筑波大学大学院修士課程
理工学研究科修士論文

JLCのためのKalman Filterによる飛跡再構成プログラムの開発

中島　泰宏

平成16年3月

Contents

Chapter 1

序論

1.1 はじめに

高エネルギー物理学とは、物質の究極の構成要素を探求し、その間に働く相互作用を解明することを目的とする学問である。高エネルギー物理学発展の歴史は、その主要な研究手段である高エネルギー加速器の進展とともにあった。新しいより高いエネルギーの加速器での衝突実験は、より小さな領域の探索を可能とし、我々の極微の世界に対する理解を深めてきたのである。現在我々は、標準理論に集約される世界像、すなわち、「自然がクォークとレプトンと呼ばれる少数の物質粒子と、その間の相互作用を媒介するゲージ粒子からなり、それらがゲージ対称性で関係づけられている」とする世界像（ゲージ原理）を手にするに至っている。この標準理論は、現在までの全ての実験事実を見事に説明し、その破れを示す確かな証拠はこれまでの所見つかっていない[1]。しかしながら、標準理論を形作るもう一つの重要な要素である自発的対称性の破れを引き起こす実体、つまり、この標準理論にあって、ゲージ粒子や物質粒子の質量を生み出すなくてはならない粒子、すなわちヒッグス粒子は未だ未発見のままである。昨今話題となっているニュートリノ振動や、B 中間子系での CP 非保存は、いずれも物質粒子（クォークまたはレプトン）とヒッグス粒子との湯川相互作用に起因すると考えられるので、その本質的な理解は、自発的対称性の破れを引き起こす実体の解明を抜きにしてありえない。現在の高エネルギー物理学の最重要課題は、このヒッグス粒子の発見とその性質の詳細な研究、さらには標準理論の枠組みを本質的に越える理論（超対称性、余次元、大統一など）の実験的な手がかりを得ることである。標準理論の構造を詳細に調べることにより、その手がかりが TeV 領域にあることが分かっている。これが、LHC を初めとする TeV 領域を調べるための将来加速器の主な建設動機である。

LHC では、広いエネルギー領域を調べる事ができるという大きな利点が有るが、構造を持った陽子・陽子の衝突であるため、反応が複雑で、素過程の精密な測定は難しい。一方、電子・陽電子線形衝突型加速器では、構造を持たない粒子同士の衝突なので、始状態が明快に分かっており、反応も素過程そのものなので、精密測定に理想的である。これまでの実験から、間接的ではあるが、標準理論のヒッグス粒子の質量について、95% の信頼度で、約 200 GeV 以下という上限値がえられており、重心系エネルギーで 400 GeV

[1]近年のニュートリノ振動の発見は、ニュートリノが有限の質量を持つ事を示したが、これは、標準理論の基本的枠組みを崩す事なく簡単に取り込む事ができる。標準理論の根幹をなす基本原理、すなわちゲージ原理はいささかもゆらいでいないのである。

程度の電子・陽電子線形衝突型加速器でその発見、研究ができると期待されるので、世界的に、建設に向けて気運が盛り上がっている。我が国でも、アジア諸国と共同で、将来計画として電子・陽電子線形衝突加速器である JLC 計画を推進しつつあり、現在加速器、測定器の両方で勢力的に開発研究が行われている。

既に述べたように、JLC の電子・陽電子線形衝突加速器の実験で最も重要な目標の一つに、ヒッグス粒子の発見と研究が挙げられる。特に、$e^+e^- \to H^0Z^0$ の反応で Z^0 の崩壊から生じるレプトン対の反跳質量分布の測定によるヒッグス粒子の測定では、その質量、崩壊巾の最も精度の高い測定を可能にするのみならず、ヒッグス粒子の崩壊モードによらない探索を可能とするので、生成断面積の崩壊分岐比によらない絶対測定が可能となる。また、ヒッグス粒子が検出不可能な粒子に崩壊した場合にも使える探索法を提供する。このレプトン対の反跳質量分布の測定で最も重要な役割を果たすのが中央飛跡検出器である。

レプトン対の反跳質量の分解能は、もともと加速中に生じるビームエネルギーの広がりと中央飛跡検出器によるレプトン対の運動量分解能とで決まる。加速器の性能を最大限に活かすためには、中央飛跡検出器によるレプトン対の運動量分解能の反跳質量分解能に対する寄与は、ビームのエネルギーの広がりの寄与に比べて無視できる程度でなくてはならない。後で詳しく述べるように、この要求から、 $\sigma_{p_T}/p_T \sim 10^{-4} \times p_T(\mathrm{GeV})$ の運動量分解能を持った高性能の中央飛跡検出器が必要となる。

また、高エネルギー電子・陽電子線形衝突型加速器実験の著しい特徴として、複数のジェットに崩壊する弱い相互作用のゲージ粒子（W/Z）、トップクォーク（t）、さらにはヒッグス（H）などの重いパートンを、ジェット不変質量法によって同定し、素過程を基本粒子のレベルで再構成する可能性（ファインマン図が見えてくる）があげられるが、そのためには、ジェット中の近接する荷電粒子飛跡を高効率で分離し、各々の運動量を高精度で測定する事、また、こうして検出された荷電粒子飛跡をカロリメータのクラスターと一対一対応させる事が必要となる。

一方、将来の電子・陽電子線形衝突型加速器実験では、ビームをナノメータサイズにまで収束させ衝突させるので、衝突するビーム同士が強い電磁力を及ぼし合い、今までにない種類のバックグラウンドが発生する可能性もある。測定器はこうしたバックグラウンドに十分耐えうるものでなくてはならない。

こうした物理の要求を踏まえ、また、加速器からの要請を加味した、中央飛跡検出器の全体設計の完成が我々 JLC CDC グループの開発研究の最終目標である。一般に、開発研究は、要素技術の開発と、それを組み合わせたシステム開発とに大別される。CDC の開発研究の場合、前者は、ワイヤーあたりの位置分解能とか、１つのドリフト・セルあたりの近接ヒット分離能といった測定器の局所的な性質にかかわる要求性能の実現可能性を見極めるためのハードウエア開発研究が中心となる。一方、後者のシステム開発は、CDC 全体としての性能評価を必要とするため、実機規模のプロトタイプ建設以前の段階においては、要素技術の開発研究の結果に基づいた、シミュレーション主体のものとなる。CDC の全体設計は、前者に対応する検出器要素（例えばドリフト・セル）の設計、後者に対応するそれらの配置（例えばレイヤー構造、全体としての大きさなど）の設計からなり、各々、基本仕様（検出器の幾何学的構造を決めるパラメータや、例えばワイヤーあたりの位置分解能など）を決める基本設計（Conceptual Design）、またそれを実際どう具体化するかを決める技術設計（Engineering Design）の段階を踏むことになる。

CDC に関する要素技術開発は、これまで主として宇宙線テスト、ビームテストなど、テストチェンバーによる実験を中心にすすめられてきた。その結果、ワイヤーあたりの位置分解能、ドリフトセルあたりの近接ヒット分離能といった測定器の局所的な性能について、その要求性能の実現可能性がほぼ確立した。一方、近年、リニアコライダーの設計提案が、JLC のみならず北米、欧州でも行われ、世界的に早期建設の気運が盛り上がっている。そこで、これまでのテスト実験の結果に基づく、CDC を含む測定器全体に関す

る詳細なモンテカルロ・シミュレーションを早急に行い、その基本設計を固めるとともに技術設計を含めた全体設計を急ぐ必要が出てきた。

このような情勢をうけて、現在測定器全体のシステム開発に不可欠な測定器のフル・シミュレータの開発が、急ピッチで進められている。この開発中のシミュレータは、JUPITER（JLC Unified Particle Interaction and Tracking EmulatoR）と呼ばれるGeant4[2]に基づく部分と、ROOT[3]に基づくその衛星プログラム（Satellites）からなる。いずれも、C++言語を用いたオブジェクト指向技術を最大限に活用した設計になっている。

本研究は、このフル・シミュレータ開発の一環として行われたものである。すでに述べたようにJLCでは、高運動量分解能、高近接飛跡分離能、高バーテックス分解能及び高検出率、異なるバンチからのバックグラウンドトラックを区別するためのタイム・スタンピング能力が必要となる。そのため、CDCから比較的密度の高い物質層を越えてバーテックス検出器のトラックと連結しトラックパラメータの測定制度を向上させること、複数のトラックを組み合わせ、崩壊点や衝突点の位置測定制度を向上させることが課題となる。Kalman Filterはこのような問題に有用な手法である。本研究では、Kalman Filterをこれらの問題に応用するための基本クラスを作成し、それをもとに飛跡再構成プログラムを開発する。さらにその応用としてタイム・スタンピング性能を研究する。

