修　士　学　位　論　文

高輝度 LHC 実験の飛跡再構成における カルマンフィルターの 計算精度向上による改良

令和 8 年 2 月 6 日

専 攻 名　物理学専攻
学籍番号　243S101S
氏　　名　浅見 優輝

神戸大学大学院理学研究科博士課程前期課程

概要

LHC-ATLAS 実験は、欧州原子核研究機構 (CERN) で行われている国際共同素粒子実験である。陽子-陽子衝突型加速器 LHC(Large Hadron Collider) を用いて、陽子をバンチと呼ばれる塊にして加速し、重心系エネルギー 13 TeV で正面衝突させ、衝突点を ATLAS 検出器で囲うことで衝突による素粒子現象を観測している。ATLAS 実験は、2012 年のヒッグス粒子の発見を始め、素粒子標準模型の精密測定等の素粒子物理学の発展に大きく貢献してきた。しかし、現在に至るまでに素粒子標準模型を超える物理の兆候はほとんど発見できていない。そこで、標準模型を超える物理の発見を目指して、2026 年から LHC の一度のバンチ交差あたりの陽子衝突数（パイルアップ）を増加させる高輝度化アップグレードが予定されている。

LHC-ATLAS 実験では、トリガーと呼ばれるオンライン（実時間）での事象選別機構を用いて興味のある事象を選別し保存している。トリガーでは粒子飛跡の情報を用いるため、トリガー装置においては高速に粒子飛跡を再構成する必要がある。ATLAS 実験では，この飛跡の再構成に一般の PC で用いられる CPU を用いている。しかし、高輝度化によるパイルアップの増加と ATLAS 検出器のアップグレードに伴い、飛跡再構成の計算量が大幅に増加し、必要な計算資源の大幅な増加が見込まれ、現行のトリガーシステムでは対応が困難である。そこで ATLAS 実験のトリガーシステムでは新たな飛跡再構成手法が検討されており、Graphics Processing Unit(GPU) と呼ばれる画像処理装置を用いた飛跡再構成がその候補の一つになっている。

GPU は並列処理に優れる処理装置で、飛跡再構成を、多数の検出器ヒットを並列に処理することで高速に実行できる可能性がある。また、一般に GPU では倍精度よりも単精度の方が処理が高速であるため、可能な限り単精度で計算処理を行いたい。本研究では、単精度の使用に伴う飛跡再構成の効率低下、精度低下の原因の一つである、カルマンフィルターと呼ばれる飛跡再構成・パラメーター抽出アルゴリズムでの計算における問題の緩和に取り組んだ。

本論文では、GPU を用いた飛跡再構成における数値精度による再構成性能の比較、および上記で述べた飛跡再構成で用いられるカルマンフィルターにおける共分散行列の更新手法の改良とその再構成性能について論じる。

目次

第 1 章

序論

1.1　素粒子標準模型

素粒子物理学とは、この世界を構成する最小単位である素粒子とそれらの相互作用について調べる学問である。現在までに 12 種類のフェルミオン（物質を構成する粒子)、4 種類のゲージボソン(力を媒介する粒子)、ヒッグス粒子の計 17 種類の粒子が発見された。これらの粒子は素粒子と考えられており、素粒子標準模型と呼ばれる理論体系にまとめられている。素粒子標準模型では、強い相互作用（グルーオンが媒介する相互作用)、弱い相互作用（W ボソンと Z ボソンが媒介する相互作用)、電磁相互作用（光子が媒介する相互作用）の 3 つの基本的な力を用いて物理現象を記述する。素粒子標準模型はこれまで行われた素粒子実験の結果のほとんどすべてを説明することができる。しかし、暗黒物質、ニュートリノの質量問題、物質と反物質の非対称性等、素粒子標準模型では説明できない物理現象も存在し、世界中で標準模型を超える物理の探索が行われている。LHC 加速器実験もそのような探索の一つである。

1.2　LHC 加速器

LHC はスイス・ジュネーヴ郊外にある CERN(欧州素粒子原子核機構) の地下約 100 m に建設された世界最大の円型加速器である。LHC は全周約 27km、陽子ビームの衝突の重心系エネルギーは 13.6 TeV と世界最高である。LHC はこの世界最高エネルギーを活かし、2012 年のヒッグス粒子発見をもたらした。LHC では、陽子をバンチと呼ばれる塊にまとめ、時計回りと反時計回りに加速させ 25 ns ごとにバンチ同士を交差させ、陽子を衝突させている。その陽子衝突点に検出器を設置することで、物理現象を観測する。LHC の周上には 4 つのバンチ交差点があり、各交差点では主に ATLAS、CMS、LHCb、ALICE の 4 実験が行われている。ATLAS 実験と CMS 実験は、汎用型検出器を用いて、標準模型の精密測定から余剰次元や暗黒物質等の新物理まで、幅広い物理を研究対象とする実験である。LHCb 実験は、b クォークの性質の研究に特化した実験であり、物質と反物質の非対称性の解明を目的としている。ALICE 実験は、重イオンを衝突させ、クォーク・グルーオンプラズマを生成し、その性質の解明を目的としている実験である。

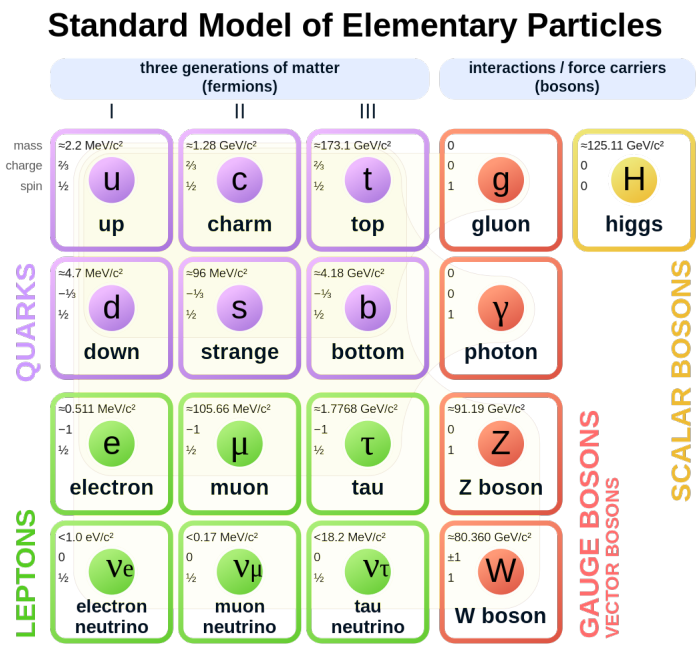


図 1.1: 素粒子標準模型 [1]。現在までに 12 種類のフェルミオン（物質を構成する粒子）、4 種類のゲージボソン（力を媒介する粒子）、ヒッグス粒子の計 17 種類の粒子が発見された。

LHC は 2010 年から運転を開始しアップグレードを繰り返しながら、2010 年から 2012 年までの Run 1 では重心系エネルギー 7 – 8 TeV、2015 年から 2018 年までの Run 2 では重心系エネルギー 13 TeV、2022 年から 2025 年までの Run 3 では重心系エネルギー 13.6TeV で運転が行われた。そして 2026 年からは LHC の高輝度化アップグレードを行い、2030 年から Run 4 として運転開始予定である。この高輝度化アップグレードについて、以下で詳しく述べる。

1.3　LHC の高輝度化アップグレードが目指す物理

LHC は現在のところヒッグス粒子を生成できる唯一の加速器であるため (図 1.3)、LHC ではヒッグス粒子についての詳細な研究が重要な課題である。ヒッグス粒子は、物質に質量を与えている粒子である。ヒッグス粒子は質量の大きい粒子とより強く結合するため、主に重い粒子を介して生成される。トップクォークや未知の重い素粒子とも結合が大きい可能性があり、新物理への手掛かりと考えられている。そのためヒッグス粒子は 2012 年の発見以来様々な研究が行われてきたが、現在でも分かっていないことは多い。ヒッグス粒子の自己結合定数もその一つである。ヒッグス粒子の自己相互作用（図 1.4）は、現在あまり理解が進んでいないヒッグスポテンシャルを測定するための有力な方法であり、LHC で測定することが期待される。しかし、ヒッグス粒子の生成断面積は小さく、自己結合に感度のあるヒッグス対生成事象はさらにまれである。そのため、これらの物理の研究には、現在までに取得されたデータよりも多くのデータが必要となる。そこで統計

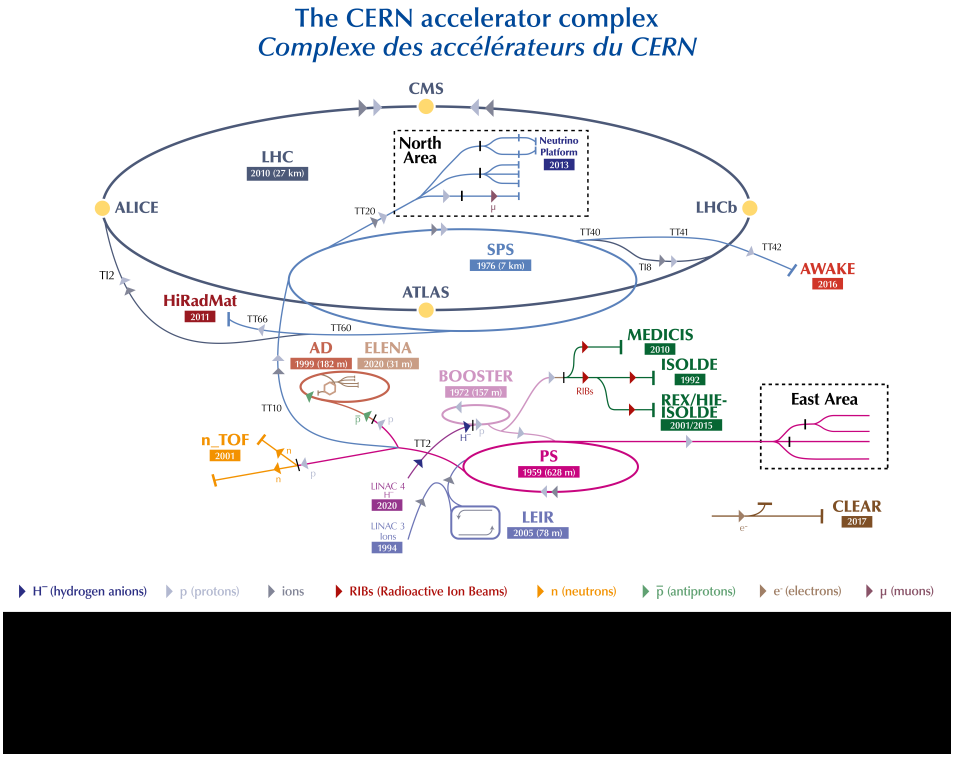


図 1.2: LHC と前段加速器群 [2]

量を大幅に増加させるために、LHC の高輝度化アップグレードを行っている。

1.4 LHC の高輝度化アップグレード

LHC の主要な研究テーマであるヒッグス粒子の精密測定や新粒子生成等の素粒子現象は生成断面積が小さく、上記で述べたように高統計が必要である。そのため、陽子の衝突頻度（ルミノシティー）を高めて、より多くの衝突を起こし、それを取得する必要がある。具体的には、バンチ当たりの陽子数を増加させること、陽子ビームを絞ることで、ルミノシティーを増加させる。瞬間ルミノシティーは、$2 \times 10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ から $5 - 7.5 \times 10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ に増加させ、積分ルミノシティーは Run3 の $450\mathrm{fb}^{-1}$ から 2030 年から 10 年間の運転で $3000 - 4000\mathrm{fb}^{-1}$ を達成する計画である。

1.5 LHC 実験における飛跡検出器とトリガーシステム

この論文では，飛跡検出器で測定した飛跡の再構成手法の改良について述べる。高エネルギー衝突型加速器実験の飛跡検出器とトリガーシステムの例として、著者が所属する ATLAS 実験を例に説明する。ATLAS 検出器の全体像 (Run 3) を図 1.6 に示す。ATLAS 検出器は、直径 22 m、長さ 44 m、重さ 7000 t の大型検出器である。内側から、内部飛跡検出器、電磁カロリメーター、ハ

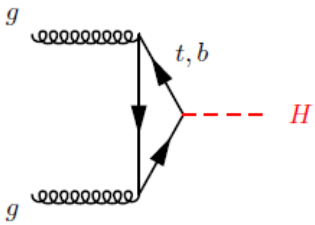
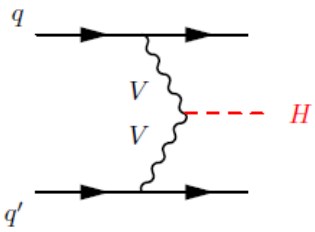
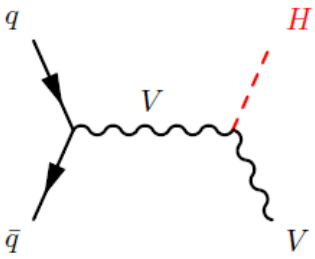
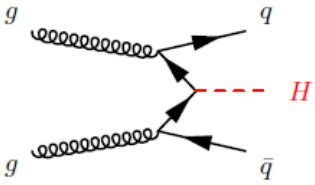
Production mode	LO diagram
ggF production	
VBF production	
VH production	
$q\bar{q}H$ production	

図 1.3: ATLAS 実験におけるヒッグス粒子の主な生成過程のダイアグラム [3]。陽子内のグルーオン衝突によるグルーオン融合 (ggF) による生成が支配的であり、これにベクトルボソン融合 (VBF) が続く。

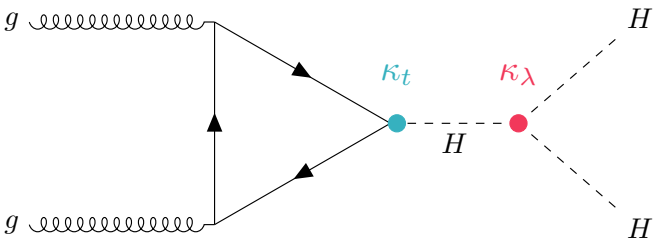


図 1.4: ヒッグス粒子の三点結合のダイアグラム [4]。

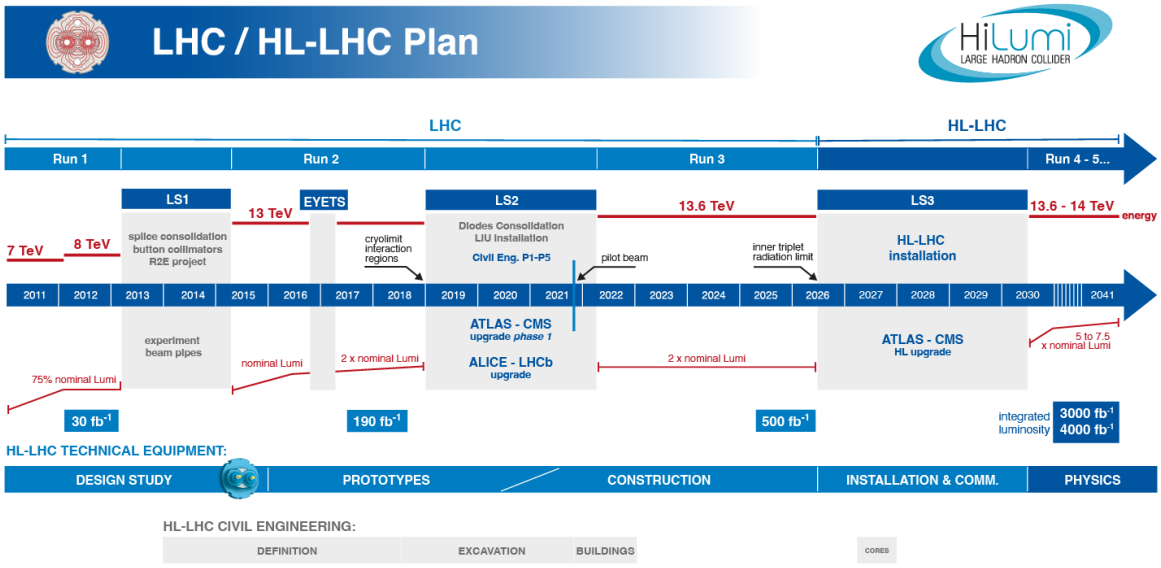


図 1.5: LHC の高輝度化計画 [5]。2026 年半ばから加速器や各検出器のインストールが開始され、2030 年から Run 4 として運転開始予定である。

ドロンカロリメーター、ミューオン検出器の順で構成されている。飛跡検出器と電磁カロリメータの間にはソレノイド磁石が設置されており、飛跡検出器領域内には、ビーム軸方向に約 2 T(テスラ) の磁場がかけられている。その磁場によって、検出器を通過する荷電粒子が曲げられ、その曲率から横運動量 p_T(ビーム軸に垂直な平面内の運動量) を測定する。

ATLAS 実験で使用している座標系を図 1.7 に示す。ATLAS 検出器では、陽子衝突点を原点とし、LHC の中心方向を正とするように x 軸、ビーム軸に沿うように z 軸、そして x 軸と z 軸に対して直交座標系 (右手系) を成すように y 軸を取る。ビーム軸周りの方位角を $\phi\,(-\pi < \phi < \pi)$、ビーム軸からの天頂角を $\theta\,(0 < \theta < \pi)$ とする。ATLAS 実験では、θ 方向の変数として、擬ラピディティ

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right) \tag{1.1}$$

が用いられる。η は質量がゼロとみなせる粒子について、ローレンツ変換に対して加算的に変換する量である。第 3 章で説明する ODD 検出器でも同様の座標系を用いる。

