline trackがFTKで定義
されているPixlayerに何層
hitしたかを見ている
(IBL含む)



✓ tailの広がりからhit数が
少ないほどresolutionは
悪い



x軸 : resolution
y軸 : Pixel (IBL含む) のhit
z軸 : Entry数 (log)
y軸に関して (hit数毎に)
normalize

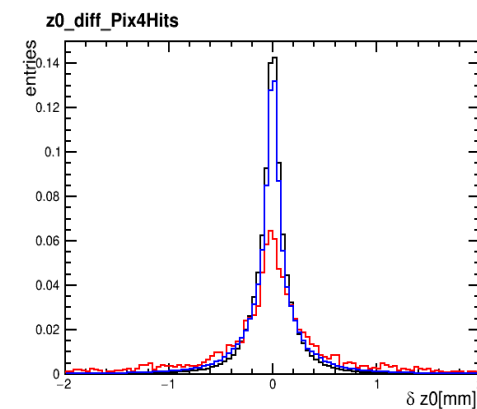
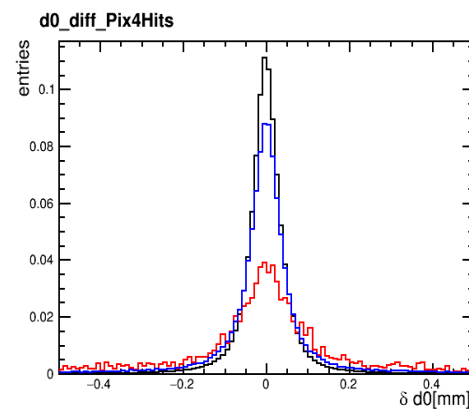
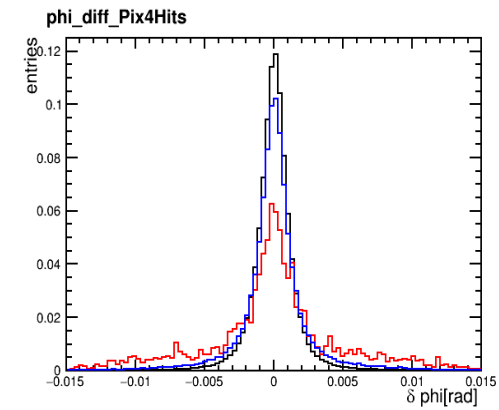
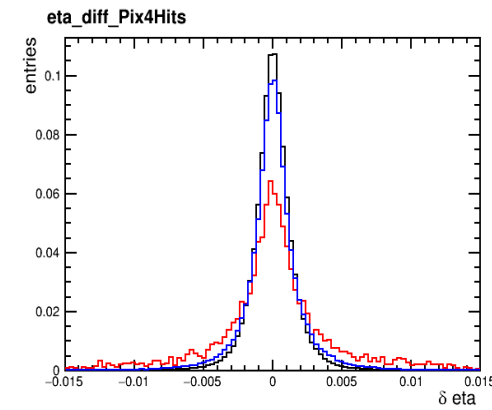
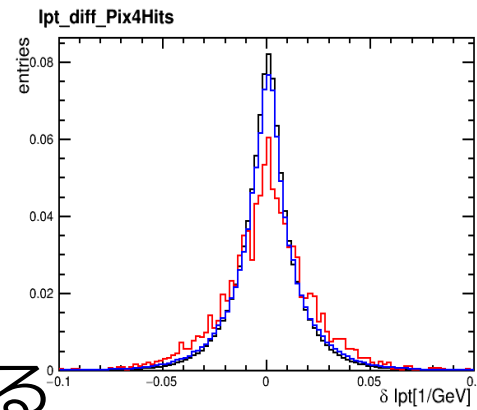
Resolution vs nPixHits on offlinetrack

hit数毎にresolutionを比較 (108領域をmerge)

✓ Pixに対するhit数毎にresolutionを比較(面積 1 にnormalize)

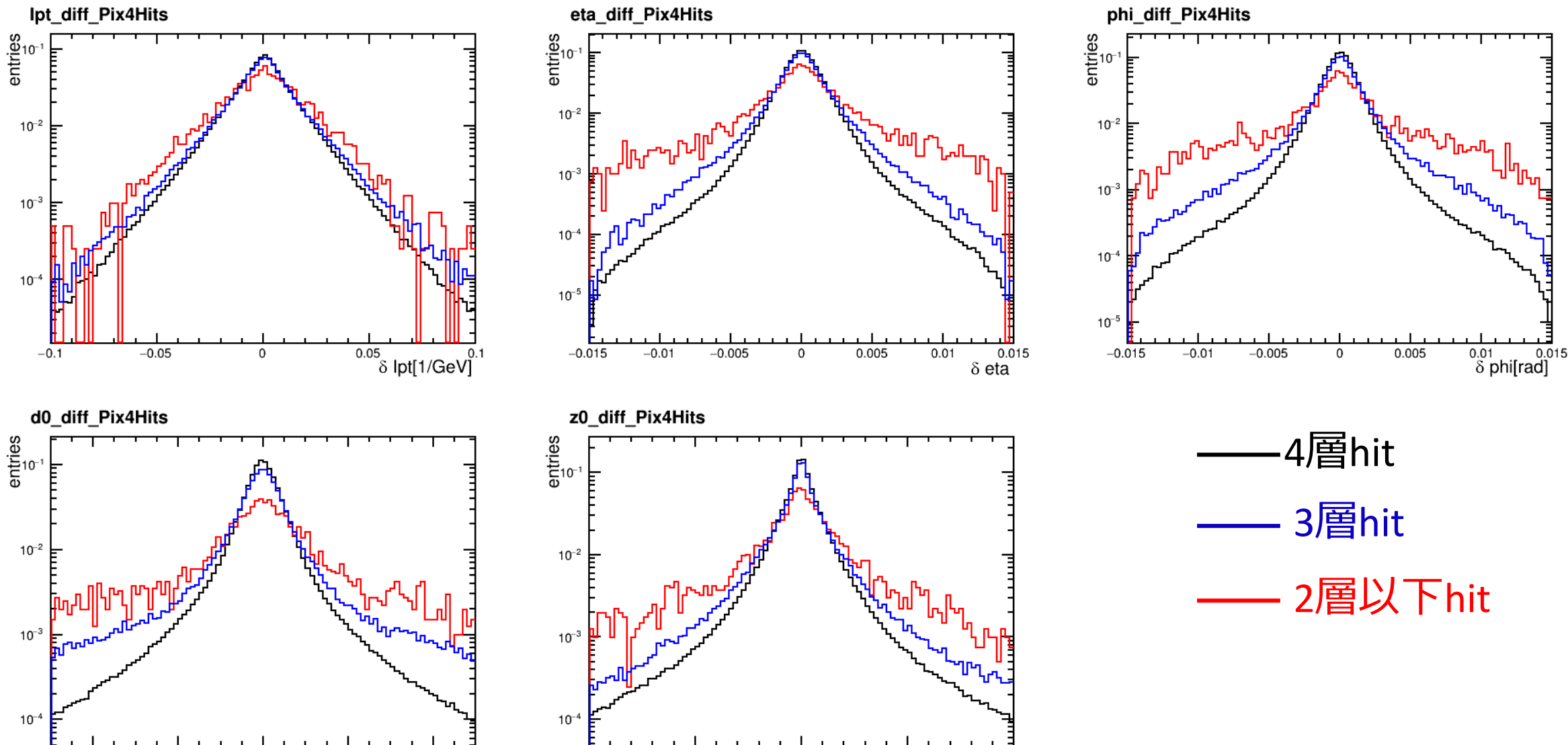
✓ hit数が多い程resolutionは良い
✓ tailは2層hitが大きく影響している