本論文の構成は以下の通りである。まずこの章の残りを使って、JLC測定器の全体設計の思想を概説した後に、中央飛跡検出器に課せられる要請を考え、そのためにどのようなシミュレーションが必要になったかを概説する。そして次章において高エネルギーにおけるシミュレーションの役割、その仕組みについて述べ、本研究がそのどの部分を担っているのかの位置付けを行う。第3章で現在開発中のオブジェクト指向技術を最大限に活用したGeant4に基づく測定器シミュレーションの枠組みについて、その設計理念、構造を詳説する。第4章ではKalman Filterの原理について述べ、基本クラスの作成、飛跡再構成プログラムへの応用について述べる。第5章ではその飛跡再構成プログラムを用いて、タイム・スタンピング能力について述べる。最後の章では、これらの開発研究の現状をまとめるとともに今後の開発研究の方向を展望する。

1.2 JLC測定器の概要とCDCの役割

1.2.1 全体設計

JLCの衝突点は当面一ケ所であるため、そこに設置される測定器は、想定されるあらゆる物理だけでなく、予想外の物理に対しても十分対応可能な高性能かつ汎用の物でなければならない。具体的にはまずW粒子やZ粒子のジェットを用いた再構成が精度よく出来なければならない。また、ヒッグス粒子の発見及び研究のため、非常に良い質量分解能が要求される。またトップクォークの研究やヒッグス粒子に対するバックグラウンドを抑えるため、bジェット識別が効率良く行える必要がある。さらには、超対称粒子探索のため十分広い立体角を隙間なく覆う測定器でなければならない。これらの条件をシミュレーションを行って検討した結果、「標準測定器」として図1.1に示すような構成の測定器を考え、表1.1に示すような性能を達成することを目標としている。

[2]Geometory ANd Tracking simulator の略 [4]
[3]C++を用いた大規模データ解析のためのフレームワーク [6]

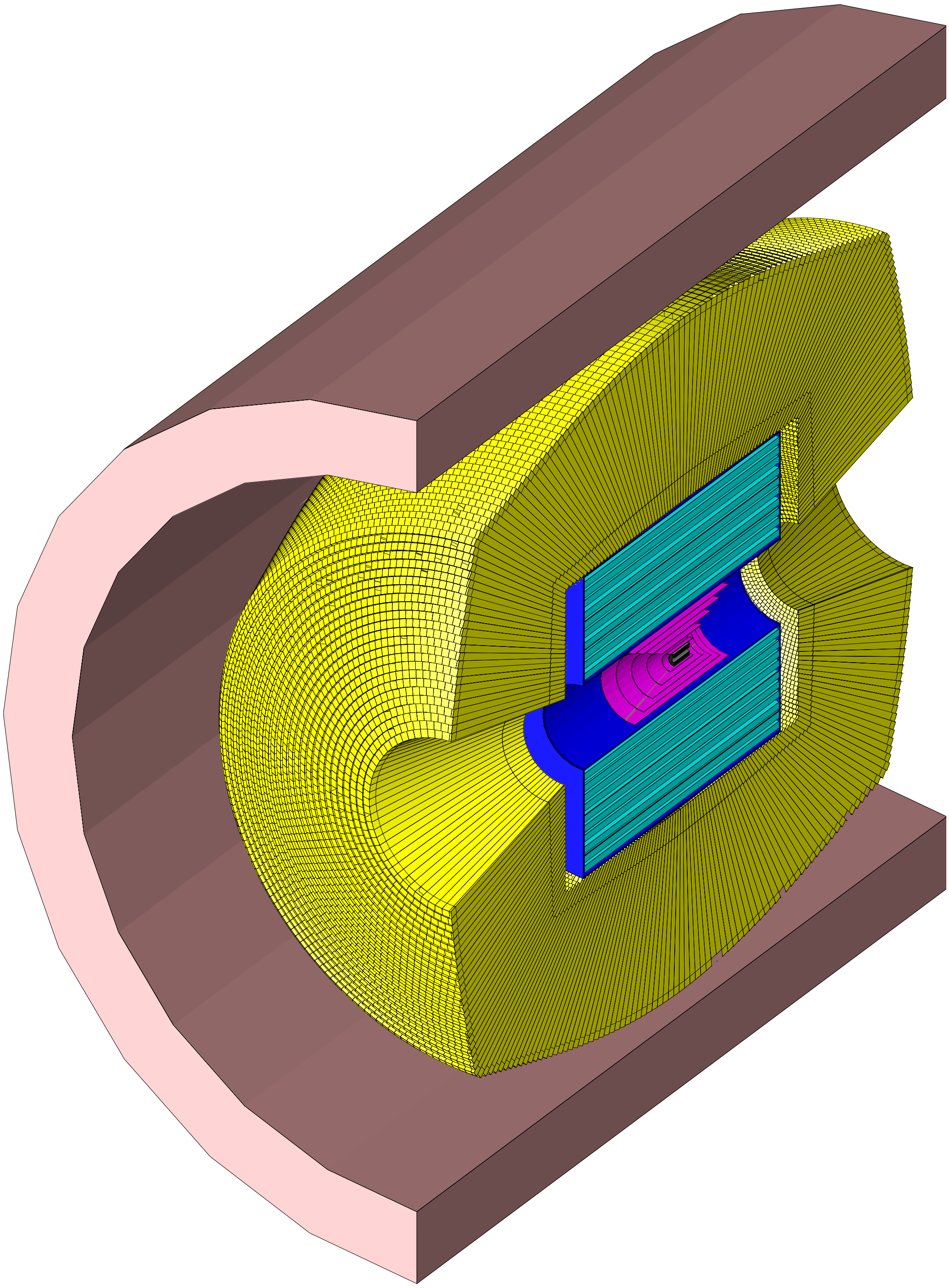


Figure 1.1: JLC 測定器の概略図

Detector		Configuration	Performances	Channels and Data Size
PM (3D Active Pixel)		θ = 11 - 48mrad (r=2-8.5cm) 300μm-thick x 2 layers pixel size=100μm	Under Study	Number of pixels = 8.6M Readout channel = 156ch Data size = 12k bytes/sec
LM (W/Si)		θ = 50-150mrad 43Xo x 16samplings Nr = 32, $N\phi$ = 16	Under Study	Number of pads = 16.4k Readout channel = 128ch Data size = 3.3k bytes/train
AM (W/Si)		θ = 150-200mrad 23Xo x 8samplings Nr = 10, $N\phi$ = 32	Under Study	Number of pads = 5.1k Readout channel = 16 Data size = 5.1k bytes/train
FT		TBD	Unknown	
VTX (CCD)		cosθ < 0.90 pixel size=25μm, thickness=300μm 4 layers at r = 2.4, 3.6, 4.8, 6.0cm	**σ = 4.0μm** $\delta^2 = 7^2 + (20/p)^2/\sin^3\theta$ [μm] **ε_b=50% @ purity=93%**	Number of pixels = 320M Readout channel = 2.4k Data size = 1.4M bytes/train
IT (Si-strip)		cosθ < 0.90 strip width=100μm, thickness=300μm 5 layers at r = 9, 16, 23, 30, 37cm	**σ = 40μm** Tracking Performance Under Study	Number of strips = 522k Readout channel = 1.0k Data size = under study
CDC (Mini-jet)	common	cosθ < 0.70 (full sample) cosθ < 0.95 (1/5 samples)	**σ_z = 1mm** **2-track separation = 2mm**	200MHz FADC depth = 1k words
	2Tesla	r = 45 - 230cm, L = 460cm N_{sample} = 80	**σ_x =100μm** **σ_{Pt} /Pt = 1x10^{-4}Pt + 0.1%**	Readout channel = 13k Data size = 5.2M bytes/train
	3Tesla	r = 45 - 155cm, L = 310cm N_{sample} = 50	**σ_x =85μm** **σ_{Pt} /Pt = 3x10^{-4}Pt + 0.1%**	Readout channel = 8.1k Data size = 3.3M bytes/train
Trackers Combined			**σ_{Pt} /Pt = 1x10^{-4}Pt + 0.1%**	
CAL (Pb/Sci)	common	EM = 27Xo (3sections) HAD = 6.5λo (4sections) $\Delta_{\theta,\phi}$ = 24mrad (EM), 72mrad (HAD)	**σ/E=15%/√E+1% (EM)** **σ/E=40%/√E+2% (Had)** **e/π ID = 1/1000**	Number of cells = 144k Readout channel = 5k Data size = 3k bytes/train
	2Tesla	cosθ < 0.985 (full thickness) r = 250 - 400 cm, z = +- 290cm		
	3Tesla	cosθ < 0.966 (full thickness) r = 160 - 340 cm, z = +- 190cm		
SHmax		scin.strip (1cm-wide) or Si-pad (1cm x 1cm)	**σ = 3mm/√E**	Readout channel = 5k Data size = 40k bytes/train
MU (SWDC/RPC/TGC)		cosθ <0.998 6 SuperLayers	**σ = 0.5mm** Muon ID under study	Readout channel = 10k
Yoke	2Tesla	r = 5.5m - 7.5m, Z = 5.0m - 7.9m		
	3Tesla	r = 4.5m - 7.0m, Z = 3.9m - 6.5m		