ここからは、ATLAS 検出器の飛跡検出器とトリガーシステムについて述べる。

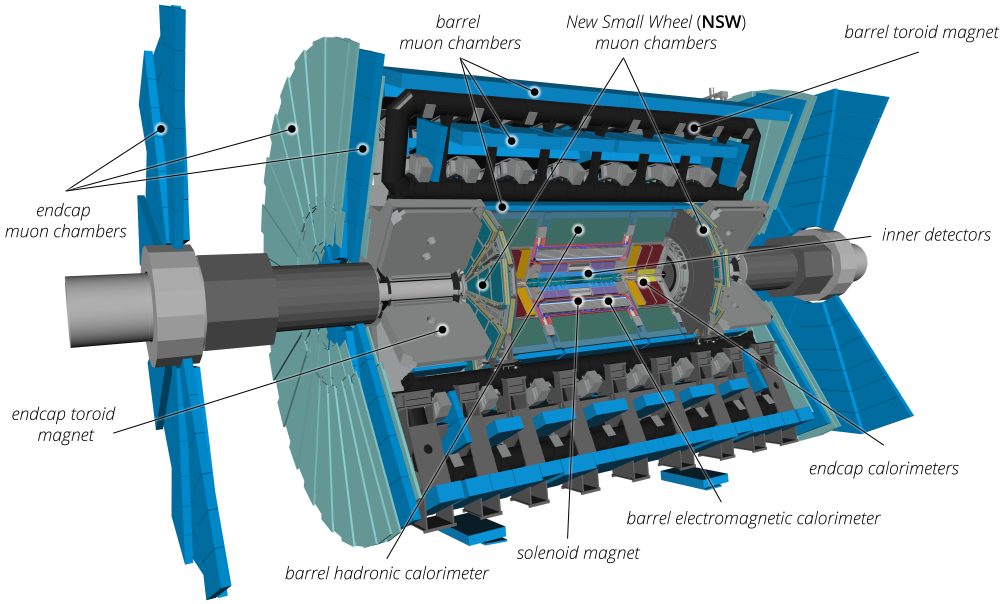


図 1.6: ATLAS 検出器の全体像 (Run 3)[6]。直径 22 m、長さ 44 m、重さ 7000 t の大型検出器である。陽子衝突点を覆うように、内側から、内部飛跡検出器、電磁カロリメーター、ハドロンカロリメーター、ミューオン検出器の順で構成されている。飛跡検出器と電磁カロリメータの間にはソレノイド磁石が設置されており、飛跡検出器領域内では、ビーム軸方向に約 2 T(テスラ) の磁場がかけられている。

1.5.1 ATLAS 内部飛跡検出器 ITk (Inner Tracker)

ATLAS 実験では、LHC の高輝度化に伴い、Run 3 まで使用していた内部飛跡検出器 (ID) を総入れ替えし、新たに ITk と呼ばれるすべてシリコンベースの半導体検出器で構成されたの内部飛跡検出器を導入予定である。ITk はシリコンピクセル検出器とシリコンストリップ検出器から構成される。表 1.1 に示されているように、ITk の各検出器は ID よりも細分化されている。検出器全体としては、η 方向のカバー領域は 2.7 から 4.0 へと大幅に増加する。また、ITk は全体で 50 億以上のチャンネルを持ち、ID を構成するピクセル検出器とストリップ検出器の約 1 億チャンネルから大幅に増加する。図 1.8 に、ITk レイアウトの概略図を示す。また、図 1.9 に ITk のジオメトリの全体図を示す。本論文では、円筒形の側面部分をバレル、底面部分をエンドキャップと呼ぶ。

図 1.10 は、$p_T = 1\,\mathrm{GeV}$ のミューオンが ITk に残したヒット点の個数の分布である。η のほとんど全域で 9 ヒット以上残していることが確認できる。

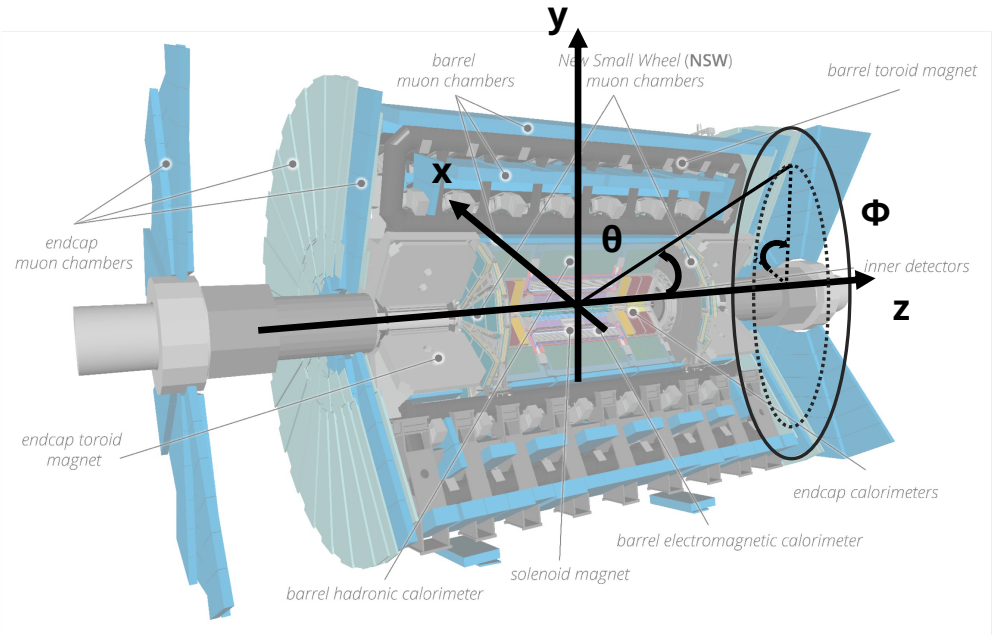


図 1.7: ATLAS 検出器の座標系

表 1.1: ID と ITk のセンサーサイズ。ITk は ID から、センサーサイズが細分化していることがわかる。

	ピクセル検出器	ストリップ検出器
ID	$50 \times 400\,\mu\mathrm{m}^2$	$80\,\mu\mathrm{m} \times 12.8\,\mathrm{cm}$
ITk	$50 \times 50\,\mu\mathrm{m}^2$, $25 \times 100\,\mu\mathrm{m}^2$	$75\,\mu\mathrm{m} \times 2.5\,\mathrm{cm}$, $75\,\mu\mathrm{m} \times 5.0\,\mathrm{cm}$

ITk ピクセル検出器

ITk ピクセル検出器は、センサーサイズ $50 \times 50\ \mu\mathrm{m}^2$ と $25 \times 100\ \mu\mathrm{m}^2$ のシリコンセンサーが含まれ、ID のピクセル検出器のセンサーサイズ $50 \times 400\ \mu\mathrm{m}^2$ よりも小型化している。$|\eta| < 4.0$ までの範囲をカバーしており、バレル領域に 5 層、エンドキャップ領域に多数設置されている。

ITk ストリップ検出器

ITk ストリップ検出器は、二本のストリップをずらして張り合わせてステレオ化することによって、表と裏の両面の情報から、位置を測定している。ITk では、幅約 75 μm、長さ 2.5 cm または 5.0 cm のストリップが使用されている。ストリップ検出器は $|\eta| < 2.7$ までの範囲を覆っており、

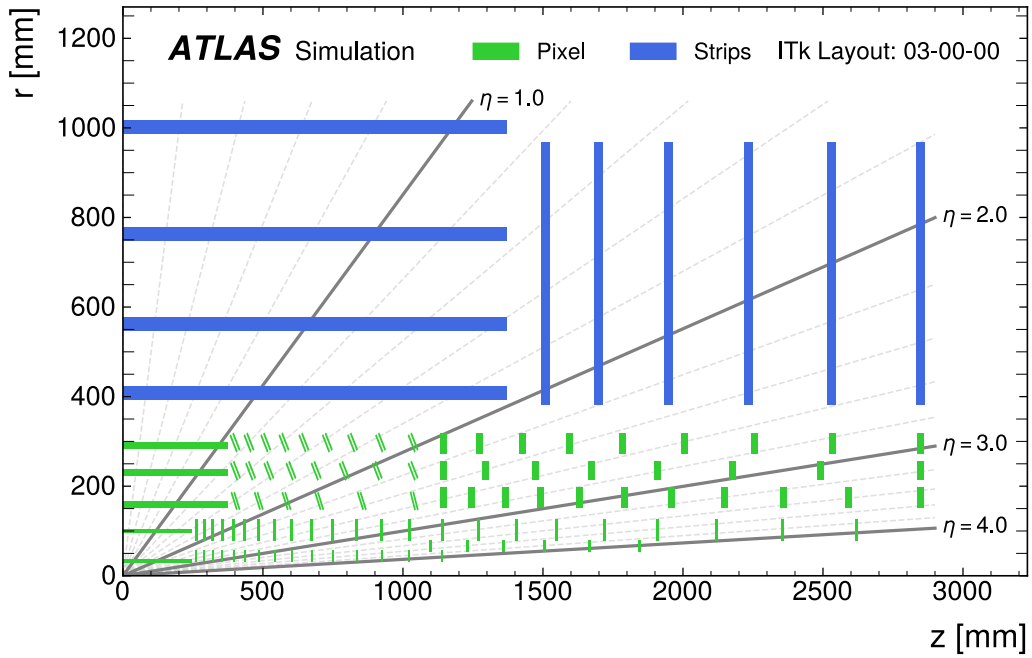


図 1.8: ATLAS 内部飛跡検出器のレイアウト [7]。検出器全体としては、η 方向のカバー領域は ID の 2.7 から 4.0 へと大幅に増加する。

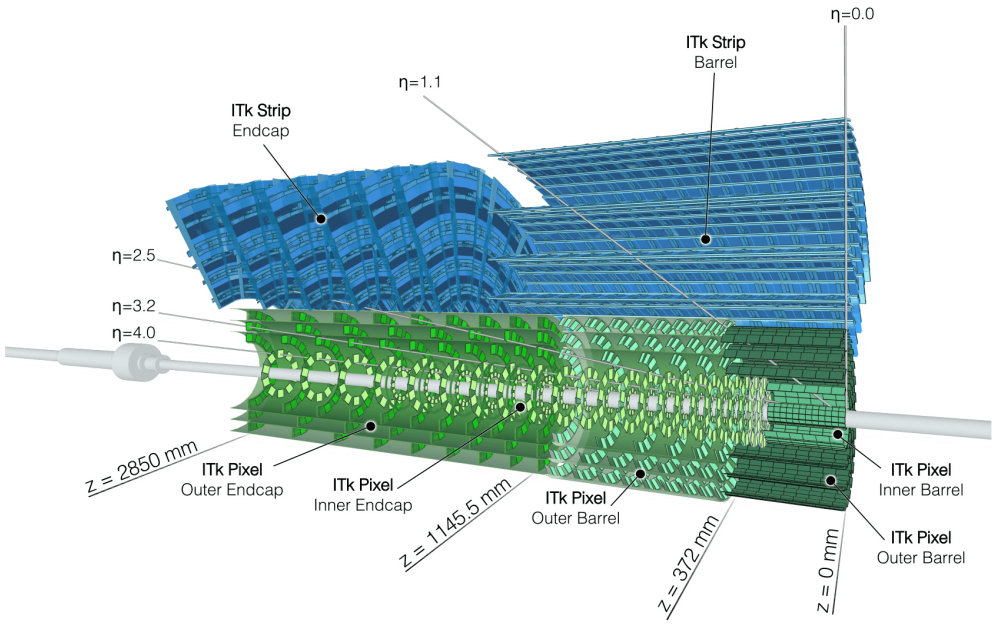


図 1.9: ITk の全体像 [7]。ITk は、$|\eta| < 4.0$ までの範囲をカバーしており、バレル領域に 5 層、エンドキャップ領域に多数設置されている。

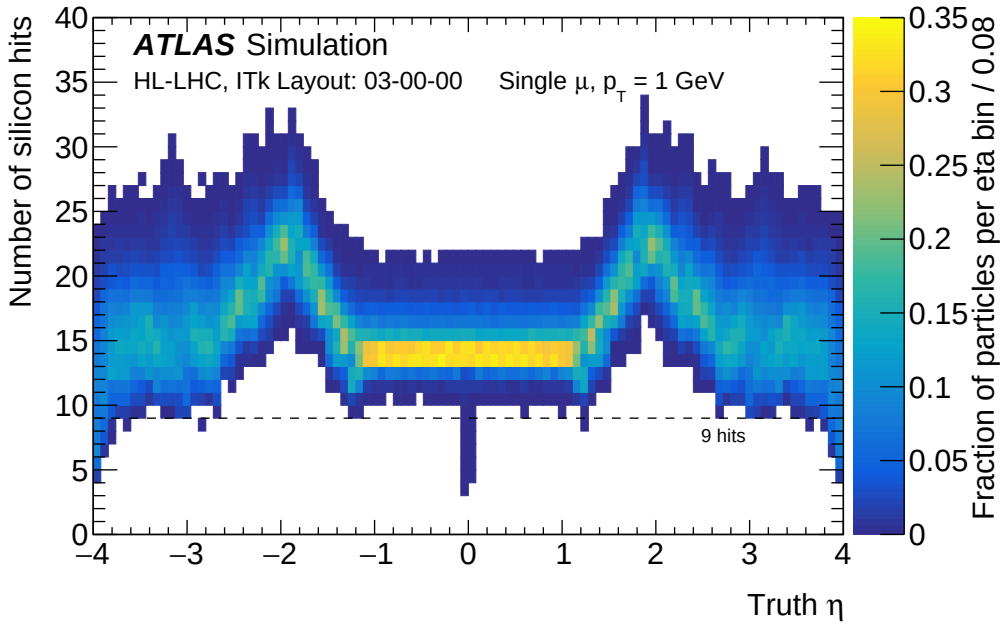


図 1.10: $p_T = 1\,\mathrm{GeV}$ のミューオンが ITk に残したヒット点の個数の分布（シミュレーション [7]。ミューオンはビームラインに垂直な面での距離が 0　2 mm の範囲で一様に分布するように生成されており、$z = -15\,\mathrm{cm}, 0\,\mathrm{cm}, 15\,\mathrm{cm}$ の固定された位置において、それぞれ同数ずつ生成されている。

バレル領域では 4 層、エンドキャップ領域では 6 層存在する。

1.5.2　ATLAS トリガーシステム

ATLAS 実験では、生成されるデータ量が膨大（約 100TB/s）であり、全てのデータを保存することができないため、トリガーシステムを用いて興味のある物理事象を選別し保存している。ATLAS トリガーシステムは、図 1.11 に示されるように、大きく二段階に分けられている。第一段階目 (Level-0 Trigger) は、ハードウェアベースの高速で粗い選別を行う。その後、第二段階目 (Event Filter) でソフトウェアベースの詳細な選別を行い、最終的に 40 MHz のデータを 10 kHz 程度にまで削減する。本研究の対象は、二段階目のソフトウェアベースのトリガーである Event filter における飛跡再構成である。Event filter では Level-0 では用いなかった中央飛跡検出器の情報から飛跡を再構成し、他の検出器と組み合わせて高度な事象選別を行うことで事象を大幅に削減する。

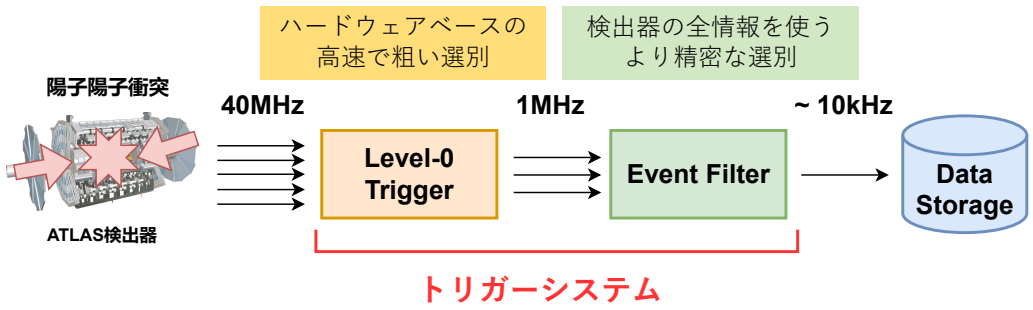


図 1.11: ATLAS トリガーシステム。トリガーシステム内の名称は 2030 年開始予定の Run 4 のものである。ATLAS 検出器からのデータに対して、Level-0 Trigger において、ハードウェアベースの高速で粗い選別を行う。その後、Event Filter でソフトウェアベースの詳細な選別を行い、最終的に 40 MHz のデータを 10 kHz 程度にまで削減する。

1.6 高輝度化による課題

高輝度化による課題の一つが、飛跡再構成である。飛跡再構成は、Event Filter での計算資源の多くを使用する。図 1.12 に示すように、今後 ATLAS 実験では、イベント再構成に必要な CPU 資源の量は、利用可能な量を少なくとも 2 倍以上上回ると予想されている。

飛跡再構成は、荷電粒子数の増加に対して組み合わせ的に演算回数が増加するため、衝突イベントの再構成において、最も計算量の多い部分である。LHC に代表される陽子陽子衝突型加速器では、荷電粒子数の増加は主にパイルアップの増加に起因しており、多くの再構成アルゴリズムでは、実行時間は荷電粒子の数のおよそ二乗に比例して増加する。

高輝度 LHC ではルミノシティーの増加によりパイルアップ（1 バンチ交差あたりの陽子対衝突数）も大幅に増加する（図 1.13）。それに伴い検出器内のヒット（検出器と粒子との相互作用点）の数が増加し飛跡再構成の計算量もヒットの組み合わせ数に応じて増加する。例えば、LHC-ATLAS 実験では、LHC の高輝度化と 1.5.1 節で述べた ATLAS 内部飛跡検出器のアップグレードによる計算量の増加に伴い、必要な計算機資源も大幅に増加し、計算機資源の制限から現行のトリガーシステムにおける CPU を用いた飛跡再構成手法では対応が困難になることが見込まれている。これについて、次章で詳しく述べる。

1.7 本論文の構成

本論文では、第 2 章で高エネルギー実験における飛跡再構成アルゴリズムについて詳しく説明する。第 3 章では、飛跡再構成の並列化とその利点と課題を明らかにする。第 4 章では、前章で挙げた並列化の課題を解決するための手法について述べ、その手法を飛跡再構成アルゴリズムに実装した結果を示す。第 5 章では、本研究のまとめと今後の展望について述べる。

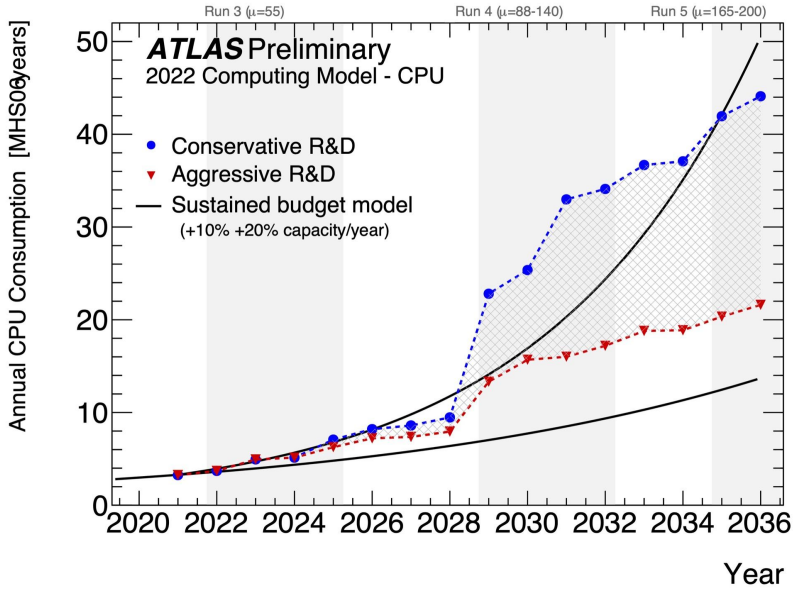


図 1.12: 2020 年から 2036 年にかけての ATLAS 実験における CPU の推定使用量の推移 [8]。青は保守的な研究開発を行った場合のシナリオ、赤が積極的な研究開発を行った場合のシナリオを意味している。黒線は毎年行われる CPU 資源の増加と新しいハードウェアの性能向上による影響を示しており、それらを合わせた結果として、計算能力が 10 %(下線)、20 % 増加する場合を示している。なお、縦方向の網掛けの領域は ATLAS 実験がデータ取得を行う期間を表す。ただし、これは 2022 年時点のものであり、現在はデータ取得期間等に変更がある。