⇒ regionごとに同様の比較

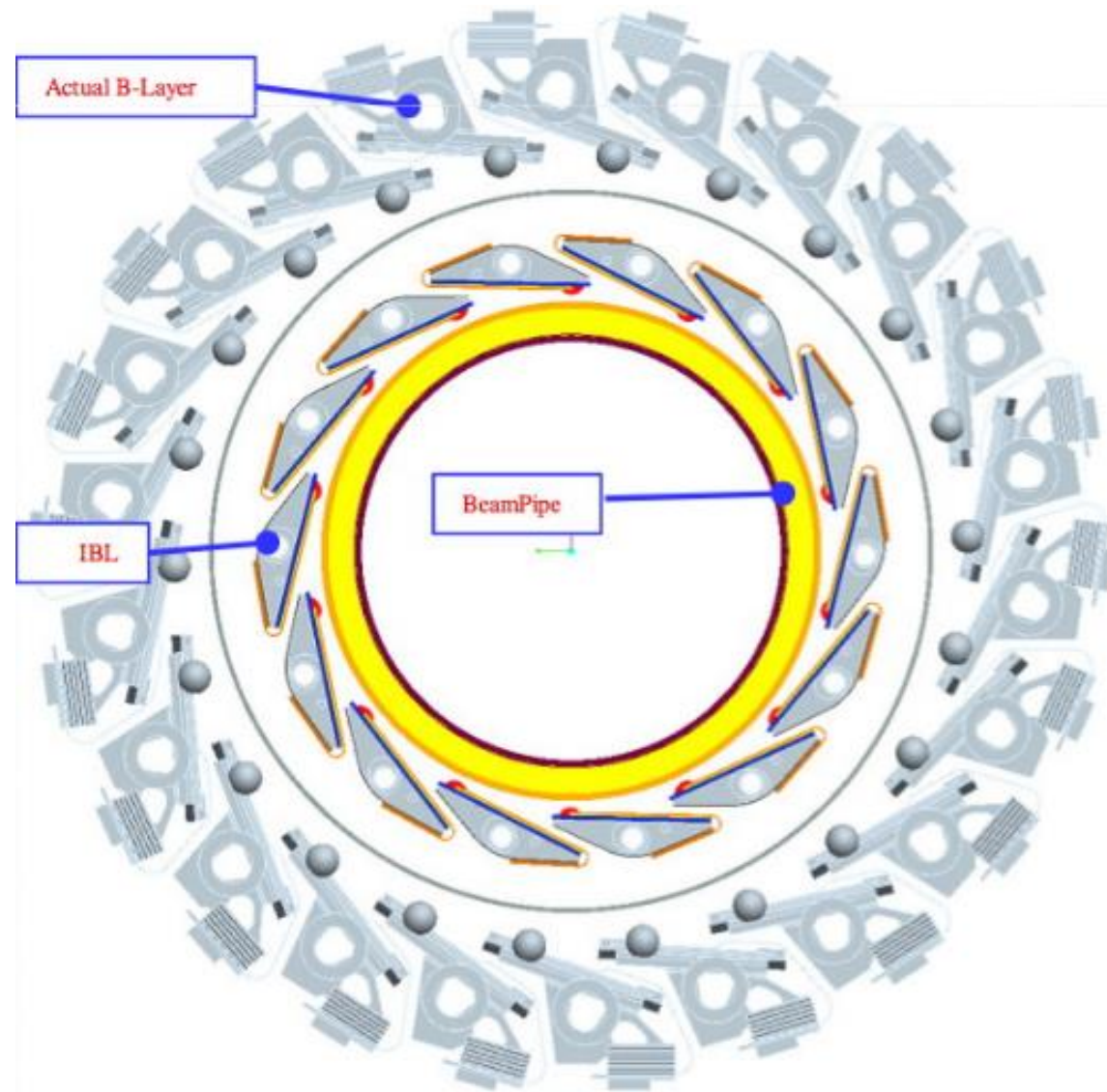


— 4層hit
— 3層hit
— 2層以下hit

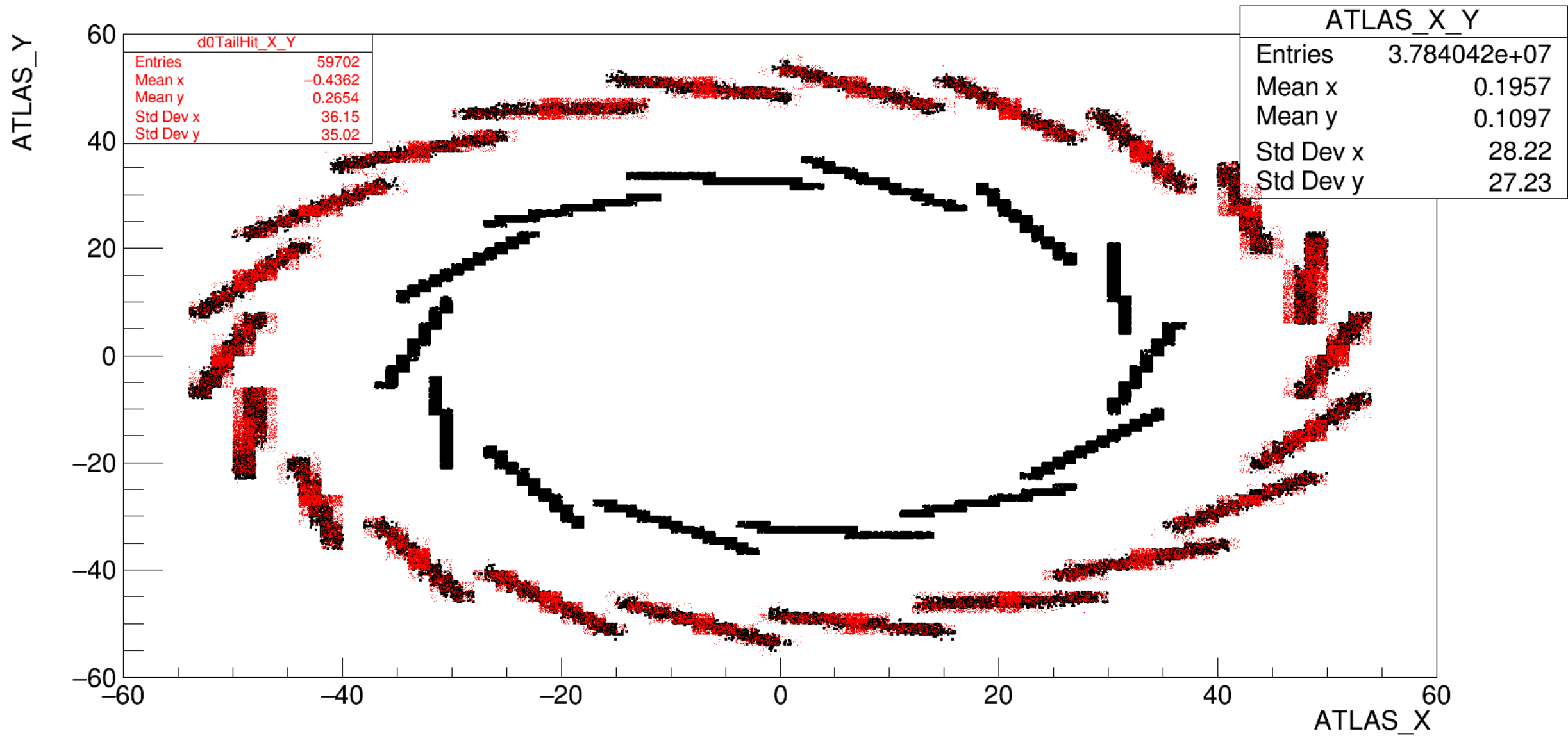
Resolution vs nPixHits on offlinetrack



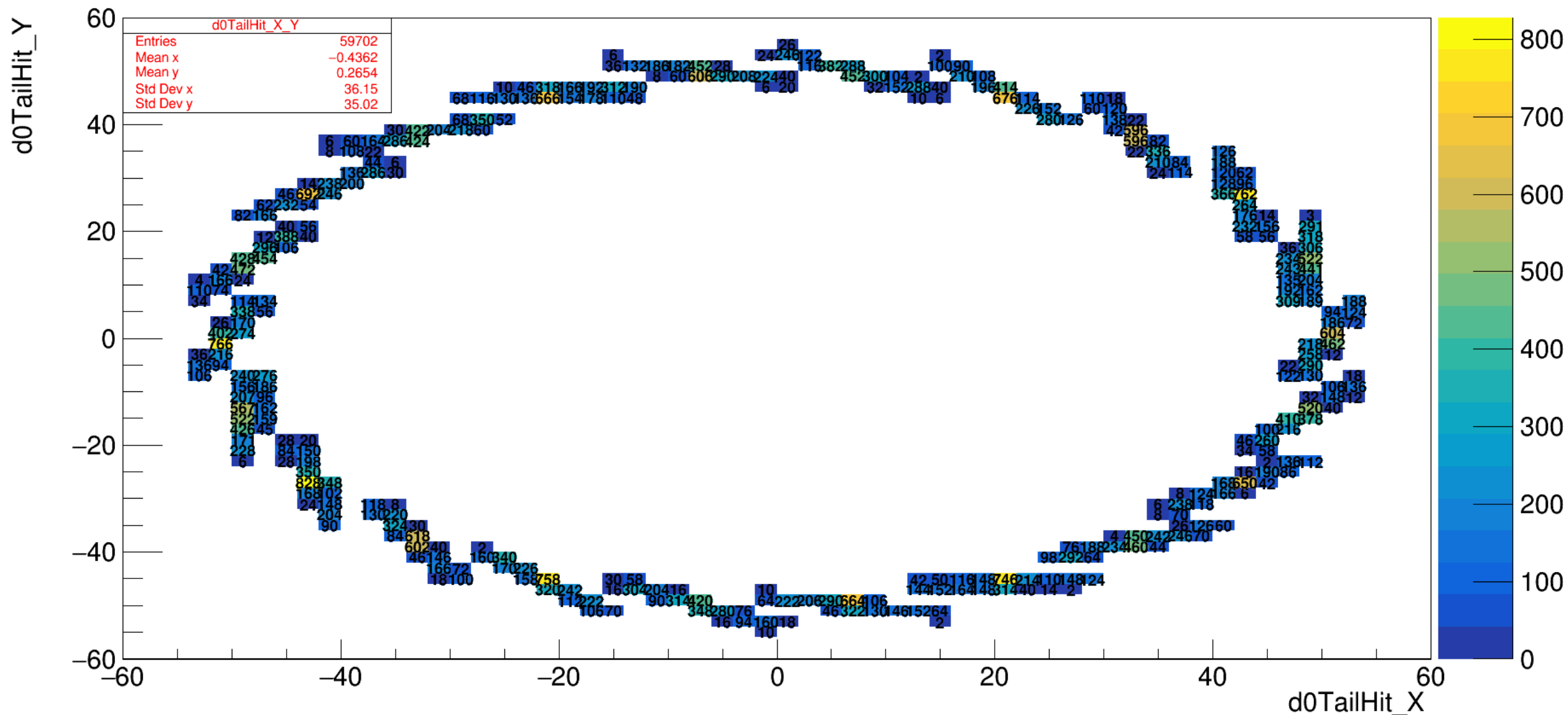
Cross section of B-layer