Table 1.1: JLC 測定器の諸性能

1.2.2 測定器の構成

バーテックス検出器

衝突点の極めて近傍で、荷電粒子の飛跡を精密に測定する検出器で、B 中間子や D 中間子の崩壊点を測定して、b クォークや c クォークの同定をする役割をしている。通常バーテックス検出器に使用されるシリコンストリップ型ではなく、2 次元的に読み出し可能なピクセル型の Si-CCD を使用する。多量のバックグランド中でも十分な機能を果たす為に、4 層の CCD 検出器より構成され、それぞれビーム軸から 2.4、3.6、4.8、6.0 cm の位置に設置される。総数 3.2×10^8 個の 25 μm × 25 μm のピクセルで構成され空間分解能は 4 μm である。

中央飛跡検出器 (CDC)

寿命の長い荷電粒子 (電子、ミュー粒子、荷電 π 中間子、荷電 K 中間子、陽子) の飛跡を検出する装置で、超電導磁石による磁場によって曲げられた荷電粒子の飛跡の曲率半径からその粒子の運動量を知ることができる。詳細は後述。

カロリメータ

飛跡を残さない中性粒子 (光子、中性子、K_L^0 など) のエネルギーを測定する役割をもつ。カロリメータはビーム軸の周りの円筒状のバレル部とそれに蓋をするような形の端部より成り $|\cos\theta| < 0.99$ の領域を覆っている。検出部分は鉛とプラスチックシンチレーターの多層サンドイッチ構造をしていて 1GeV の電子や光子に対するエネルギー分解能は 15%、1GeV のハドロン粒子に対しては 40%である。

ミュー粒子検出器

カロリメータで吸収されることなく全ての測定器を通り抜けるミューオンを検出する役割をもつ。このミュー粒子検出器は運動量の測定ではなくミュー粒子の識別に用いられるので空間分解能は 500 μm 程度でよい。この検出器は、6 層 (Supre-Layer) からなり、1 つの層 (Supre-Layer) には単線のワイヤーチェンバーから成る 4 つの層構造になっていて飛跡の方向を求めることができる。またカロリメータ及び鉄中でのエネルギー損失のため識別可能なミュー粒子の運動量は 3.5GeV 以上である。

1.3 中央飛跡検出器 (CDC) への要請

1.3.1 物理からの要請

多くの実験による精力的な探索にも係わらず、標準模型の要であるヒッグス粒子は未だ見つかっていない。その探索と性質の研究は JLC における物理の最も重要な課題の一つである。標準模型では、ヒッグス粒子の質量はパラメーターにすぎない。従って、その予言は標準模型を越える理論でのみ可能である。軽い ($\lesssim$ 200 GeV) ヒッグス粒子の存在は大統一、大砂漠を基礎とする模型の一般的な帰結である。これに対し、テクニカラーなどの複合ヒッグス模型ではヒッグス粒子は重くて良い。前者のシナリオでは、自然さの問題を考えれば、TeV 以下に多くの超対称粒子が存在し得る。一方、後者のシナリオでは、その背後にある新しい力学の全容を明らかにするには、TeV を越えたエネルギーが必要となる。この意味で、JLC における

軽いヒッグス粒子の探索は、高エネルギー物理の今後の方向を決定する分岐点となるであろう。既に述べたように、標準理論の枠組みの中で、これまでに得られたデータをフィットすると、ヒッグス粒子の質量に対して約 200 GeV の上限値がえられる。これは、データが軽いヒッグスのシナリオを指示していると見る事ができるが、重いヒッグスの可能性はまだ残されており、最終的な決着は、直接探索の結果を待たねばならない。以下にヒッグス粒子探索からの中央飛跡検出器の性能に対する要求をまとめる。

ヒッグス粒子の探索法

高エネルギー電子陽電子衝突反応で、標準模型のヒッグス粒子を作る反応としては、(1) $e^+e^- \to H^0_{SM}Z^0$、(2) $e^+e^- \to \nu\bar{\nu}H^0_{SM}$、(3) $e^+e^- \to e^+e^-H^0_{SM}$ 反応などがある。このうち、(2) と (3) の反応は 1 TeV 以上の高エネルギーで断面積が大きくなるので特に重いヒッグスの探索の場合に適している。一方、(1) の反応は比較的軽いヒッグスの場合に適しており、LEP での探索などで利用されている。JLC-I のエネルギー領域でも、主に (1) の反応を利用して、ヒッグスの探索を行なうことになる。

一方、ヒッグス粒子とフェルミオン、ウィークボソンとの結合は質量に比例し、よって、その崩壊の部分巾は質量の自乗に比例する。従って、ヒッグス粒子は、質量が半分以下の粒子 ($< \frac{1}{2}M_{H^0_{SM}}$) のうち最も重い粒子への崩壊巾が最も大きい。実際 140 GeV 以下では確かに $H^0_{SM} \to b\bar{b}$ モードの崩壊の分岐比が最大である。しかしながら、ウイークボソンへの崩壊巾と b クォークへの崩壊巾は 3 桁近く異なるので、140 GeV 以上では、$H^0_{SM} \to W^*W$ モードの崩壊の分岐比が最も高くなる。超対称性理論が予言するような軽いヒッグス（140 GeV 以下）の場合は、主として $H^0_{SM} \to b\bar{b}$ モードの崩壊を探索することになる。

以上のことから、$e^+e^- \to H^0_{SM}Z^0$ 反応によるヒッグス粒子生成事象の検出は、終状態として、Z^0 の崩壊モードにより (1)$\nu\bar{\nu}b\bar{b}$、(2)$l^+l^-b\bar{b}$、および (3)$q\bar{q}b\bar{b}$ の三つの型にに分類出来る。

典型的な事象の例を図 1.2 に示す。

このようなヒッグス粒子は次の二つの方法で探索できる。まず、上記三つの場合によって、2 ジェット、レプトン対あるいは四次元運動量欠損の不変質量が Z 粒子のそれと一致するという要求をすると、ヒッグス粒子は残りの b クォークによる 2 ジェット系の不変質量分布にピークとなって現れる。このピークを探すことにより発見出来る、2 ジェット不変質量分布を用いた探索。もう一つは、$l^+l^-b\bar{b}$ モードを使い、検出した l^+l^- から l^+l^- 以外の系の質量を求めることにより、ヒッグスの崩壊モードに無関係に探索を行なうことが出来る、レプトン対に対する質量欠損を用いた探索である。

CDC への性能要請

<u>2 ジェット不変質量分布を用いた探索からの要請</u>

電子陽電子衝突過程は全重心系エネルギーが反応の素過程に使用されるために、終状態の識別が容易であり、確実な新粒子探索や精密実験ができるという特徴がある。これに加えて、JLC のエネルギー領域ではジェットのエネルギー集中がますます顕著になり、また Calorimeter のエネルギー分解能も良くなるので、トップ以外のクォークが鋭いジェットとして見えるようになるのみならず、ジェット不変質量法によるゲージボソンやトップクォークの再構成が可能となる。つまり、JLC では、ファイマン図を見るが如く、反応の終状態を基本粒子すなわちレプトン、クォーク、ゲージボソンの単位で捉えることができるようになるのである。これは、全く新しい加速器実験の幕開けである。この特筆すべき可能性を現実のものとし、加速器