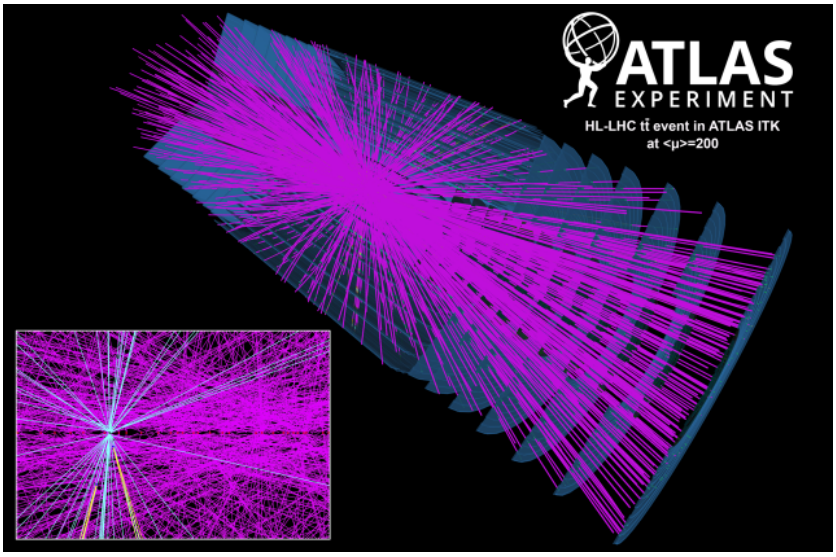


図 1.13: 高輝度 LHC において、一度のバンチ交差で 200 個の陽子が衝突しているシミュレーションの様子。以降、一度のバンチ交差によって n 個の陽子が衝突している場合、それをパイルアップ n (パイルアップ 200 など) と呼ぶ。

第 2 章

高エネルギー実験における飛跡再構成

高エネルギー実験における飛跡再構成とは、検出器と粒子の相互作用によって生じる一連の測定点 (measurement) から，一つの粒子からなる軌跡であることが確からしいものを抜き出し，そこから飛跡のパラメーター（運動量，飛跡の生成位置に関連する情報など）を取り出す作業である。

以下 2.1 節では高エネルギー実験で用いられる汎用飛跡再構成アルゴリズム ACTS について詳しく説明し、2.2 節では高エネルギー実験の飛跡再構成で一般に用いられているカルマンフィルターについて詳しく述べる。

2.1 汎用飛跡再構成アルゴリズム ACTS (A Common Tracking Software)

飛跡再構成において、アルゴリズムの調整は各実験の検出器のジオメトリの詳細に強く依存する一方で、使用されるアルゴリズムは多くの共通した特徴を持っている。また、高輝度 LHC や今後の実験における極めて過密な環境では、これらのアルゴリズムにさらに大きな計算負荷がかかることが予想され、より高性能なアルゴリズムの開発が求められる。ACTS は、そのような背景から開発された実験に依存しない飛跡再構成ソフトウェアである [9][10]。ACTS のコードは C++ ベースで書かれており、スレッドセーフ（マルチスレッド環境下で、複数の処理を同時に実行したり、同じデータを扱ったりしても問題ない）になるように設計され、並列実行をサポートしている。また、データ構造はベクトル演算に最適化されており、計算機のベクトル演算機能を用いることによる線形代数の演算の高速化が図られている。また、ACTS はアルゴリズムの拡張が容易なように設計されており，新しいアルゴリズムの開発プラットフォームとして用いることができるようになっている。

飛跡再構成は、大きくグローバルな手法とローカルな手法に分けることができる。グローバルな手法は、検出器全体の測定点を一つのまとまりとして捉えて飛跡を見つける手法であり、共形写像（conformal mapping）やホフ変換等の空間情報から飛跡パラメーター空間への変換手法や、ニューラルネットワークを用いた手法がある。一方、ローカルな手法は、シード (seed) と呼ばれる、測定点をいくつかまとめた飛跡を形成する測定点の部分集合の候補を作り、それを完成させる

ために、それらに繋げる新たな測定点を探索する手法である。以下で説明する ACTS はローカルな手法を用いた飛跡再構成アルゴリズムである。

ATLAS 実験では、検出器内にビーム軸方向に沿った磁場が掛けられており、荷電粒子は磁場に巻き付くように曲がる。よって、荷電粒子の飛跡は運動量の大きさに従ったらせん軌道を描くことになる。本研究は、ATLAS 実験と同様の環境下で行っているため、本研究の再構成対象の荷電粒子もらせん軌道を描く。このような軌道を一般に図 2.1 に示すように、5 つのパラメーターで表現する。

d_0　：横方向 (ビーム軸に垂直な平面内) のインパクトパラメータ。ビーム軸に対する荷電粒子の飛跡の x-y 平面内における最近接点とビーム軸の距離を表す。

z_0　：縦方向（ビーム軸に平行な方向）のインパクトパラメータ。ビーム軸に対する荷電粒子の飛跡の最近接点の z 座標を表す。

ϕ　方位角。ビーム軸に対する荷電粒子の飛跡の最近接点における $x-y$ 平面での粒子の運動量方向を表す角度。

θ　：極角。ビーム軸に対する荷電粒子の飛跡の最近接点におけるビーム軸（z 軸）からの粒子の運動量方向の角度。

q/p　：粒子の電荷 q と運動量の大きさ p の比。磁場中での粒子の曲率を表す。

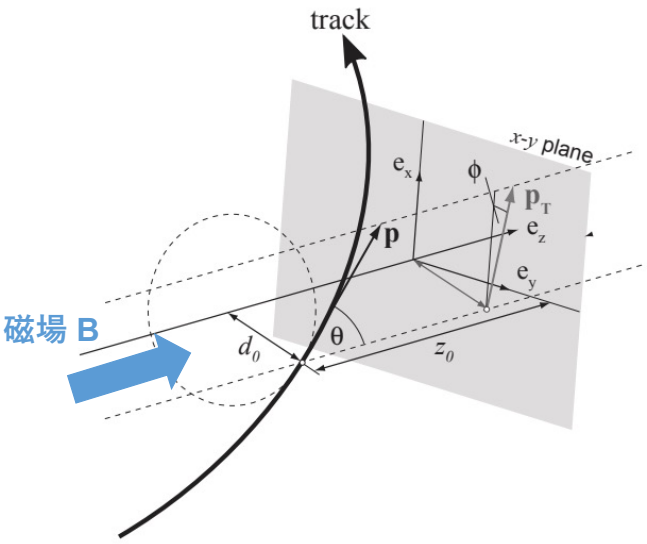


図 2.1: トラックパラメーター [11]。らせん軌道を (d_0, z_0, ϕ, θ, q/p) の 5 つのパラメーターで表す。

ACTS で採用されている飛跡再構成アルゴリズムの流れを図 2.2 に示した。以下ではこのアルゴリズムについて説明する [12]。

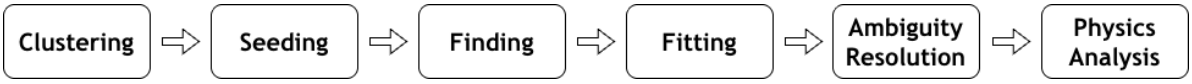


図 2.2: ACTS におけるトラッキングのパイプライン。左から右に流れていき、検出器からの生信号から粒子が再構成されていく。Ambiguity resolution と Fitting は順序を逆にすることもある。

Clustering

検出器の読み出し信号をヒットに変換するアルゴリズムである。粒子が検出器を通過すると検出器に信号が与えられるが、信号と通過した粒子の数は必ずしも 1 対 1 に対応しているわけではない。例えば、検出層に対して斜めに通過した場合のように、一つの粒子がある検出層を通過した際に、複数のセグメント (検出器の位置分解能の最小単位) に対して信号を与える場合がある。そのような場合でも、複数の粒子が通過したとみなさずに、一つの粒子が通過したと判別する必要がある。そこで、信号のあった隣接する同一検出層上のセグメントを同じ粒子によるヒットとみなして結合し、クラスターを構成する。その後、クラスターの有効位置（粒子がセンサーと交差したと推定される位置）を計算し、その位置をヒットとする。また、有効位置の計算方法にはセグメントの幾何学的位置のみから算出する方法や各セグメントで収集された電荷量で重み付けして算出する方法等がある。本研究では既にクラスタリングを行ったデータを使用している。

Seeding

飛跡として矛盾しないヒット 3 個を見つけて一組にまとめるアルゴリズムである。seeding アルゴリズム内では、組み合わせ的に生成された各 3 つのヒットの組（トリプレット）に対して、横運動量 p_T およびインパクトパラメーター d_0 を推定し、それらの値に応じて段階的にフィルタリングを行う。図 2.3 に示すように、必ずしも隣り合う検出層のヒットを繋がなくても良い。このフィルタリングで得たトリプレットのことをシード (seed) と呼ぶ。

Finding

Seed と矛盾しないようなヒットを繋げていき、最終的にトラックの候補を出力するアルゴリズムであり、組み合わせカルマンフィルター (Combinatorial Kalman filter) と呼ばれる。

これは以下のように行われる図 2.4。まず、seed をフィットすることによって得たトラックパラメーターと共分散行列から、物質との相互作用も考慮しながら隣の検出器層に外挿する。隣接する検出層に n 個の無矛盾なヒットがある場合、直前のトラック候補 (オリジナルと呼ぶ) のコピーを n 個作成し、後述するカルマンフィルターを使用して各コピーに対して隣接する層で見つけたヒットをそれぞれ一つずつ割り当て、各々トラックパラメーターを更新する。また、オリジナルのトラック候補は、その検出層にヒットがなかったとしてそのまま保持され、オリジナル 1 個とコピー n 個の合計 $n+1$ 個のトラック候補が次の層に引き継がれる。この手順を検出器の最外層に到達、または（あるトラック候補に対して）missing ヒットの数が閾値を超えるまで続ける。ここで

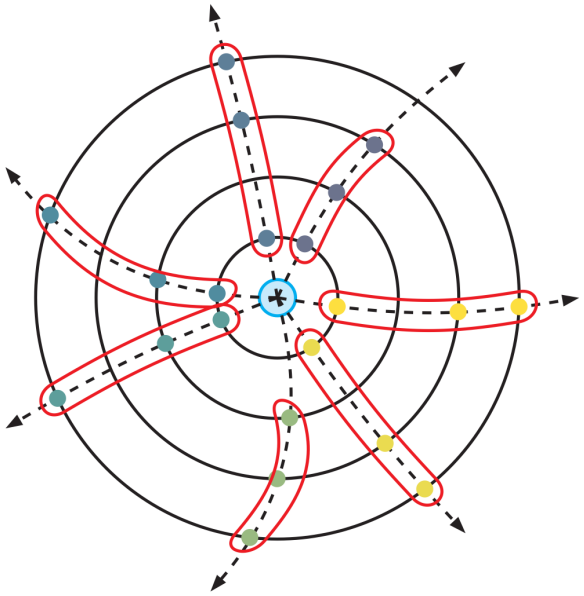


図 2.3: Seeding[12]。飛跡として矛盾しないヒット 3 個を見つけて一組にまとめ、seed と呼ばれるトリプレットを作成する。この seed は finding の入力になる。

missing ヒットとは、存在すると想定される位置にヒットが存在しないことを意味する [13]。

Fitting

Finding で得られたトラック候補から、各トラックパラメータを得るアルゴリズムである。fitting の方法はいくつかあるが、ACTS ではカルマンフィルターを用いた fitting が行われる。これについては 2.2 節で述べる。

Ambiguity resolution

各トラック候補に点数を付け、その点数に応じてトラック候補を除去していき、フェイクトラック（再構成したものに対応する粒子がないトラック）を取り除くアルゴリズムである。finding における組み合わせの段階では，測定位置の不定性，トラック外層の不定性などから，次の層のヒットの候補を一つに絞れない場合も多い。高い再構成効率を達成するには、finding で得られるトラック候補数はしばしば多くなり、その中にはフェイクトラックも場合によっては多数含まれることになる。Ambiguity resolution はこのようなフェイクトラックを取り除くアルゴリズムの一つである。例えば、より長い距離を整合したヒット列は異なる粒子由来のヒットの組み合わせである可能性が低いため、ヒット数が多いほど高得点になる。また、他の候補とヒットを共有している場合やヒットがあると期待されるセンサーにヒットがない場合は減点される。さらに、χ^2、運動量等も考慮される。このような判定により各トラックに点数が付けられ、低い点数のトラックは除去される。なお、Ambiguity resolution と Fitting は順序を逆にする場合もあり、図 2.2 に示すような本研究でも Ambiguity resolution と Fitting の順序を逆にしたパイプラインを使用した。

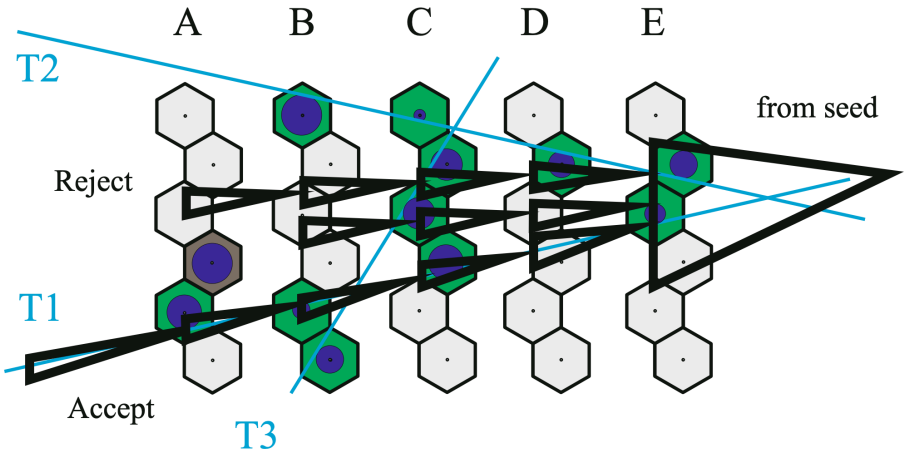


図 2.4: 組み合わせカルマンフィルターのイメージ図 [13]。E 層から A 層に向かって、飛跡として矛盾しないヒットを seed から繋げていく様子を表している。緑色のセルは信号のあった検出器を表しており、水色の直線は粒子を表している。この図では、一つのトラック候補に対して、missing ヒットが 2 つある場合に reject される。この図では外層 (E 層) から内層 (A 層) に向けてカルマンフィルターが進んでいるが、本研究で用いている組み合わせカルマンフィルターは、内層から外層に向けて進んでいる。

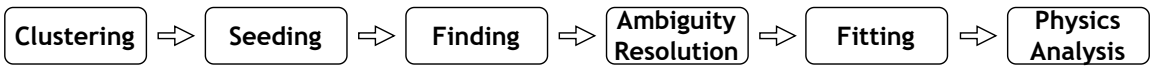


図 2.5: 本研究で使用したトラッキングパイプライン。図 2.2 とは Ambguity resolution と Fitting の順序が逆になっている。

2.2 カルマンフィルター

現実世界の検出器内では、散乱や電離・制動輻射によるエネルギー損失によって荷電粒子の運動は変化する。また、検出器には誤差やノイズがある。これらにより、ヒット点をらせん曲線で結ぶだけでは、飛跡の再構成効率や飛跡パラメーター制度の低下を引き起こす可能性がある。そこで、高エネルギー実験における飛跡再構成では、カルマンフィルターと呼ばれるアルゴリズムが広く用いられる。カルマンフィルターは、誤差のある時系列に並んだ観測値から、次のある時間のシステムの状態を推定する数学的アルゴリズムである。ACTS では、finding と fitting で用いられる。カルマンフィルターを使用したトラッキングでは、運動方程式によって磁場中の荷電粒子の運動を計算し、次の検出層における粒子の位置を予測する。そして、予測の不定性と測定の不定性で重み付けをしながら実際の測定位置と比較し、その検出層での推定位置を更新する。これを各層で繰り返

していく事により、粒子の軌道（トラックパラメーター）を求めている。

カルマンフィルターは、Prediction と Filtering に分けられ、これらは各ステップで交互に行われる (図 2.6)。以下の表記法は [14] にならっている。

Prediction

現在のステップ i（i 番目の検出層）での情報に基づいて、ステップ $i+1$ の状態ベクトルと共分散行列（推定の不確かさ）を予測する。ここで、文字の右下の i はステップ数 (時間に対応)、文字の右上の $-$ は予測値、$+$ は更新値、文字の上の $\bar{}$ は平均値を表す。

$$\hat{x}_i^- = \Phi_{i,i-1}\hat{x}_{i-1}^+, \quad \hat{x}_0^+ = \bar{x}_0 \tag{2.1}$$

$$P_i^- = \Phi_{i,i-1}P_{i-1}^+\Phi_{i,i-1}^T + S_i, \quad P_0^+ = P_0 \tag{2.2}$$

ここで、

$\hat{x}_i^-$ ：ステップ i での予測された状態ベクトル（今回はトラックパラメーター）

$\hat{x}_{i-1}^+$ ：ステップ $i-1$ での更新された状態ベクトル

P_i^- ：ステップ i での予測された共分散行列

P_{i-1}^+ ：ステップ $i-1$ で更新された共分散行列

$\Phi_{i,i-1}$ ：ステップ $i-1$ からステップ i への状態遷移行列（次の検出層へのトラックパラメーターの変化を線形近似して、行列で表したもの。今回は磁場中の運動方程式を数値的に解いた結果に対応する）