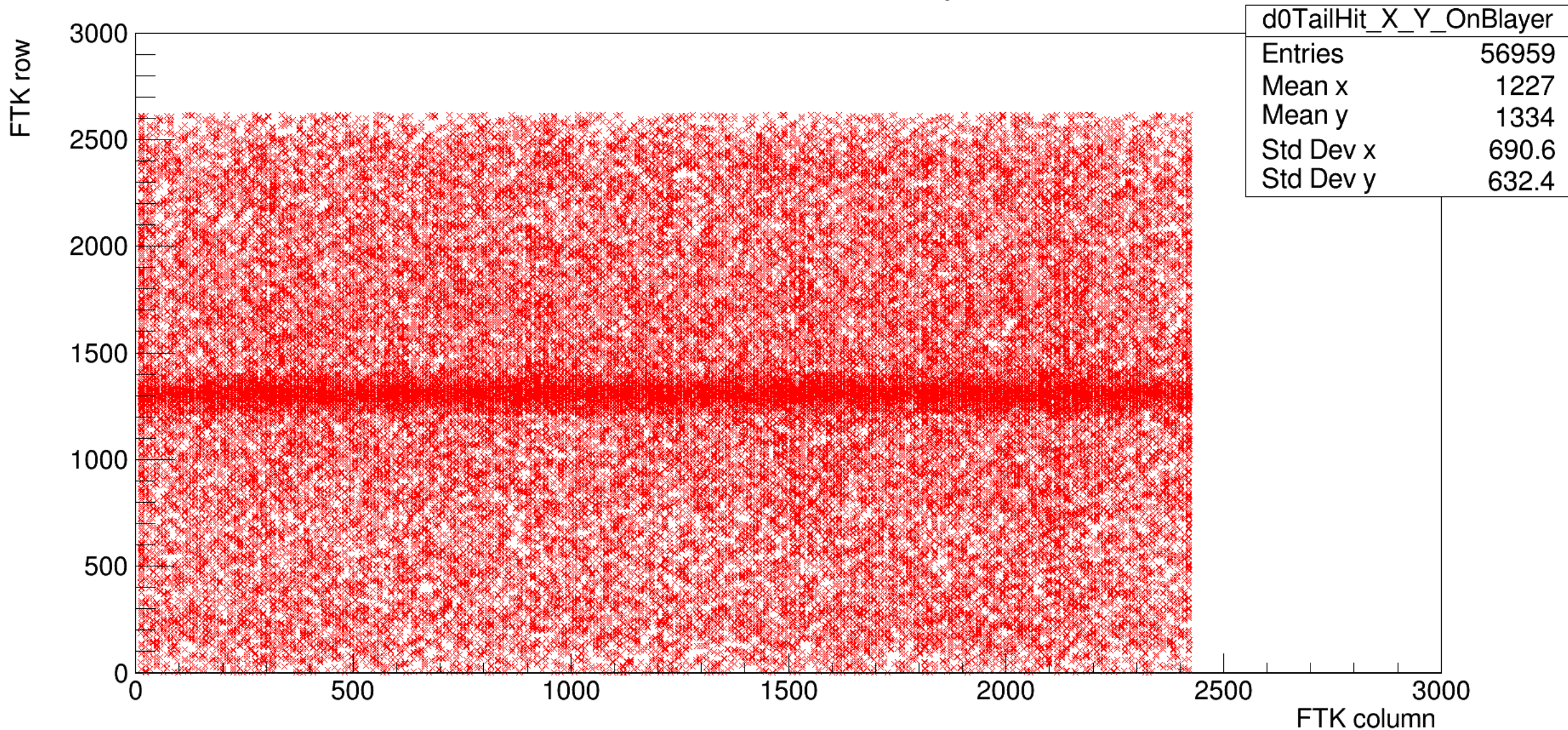
ATLAS_X_Y



d0TailHit_X_Y

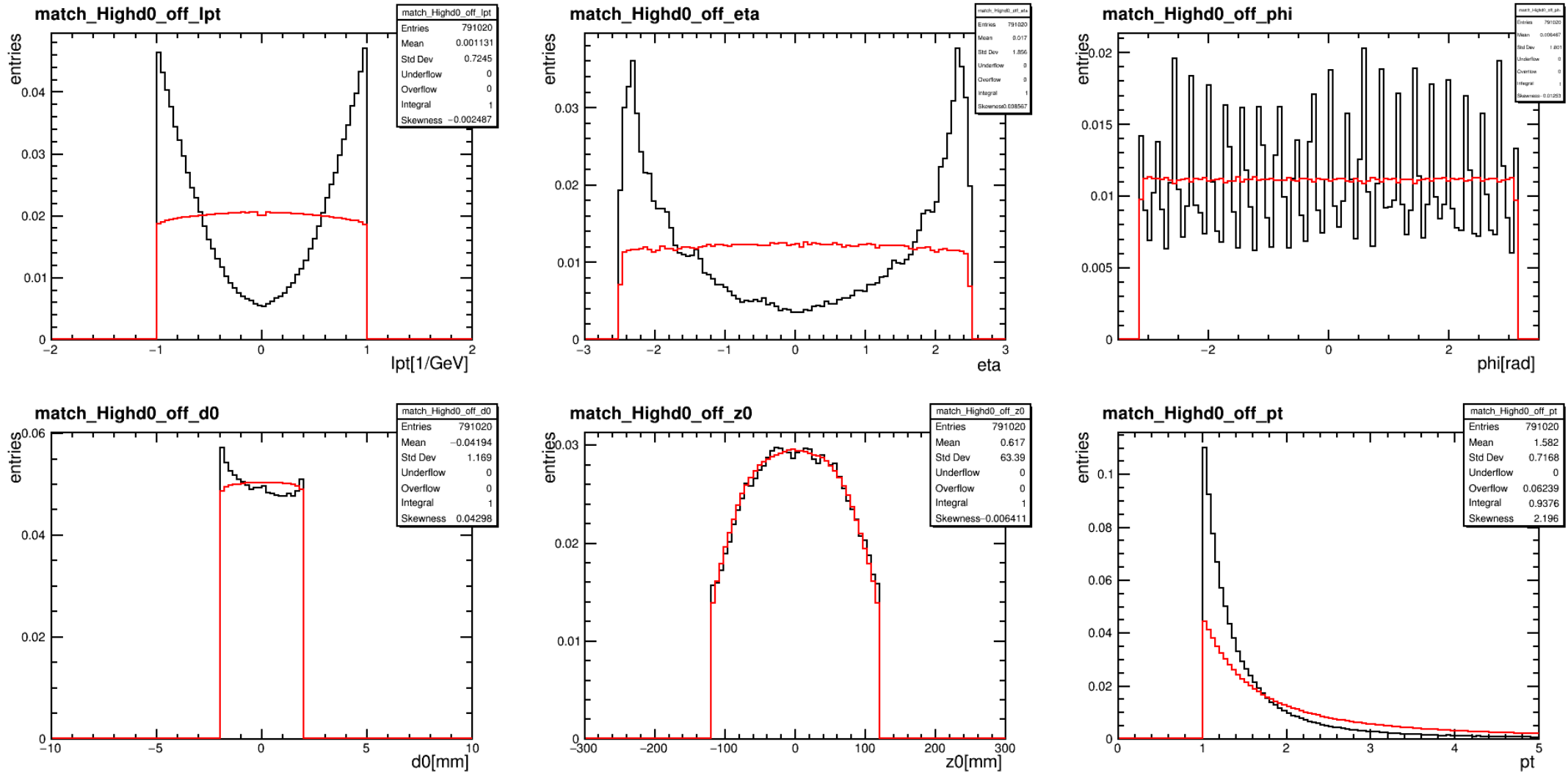


d0TailHit_X_Y_OnBlayer



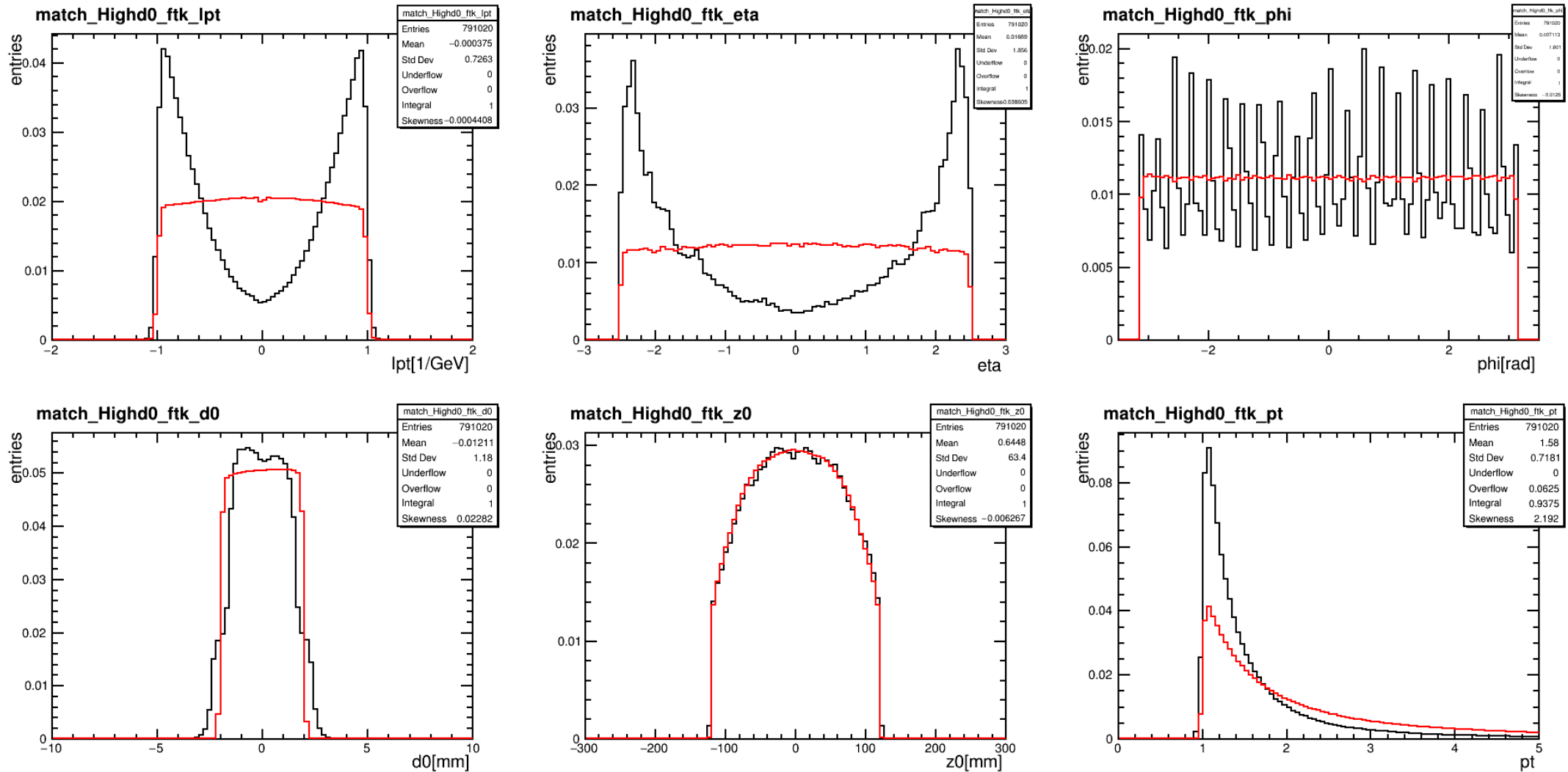
Compare tail and bulk

Offline tracks parameters



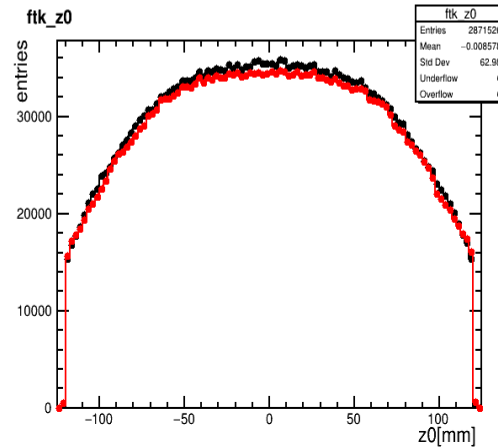