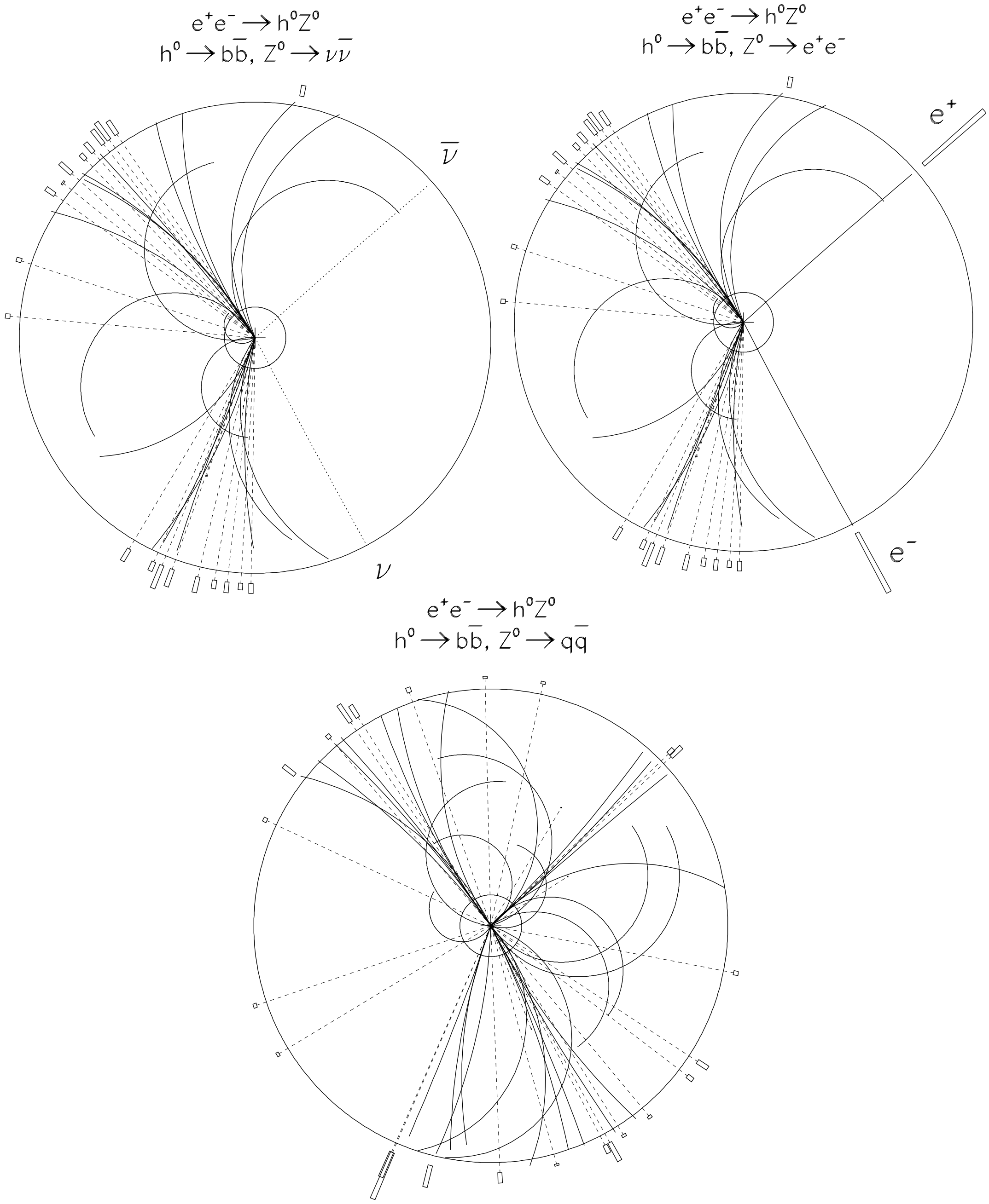


Figure 1.2: 典型的な $e^+e^-\!\!\to H^0_{SM}Z^0$ 事象の例。$m_{H_{SM}} = 120$ GeV、$\sqrt{s} = 300$ GeV とした。(a) $H^0_{SM} \to b\bar{b}$ 、 $Z^0 \to \nu\bar{\nu}$、(b) $H^0_{SM} \to b\bar{b}$ 、 $Z^0 \to e^+e^-$、(c) $H^0_{SM} \to b\bar{b}$ 、 $Z^0 \to q\bar{q}$。図 (a) および (c) では、実線は２テスラの磁場中におかれた内径 0.3 m、外径 2.3 m の飛跡検出器により測定される荷電粒子の軌跡を、点線は光子を示す。外側の箱は電磁 Calorimeter を表し箱の大きさはエネルギーに対応している。

の潜在能力を 100% 引き出すためには、終状態に生成されるニュートリノを除く全ての粒子を精度よく検出する、高性能の測定器が必要である。

そこで、W ボソンと Z ボソンは、主要な崩壊モードであるクォークジェットへの崩壊において識別可能であることを要求する。Z ボソンは W ボソンより 10 GeV 程度重く、それぞれ 2.5 GeV と 2.0 GeV の崩壊巾をもつ。従って、W と Z が 2 ジェット不変質量で分離可能であるためには、その分解能はこれらの崩壊巾と同程度でなければならない。

測定器の性能を最大限に生かしてジェット不変質量の分解能を出来る限り改善するためには、高分解能のハドロン Calorimeter を建設するのみならず、中央飛跡検出器から得られる運動量情報を利用することも重要である。特に、前節で述べたようなヒッグス粒子のレプトン対質量欠損法による測定から要求されるような高い分解能の中央飛跡検出器がある場合には、荷電ハドロン粒子のエネルギーとして Calorimeter でなく中央飛跡検出器の情報を用いた方が測定精度が向上する。すなわち、荷電粒子に関しては中央飛跡検出器、中性粒子に関しては Calorimeter と言うように、役割を分担できるのが理想である。この場合、Calorimeter のクラスターと中央飛跡検出器で検出された荷電粒子の飛跡とを対応（クラスター・トラックマッチング）させ、対応しないものに関してだけ中性粒子として Calorimeter 情報を使うことになる。Calorimeter には位置測定用のシリコンパッド（$1\mathrm{cm} \times 1\mathrm{cm}$）が装着されているが、中央飛跡検出器で検出された荷電粒子の飛跡をこのパッドの位置まで延長した際に間違いなく対応できることが必要となる。

また、ジェット中の荷電粒子に対する高分解能の実現には、複数の飛跡が重なり合って分離できない場合が生ずるという困難がある。特に、飛跡の一部の測定点が使いものにならなくなると、実質的な測定点の数と測定領域の大きさが減少し、運動量分解能を悪化させる。近接した 2 本の飛跡の分離性能に対する要求は、ジェットの混み具合がエネルギーによるので、エネルギーとともに変わるが、JLC では、2 mm 程度の距離であれば分離できることが望ましい。

<u>レプトン対に対する質量欠損法を用いた探索からの要請</u>

レプトン対に対する質量欠損法とは、初期状態の重心エネルギー（E_{CM}）が良く分かっているとき、終状態 2 体のうちの一方（今の場合 Z ボソン）のエネルギーや運動量（E_Z, p_Z）から、他方（ヒッグス粒子）の質量（M_h）をエネルギー運動量保存則より計算する方法である。すなわち、

$$\begin{aligned} M_h^2 &= (E_{CM} - E_Z)^2 - \vec{p}_Z^2 \\ &= E_{CM}^2 - 2E_{CM}(|\vec{p}_{\ell_1}| + |\vec{p}_{\ell_2}|) + 2|\vec{p}_{\ell_1}||\vec{p}_{\ell_2}|(1-\cos\theta) \end{aligned} \tag{1.1}$$

ここで、

$$\vec{p}_z = \vec{p}_{\ell_1} + \vec{p}_{\ell_2}$$

で、θ は実験室系でのレプトン対の運動量間の角度である。従って、ヒッグス質量の分解能（ΔM_h）は、(1.1) 式より角度の誤差を無視すれば

$$(\Delta M_h^2)^2 = \left(\frac{\partial M_h^2}{\partial|\vec{p}_{\ell_1}|}\right)^2 (\Delta|\vec{p}_{\ell_1}|)^2 + \left(\frac{\partial M_h^2}{\partial|\vec{p}_{\ell_2}|}\right)^2 (\Delta|\vec{p}_{\ell_2}|)^2$$

ただし、

$$\frac{\partial M_h^2}{\partial|\vec{p}_{\ell_1}|} = -2\left(E_{CM} - 2|\vec{p}_{\ell_2}|\sin^2\frac{\theta}{2}\right)$$

$$\frac{\partial M_h^2}{\partial |\vec{p}_{\ell_2}|} = -2\left(E_{CM} - 2|\vec{p}_{\ell_1}|\sin^2\frac{\theta}{2}\right)$$

となる。

ここで、しきい値近くでは $\theta \simeq 180^\circ$ かつ $|\vec{p}_{\ell_1}| \simeq |\vec{p}_{\ell_2}|$ である事を考慮すれば、ヒッグス質量の分解能（ΔM_h）はレプトンの運動量分解能（$\Delta|\vec{p}_\ell|$）に比例し、

$$\Delta M_h \simeq \sqrt{2}\frac{E_{CM} - 2|\vec{p}_\ell|}{M_h} \cdot \Delta|\vec{p}_\ell| \tag{1.2}$$