S_i ：ステップ i における多重散乱などによるノイズ行列

である。

Filtering

ステップ $i+1$ での測定値を予測に組み込み、状態ベクトルと共分散行列を更新（抽出、filter）する。

$$\hat{x}_i^+ = \hat{x}_i^- + K_i(y_i - H_i\hat{x}_i^-) \tag{2.3}$$

$$P_i^+ = P_i^- - K_iH_iP_i^- = (I - K_iH_i)P_i^- \tag{2.4}$$

ここで K_i は最適カルマンゲインで、以下のように表せる。ここで最適とは、推定誤差を最小化するという意味である。

$$K_i = P_i^-H_i^T(H_iP_i^-H_i^T + R_i)^{-1} \tag{2.5}$$

y_i ：ステップ i における測定（観測）ベクトル（検出器の位置座標）

H_i　：ステップ i における状態空間から測定（観測）空間へのマッピング行列（状態 x と測定値 y はそのままでは比較できないので、変換行列 H によって同じ空間に変換している）

R_i　：ステップ i における測定ノイズの共分散 (検出器の位置分解能)

カルマンゲインは予測と測定のどちらを信頼するかを、それらの誤差の大きさに基づいて決定する。予測誤差 $P_i^- >$ 測定誤差 R_i の場合、カルマンゲイン K は 1 に近づき、予測値は強い修正を受けることになるのに対し、予測誤差 $P_i^- <$ 測定誤差 R_i の場合、カルマンゲイン K は 0 に近づき、予測値は弱い修正を受ける。

χ^2 は以下の式で計算する。

$$r_i = y_i - H_i \hat{x}_i^+ \tag{2.6}$$

$$\chi_+^2 = r_i^T [(I - H_i K_i) R_i]^{-1} r_i \tag{2.7}$$

ここで、r_i は、ステップ i における残差（residual）である。

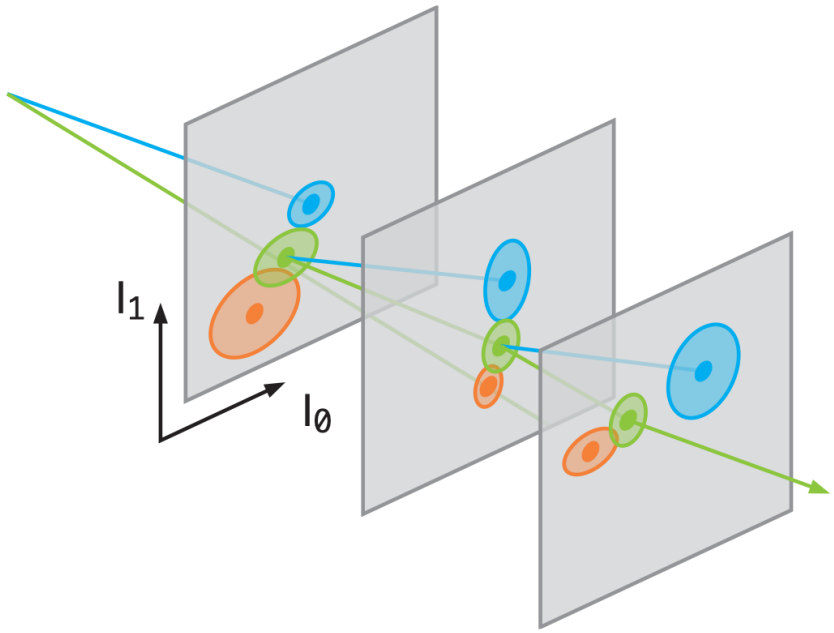


図 2.6: カルマンフィルターのイメージ図 [12]。青が予測値、赤が観測値、緑が予測値と観測値を考慮した更新値を示す。各円の大きさは予測値や観測値の誤差の大きさを表し、これらの大きさによって、予測と観測のどちらに重みを置くかが決まる。また、各平面は検出層を表す。

飛跡再構成におけるカルマンフィルターの数値計算上の問題点

ここでの議論のために、半正定値行列の定義を以下に示す。

実対称行列 A が任意の実ベクトル $\vec{x}$ に対して、内積 $(\vec{x}, A\vec{x})$ が、

$$(\vec{x}, A\vec{x}) \geq 0 \tag{2.8}$$

を満たすとき、A を半正定値行列と呼ぶ。また共分散行列は一般に半正定値行列であることが知られている。また、任意の正定値行列に対して、任意の正則行列 B に対して

$$BAB^T \tag{2.9}$$

も半正定値行列である。

共分散行列は半正定値行列である。しかし、以上で紹介した形式では、式 (2.4) が式 (2.9) の形で変換されておらず、P_i^+ の半正定値は失われてしまう可能性がある。P_i^+ の半正定値が失われることは、共分散行列として計算を進めていく上で問題が生じる。例えば、(半) 正定値性が失われることで、本来は正であるはずの χ^2(式 (2.7)) が負になることがある。これが避けられているカルマンフィルターの式について，第 4 章で議論する。

第 3 章

飛跡再構成の並列化と計算速度および計算精度

飛跡再構成は並列化可能なアルゴリズムで、並列化による処理の高速化が期待される。本論文では、特に GPU を用いた飛跡再構成の並列化について論じる。

3.1 GPU (Graphics Processing Unit)

GPU は画像処理を目的として開発された演算処理装置である。

CPU (Central Processing Unit) との違い

CPU と GPU の仕様の比較の 1 例として、2017 年時点でデスクトップ向け PC 向けのハイエンド CPU である Intel Core i7 - 9700KF CPU と NVIDIA のゲーム向けのハイエンド GPU であった RTX2080 Ti の仕様を表 3.1 に示す [15]。表 3.1 から GPU がプロセッサコア数は 8 倍以上で同時に実行するスレッド数では 500 倍以上と圧倒的に並列度が高いことが分かる。そのため、FP32(以下の「計算機における数値精度」を参照) のピーク演算性能は約 6.4 倍になっている。ただし、GPU コアの性能と CPU コアの性能は異なり、1 つの CPU コアは 1 つの GPU コアよりも大幅に高い性能を持っている。また、クロック周波数は、CPU が 3.6 GHz に対して、GPU では 1.350 GHz と CPU の 4 割弱となっている。

CPU は汎用の処理を行うためのプロセッサで、条件分岐を含んだ繰り返し処理などを処理を短い時間で実行できるように工夫されている。それに対して、GPU は大量の命令を (同期せずに) 並列処理することで、全体としての処理時間を短縮するという考えの基に作られている。そのため、GPU は並列に実行可能な処理に対しては高い性能を発揮するが、1 つの仕事を実行する速度はクロック周波数が低い分遅くなる。また、CPU のコアは 1 つの仕事を高速に実行するための機能（分岐予測等）が用意されているが、GPU はそのような工夫はなく命令を順番に実行していく。よって、1 つの仕事を処理するのに要する時間は、CPU の処理時間と比較して、クロック周期の比よりも長くなる。

表 3.1: CPU と GPU の仕様比較

	Intel Core i7 - 9700KF (CPU)	RTX2080 Ti (GPU)
コア数	8	68
スレッド数	8	4352
演算器数 (FP64)	4×8 (32)	68×2 (136)
演算器数 (FP32)	4×16 (64)	68×2 (4352)
クロック周波数 ※ () 内はターボ時	3.6 (4.9) GHz	1.350 (1.545) GHz

GPU は、一般に OS を動かすことができないため、CPU と組み合わせて使用することになる。具体的には、CPU で OS を走らせ、I/O やネットワークをサポートして、データの入力を行い、そのデータを GPU に供給する。そして GPU 上で大量の計算を行い、計算結果を再度 CPU に戻して、ディスク等の I/O に書き出したり、ネットワークを用いて外部に送るなどして使用する [15]。

科学技術計算への応用

画像処理の際に行われる 3 次元座標変換では行列の乗算が用いられるため、GPU は行列の乗算に優れている。この点に着目され、GPU が科学技術計算でも用いられるようになっていった。画像処理以外の計算への応用を目指して開発された GPU は、GPGPU(General Perpose GPU) と呼ばれる。最近では GPU の用途が広がってきたこともあり、単に GPU と呼ぶことが一般的になっている。本論文でも以降は GPGPU を区別せずに GPU と呼ぶ。

計算機における数値精度

現在、浮動小数点数の形式には、IEEE(Institute of Electrical and Electronics Engineers) の規格検討部門が定めた IEEE 754-2008 という規格が使用されている。浮動小数点数は、式 (3.1) のように定義され、Sign(符号、S)、Exponent(指数、Exp)、Fraction(仮数、Frac) と Bias から成る。

$$(-1)^{\mathrm{S}} \times 2^{\mathrm{Exp}-\mathrm{Bias}} \times 1.\mathrm{Frac} \tag{3.1}$$

IEEE 754 規格では、16 ビット長の半精度 (FP16)、32 ビット長の単精度 (FP32)、64 ビット長の倍精度 (FP64)、128 ビット長の 4 倍精度 (FP128) が定義されているが、本論文では 32 ビット長の単精度と 64 ビット長の倍精度を用いる (図 3.1)。仮数部の 10 進数での精度 (有効桁数) は、単精度が 6 桁、倍精度が 15 桁である。

IEEE 754 規格は全体のビット数が長くなるにつれて、Exp のビット数と Frac のビット数の両方が長くなっており、精度の高い形式の方が表せる数値の精度が高くなることに加えて、表現可能な数値の範囲も広くなる表現形式になっている。

グラフィックスでは一般に 32 ビット長の単精度の数値で十分であるが、多くの科学技術計算では、計算誤差を小さくするために 64 ビット長の倍精度演算を用いる。

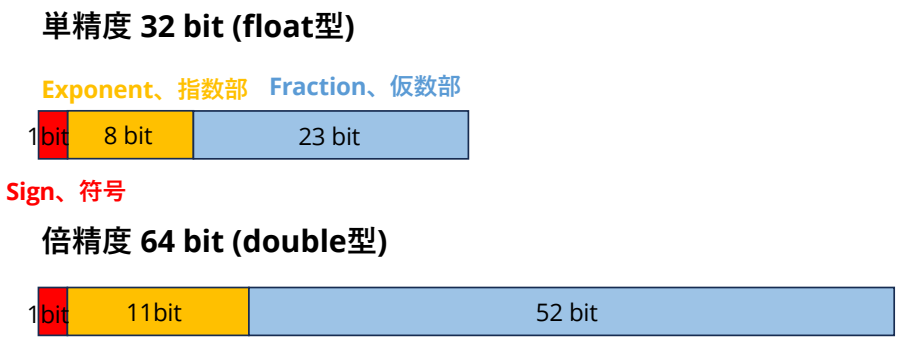


図 3.1: 浮動小数点数の形式。仮数部の 10 進数での精度 (有効桁数) は、単精度が 6 桁、倍精度が 15 桁である。

3.2 ACTS の GPU における実装

2.1 節で述べたように、ACTS はスレッドセーフ、つまり並列処理可能なように設計されている。現在、ACTS を並列化することを目的とした ACTS parallelization project が進行中であり、その中の一つに traccc と呼ばれる GPU 上で飛跡再構成を行うプロジェクトがある。traccc では、seeding における各 seed や finding におけるトラック候補を並列に計算している。traccc は CUDA や CYCL 等の様々な GPU コンピューティングプラットフォームで開発されている。traccc は C++ プログラミング言語をベースとした、GPU 向けの飛跡再構成ライブラリである。CUDA は最も広く使用されている GPU コンピューティングプラットフォームであり、NVIDIA 製の GPU で動作する。CYCL は AMD 製や Intel 製の GPU でも動作するプラットフォームである。本研究では CUDA の traccc コードを使用している。traccc は、目標の一つとして ATLAS Event filter での採用を掲げているが、あらゆるタイプの検出器に対応可能なように開発されている。traccc は現在、メモリ不足等で止まることなく最後まで動作する。traccc は飛跡再構成全体をカバーしているが、traccc 再構成アルゴリズムの一部（例えば、seeding だけ）を取り出して、他のアルゴリズムは CPU で動かすという運用方法も行われている。今回は、seeding, finding, fitting すべてを並列化するものを試した。

3.3 本研究の目的

上記のように、飛跡再構成の高速化には GPU は有利である可能性がある。ただし、GPU の演算機は単精度コアが多く、倍精度コアを積んだ GPGPU は高価であり、大規模な計算能力を必要とする ATLAS 実験での計算は、できるだけ単精度で行いたい。本研究は、GPU による並列化でどのように速度が向上するか、また GPU による計算精度の飛跡再構成に与える影響を調べ、その

問題点に対処することを目的とする。

3.4 ACTS の性能評価に用いた環境

3.4.1 使用したプロセッサ

表 3.2 に、本研究で使用した CPU と GPU の仕様を示す。

表 3.2: 使用した CPU と GPU

	Intel Xeon Gold 5318Y (CPU)	NVIDIA RTX A6000 (GPU)
コア数	24	10752
理論演算性能 (FP32)	3.23 TFLOPS	38.7 TFLOPS
理論演算性能 (FP64)	1.61 TFLOPS	0.60 TFLOPS
クロック周波数	2.10 GHz	1.410 GHz
発売年	2021 年	2020 年

3.4.2 使用した検出器

今回使用した Open Data Detector (ODD) と呼ばれる高輝度 LHC 向けの汎用検出器モデルを図 3.2 に示す (ただし、図 3.7 のみ ITk ジオメトリを用いている)。図 3.3 に、ODD レイアウトの概略図を示す。ODD は、ピクセル検出器と二種類のストリップ検出器から構成され、内側からピクセル検出器、ショートストリップ検出器、ロングストリップ検出器からなる。また ODD は、高輝度 LHC の ATLAS 内部飛跡検出器 ITk に着想を得ている。

図 3.4 は、ODD で得られるヒット点の数の分布である。この図から η のほとんど全域で 12 点以上のヒットが得られることが分かる。

3.4.3 使用した物理サンプル

今回は、物理サンプルとして LHC 実験で広く用いられている $t\bar{t}$ のパイルアップ 200 のサンプルを使用した。図 3.5 にクォーク対の衝突によって生成される $t\bar{t}$ 事象のファインマン・ダイアグラムを示す。今回は 10 イベントの $t\bar{t}$ サンプルを使用した。この $t\bar{t}$ の分布を図 3.6 に示した。1 イベントに含まれる荷電粒子の数は 2800 程度である。

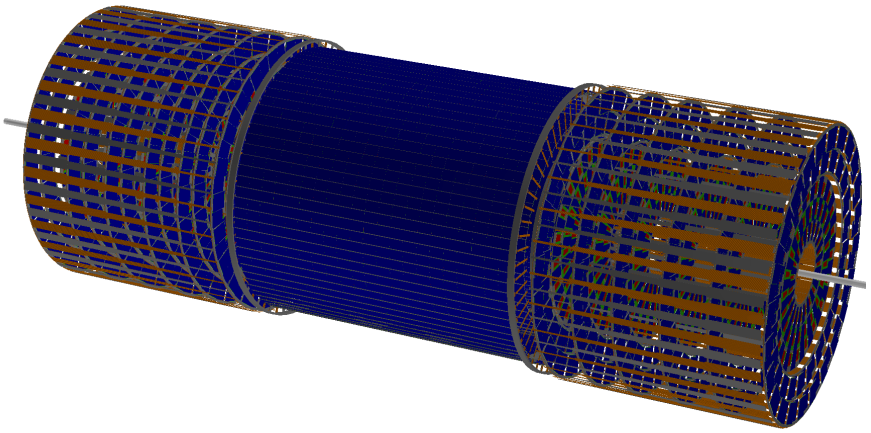


図 3.2: ODD の全体像

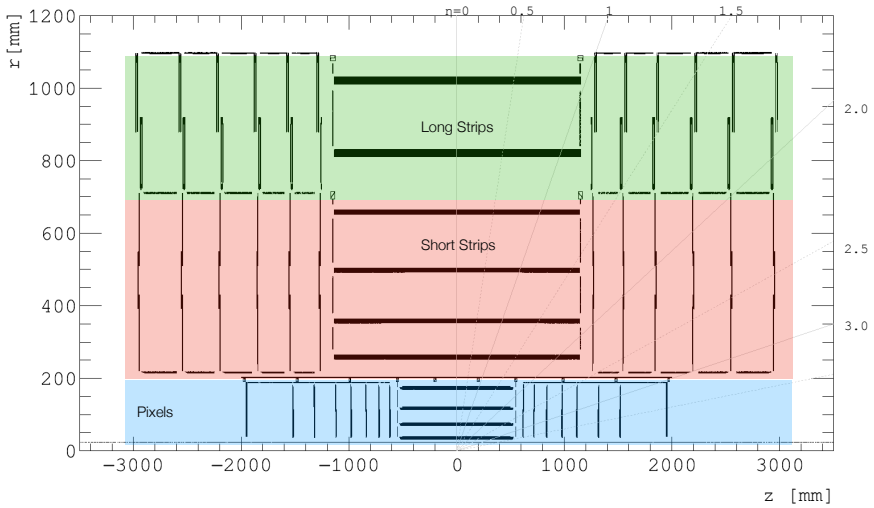


図 3.3: ODD のレイアウト。座標系は図 1.7 の ATLAS 検出器と同じである。

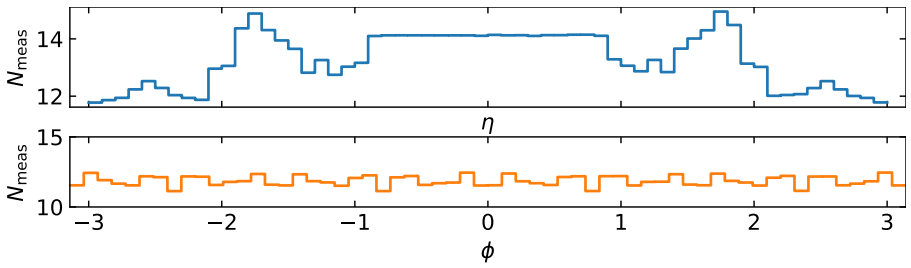


図 3.4: ODD のヒット点の分布 [16]