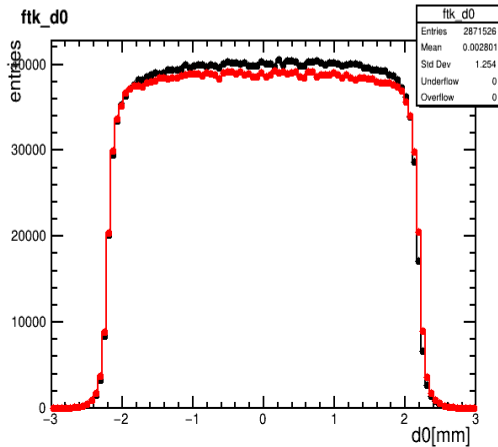
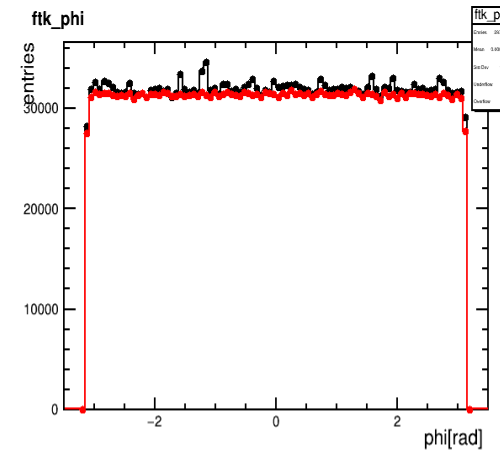
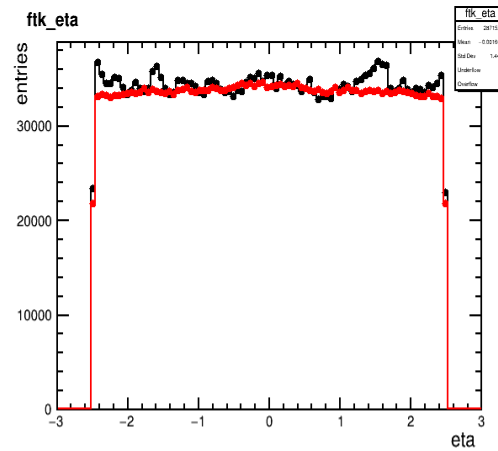
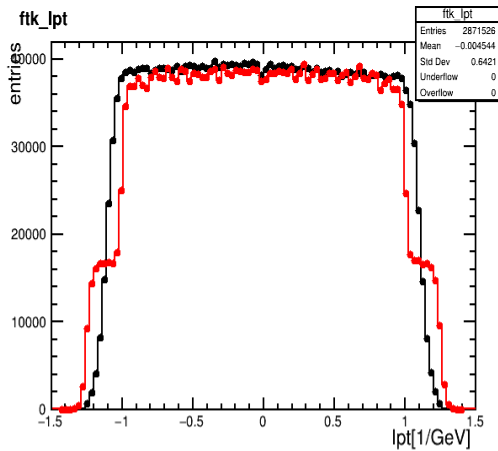
Compare tail and bulk

FTK tracks parameters



FullSimとFastSimのParameter分布の比較