と表すことができる。

一方、ビームエネルギーの広がり（$\Delta E = \Delta E_{CM}/2$）は、0.2% 程度まで小さくできると期待でき、この広がりのヒッグス質量の分解能への寄与は

$$\begin{aligned}\Delta M_h &= \frac{E_{CM} - (|\vec{p}_{\ell_1}| + |\vec{p}_{\ell_2}|)}{M_h} \cdot \Delta E_{CM} \\ &\simeq \frac{E_{CM} - 2|\vec{p}_\ell|}{M_h} \cdot \Delta E_{CM}\end{aligned}$$

で近似できる。これは、$M_h = 100$ GeV、$E_{CM} = 250$ GeV とすると、$\Delta M_h \simeq 0.75$ GeV の寄与である。運動量分解能のから来る誤差が無視できるためには、(1.2) 式で得られる誤差がその 1/2 程度以下でなくてはならない。すなわち、50 GeV に対して 0.4% の運動量分解能が必要となる。

1.3.2 加速器からの要請

加速器

円型加速器では、電子と陽電子は円型の軌道を何周も回る。そのため、比較的弱い加速装置でも、粒子が軌道を何周もする間に、少しずつエネルギーを大きくすることができる。また、同じビームが何度も衝突するので、比較的小さなビーム強度でも反応確率を高くすることが出来る。例えばつくばのトリスタンや欧州の LEP などは、この方式の加速器である。しかしこの方式では、高エネルギーの電子が曲げられるとき放射光を発生してエネルギーを失うため、到達できるエネルギーには限界があり、LEP-II 以上のエネルギー、すなわちで 200 GeV を超える重心系エネルギーを実現するのは困難である。一方、線形加速器は、前段部のごく低エネルギーの部分をのぞいては、曲線部を持たない。このため、放射光によるエネルギー損失は原理的にない。従って、これまで円型加速器では、到達できなかった高エネルギーを実現できる。しかし、直線である為、電子や陽電子は加速部分を１度しか通らないし、それらは１回の衝突にしか使えない。そのため

- これまでの加速器の約 10 倍、強く加速すること、
- 非常に多数のバンチ (電子、陽電子の塊) を次々と発生させ、それを加速すること、
- ビームを衝突させるときに非常に小さく (240 nm × 3 nm) 圧縮すること、

が必要である。現在 JLC では、図 1.3 で示されているような、加速器が考えられている。また、パラメーターは、表 1.2 に示す。

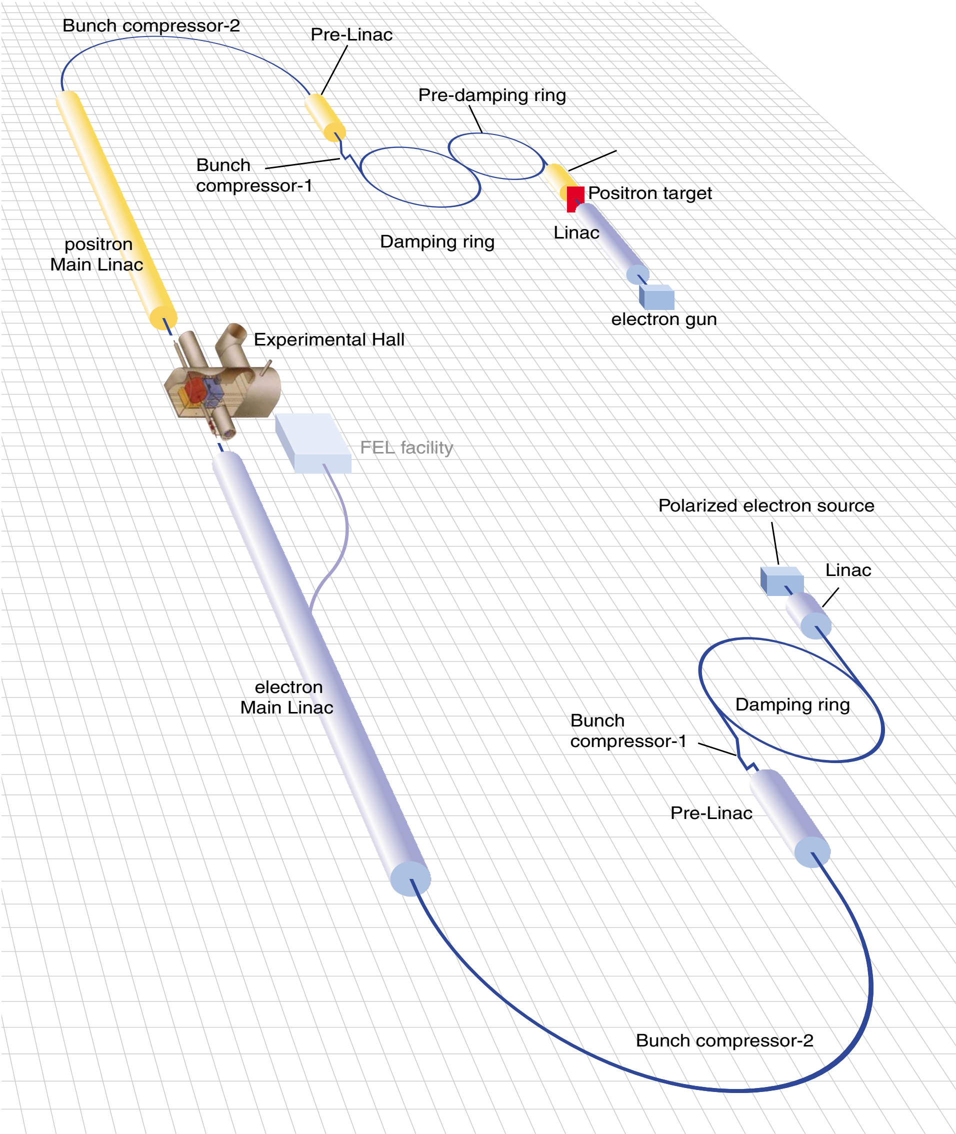


Figure 1.3: JLC の加速器の概略図

Table 1.2: E_{CM} =500 GeV におけるパラメーター

		A	B	C	X	Y	
Beam parameters							
Center-of-mass energy	E_{CM}	535	515	500	497	501	GeV
Repetition rate	f_{rep}			150			Hz
Number of particles per bunch	N	0.75	0.95	1.10	0.55	0.70	10^{10}
Number of bunches/RF Pulse	n_b		95		190		
Bunch separation	t_b		2.8		1.4		ns
R.m.s. bunch length	σ_z	90	120	145	80	80	μm
Normalized emittance at DR exit	$\gamma\varepsilon_x$		300		300		10^{-8}m·rad
	$\gamma\varepsilon_y$		3.0		2.0		10^{-8}m·rad
Main Linac							
Effective Gradient[1)]	G_{eff}	59.7	56.7	54.5	54.2	50.2	MV/m
Power/Beam	P_B	4.58	5.58	6.28	6.24	7.99	MW
Average rf phase	ϕ_{rf}	10.6	11.7	13.0			deg.
Linac Tolerances	y_c	16.1	15.2	14.6	18.	14.	μm
Number of DLDS nonets			23			25	
Number of structures per linac			2484			2700	
Number of klystrons per linac			1656			1800	
Active linac length			4.47			4.86	km
Linac length			5.06			5.50	km
Total AC power	P_{AC}		118			128	km
IP Parameters							
Normalized emittance at IP	$\gamma\varepsilon_x$	400	450	500	400	400	10^{-8}m·rad
	$\gamma\varepsilon_y$	6.0	10	14	4.0	4.0	10^{-8}m·rad
Beta function at IP	β_x	10	12	13	7	7	mm
	β_y	0.10	0.12	0.20	0.08	0.08	mm
R.m.s. beam size at IP	σ_x	277	330	365	239	239	nm
	σ_y	3.39	4.88	7.57	2.57	2.55	nm
Disruption parameter	D_x	0.0940	0.117	0.136	0.0876	0.112	
	D_y	7.67	7.86	6.53	8.20	10.43	
Beamstrahlung param	$\langle\Upsilon\rangle$	0.14	0.11	0.09	0.127	0.163	
Beamstrahlung energy loss	δ_B	4.42	4.09	3.82	3.49	5.22	%
Number of photons per e^-/e^+	n_γ	1.10	1.20	1.26	0.941	1.19	
Nominal luminosity	$\mathcal{L}_{00}$	6.82	6.41	4.98	11.15	18.20	10^{33}cm^{-2}s^{-1}
Pinch Enhancement[2)]	H_D	1.444	1.392	1.562	1.389	1.483	
Luminosity w/ IP dilutions	$\mathcal{L}$	9.84	8.92	7.77	15.48	27.0	10^{33}cm^{-2}s^{-1}

1) Effective gradient includes rf overhead (8%) and average rf phase $\langle\cos\phi_{rf}\rangle$.