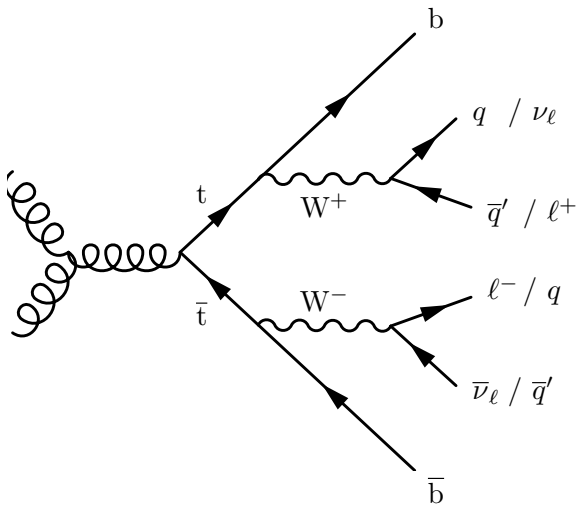


図 3.5: 主な $t\bar{t}$ 生成事象のファインマン・ダイアグラム

3.5 GPU による並列化と計算速度

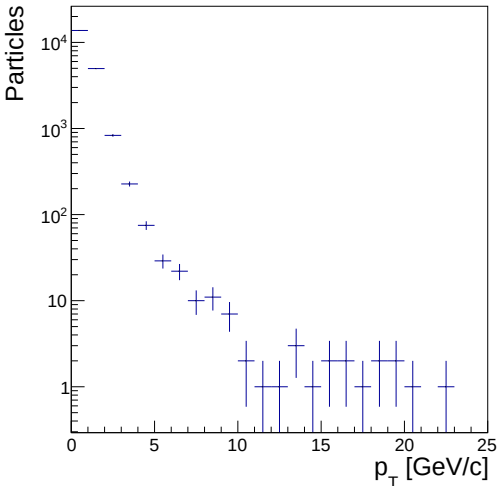
飛跡再構成を GPU に実装する大きな利点の一つは、計算処理の高速化である。ここではその計算時間を検証する。

まず、図 3.7 に、CPU の実行時間に対する GPU の実行時間の比を示した。GPU で、1 イベント当たり 10, 100, 1000 トラック（パイルアップなし）を含む事象をそれぞれ 100 イベント実行した時間を、CPU が同じ処理をしたときの実行時間に対する比として表した。ただし、ここで使用しているトラックは、トイトラック (Toy track) と呼ばれるもので、質量 0 で散乱を起こさない、電荷-1 の仮想粒子である。この図から、トラック数が増加するにつれて、GPU が CPU に比べて優位になっていることが分かる。また、GPU 同士で比較すると、倍精度よりも単精度の方が高速であることがわかる。これは、一般に GPU に含まれている演算器は単精度 (FP32) が大きい割合を占めており、倍精度演算器が少ないためである。今回使用した NVIDIA RTX A6000 GPU は、単精度演算器と倍精度演算器の割合は 32:1 である。

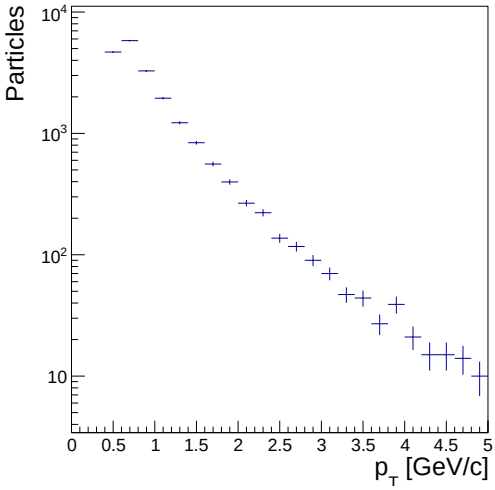
次に図 3.3 と表 3.3 に、LHC 実験における飛跡再構成のベンチマークとして広く用いられる $t\bar{t}$ 事象 (toy track ではない) における GPU の単精度と倍精度の飛跡再構成の処理時間の比較を示した。これらの図や表からも、単精度 GPU の方が倍精度 GPU よりも高速であることが分かる。

3.6 GPU の単精度計算に伴う問題点

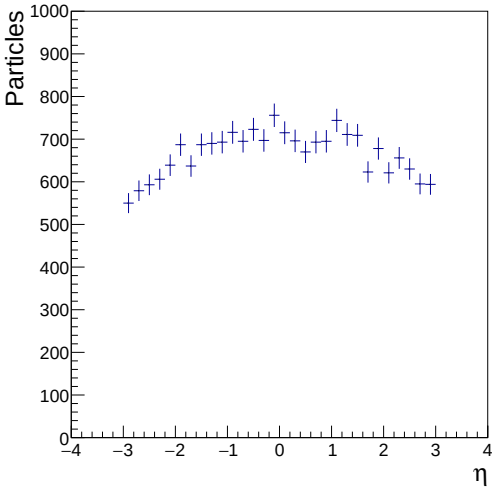
前節では、単精度で処理が高速になるという利点について説明した。しかし、GPU を単精度で使用する場合、数値精度の不足によって計算性能が低下する場合がある。今回は計算性能低下の例



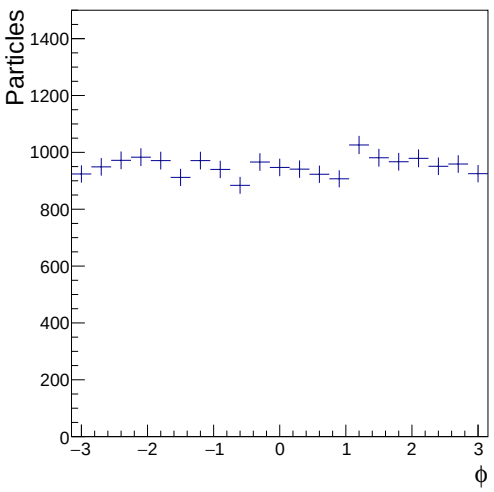
(a) p_T 分布 ($0 < p_T < 25\,\mathrm{GeV}$)



(b) p_T 分布 ($0 < p_T < 5.0\,\mathrm{GeV}$)



(c) η 分布



(d) ϕ 分布

図 3.6: 真の粒子の分布。$p_T > 500\,\mathrm{MeV}$ のカットが掛けられている。

表 3.3: 単精度と倍精度の処理時間。合計時間には、seeding、finding、fitting 以外の時間（ファイルの読み込み時間等）も含まれている。

手法	Seeding [ms]	Finding [ms]	Fitting [ms]	Total [ms]
単精度	$(11 \pm 2) \times 10$	$(68 \pm 4) \times 10$	263 ± 6	$(944 \pm 7) \times 10$
倍精度	$(77 \pm 2) \times 10$	$(279 \pm 4) \times 10$	$(105 \pm 1) \times 10$	$(1306 \pm 5) \times 10$

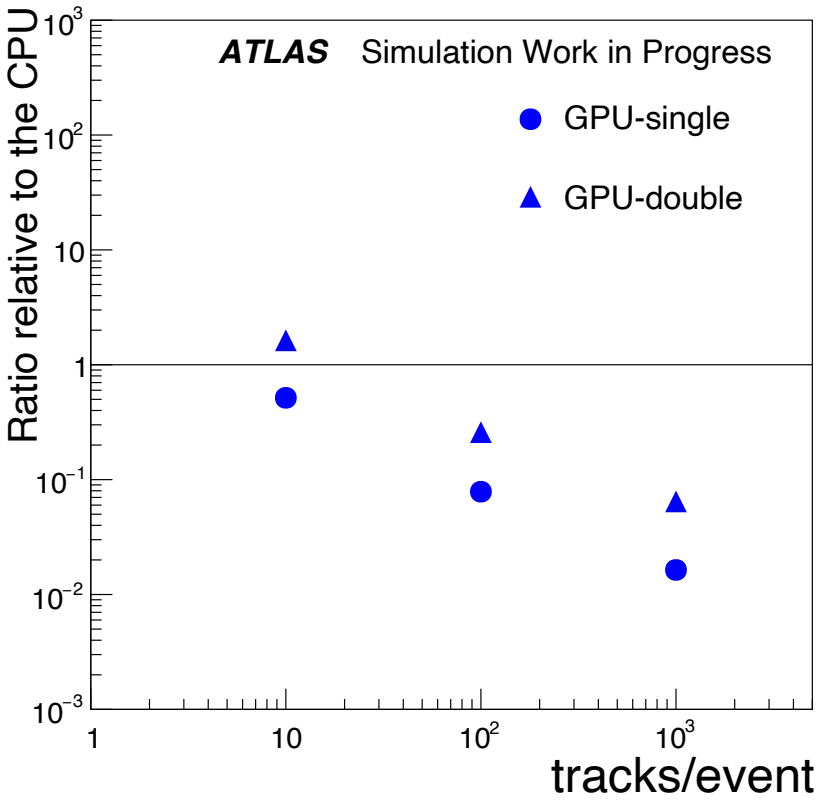
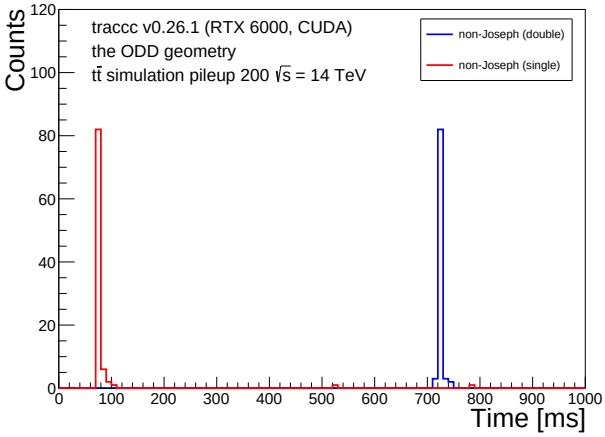


図 3.7: 飛跡再構成の処理時間の比較。横軸は 1 イベント当たりのトラック数、縦軸は CPU の実行時間に対する GPU の実行時間である。CPU と GPU で、1 イベント当たり 10, 100, 1000 トラック（パイルアップなし）を含む事象をそれぞれ 100 イベント実行した。ただし、ここで使用しているトラックは、トイトラック (Toy track) と呼ばれるもので、質量 0 で散乱を起こさない、電荷-1 の仮想粒子である。このグラフでのみ検出器は ITk ジオメトリを用いている。また、$10\,\mathrm{GeV} < \mathrm{p} < 100\,\mathrm{GeV}$, $-4.0 < \eta < 4.0$ の範囲で生成した。

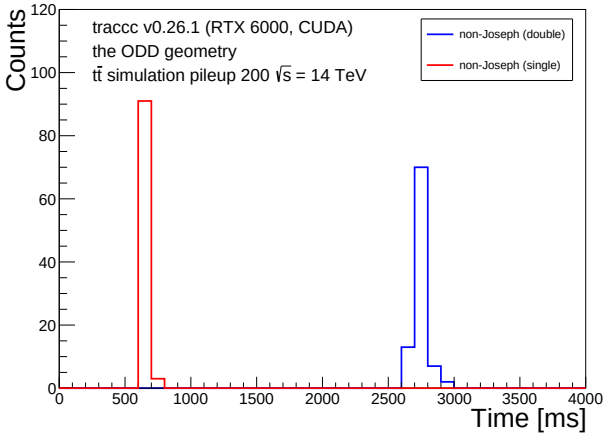
として、表 3.4 に GPU を用いた飛跡再構成のカルマンフィルターにおける正の値である χ^2 が負となって、その飛跡候補の finding あるいは fitting が停止するエラー数を、計算精度別に示した。この表に示されるように、単精度ではカルマンフィルターにおいて $\chi^2 < 0$ のエラーが起きやすい。2.2 節で述べたように、χ^2 は最適カルマンフィルターで負になる可能性があるが、一般に倍精度演算器を多く持つ GPU は高価であるため、3.3 節高速化に加えて経済的な観点からも倍精度ではなく単精度を可能な限り用いたい。

Matching の定義

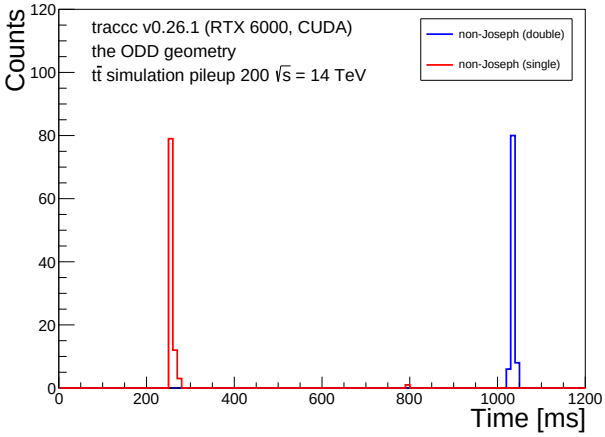
finding や fitting などの結果見つかったトラックに対し、各トラックに含まれているヒットがどの粒子由来のものであるかの割合を計算することによってシミュレーションのもととなる真の粒子情報との対応をとり、ある再構成された飛跡が真の粒子由来である可能性が高いものを取り出し



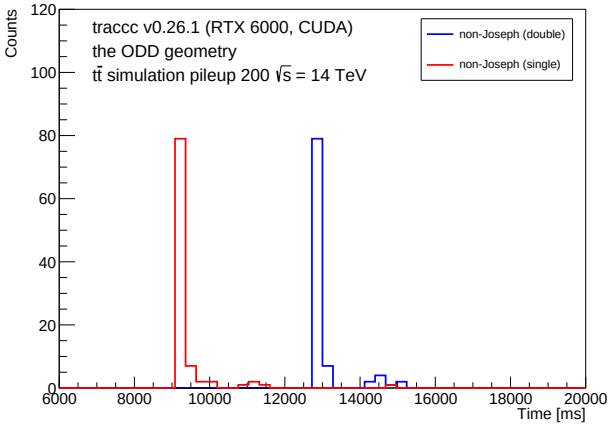
(a) seeding



(b) finding



(c) fitting



(d) 全体

図 3.8: 単精度と倍精度の処理時間の比較

表 3.4: GPU を用いた飛跡再構成のカルマンフィルターにおけるエラー ($\chi^2 < 0$) 数の比較。$t\bar{t}$ のパイルアップ 200 のサンプル、10 イベント（3.4 節）で生じた当該エラー数である。

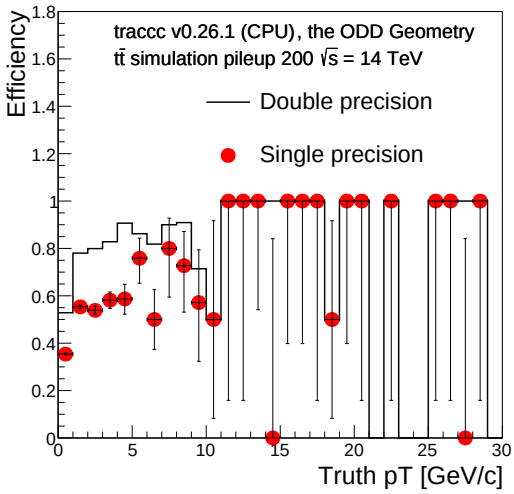
	単精度	倍精度
Finding でのエラー数	480	0
Fitting でのエラー数	2032	0
Fit を行ったトラック数	43,279	43,304

て、それについて再構成の効率、真の粒子のトラックパラメーターとの違い（残差）などを評価している。以下でこの matching について説明する。

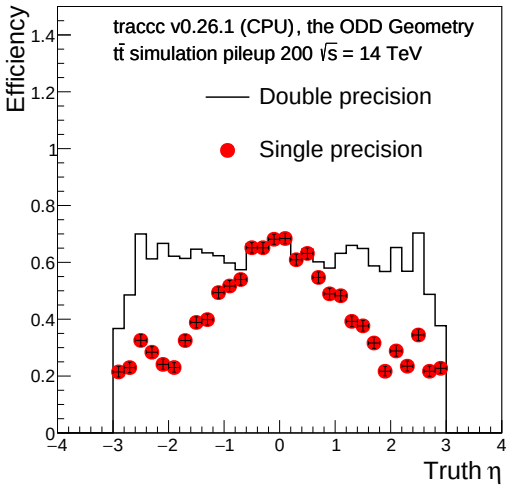
あるトラックに含まれるヒットの集合（集合に含まれるヒットを N_t 個とする）に対して、各ヒットがどの粒子由来であるかを調べ、最も多くのヒットを残した粒子 (major particle) を特定する（そのヒットの個数を N 個とする)。また、その major particle を構成するヒットの数を N_p 個とする。このとき、$N > N_t/2$ かつ $N > N_p/2$ を満たす場合に、そのトラックは粒子と match したとみな。この matching 法は、double matching と呼ばれる。

Fitting における efficiency

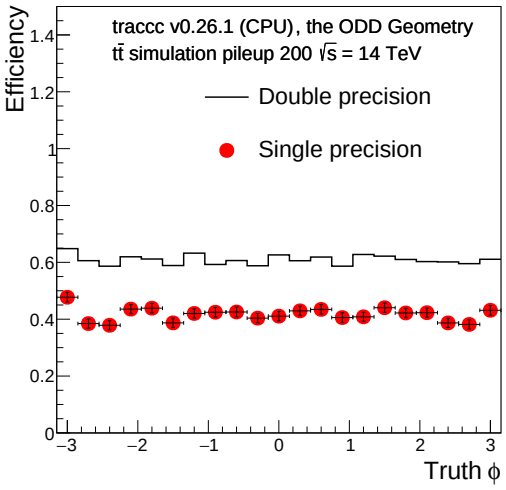
Fitting における単精度と倍精度の efficiency の比較を図 3.9 に示す。ここで、efficiency の分母となるのは、図 3.6 に示した、すべての真の粒子 ($p_T > 500\,\mathrm{MeV}$ のカットが掛けられている) であり、分子となるのは、fitting に成功した (後述する、fitting の過程でエラーが起きずに成功したもの) かつ、finding で match した飛跡に対応する真の粒子である（4.2.2 節)。このグラフから、単精度と倍精度には efficiency に大きな差があること分かる。前述のことと合わせると、カルマンフィルターでの計算エラーの低減が、この効率の改善につながると考えた。次章では，この低減手法とその飛跡再構成に与える影響について述べる。



(a) p_T 分布



(b) η 分布



(c) ϕ 分布

図 3.9: Fitting における単精度と倍精度の efficiency の比較。Efficiency の分子に加える条件は、前節の double matching と fitting に成功したことである。

第 4 章

カルマンフィルターにおける共分散行列の更新手法の改良

前章では、GPU での計算の高速化に必要な倍精度から単精度計算への移行により、fitting においてカルマンフィルターの段階で失敗が起きること、その一つが共分散行列の半正定値性が単精度化により失われ、χ^2 が負になる場合であり、それが多く見られることがわかった。この章では、共分散行列の更新手法式 (2.4) を改良することによってこれらのエラーを削減し、fitting の成功確率を向上させることを目指す。また、それが飛跡再構成全体、特に fitting の結果にどう影響を及ぼすかを見る。