✓ FullSimの再構成するすべてのphase spaceをFastSimで再構成できているかの確認



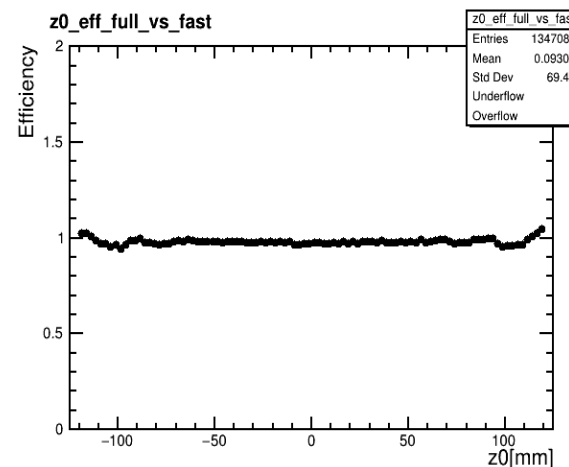
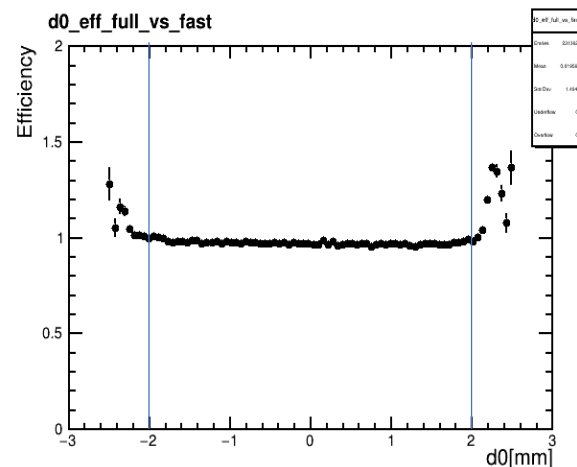
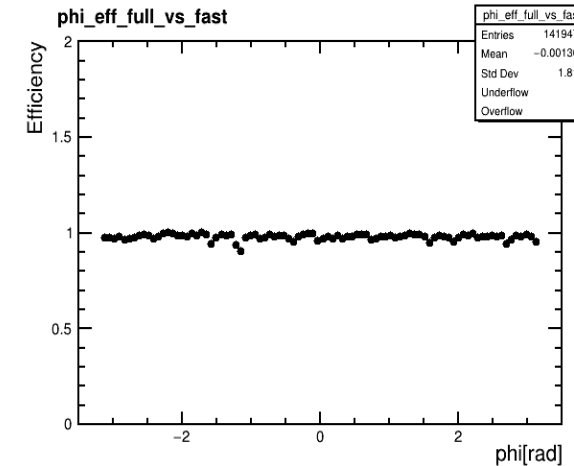
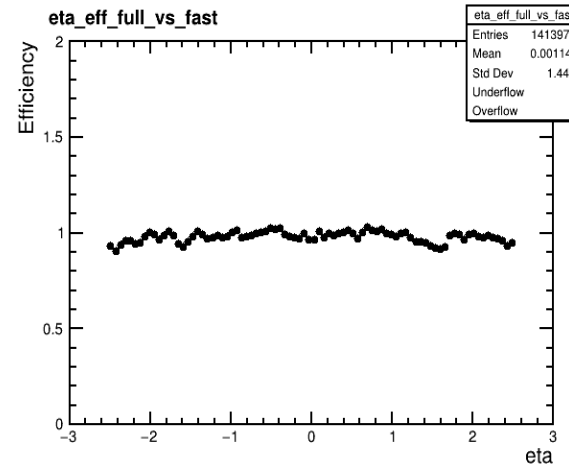
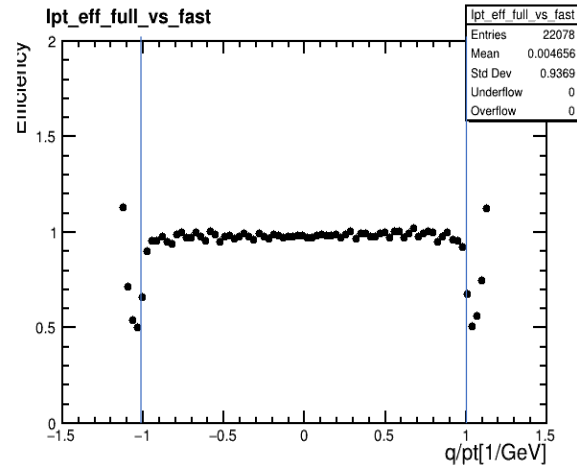
— full sim
— fast sim