2) H_D includes geometric reduction (hour-glass) and dynamic enhancement. The focal points of the two beams are made separated to each other by about $1\sigma_z$ for higher H_D (~10%)

加速器からの要請

以上に述べたように、電子・陽電子線形衝突型加速器には、今までの円形加速器にない特徴があり、これが中央飛跡検出器に今までにない制約を与えることになる。特に、ビームをナノメータ程度まで絞り込むための最終収束電磁石系は、衝突点を囲む測定器システムの中に入り込み、その一部として扱われる。ナノメータのビームを安定に衝突させるためには、最終収束系にもナノメータの精度の安定性が要求される。特に電子側の電磁石と陽電子側の電磁石の相対位置がナノメータの精度で制御されていなくてはならない。そのため、現在の JLC の最終収束系の設計では、電子側の最終収束電磁石と陽電子側の最終収束電磁石は同一の CFRP 製の円筒（サポートチューブ）の中に納められることになっている。このサポートチューブの半径は約 40 cm なので、中央飛跡検出器の内径はそれ以上となる。これが第一の要請である。

第二の要請は、ビームのバンチ構造から来る。すでに述べたように、JLC では反応確率を高めるため、一度に多くの（200 個程度）のバンチ（電子あるいは陽電子のビームのかたまり）を加速し、衝突させる。これらのバンチの間隔は、現在の X-バンドの主線形加速器の設計では 1.4 nsec であり、これらの約 200 個のバンチの列（バンチトレイン）が 150 Hz で交差する。JLC では、ビームの大きさがあまりに小さいために、一度のバンチトレインの交差の際に、どれかのバンチでいわゆるミニジェット反応が起きる確率が無視できない。バンチトレイン内のバンチが時間的に区別できなければ、これら約 200 個のバンチの中で起きた反応は全て重なり合って測定されることになる。これは、電子・陽電子衝突の特徴である反応のきれいさを大きく損なうので、是非とも避けなくてはならない。

もし、中央飛跡検出器によって測定された荷電粒子の飛跡がどのバンチから来たものかを特定できれば、バックグラウンドであるミニジェット反応が、目的とする信号反応と同じバンチで起こった場合を除いて、これを分離できる。計算によれば、ミニジェットバックグラウンドを注目する物理イベントあたり 1 イベント以下に抑えようとすると、中央飛跡検出器単独では、最低 10 バンチを分離できるバンチ分離能が必要になる。

1.4 中央飛跡検出器 (CDC) の基本設計

1.4.1 中央飛跡検出器 (CDC) の幾何学的構造

1.4.2 要求されるワイヤーあたりの局所的性能

チェンバーの全体構造が決まったので、次にこの JLC CDC 全体への性能要求を、達成すべきワイヤーあたりの性能に焼き直す。

検出器の性能の下限は物理的要請から決まってしまうので、ここで考慮すべき問題は、その性能を保証するための測定精度を達成出来るか否か、である。計算の結果、その測定精度が、現在の技術力からみて、明らかに達成不可能なものであれば、設定された基礎デザインに問題がある。ただちに再度物理要請に立ち戻り、基礎デザインを練り直さなければならない[4]。

一方、要求される測定精度が達成可能と見積もられる場合には、その測定精度を実機で達成出来るか否かを実験によって確かめなくてはならない。ここに至って、テストチェンバー等を使用した R&D 実験が開

[4]実際には、基礎デザインを練る段階で当然このことは考慮されている。基礎デザインの設定は物理的要求と実現可能な測定精度を天秤にかけながら行われるものであり、基礎デザインが完成した時点で、測定精度に対する境界条件は満たされているべきである。

始される。R&D の結果、必要な測定精度が満たせないことが判明した場合には、基礎デザインにたち戻って再度デザインの検討を行わなければならない。

R&D 項目及び R&D 現状については次節に述べることとし、ここでは基礎デザインから予想される JLC-CDC の性能と、その性能を達成するために必要な測定精度について列挙する。

運動量分解能

運動量分解能に対する最も厳しい要求は、$e^+e^- \to ZH$ の反応でつくられた Z がレプトン対に崩壊した時の、レプトン対から計算される質量欠損の分解能に対する要求から決まる。標準模型で期待されるヒッグスの崩壊幅は MeV のオーダーであるので、検出器側も出来る限りその大きさに近付きたいところである。実際には、ビームエネルギーの幅が 200MeV 程度存在するので、検出器側では少なくともこのビームエネルギーの広がりと同程度の運動量分解能を持たねばならない。ヒッグス質量の分解能 ΔM_h はレプトンの運動量分解能 ΔP_ℓ に比例し、$\Delta M_h \simeq 2P_\ell/M_h \cdot \Delta P_\ell$ と表すことができるため、M_h =100 GeV に対する ΔM_h =200 MeV の要求は、$P_\ell \simeq$50 GeV として、ΔP_ℓ =200 MeV すなわち 50 GeV に対して 0.4% の運動量分解能の要求となる。この運動量分解能が達成されれば、5TeV の荷電粒子に対してその電荷を判別出来ることになる。

横方向の運動量 P_t の逆数 $\kappa = 1/P_t$ に対してその分解能 σ_κ は

$$\sigma_\kappa^2 = (\sigma_\kappa^{meas})^2 + \left(\sigma_\kappa^{MS}\right)^2 \tag{1.3}$$

で与えられる。ここで右辺第一項は位置測定の誤差、第二項はチェンバー ガスでの多重散乱によるものであり、各々

$$\begin{array}{ll} \sigma_\kappa^{meas} & \simeq \left(\frac{\alpha\sigma_x}{Bl^2}\right)\sqrt{\frac{720}{n+4}} \\ \sigma_\kappa^{MS} & \simeq \left(\frac{\alpha C}{Bl}\right)\sqrt{\frac{10}{7}\left(\frac{X}{X_0}\right)} \cdot \kappa \end{array} \tag{1.4}$$

と書ける。ここで $\alpha = 333.56$ ($\mathrm{cm \cdot T \cdot GeV^{-1}}$)、$C = 0.0141$ (GeV)、 L は測定される飛跡の長さ (cm)、B は磁場の強さ (T)、$\left(\frac{X}{X_0}\right)$ は輻射長で表したチェンバー ガスの厚さ、n は測定点の数である。

ここに、磁場 2 T のパラメータとして、B=2Tesla、$\sigma_x = 100\mu$m、n=80、X/X_0=1.1% (CO_2/isobutane)、l=200cm を選ぶ。$\sigma_x = 100\mu$m というのは、CO_2/isobutane ガスで約 5cm のドリフト距離を持つセル構造の場合の平均位置分解能であり、l=200cm は 2T の場合の CDC の外筒から内筒までの距離である。すると、σ_κ^2 及び $\left(\frac{\sigma_{p_T}}{p_T}\right)^2$ は、

$$\sigma_\kappa^2 = (1.1 \times 10^{-4})^2 + (1.5 \times 10^{-3} \cdot \kappa)^2 \tag{1.5}$$

$$\left(\frac{\sigma_{p_T}}{p_T}\right)^2 = (1.1 \times 10^{-4} \cdot p_T[GeV])^2 + (1.5 \times 10^{-3})^2 \tag{1.6}$$

となる。

同様にして、3T では、B=3Tesla、n=50、測定点が減った分空間分解能の上限を厳しくして $\sigma_x = 85\mu$m、l=110cm とすると、

$$\sigma_\kappa^2 = (2.9 \times 10^{-4})^2 + (1.7 \times 10^{-3} \cdot \kappa)^2 \tag{1.7}$$

$$\left(\frac{\sigma_{p_T}}{p_T}\right)^2 = (2.9 \times 10^{-4} \cdot p_T[GeV])^2 + (1.7 \times 10^{-3})^2 \tag{1.8}$$

となり、3T では若干悪化するが、この悪化分は衝突点座標拘束 (IP Constraint) を課すと回復する。IP Constraint を課すことは、Vertex の情報を用いることと同等の効果であるので、実験的には Vertex とトラックをつなぐことによって、運動量分解能を向上することができる。

以上より、R&D 項目としては、ドリフト領域全域にわたって、磁場２ T では平均 $\sigma_x = 100\mu$m、3T では平均 $\sigma_x = 85\mu$m の空間分解能が保証されれば、運動量分解能に対する CDC への要求を満たすことができる。