4.1 共分散行列の更新手法の改良

まず以下で、最適カルマンフィルターにおける共分散行列の更新式（式 (2.4)）を導出する過程から、共分散のより一般的な形式（後述の式 (4.5)）を導く。そのうえで、式 (2.4) を導出する過程から、式 (2.4) はこの一般的な形式が式 (2.5) を満たす場合に成立することを示す。この導出の準備のため、$\hat{x}_i^+$ と y_i の定義を以下に再掲する。式 (2.3)

$$\hat{x}_i^+ = \hat{x}_i^- + K_i(y_i - H_i\hat{x}_i^-) \tag{2.3}$$

のかっこの中に対応する式

$$y_i = H_i x_i + v_i \tag{4.1}$$

は，ある測定 y_i が，そこにおける状態ベクトルと測定ノイズから成り立つことを示す。ここで，各変数は以下の通り定義される。

$\hat{x}_i^+$ ：ステップ i での更新された状態ベクトル
$\hat{x}_i^-$ ：ステップ i での予測された状態ベクトル（今回はトラックパラメーター）
K_i ：カルマンゲイン (ここでは最適カルマンゲイン式 (2.5) を仮定しない)
y_i ：ステップ i における測定（観測）ベクトル（検出器の位置座標）

H_i　　：ステップ i における状態空間から測定（観測）空間へのマッピング行列（状態 x と測定値 y はそのままでは比較できないので、変換行列 H によって同じ空間に変換している）

v_i　　：測定ノイズ

ここで、測定ノイズ v_i はガウス分布に従うと仮定する ($E[v_i] = 0, \quad E[v_i v_j^T] = R_{ij}\delta_{ij}$($\delta_{ij}$ はクロネッカーのデルタ))。

ステップ i で更新された共分散行列 P_{i-1}^+ は、

$$P_i^+ = E[(x_i - \hat{x}_i^+)(x_i - \hat{x}_i^+)^T] \tag{4.2}$$

と書ける。これに式 (2.3) と式 (4.1) を代入すると、

$$P_i^+ = E[\{x_i - (\hat{x}_i^- + K_i(y_i - H_i\hat{x}_i^-))\}\{x_i - (\hat{x}_i^- + K_i(y_i - H_i\hat{x}_i^-))\}^T] \tag{4.3}$$

となる。$x_i - \hat{x}_i^-$ に対して因数分解を行い、その後、期待値 E 内を展開し、その際 $I - K_iH_i$ は確率変数ではないことに注意すると、

$$\begin{aligned} P_i^+ &= E[\{(I - K_iH_i)(x_i - \hat{x}_i^-) - K_iv_i\}\{(I - K_iH_i)(x_i - \hat{x}_i^-) - K_iv_i\}^T] \\ &= (I - K_iH_i)E[(x_i - \hat{x}_i^-)(x_i - \hat{x}_i^-)^T](I - K_iH_i)^T \\ &\quad - (I - K_iH_i)E[(x_i - \hat{x}_i^-)v_i^T]K_i^T - K_iE[v_i(x_i - \hat{x}_i^-)^T](I - K_iH_i)^T \\ &\quad + K_iE[v_iv_i^T]K_i^T \end{aligned} \tag{4.4}$$

となる。

ステップ i での予測された共分散行列は、$P_i^- := E[(x_i - \hat{x}_i^-)(x_i - \hat{x}_i^-)^T]$、ステップ i における測定ノイズの共分散行列は、$R_i := E[v_iv_i^T]$ で定義される。また、状態ベクトル x_i と測定ノイズ v_i が独立である ($E[(x_i - \hat{x}_i^-)v_i^T] = E[v_i(x_i - \hat{x}_i^-)^T] = 0$) と仮定すると、

$$P_i^+ = (I - K_iH_i)P_i^-(I - K_iH_i)^T + K_iR_iK_i^T \tag{4.5}$$

式 (4.5) で表される共分散行列の更新式を Joseph 形式と呼ぶ。

そして式 (4.5) に、最適カルマンゲイン (推定誤差を最小にする) の条件式 式 (2.5) を代入すると、式 (2.4) が導かれることを確認する。まず、式 (4.5) を展開すると、

$$\begin{aligned} P_i^+ &= (I - K_iH_i)P_i^-(I - K_iH_i)^T + K_iR_iK_i^T \\ &= (I - K_iH_i)P_i^- + P_i^-H_i^TK_i^T - K_i(H_iP_i^-H_i^T + R_i)K_i^T \end{aligned} \tag{4.6}$$

となる。ここに最適カルマンゲイン (推定誤差を最小にする) の条件式 式 (2.5) を代入すると、

$$
\begin{aligned}
P_i^+ &= (I - K_i H_i) P_i^- + P_i^- H_i^T K_i^T - P_i^- H_i^T (H_i P_i^- H_i^T + R_i)^{-1} (H_i P_i^- H_i^T + R_i) K_i^T \\
&= (I - K_i H_i) P_i^- + P_i^- H_i^T K_i^T - P_i^- H_i^T K_i^T \\
&= (I - K_i H_i) P_i^-
\end{aligned}
\tag{4.7}
$$

以上のように式 (2.4) が導かれることから、式 (2.4) は式 (2.5) の最適カルマンゲインの条件を、共分散の定義である式 (4.5) に代入することによって求まる。Joseph 形式に対して、式 (2.4)(式 (4.7)) を本論文では non-Joseph 形式と呼ぶこととする。

しかし、以下に示すように，最適カルマンゲインを仮定した共分散行列式 (2.4) は，半正定値性が保証されず，χ^2 が負となり，飛跡再構成がカルマンフィルターの段階で止まってしまう可能性がある。

4.1.1 Joseph 形式の利点

本節では、Joseph 形式 $(I - K_i H_i) P_i^- (I - K_i H_i)^T + K_i R_i K_i^T$ の性質について説明する。まず、2.2 節で説明したように、non-Joseph 形式では、半正定値性が失われやすい。それに対し、Joseph 形式は対称性と加算的な形のため、式 (2.4) に比べて、非正定値となりにくい (ただし、完全にではない)[14]。これは、共分散行列として重要な点である。以下では、さらに Joseph 形式の利点を説明する。

最適カルマンゲインの誤差が共分散行列に与える影響

最適カルマンゲイン K が数値計算の誤差などにより真の値からずれた場合、Joseph 形式のほうがその誤差が共分散に与える影響が小さいことを以下に示す。K を δK だけ変化させることを考えると、non-Joseph 形式 $P_i^+ = (I - K_i H_i) P_i^-$ では、

$$
\delta P^+ = -\delta K H P^- \tag{4.8}
$$

と δK の一次に比例した真の値からのずれが生じる。

それに対して、Joseph 形式 $(I - K_i H_i) P_i^- (I - K_i H_i)^T + K_i R_i K_i^T$ では、δK の二次の項を無視すると、

$$
\begin{aligned}
\delta P^+ &= -\delta K [R K^T - H P^- (I - K H)^T] + [K R - (I - K H) P^- H^T] \delta K^T \\
&= 0
\end{aligned}
\tag{4.9}
$$

δK の一次に比例したずれは生じない。

この点からも数値計算では、Joseph 形式が有利になると考えられる。これらのことから、カルマンゲインの形を仮定せずに導かれた Joseph 形式を使用することで、共分散行列の正定値性を確保することにより、計算途中のエラーを低減する。以下では、これにより求まる飛跡にどのような影響があるかを見る。

4.2 改良による飛跡再構成の性能評価

本節では、Joseph 形式の導入による飛跡再構成の性能を評価する。4.2.3 節の速度比較では CUDA コードを用いて GPU 上で動かしているが、それ以外の性能評価では、CUDA 用のコードにおいて ambiguity resolution が未実装のため、それ以外は CPU コードを用いている。今回これらの計算における GPU および CPU は，3.4 節で説明したように、NVIDIA RTX A6000 と Intel Xeon Gold 5318Y を使用した。

まず、カルマンフィルターにおけるエラー数の比較を表 4.1 と表 4.2 に示した。これらの表から、カルマンフィルターでは、χ^2 に関するエラーが生じることが分かる。ただし、倍精度の場合は全くエラーが生じていない。単精度における non-Joseph 形式と Joseph 形式の比較では、Joseph 形式の移行により、Finding では $\chi^2 = \infty$ のエラーはわずかに増加するものの、$\chi^2 < 0$ のエラー数は 480 回から 0 回と全くエラーが生じなくなった。また、Fitting においては、$\chi^2 = \infty$ のエラー数はほとんど変化しないものの、$\chi^2 < 0$ のエラー数は 2032 回から 326 回と約 84% 減少した。

表 4.1: Finding におけるカルマンフィルターにおけるトラック数及びエラー数の比較。$t\bar{t}$ のパイルアップ 200 のサンプル、10 イベント。θ, ϕ, q/p はトラックパラメーターを示す。なお、seed 数は同一環境、同一イベントの飛跡再構成であっても毎回異なり、それに伴って finding の出力数も毎回異なっていた。

	単精度 non-Joseph	単精度 Joseph	倍精度 non-Joseph
$\theta = 0$	0	0	0
$\phi = \infty$	0	0	0
$q/p = 0$	0	0	0
$\chi^2 < 0$	480	0	0
$\chi^2 = \infty$	2612	2638	0
Finding の入力トラック数 (seed 数)	349814	349597	349569
Finding の出力トラック数	175534	175485	175493

このエラーが再構成の効率やパラメーターの計算精度に与える影響を以下で見る。

4.2.1 Finding と Ambiguity resolution における飛跡再構成確率

図 4.1 は、finding で得たトラック候補に対する efficiency と、ambiguity resolution 後に得たトラック候補に対する efficiency を示したものである。Efficiency の定義は、3.6 節で定義したものと同じで、truth 飛跡とマッチしたトラックの再構成確率を示している。finding ではカルマンフィルターが使用されているため、Joseph 形式と non-Joseph 形式の間で efficiency に差が生じる可能性は考えられるが、今回の環境下では Joseph 形式と non-Joseph 形式による差は生じなかった。

表 4.2: Fitting におけるカルマンフィルターにおけるエラー数の比較。$t\bar{t}$ のパイルアップ 200 のサンプル、10 イベント。

	単精度 non-Joseph	単精度 Joseph	倍精度 non-Joseph
$\theta = 0$	0	0	0
$\phi = \infty$	0	0	0
$q/p = 0$	0	0	0
$\chi^2 < 0$	2032	326	0
$\chi^2 = \infty$	18352	18354	0
Fitting の入/出力トラック数	43279	43297	43303

また、ambiguity resolution においてフェイクトラックを取り除いた後の efficiency(図 4.1b、図 4.1d) に差が生じていないことは、どちらかの形式が finding において、より荷電粒子由来のヒットを見つけているわけではないということを意味している。

4.2.2 Fitting における性能評価

本節では、fitting における再構成指標を見ていく。

飛跡再構成確率

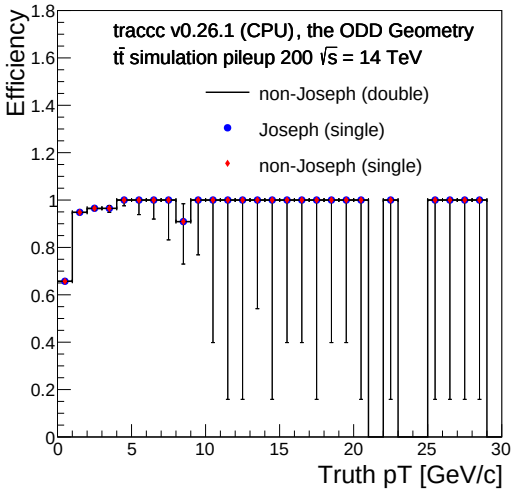
単精度における Joseph 形式と non-Joseph 形式の fitting の efficiency を図 4.2 に示す。図 4.2 から、Joseph 形式では efficiency が回復している、つまり、fitting によってトラックパラメーターを再構成できる確率が上がっていることが分かる。ただし、その確率は倍精度計算には及んでいない。これは、表 4.2 の結果と同様である。つまり、単精度 Joseph ではエラーが減少しているが、その結果が fitting の成功確率に現れていると考えられる。

Fitting の段階では、表 4.2 のカルマンフィルターエラーに加えて、各トラックが fit に成功したかどうかを示すフラグがある。表 4.3 に、それを示す。表 4.3 より、単精度 Joseph 形式では、実際に単精度 non-Joseph 形式よりも多くのトラックが fitting に成功することが分かる。これが図 4.2 にも現れている。特にトラック数の増加が大きいのは $|\eta|$ が 2 の付近である。

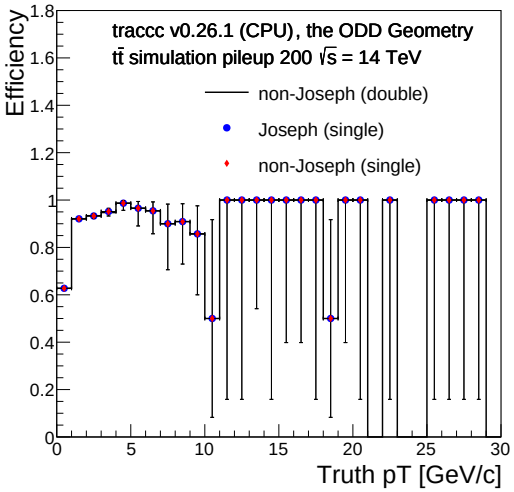
ここで、Smoothing とは、カルマンフィルターの prediction と filtering を逆方向 (外層から内層) に伝播するアルゴリズムである。prediction と filtering は、現在のステップより前の情報しか反映されないが、Smoothing によって、そのステップ以降の情報も取り入れる (ステップ $i+1$ 以降の情報を使用してステップ i の状態を更新する) ことができ、より精度の良い更新が可能になる。

トラックフィットにおける自由度 (ndf)

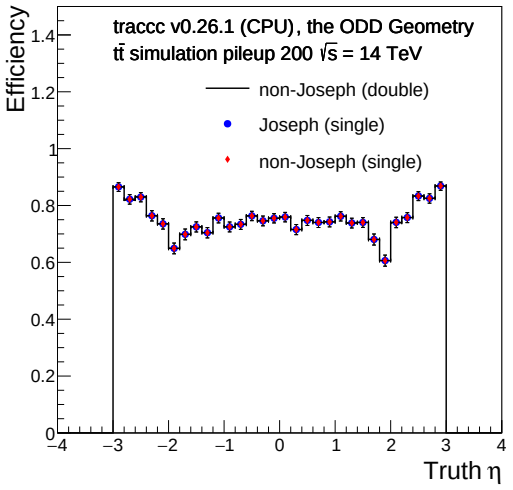
図 4.4 に、単精度 Joseph 形式と単精度 non-Joseph 形式と倍精度 non-Joseph 形式の再構成後のトラックの ndf の分布を示した。この図より単精度において、Joseph 形式の方が non-Joseph



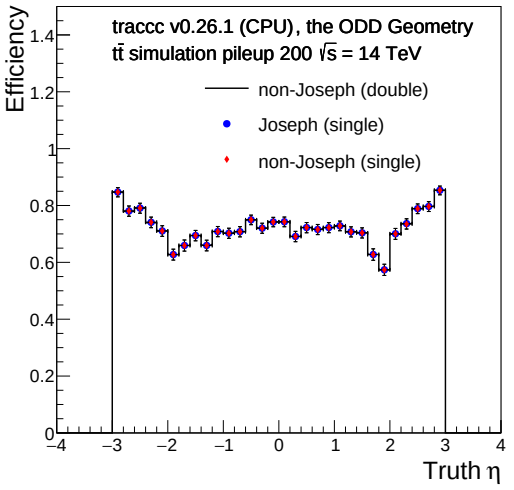
(a) p_T (finding)



(b) p_T (ambiguity resolution)

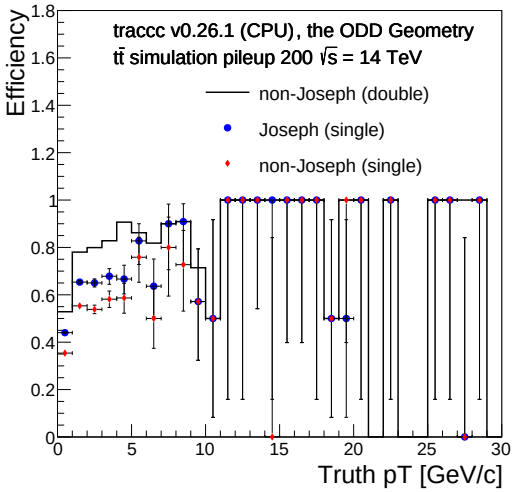


(c) η (finding)

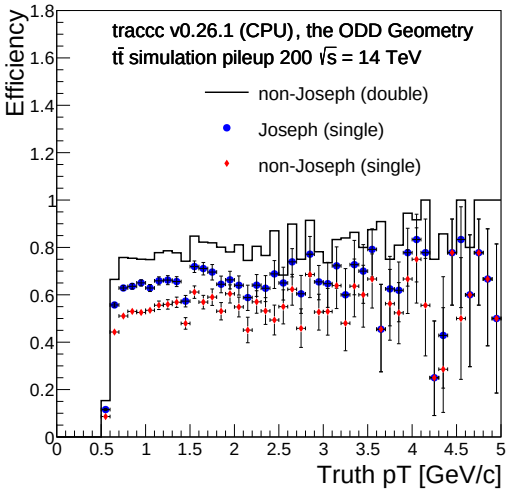


(d) η (ambiguity resolution)

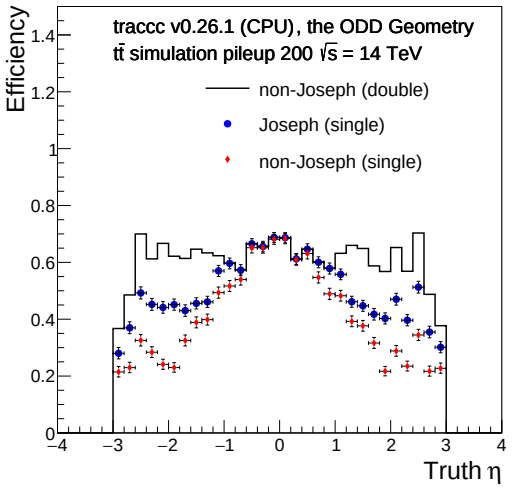
図 4.1: Finding, Ambiguity resolution における efficiency の比較 (単精度 Joseph vs 単精度 non-Joseph vs 倍精度 non-Joseph)



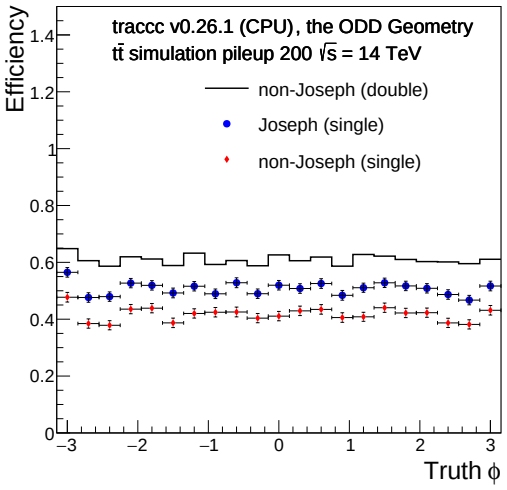
(a) p_T 分布 ($0 < p_T < 25\,\mathrm{GeV}$)