割合を見た

FullSimとFastSimのParameter分布の比較

✓ FullSimの再構成するすべてのphase spaceをFastSimで再構成できているかの確認



横軸：各パラメータ
縦軸：Fast / Full

✓ FullSimで使用のPhase spaceで90%以上で再構成可能

Fake study

✓ Kodai at Waseda has studied FTK fake track

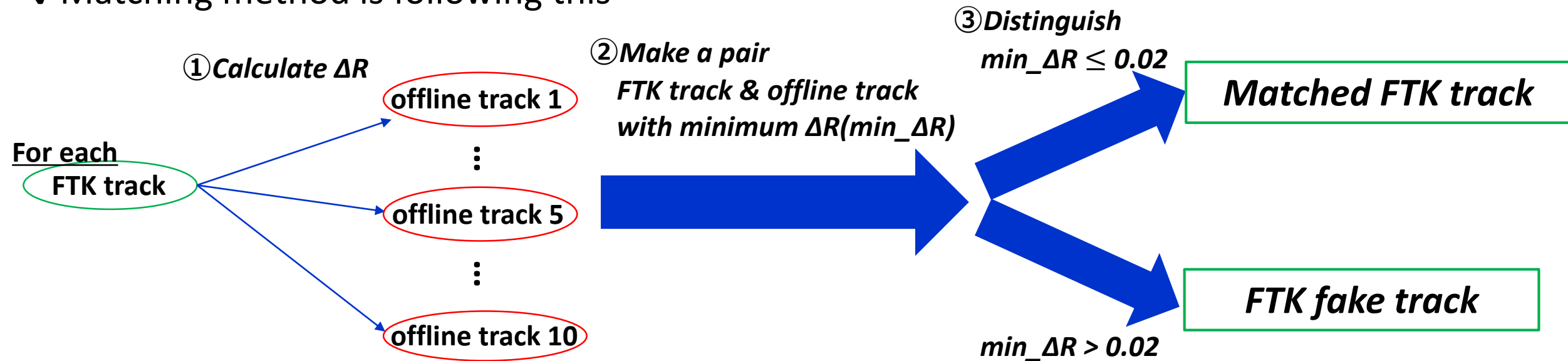
✓ I would like to introduce an interesting study in that about “duplicate track”

○ In this study

✓ Using sample

“group.trig-daq.ftk.MC16.PERF.AOD_FTK.20171207_1_EXT0 (total 5M muons)”

✓ Matching method is following this



✓ We need to optimize definition of ΔR matching threshold “0.02”

Offline software validation (*duplicate track*)

- ✓ In our matching method, there are pairs with two or more FTK tracks for one offline track with taking the minimum ΔR
- ✓ We focus on two matched FTK tracks (with $\Delta R \leq 0.02$) for one offline track

Ex.) ○ Make pair with minimum ΔR

“FTK track 1” pair with “offline track1”

“FTK track 2” pair with “offline track3”

“FTK track 3” pair with “offline track3”

⋮

“FTK track 8” pair with “offline track10”



ΔR with “FTK track 2” and “offline track3” ≤ 0.02 and

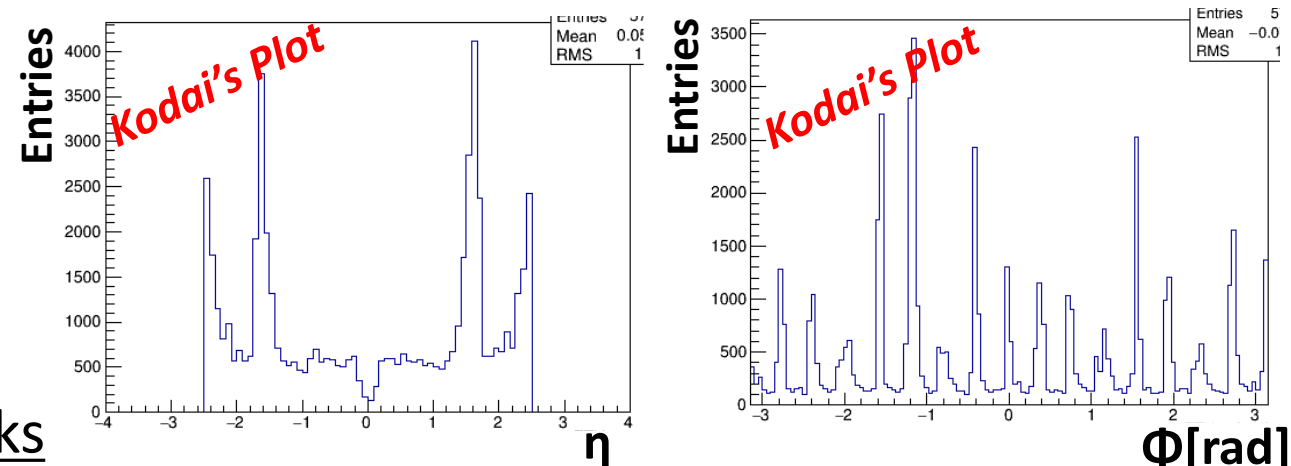
ΔR with “FTK track 3” and “offline track3” ≤ 0.02

\Rightarrow ✓ Plot “FTK track 2” and “FTK track 3” parameters

○Result

- ✓ On η and Φ , there is unique structure
- ✓ About η , there are 4 peaks
- ✓ About Φ , there are 16 peaks
- \Rightarrow “4 on η ” and “16 on Φ ”
match the number of *FTK η - Φ tower*
- \Rightarrow We check the FTK bank-ID on these FTK tracks