Dip Angle

ジェットの不変質量を精度よく求めるためには CDC の飛跡とカロリーメータのクラスターの間に正しく対応を付けることが重要である。カロリーメータでは 1cm×1cm のシリコンパッド測定器を用いてシャワーの位置を測定するので、CDC の飛跡のカロリーメータ上での位置はこれよりも良い精度で決定したい。円周方向の位置については問題なく達成出来るので、z 方向の位置精度について考える。CDC とカロリーメータの間には CFRP でできた CDC の外壁（0.5cm）が存在するがそれほどの物質量ではなく、実際 5GeV 以上の粒子に対しては多重散乱の影響はほとんどない。計算によれば、２ T で７層、3T で 6 層のステレオ層がそれぞれ z 方向に 1mm の位置測定精度を達成できればカロリーメータの表面まで延長した飛跡の z 位置精度も 1mm の精度が達成出来る。

近接飛跡分離能

質の良い飛跡再構成を行うためには２本の近接した飛跡を分離出来る分解能が重要である。この性能を 250GeV の W 粒子が崩壊して出来る粒子を用いて調べた。これによるともし 2mm 以下しか離れていない２個のヒットは区別できないとすれば、最内層においては 5%、最外層においては 1%のヒットが２個のヒットを分離することに失敗することにより失われる。この程度の損失は飛跡再構成にほとんど影響を及ぼさないので、2mm の近接ヒット分離能が達成できればよい。2mm は時間にして 100ns に相当するが、計算によれば信号のすそは立ち上がりから 100ns 後には約半分に減少しているので容易に識別できる。従って、2mm の近接ヒット分離能は十分達成可能である。

Bunch Tagging

JLC では 150Hz の１パルス中に 1.4 nsec 間隔で約 200 ビームバンチが衝突する。このとき、本物の信号事象の他に、次に述べるようにビーム・ビーム相互作用によるバックグラウンド事象や、ミニジェットと呼ばれるハドロニック・バックグラウンド事象が起こり得る。このとき、セルが互い違いに配置されているミニジェットセル型 CDC では、図 1.4 のように、現在注目しているバンチ以外で生成されたトラックは、セルの間でうまくつながらない。これは、バンチ同士が衝突した時刻 T_0 を粒子の通過位置を求めるときに使っているためである。従って、事象の発生したバンチを同定できるため、明らかに異なる T_0 をもつバックグラウンド事象は取り除くことができる。

粒子の通過位置は、その通過時間から計った各ワイヤーでの電離電子の到達時間を測定することによって定まる。電離電子のドリフト速度は、電場 1kV/cm で 7.8μm/nsec なので、各点あたり空間分解能 100μm を達成したとすると、時間分解能は 12.8 nsec になる。従ってトラックあたりの時間分解能をおおざっぱに

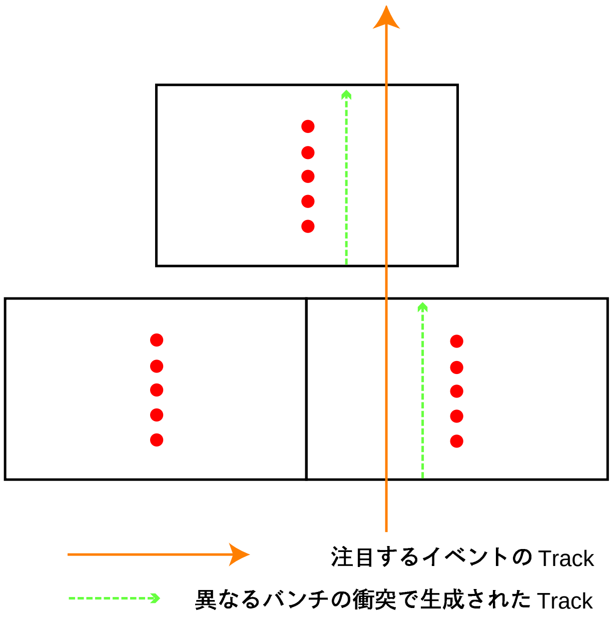


Figure 1.4: Stagger されたセルと Banch Tagging

見積もると、2T の場合には、一粒子あたり 12.8 nsec / $\sqrt{100\text{ 測定点}}$ = 1.28 nsec、3T の場合には、同様に、12.8 nsec / $\sqrt{50\text{ 測定点}}$ = 1.81 nsec となる。

Background

ビーム・ビーム相互作用によるコヒーレント及び非コヒーレントな QED 過程により、一次バックグラウンドとして電子・陽電子が発生する。これらはQ電磁石の前面やマスク等に衝突して二次バックグラウンド光子を発生する。これらの光子のエネルギースペクトルは約 100keV に幅広いピークを持つとともに、500keV には対消滅による鋭いピークが存在する。

バックグラウンドは、磁場、ビーム強度、マスクの形状によるが、現在の設計では $\sqrt{s}$=500GeV において1衝突当たり約 10^4 個の光子が CDC に飛び込むことになる。これにチェンバーガスやワイヤーでの反応断面積をかけてやると、ワイヤー1本当たりの平均ヒット数は最内層で 0.06、最外層では 0.01 以下となる。この程度のバックグラウンドは飛跡再構成にほとんど影響を及ぼさない。

ミニジェットと呼ばれる二光子過程からのハドロニック・バックグラウンドも JLC においては問題となりうる。$\sqrt{s}$ = 500GeV においては CDC のアクセプタンス内での平均ミニジェット・エネルギーは約 2.5GeV で、平均粒子数は5個である。横方向運動量が 1 GeV 以上のミニジェットの数は1バンチ衝突当たりオーダー 0.1 個と見積もられている。従って、異なるバンチが全く分離出来ないとすると、バンチトレインあたり約２００個のバンチがあるので、オーダー１０個のミニジェットが重なってしまう。以上のことから、CDC は異なるバンチを分けられる時間分解能をもつ必要がある。

1.4.3 シミュレータの必要性と本研究の目標

以上のように基礎デザインから要求される各測定量への測定精度が決まったところで、これらの測定精度が実際に達成できるか否かを見極めるための R&D が開始された。R&D ではまず、CDC の全体構造に依

存しないローカルパラメータを設定し、その後 CDC 全体の構造に直接依存するグローバルパラメータの最適化を行う。ここで、ローカルパラメータの設定は CDC の一部を取り出したテストチェンバーを作成して行うことができるが、グローバルパラメータの最適化のために実機サイズのプロトタイプを作成することは、時間、費用共両面において現実的でない。そのため、ローカルパラメータの設定からグローバルパラメータの最適化に移行する前に、シミュレータによる最適化作業が必要不可欠となるのである。シミュレータの開発段階では、同時にオフラインの解析プログラムの開発が行われる。我々の R&D はシミュレータが必要な段階まできており、現在その開発を進めている。グローバルパラメータの最適化や、測定精度を求めるためには、CDC のセル構造などを詳細に組み込んだシミュレータが必要である。我々が進めているシミュレータの構成を以下に示す。

1. イベント生成部分（粒子以前のパートンを粒子に変換）
2. Monte-Carlo Truth 生成部分（検出器シミュレーション部）
3. イベント解析プログラム（イベント再構成部）
4. 物理解析プログラム（物理解析部）

このうち、1. と 4. は既存のプログラムが存在するので、それを用いる。

2. の部分に関しては、検出器と粒子の反応に関するモンテカルロ・シミュレータである Geant4[4] をベースに開発する。ただし、Geant4 はライブラリ群であり、CDC で開発したシミュレータをやがてバーテックスやカロリーメータにも拡張することを考えると、ライブラリを生のまま用いるのは好ましくない。そこで、JLC 検出器全体を統合するようなシミュレーション・フレームワークを開発した上で、そのフレームワークを用いて CDC 部分の開発を行う。また、Geant4 ライブラリに含まれていないオブジェクトに関しては、そのつど新規開発する。

3. の部分に関しては、JLC の解析フレームワークである JSF[5] の枠組みを用いて、まず JSF の下位にシミュレータ用のフレームワークを作成する。これは、2. と同様に、他の検出器を将来統合する方向を考慮しているためである。実際の解析部分は、ローカルパラメータの測定値の解析に際して開発された解析プログラムを組み込みながら、CDC 全体のトラッキングメソッドや、シミュレータ特有の解析ルーチン（Monte-Carlo Truth の情報を利用したカンニングルーチン）を開発する。

2. の検出器シミュレータ部分は JUPITER（JLC Unified Particle Interaction and Tracking EmulatoR）と名付けられ、さらに 3. のイベント再構成部分は JUPITER の衛星プログラムとして Satellites と名付けられた。それぞれ、Geant4 と Root をベースに C++言語を用いたオブジェクト指向技術を最大限に利用した設計になっている。これまで、JUPITER はフレームワークの第一段階の開発を終え、各検出器の実装が進行中である。CDC に関しては、フィールドワイヤーを除き、アクシャル・ステレオ両レイヤーがインストール済みであり、第一段階の開発を終了している。Satellites は、Hit 作成部（HitMaker）、トラック作成部（TrackMaker）、トラックフィッティング部（TrackFitter）が実装されている。