(b) p_T 分布 ($0 < p_T < 5.0\,\mathrm{GeV}$)



(c) η 分布



(d) ϕ 分布

図 4.2: 各手法の fitting における efficiency の比較 (単精度 Joseph vs 単精度 non-Joseph vs 倍精度 non-Joseph)。

表 4.3: Fitting の出力結果の比較。$t\bar{t}$ のパイルアップ 200 のサンプル、10 イベント。

	単精度 non-Joseph	単精度 Joseph	倍精度 non-Joseph
フィットに成功	17562	27700	36644
NDF <0(失敗)	13739	5961	6344
smoothing に失敗している ヒットがある (失敗)	1785	296	316
その他の失敗	10293	9340	0
Fitting の入/出力トラック数	43279	43297	43303

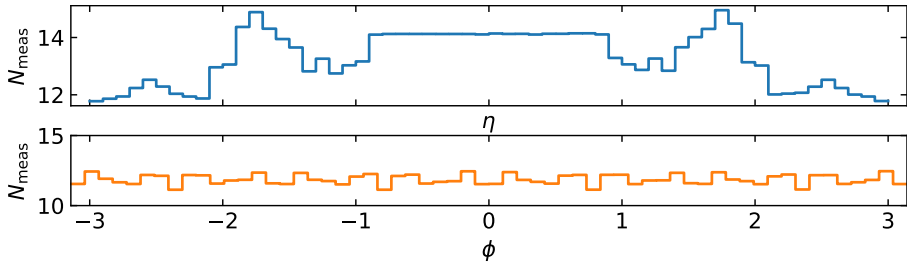


図 4.3: ODD のヒット分布 (再掲)。

形式よりも ndf の小さいトラックを多く再構成していることがわかる。また、図 4.3 より、$|\eta|$ が 2 の付近ではヒット数が少ないことから、図 4.2 で見られた $|\eta| \simeq 2$ での fit 確率の上昇は、このような NDF、つまりヒット数の少ないトラックの再構成確率が上がることを反映していると考えられる。

トラックパラメーターの残差分布

non-Joseph 形式と Joseph 形式の各トラックパラメーターの残差分布を図 4.5 に示す。ここで residual r は、fitting によって得られたパラメーター p_{fitted} と真の粒子のパラメーター p_{truth} を用いて

$$r \coloneqq p_{fitted} - p_{truth} \tag{4.10}$$

と定義する。

図 4.5 には、fitting に成功したかしないかにかかわらず各トラックのパラメーターの残差分布を実線で、また fitting に成功したパラメーターの分布を点で表している。Fitting に成功した場合のトラックパラメーターでは分布のテールが大幅に減っていることから、fitting に成功した場合はトラックパラメーターが更新され、精度が向上していることが読み取れる。これは ϕ 分布で顕著である。また、fitting に成功したトラックの計算手法および精度別の比較では、Joseph 形式への変

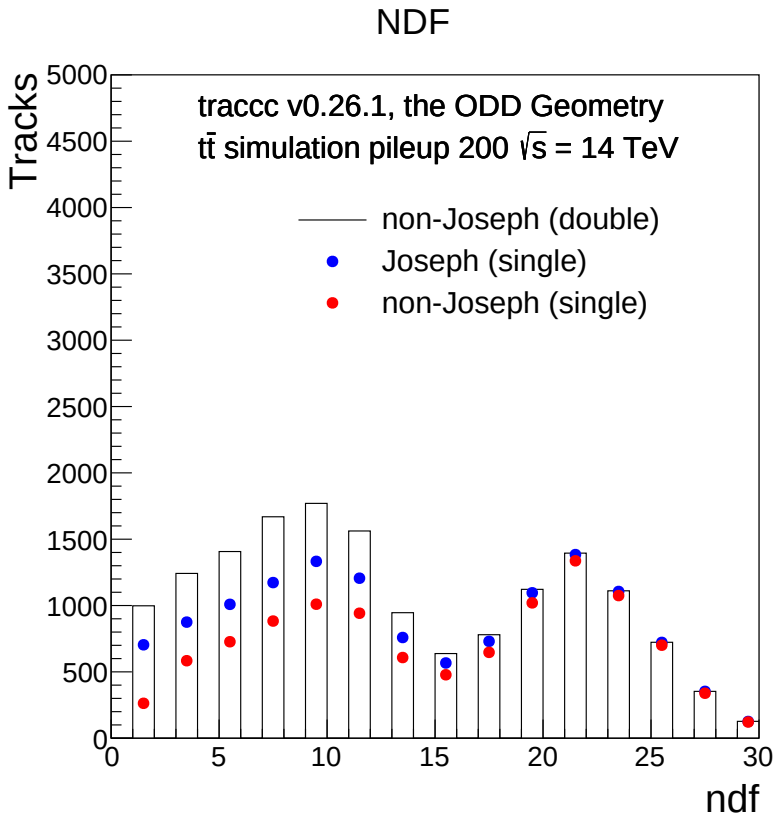
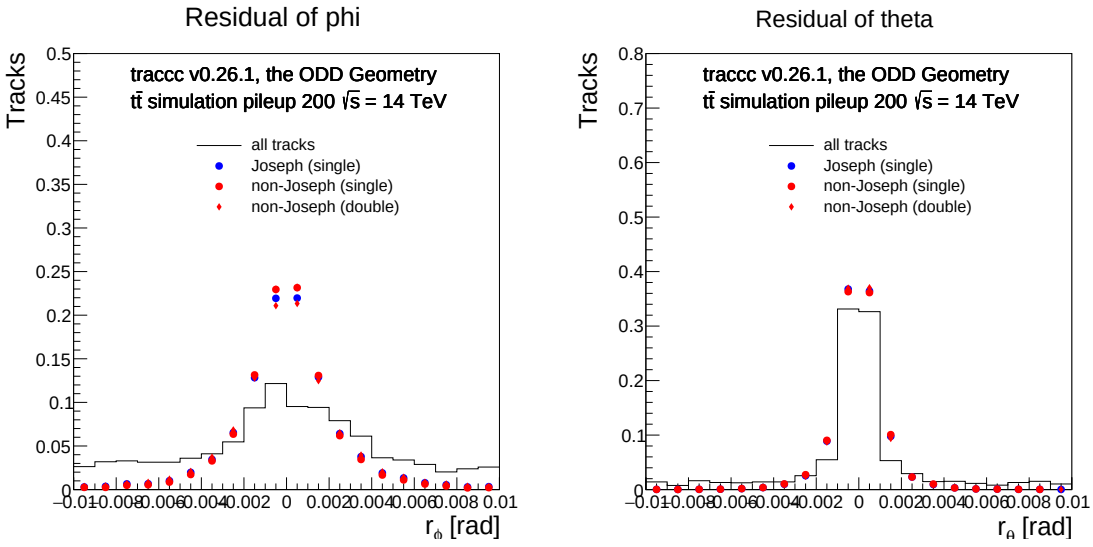
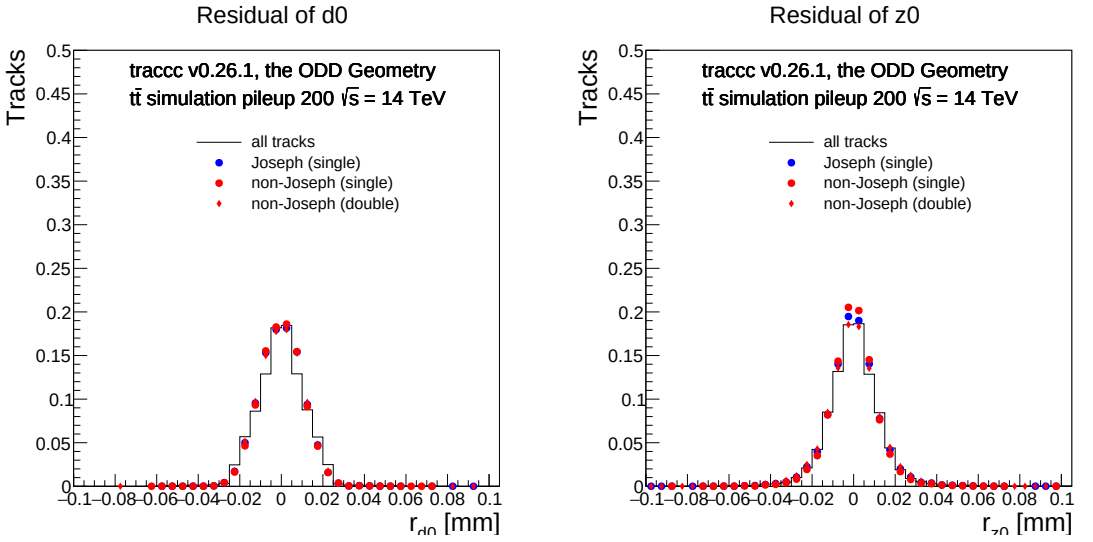


図 4.4: 単精度における Joseph 形式と non-Joseph 形式の再構成後のトラックの ndf の分布。ndf の値が 2 ずつ増えているのは、測定点 1 つが 2 次元平面である検出器上の点を表すことから、測定点一つにつき NDF が 2 つずつ増加するためである。

更によって、residual は変化していないことが分かる。



(a) ϕ (b) θ



(c) d_0 (d) z_0

図 4.5: 各手法のトラックパラメーターの residual の比較 (単精度 Joseph vs 単精度 non-Joseph vs 倍精度 non-Joseph)。各グラフの面積が 1 になるように規格化している。

4.2.3 速度比較

non-Joseph 形式が利用される理由の一つに計算量が比較的少ないことがあるが、Joseph 形式にすることによって、計算時間にどれほどの変化が生じるのかを調べた。速度比較では、CUDA のコードを使用し、GPU 上で動かしている。図 4.6 では、$t\bar{t}$ パイルアップ 200 のサンプルを 10 イ

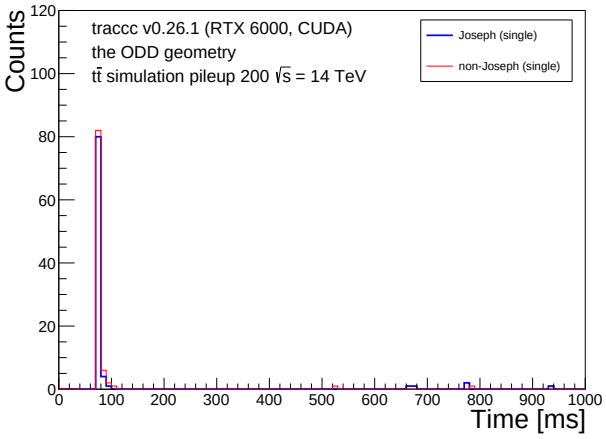
ベントを一つの run として、1 run のトラッキングに要する時間を計測したものである。ただし、トラッキング実行直後の run では計算機の環境により、実行時間が非常に長くなるため、トラッキングは 100 run のうち初めの 5 run は除外した。表 4.4 より、fitting では予想通り Joseph 形式の方が時間が掛かっているが、トラッキング全体としては 0.6% の範囲内で一致している。また、単精度と倍精度と比較すると、倍精度と単精度の間の差よりも、Joseph 形式と non-Joseph 形式の差の方が小さい。これらのことから、計算量が増加するという Joseph 形式のデメリットは、今回の環境下ではほとんど無視できると言える。また、seeding ではカルマンフィルターが使用されていないにもかかわらず、Joseph 形式と non-Joseph 形式の処理時間に 70 ms 程度の差が生じており、倍精度計算による速度低下と考えられる。

表 4.4: 単精度における Joseph 形式と non-Joseph 形式の処理時間。合計時間には、seeding、finding、fitting 以外の時間（ファイルの読み込み時間等）も含まれている。

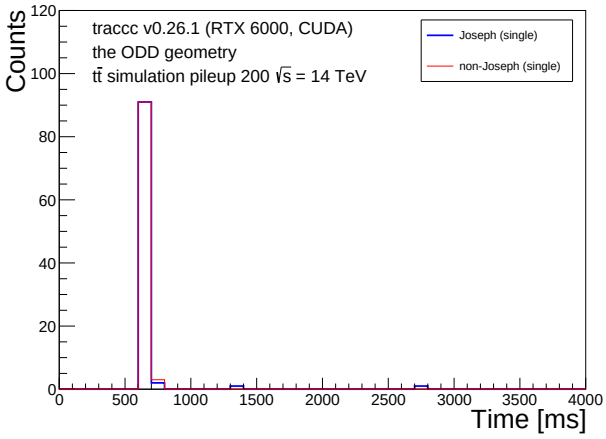
手法	Seeding [ms]	Finding [ms]	Fitting [ms]	Total [ms]
Joseph(単精度)	$(18 \pm 3) \times 10$	$(67 \pm 2) \times 10$	271.0 ± 0.4	$(950 \pm 6) \times 10$
non-Joseph(単精度)	$(11 \pm 2) \times 10$	$(68 \pm 4) \times 10$	263 ± 6	$(944 \pm 7) \times 10$
non-Joseph(倍精度)	$(77 \pm 2) \times 10$	$(279 \pm 4) \times 10$	$(105 \pm 1) \times 10$	$(1306 \pm 5) \times 10$

4.2.4 倍精度での Joseph 形式の有無による再構成精度の比較

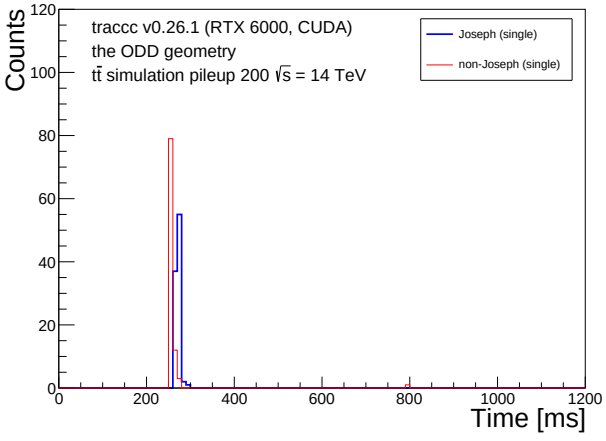
倍精度における Joseph 形式の導入による性能変化があるかを調べるため、倍精度同士での再構成性能を比較した。まず、表 4.5 と表 4.6 に、finding と fitting におけるカルマンフィルターで生じるエラー数の内訳を示す。この表から、倍精度では non-Joseph 形式と Joseph 形式ともに、エラーが生じていないことが分かる。Fitting における倍精度 non-Joseph と倍精度 Joseph の efficiency を図 4.7 に示した。この図から、倍精度においては non-Joseph 形式と Joseph 形式では、efficiency が変わらないことが分かる。したがって、今回の共分散行列の更新手法は単精度においてのみ重要であることが分かる。



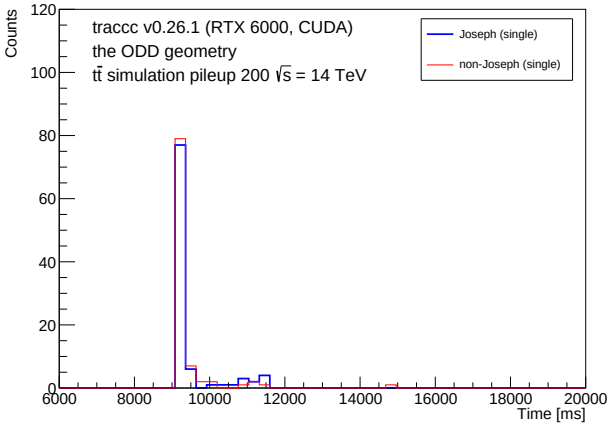
(a) seeding



(b) finding

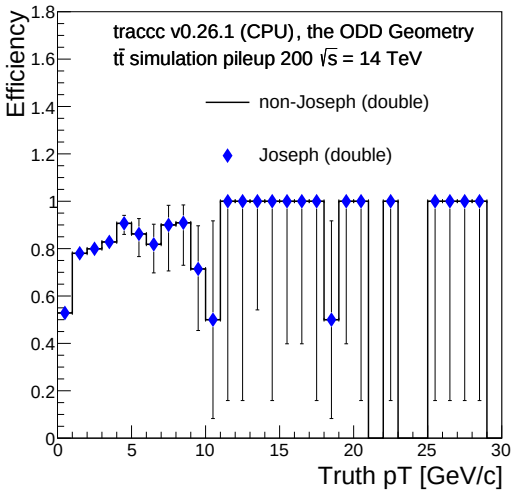


(c) fitting

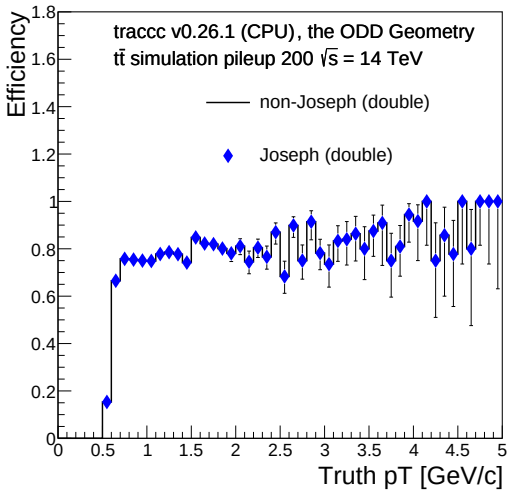


(d) 全体

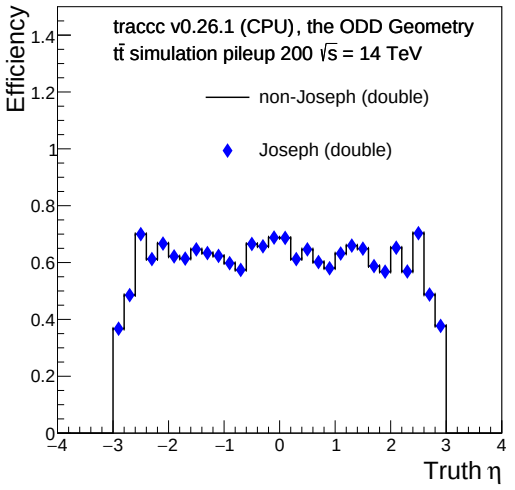
図 4.6: 単精度における Joseph と non-Joseph の処理時間の比較