Distributions of focused FTK track parameters



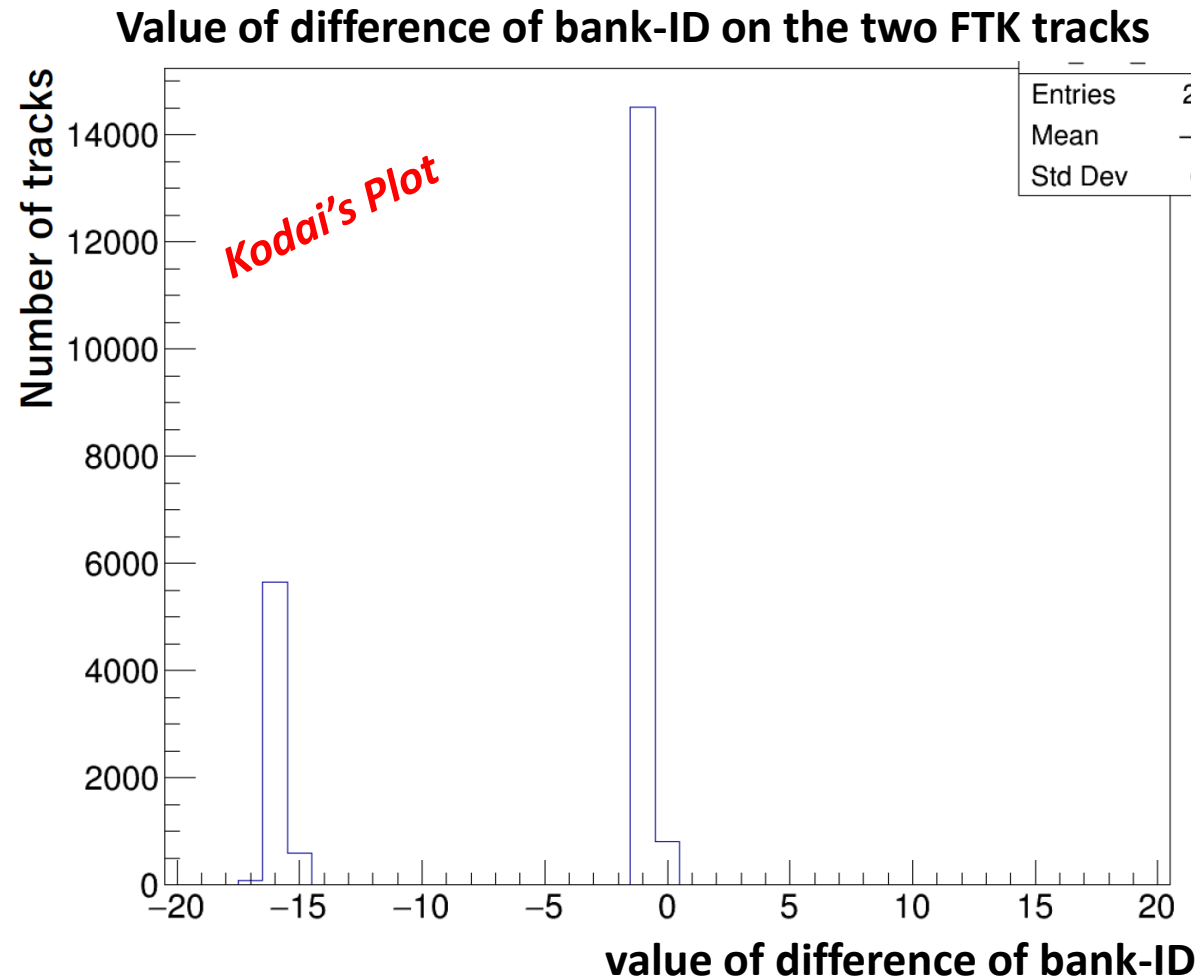
Offline software validation (*duplicate track*)

✓ We check the FTK bank-ID on two matched FTK tracks (with $\Delta R \leq 0.02$)

Result

- ✓ The value of difference = 0, -1, -15, -16, -17
- ✓ "0" : The two tracks are in the same FTK tower
- ✓ "-1" : In Φ direction, different 1 ID
- ✓ "-16" : In η direction, different 1 ID
- ✓ "-15, -17" : In η & Φ direction, different 1 ID
- ⇒ ✓ Many of one of the two tracks should be removed as "duplicate track"

✓ "0" may be due to another cause
 For example, FTK fake track ..
 ⇒ We will study it separately



Status of the implementation into Athena

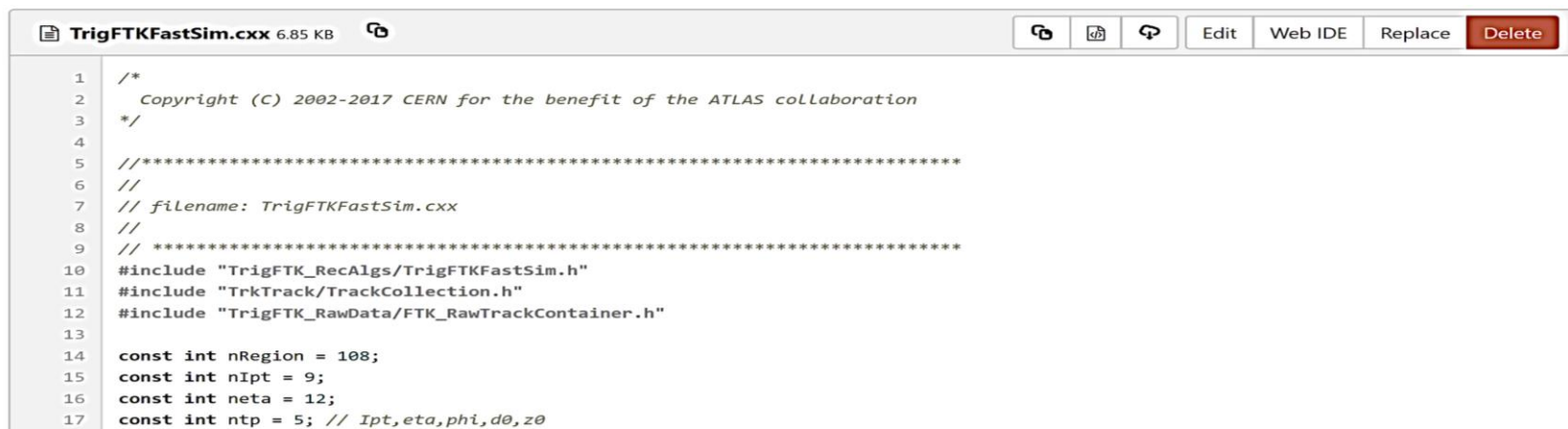
✓ Albert Kong (Adelaide) has been working on his Qualification Task to implement Offline-seeded Fast Simulation into Athena.

✓ Jira page for the implementation is prepared.

- <https://its.cern.ch/jira/browse/FTKSIM-62>

✓ The software is growing in the following link.