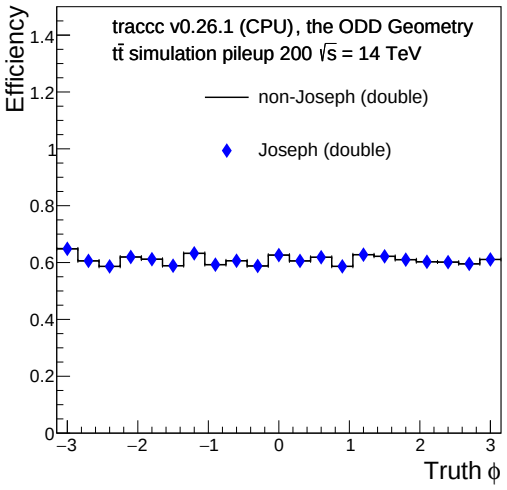
(a) p_T 分布



(b) p_T 分布 ($0 < p_T < 5.0\,\mathrm{GeV}$ を拡大したもの)



(c) η 分布



(d) ϕ 分布

図 4.7: 倍精度 Joseph と倍精度 non-Joseph の fitting における efficiency の比較

表 4.5: Finding におけるカルマンフィルターにおけるエラー数の比較。$t\bar{t}$ のパイルアップ 200 のサンプル、10 イベント。

	倍精度 non-Joseph	倍精度 Joseph
$\theta = 0$	0	0
$\phi = \infty$	0	0
$q/p = 0$	0	0
$\chi^2 < 0$	0	0
$\chi^2 = \infty$	0	0
Finding の入力トラック数 (seed 数)	349418	349519
Finding の出力トラック数	175383	175442

表 4.6: Fitting におけるカルマンフィルターにおけるエラー数の比較。$t\bar{t}$ のパイルアップ 200 のサンプル、10 イベント。

	倍精度 non-Joseph	倍精度 Joseph
$\theta = 0$	0	0
$\phi = \infty$	0	0
$q/p = 0$	0	0
$\chi^2 < 0$	0	0
$\chi^2 = \infty$	0	0
Fitting の入/出力トラック数	43303	43304

第 5 章

まとめ

2030 年から開始予定の高輝度 LHC 実験では、パイルアップが大幅に増加する。この増加により、荷電粒子の飛跡再構成の計算量が大幅に増加し、計算機資源を圧迫することが大きな課題となっている。それに対応するために新たな飛跡再構成手法の一つとして、GPU を用いた飛跡再構成 traccc が開発されている。GPU は並列処理に優れる演算装置であり、traccc は、実験や検出器に依存しない汎用飛跡再構成ソフトウェア ACTS を GPU 上で動かし、各飛跡を並列処理することによって、飛跡再構成の高速化を図る。

GPU の大きな利点の一つは、特定の処理が高速であることである。GPU は飛跡数が多くなるほど、CPU に対して再構成計算がより高速になることを確認した。また、倍精度演算器よりも単精度演算器を多く持つ一般的な GPU においては、倍精度で再構成計算を行うよりも単精度で計算を行う方が高速であることが分かった。倍精度の演算器を多く持つ GPU は一般に高価であるので、GPU を用いた飛跡再構成では単精度で再構成計算を行いたい。しかし、単精度では数値精度の不足により、計算性能が低下するという課題がある。それを finding や fitting における $\chi^2 < 0$ のエラーの発生数や fitting における efficiency が倍精度と単精度の差が大きいことで確認した。このことから、カルマンフィルターでの計算エラーの低減が、この効率の改善につながると考えた。本研究では、finding と fitting で使用されているカルマンフィルターに着目し、カルマンフィルターの改良によって再構成性能の向上を目指した。具体的には、共分散行列の更新式において、最適カルマンゲインという条件を満たした場合にのみ成り立つ non-Joseph 形式を、最適カルマンゲインを仮定しない Joseph 形式に変更することで、飛跡再構成性能にどのような変化があるかを調べた。

Joseph 形式に変更することで、単精度を維持したまま finding と fitting における $\chi^2 < 0$ のエラー数が減少し、fitting の efficiency も改善することを確認した。これによってトラックパラメーターの精度が改善する飛跡が増えた。また、Joseph 形式の導入によって、residual 等の再構成評価指標も変化していないことを確認した。このことは、Joseph 形式、あるいは倍精度により単精度の non-Joseph 形式に対して増えたトラックの再構成精度が、有意に劣るわけではなく、fitting の成功数が増えるに応じて，再構成精度の高いトラックが増えることを示している。さらに、Joseph 形式のデメリットである計算量の増加もほとんど影響がないことも確認した。また、倍精度では Joseph と non-Joseph の再構成性能に差は見られず、数値精度が大きく影響することも再確認し

た。一方で、Joseph 形式の導入により efficiency は改善したが、依然として低いままであるため、単精度の他のエラーの原因も探り、さらなる改良が必要がある。

付録 A

Joseph 形式の測定ノイズ共分散行列 (measurement noise covariance) の導出

ここでは、ステップ i における測定ノイズの共分散 (検出器の位置分解能) R_i を導出する。R_i も最適カルマンゲインを仮定するかどうかによって、Joseph 形式と同様に簡略化される。

$$y_i = H_i x_i + v_i,\ \hat{x}_i^+ = \hat{x}_i^- + K_i(y_i - H_i\hat{x}_i^-) \tag{A.1}$$

ここで、

y_i ：ステップ i における測定（観測）ベクトル（検出器の位置座標）
H_i ：ステップ i における状態空間から測定（観測）空間へのマッピング行列（状態 x と測定値 y はそのままでは比較できないので、変換行列 H によって同じ空間に変換している）
$\hat{x}_i^-$ ：ステップ i での予測された状態ベクトル（今回はトラックパラメーター）
$\hat{x}_i^+$ ：ステップ i での更新された状態ベクトル
K_i ：カルマンゲイン（式 (A.3) では最適カルマンゲインを仮定しない）
v_i ：測定ノイズ

である。

まず、導出の準備として式 (A.1) を考えると、

$$\begin{aligned}
r &= y_i - H_i\hat{x}_i^+ \\
&= H_i x_i + v_i - H_i\left(\hat{x}_i^- + K_i(y_i - H_i\hat{x}_i^-)\right) \\
&= H_i x_i + v_i + -H_i\hat{x}_i^- - H_i K_i(y_i - H_i\hat{x}_i^-) \\
&= H_i x_i + -H_i\hat{x}_i^- - H_i K_i\left(H_i x_i + v_i - H_i\hat{x}_i^-\right) \\
&= H_i(x_i - \hat{x}_i^-) + v_i - H_i K_i H_i(x_i - \hat{x}_i^-) - H_i K_i v_i \\
&= (I_m - H_i K_i)v_i + H_i(I_{66} - K_i H_i)(x_i - \hat{x}_i^-)
\end{aligned} \tag{A.2}$$

以下から導出する。

$$
\begin{aligned}
R_i &= Cov(r) \\
&= E[\{(I_m - H_iK_i)v_i + H_i(I_{66} - K_iH_i)(x_i - \hat{x}_i^-)\} \\
&\quad \{(I_m - H_iK_i)v_i + H_i(I_{66} - K_iH_i)(x_i - \hat{x}_i^-)\}^T] \\
&= H_i(I_{66} - K_iH_i)E[(x_i - \hat{x}_i^-)(x_i - \hat{x}_i^-)^T](I_{66} - K_iH_i)^T H_i^T \\
&\quad + (I_m - H_iK_i)E[v_i(x_i - \hat{x}_i^-)^T](I_{66} - K_iH_i)^T H_i^T \\
&\quad + H_i(I_{66} - K_iH_i)E[(x_i - \hat{x}_i^-)v_i^T](I_m - H_iK_i)^T \\
&\quad + (I_m - H_iK_i)E[v_iv_i^T](I_m - H_iK_i)^T \\
&= (I_m - H_iK_i)E[v_iv_i^T](I_m - H_iK_i)^T \\
&\quad + H_i(I_{66} - K_iH_i)E[(x_i - \hat{x}_i^-)(x_i - \hat{x}_i^-)^T](I_{66} - K_iH_i)^T H_i^T \\
&\quad (\because E[(x_i - \hat{x}_i^-)v_i^T] = E[v_i(x_i - \hat{x}_i^-)^T] = 0) \\
&= (I_m - H_iK_i)R_i(I_m - H_iK_i)^T + H_i(I_{66} - K_iH_i)P_i^-(I_{66} - K_iH_i)^T H_i^T
\end{aligned}
\tag{A.3}
$$

以下に、Joseph 形式の $R(Cov(r))$ から non-Joseph 形式の $Cov(r)$ の導出（共分散行列の更新式と同様に、式 (2.5) の最適カルマンゲインを仮定した場合に Joseph 形式から non-Joseph の形式が導かれる事) を示す。

準備

まず、準備として、以下の式 (A.4) を示す。

$$
P_i^- H_i^T K_i^T = K_i H_i P_i^- \tag{A.4}
$$

$S = H_iP_i^-H_i^T + R_i$ とおくと、P_i^- と R_i は対称行列であるから、S は対称行列である。したがって、S^{-1} も対称行列である。また、S を用いると、$K_i = P_i^- H_i^T S^{-1}$ と表せるので、

$$
\begin{aligned}
\text{(式 } (A.4) \text{ の左辺)} &= P_i^- H_i^T K_i^T \\
&= P_i^- H_i^T S^{-1} H_i (P_i^-)^T
\end{aligned}
\tag{A.5}
$$

ここで、式 (A.4) の左辺は対称行列であるから、$P_i^- H_i^T K_i^T = (P_i^- H_i^T K_i^T)^T$ となることを考えると、

$$
\begin{aligned}
\text{(式 (A.4 の) 左辺)} &= P_i^- H_i^T K_i^T \\
&= (P_i^- H_i^T K_i^T)^T \\
&= K_i H_i (P_i^-)^T \\
&= K_i H_i P_i^- (\because P_i^- \text{は対称行列})
\end{aligned}
\tag{A.6}
$$

よって、式 (A.4) が示された。

導出

$$
\begin{aligned}
Cov(r) &= (I_m - H_iK_i)R_i(I_m - H_iK_i)^T + H_i(I_{66} - K_iH_i)P_i^-(I_{66} - K_iH_i)^T H_i^T \\
&= (I_m - H_iK_i)R_i - (I_m - H_iK_i)R_iK_i^T H_i^T \\
&\quad + H_i(I_{66} - K_iH_i)P_i^- H_i^T + H_i(I_{66} - K_iH_i)P_i^- H_i^T K_i^T H_i^T \\
&= (I_m - H_iK_i)R_i - R_iK_i^T H_i^T + H_iK_iR_iK_i^T H_i^T \\
&\quad + H_iP_i^- H_i^T - H_iK_iH_iP_i^- H_i^T - H_iP_i^- H_i^T K_i^T H_i^T + H_iK_iH_iP_i^- H_i^T K_i^T H_i^T \\
&= (I_m - H_iK_i)R_i - (H_iP_i^- H_i^T + R_i)K_i^T H_i^T + H_iK_iR_iK_i^T H_i^T \\
&\quad + H_iP_i^- H_i^T - H_iK_iH_iP_i^- H_i^T + H_iK_iH_iP_i^- H_i^T K_i^T H_i^T \qquad \text{(A.7)} \\
&= (I_m - H_iK_i)R_i + H_iK_i(H_iP_i^T H_i^T + R_i)K_i^T H_i^T - H_iK_iH_iP_i^- H_i^T \\
&\quad (\because P_i^- \text{は対称行列}) \\
&= (I_m - H_iK_i)R_i + H_iP_i^- H_i^T K_i^T H_i^T - H_iK_iH_iP_i^- H_i^T \\
&\quad (\because K_i = P_i^- H_i^T (H_iP_i^- H_i^T + R_i)^{-1}) \\
&= (I_m - H_iK_i)R_i + H_i(P_i^- H_i^T K_i^T - K_iH_iP_i^-)H_i^T \\
&= (I_m - H_iK_i)R_i \quad (\because \text{式 } (A.4))
\end{aligned}
$$

よって、Joseph 形式の測定ノイズ共分散行列 R から non-Jospeh 形式の測定ノイズ共分散行列 R が導かれた。

謝辞

山崎先生、最後まで修論の添削をしてくださり、本当にありがとうございます。休日もミーティングをしてくださり、ありがとうございます。お手数をおかけしました。CERN 出張に行かせていただきありがとうございました。とても良い思い出になりました。前田先生、学部の卒業研究から修士での ATLAS まで大変お世話になりました。卒業研究では、検出器の製作を夜中まで手伝っていただいたり、解析を夜中まで手伝ってくださり、ありがとうございました。本当に心強かったです。これを書いている修論提出の日も朝まで残ってくださりありがとうございました。お体に気をつけてください。藏重先生、研究について常に鋭いご指摘をしていただき、様々な知見を頂きました。ありがとうございました。副査も引き受けてくださり、ありがとうございます。今年退官されるとのことですが、これまでお疲れ様でした。粒子物理学研究室のスタッフの方々、いつもご指導ありがとうございました。北川さん、出張などの手続きをしてくださりありがとうございました。伊藤飛鳥先生、副査を引き受けていただきありがとうございます。

清水さん、リモートが中心でしたが、丁寧なご指導ありがとうございました。白岩くん、共同研究者として色々教えてくれてありがとう。広島学会での飲み会は楽しかったです。

Stephen, thank you for teaching and answering my questions. I really enjoyed my CERN life, thanks to you!

水引さん、ATLAS の先輩として普段からお世話になりました。研究内容への鋭いコメントが非常に勉強になりました。CERN で沢山奢ってくださりありがとうございます。笹田くん、研究室の隣の席で色々テクノロジーを教えてくれてありがとう。今後も役に立てていこうと思います。CERN も楽しかったです。佐倉くん、さすがのワードセンスありがとうございます。研究室から一緒に帰ることが多かったですが、とても支えになりました。佐野くん、靴下を食べたりして笑わせてくれてありがとう。とても元気が出ました。CERN でも助けてくれてありがとう。田中くん、スライドの作り方がうまいなといつも思っています。博士課程でも頑張ってください。張さん、ニコニコ話しかけてくれてありがとう。今度中国を案内してください。あと、出会ったときよりも日本語が格段にうまくなっていて尊敬しています。山口くん、奇想天外な発想でいつも笑わせてくれてありがとうございます。博士課程でも頑張ってください。神吉くん、橋本さん、二人にはとても笑わせてもらいました。これからもそのまま頑張ってください。応援しています。

高橋さん、私が B4 の時から様々な面でお世話になりました。物理や研究に関する的確なアドバイスは、研究において非常に支えになりました。ありがとうございました。鐘さん、生井さん、鈴

木大夢さん、様々なアドバイスをくださりありがとうございました。鈴木啓司くん、学部 1 年の時から沢山遊んでくれてありがとう。曽根くん、また家でゲームをやろうね。谷口くん、これからもよろしくね。和田くん、マスターボールのクッションにはいつも助けられました。M1 の青山くん、稲葉くん、門田くん、遠山さん、西上さん、西田さん、野村くん、番原くん、ズーくん、大部屋で楽しませてくれてありがとう。特に忘年会・新年会などを色々計画してくれてありがとうございました。クレープやお刺身があり、豪華で非常に楽しい会でした。B4 の井上くん、畑中くん、豊福くん、原くん、佐村くん、小川くん、横川くん、奥くん、西川くん、矢野くん、足立くん、三好くん、木下くん、いつも挨拶してくれたり、フットサルに参加してくれてありがとう。

参考文献

[1] CERN, ATLAS Open Data, The Standard Model of Particle Physics and Beyond 2025, URL: https://opendata.atlas.cern/docs/documentation/introduction/SM_and_beyond.

[2] CERN, The CERN accelerator complex, layout in 2022 2022, URL: https://cds.cern.ch/record/2800984.

[3] The ATLAS Collaboration, ATLAS Open Data URL: https://opendata.atlas.cern/docs/documentation/introduction/SM_and_beyond/.

[4] The ATLAS Collaboration, Study of Higgs boson pair production in the HH → bb γ γ final state with 308 fb − 1 of data collected at s √=13 TeV and 13.6 TeV by the ATLAS experiment 2025, URL: https://arxiv.org/abs/2507.03495.

[5] CERN, HL-LHC PROJECT URL: https://project-hl-lhc-industry.web.cern.ch/content/project-schedule.

[6] The ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider: A Description of the Detector Configuration for Run 3 2024, URL: https://arxiv.org/pdf/2305.16623.

[7] The ATLAS collaboration, Expected tracking performance of the ATLAS Inner Tracker at the High-Luminosity LHC 2025, URL: https://iopscience.iop.org/article/10.1088/1748-0221/20/02/P02018/pdf.

[8] ATLAS collaboration, ATLAS Software and Computing HL-LHC Roadmap 2022, URL: https://cds.cern.ch/record/2802918?ln=ja.

[9] Xiaocong Ai, Corentin Allaire, Noemi Calace, Ang ´ ela Czirkos, Markus Elsing, Irina Ene, Ralf Farkas, Louis-Guillaume Gagnon, Rocky Garg, Paul Gessinger, Hadrien Grasland, Heather M. Gray, Christian Gumpert, Julia Hrdinka, Benjamin Huth, Moritz Kiehn, Fabian Klimpel, Bernadette Kolbinger, Attila Krasznahorkay, Robert Langenberg, Charles Leggett, Georgiana Mania, Edward Moyse, Joana Niermann, Joseph D. Osborn, David Rousseau, Andreas Salzburger, Bastian Schlag, Lauren Tompkins, Tomohiro Yamazaki,Beomki Yeo, Jin Zhang, A Common Tracking Software Project 2021, URL: https://arxiv.org/abs/2106.13593.

[10] the Acts project, ACTS GitHub URL: https://github.com/acts-project/acts.

[11] The ATLAS collaboration, ATLAS Software Documentation 2024 (last update),
URL: https://atlassoftwaredocs.web.cern.ch/internal-links/tracking-tutorial/idoverview/.

[12] the Acts project, ACTS Tracking in a nutshell
URL: https://acts.readthedocs.io/en/latest/tracking.html.

[13] Rudolf Frühwirth , Are Strandlie, Pattern Recognition, Tracking and Vertex Reconstruction in Particle Detectors 2021,
URL: https://link.springer.com/book/10.1007/978-3-030-65771-0.

[14] J. Russell Carpenter and Christopher N. D' Souza, Navigation Filter Best Practices 2018,
URL: https://ntrs.nasa.gov/api/citations/20180003657/downloads/20180003657.pdf.

[15] Hisa Ando, GPU を支える技術 超並列ハードウェアの快進撃 [技術基礎] 2021.

[16] Paul Gessinger-Befurt, Andreas Salzburger, Joana Niermann, The Open Data Detector Tracking System
URL: https://iopscience.iop.org/article/10.1088/1742-6596/2438/1/012110/pdf.