- https://gitlab.cern.ch/akong/athenaprivate1/blob/smearing-21.0/Trigger/TrigFTK/TrigFTK_RecAlgs/src/TrigFTKFastSim.cxx



```
TrigFTKFastSim.cxx 6.85 KB
1  /*
2   Copyright (C) 2002-2017 CERN for the benefit of the ATLAS collaboration
3  */
4
5  //*****
6  //
7  // filename: TrigFTKFastSim.cxx
8  //
9  // *****
10 #include "TrigFTK_RecAlgs/TrigFTKFastSim.h"
11 #include "TrkTrack/TrackCollection.h"
12 #include "TrigFTK_RawData/FTK_RawTrackContainer.h"
13
14 const int nRegion = 108;
15 const int nIpt = 9;
16 const int neta = 12;
17 const int ntp = 5; // Ipt, eta, phi, d0, z0
```

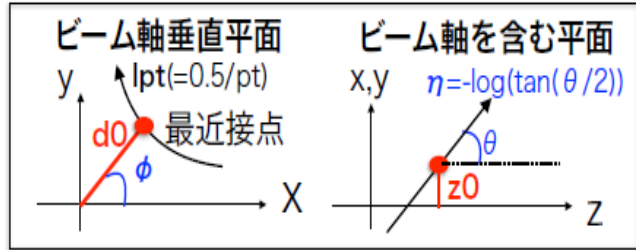
✓ We plan to test triggers with this software in the next few weeks.

Truth-seeded

変数 : $lpt, d0, \phi, \eta, z0$

事象 : 1 個のミュオン

(各変数が等確率に分布)

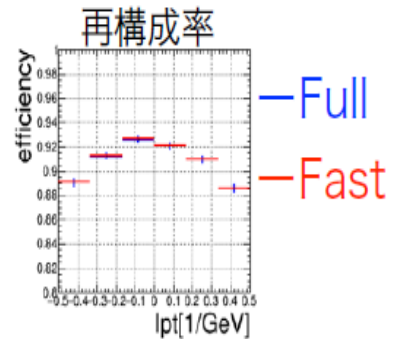


1.再構成率(ヒット共有率50%以上)

Truthの $lpt, \eta, d0, z0$ に依存

$lpt, \eta, d0, z0$ の領域ごとに測定し

その確率で飛跡の生成を行う

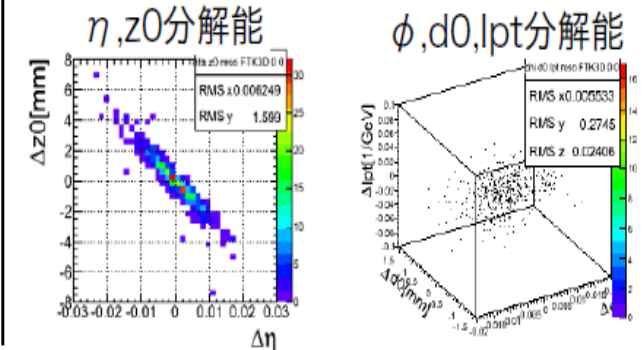


2.分解能(Truthとの差)

Truthの lpt, η に依存し $\eta, z0$ 及び $\phi, d0, lpt$ が相関

lpt, η の領域ごとに $\eta, z0$ を2D、 $\phi, d0, lpt$ を3Dの

正規分布で近似し、乱数をとってパラメータを決定



→1事象につき~10msの時間で生成が可能(※飛跡1000本を仮定)

1. Fast Simulation を開発する。

2. 1本の飛跡からなる事象(1つのミュオンが発生する事象など)について適用し、Full

Simulation との再構成率や分解能の違いを調べ、必要なら補正する。

3. 複数の飛跡からなる事象(複数の点からミュオンが発生する事象など)について適用し、飛跡が増えたときの影響を調べる。

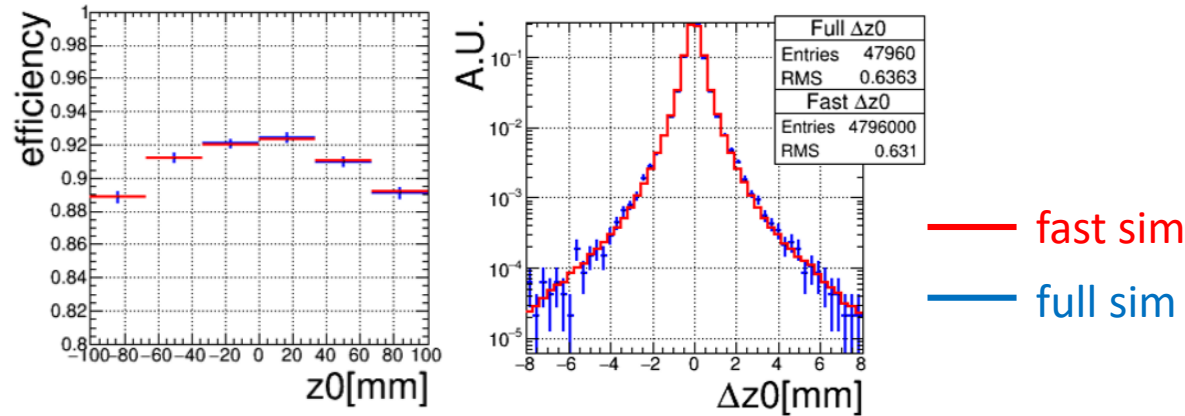
4. 実際に物理解析に利用する事象(H! 事象など)について適用し、事象トポロジーによる影響を調べる。

5. 適用してできたFast Simulation の飛跡をトリガーに利用し(一次衝突点の再構成やトリガーなど)、Full Simulation との取得率などの違いを調べる。

Truth-seeded

手法 : truthの飛跡パラメータを元にFTK trackのパラメータを乱数で決定

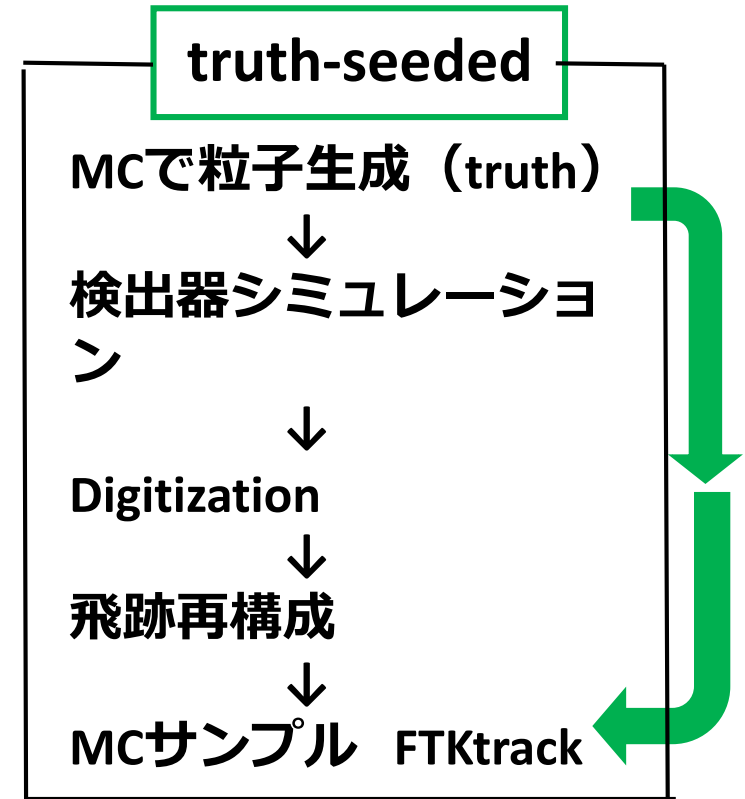
結果 : ①



➡ 飛跡再構成率、分解能共にfull simとよく一致

② ~ 10 (ms/事象) でFTK trackを生成可能

➡ 非常に高速



fake : full simで発生する
本来は存在しない飛跡

改善の余地 : • 偽飛跡 (fake) が再現できない

• full simとの相関がない

(full simで再構成されたtrackパラメータと完全に一致するtrackを再構成できない)