しかし、既存の TrackFitter は、1 つのトラックパラメータでフィッティングを行うので、運動量の低い粒子が生成するトラックの場合には精度が悪くなってしまう。したがって本研究では、測定点ごとにトラックパラメータを決めることのできる Kalman Filter でのトラックフィッティングを作成することを目的とする。第 4 章で Kalman Filter の基本クラスとそれを用いた飛跡再構成プログラムを作成し、その応用として第 5 章でタイム・スタンピング能力について述べる。

Chapter 2

シミュレータについて

2.1 役割と構造

ここでは、一般的に高エネルギー物理実験で用いられるシミュレータの役割について説明する。

高エネルギー物理実験で使われるシミュレータはモンテカルロシミュレータである。これは高エネルギー物理実験においてもっとも基本的な部分を構成している。JLC を例に取ると、まず始めに目標とする重要な物理を明らかにしなければならない。このターゲットが決まると、次はビームエネルギー、ルミノシティ、バックグラウンドなどのマシンパラメータの設定、さらにそれを受けて運動量分解能、飛跡再構成の精度、エネルギー分解能、粒子同定の精度など、検出器のパラメータの設定が行われる。これらすべてがモンテカルロシミュレータの最初の仕事である。こうして得られたパラメータは、数々の R&D を経て最終的には TDR(Technical Design Report）に統合される。モンテカルロシミュレータは、プロジェクトの重要な３つの根幹である物理、加速器、検出器のパラメータを相関づける唯一の手段であり、それゆえに十分な計算速度と精度をもって、リニアコライダー実験の特性をシミュレートできるものでなくてはならない。

モンテカルロシミュレータの構造を図 2.1 に示す。まずはじめに、ビーム相互作用と Initial State Radiation を経て生成されるパートン（４元運動量で表される）からスタートする。パートンはすぐにシャワー、ハドロン化、崩壊などを起こし、最終的には安定粒子になる。これを２つの４元ベクトル (x^μ, p^μ) で表す。これらの粒子は、検出器の中を、検出器中の様々な物質と相互作用しながら走って行く。このとき粒子は、飛跡検出器中では（荷電粒子であれば）通常 Helix のパラメータで表される螺旋を描き、ヒットの形で飛跡を残す。カロリーメータの中では、それぞれ粒子の特性に従ってエネルギークラスターの形でシャワーの痕跡を残す。更に、必要であれば、これらのヒットやクラスターを ADC や TDC の信号の形に変換することもできる。

ところで、この全く逆を辿ることが可能であることは、明らかであろう。この逆の過程は、ADC や TDC の信号から始まって、最終的にはビーム衝突点で反応が起こった直後のパートンにまで戻っていく。これが、イベント再構成と解析の作業である。飛跡検出器の場合には、まず ADC と TDC のデータから、検出器のどこを粒子が通ったかについての情報（ヒット点）を作成する。次に、どのヒット点が同一の粒子から作られたものかについてのグループ分けを行い（Track Finding)、更に、グループ分けした点の集合を Fitting して Helix パラメータに変換する（Track Finding)。同様にして、カロリーメータでも個々のタワー

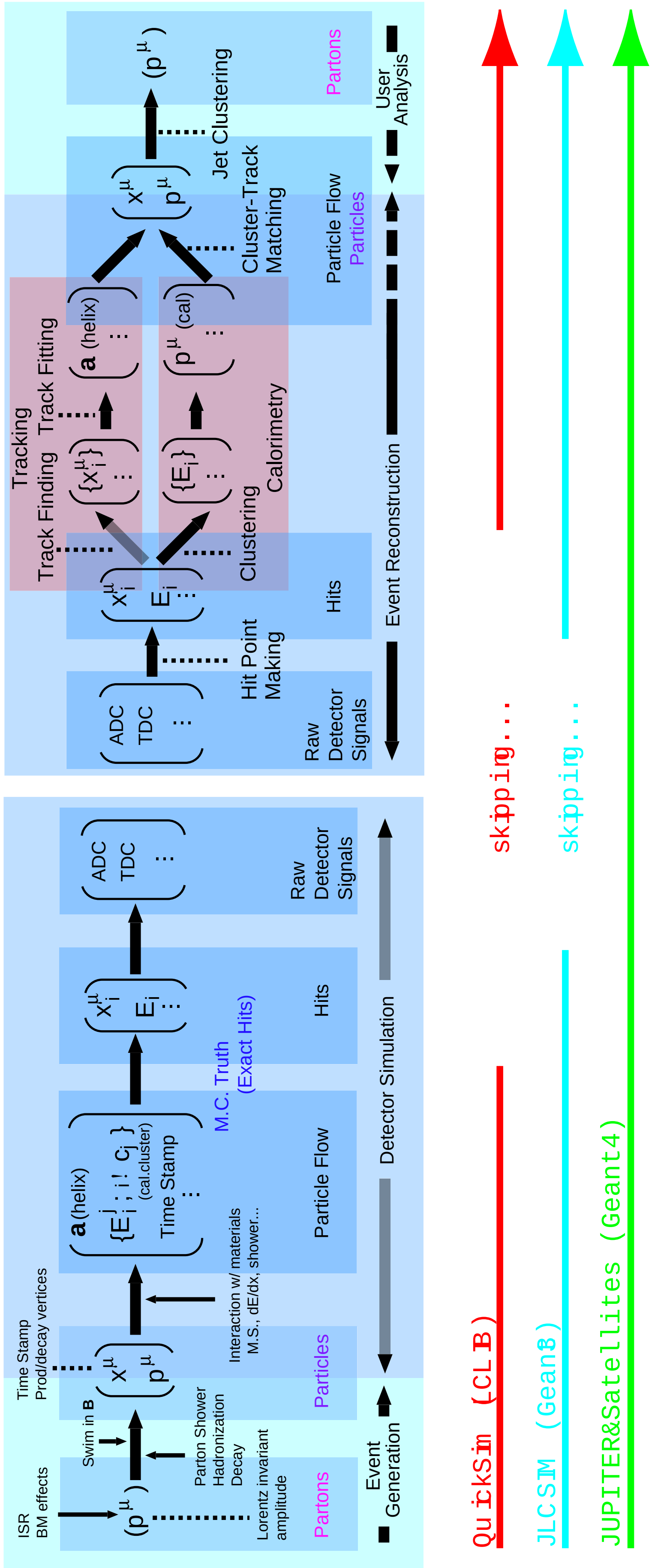


Figure 2.1: モンテカルロシミュレーションの流れ

の信号からもクラスターを再構成する（Clustering)。このようにして再構成された飛跡とクラスターを対応させ、Energy Flow の分解能を向上してから、再び２組の４元ベクトルで粒子の初期状態を表す。このような粒子の情報を多数集めて、ジェットの再構成を行い、最終的にはもとの 4 元運動量で表されるパートン再構成を目指す。

以上が、実験を、可能な限り忠実にモンテカルロシミュレーションに焼き直したときの手順である。このようなシミュレーションをフルシミュレーションと呼ぶ。しかし、シミュレータの使命はただ実験を忠実に再現することだけではない。このように計算機技術が発展した現在でさえ、コンピュータが１つのイベントを生成する速度は自然のそれに敵うべくもないが、それでもいくつかの過程を省略することによって、若干シミュレーション時間を早くすることが出来る。このとき、どの部分を省略するかによって、シミュレーションのレベルを変えることが出来る。JLC では、図 2.1 の下部に示した通り、シミュレーションレベルによって、QuickSim、JIM [7]+Analysis Program、JUPITER&Satellites（開発中）などのシミュレータが存在する。

2.2 検出器シミュレータの種類

2.2.1 クイック シミュレータ

クイック シミュレータとは、文字どおり、計算時間の短いシミュレータである。短い時間で統計を多くためる必要のある物理解析や、加速器や測定器の基本パラメータの設定等、パラメータをふりながらおおまかなデザインの設定を行うときなどに用いられる。クイック シミュレータが早いのは、図 2.1 に示したとおり、測定器に関するシミュレーションの多くをスキップしているからであるが、そのときにスキップした場所で混入し得るエラーをどのように組み込むかによって、２通りのアプローチが存在する。ここでは、そのいずれも、Helix パラメータ以降のシミュレーションを行わないものとし、それら二つの特徴と役割について、主に飛跡再構成部分を例に述べる。

まず第１は、Helix パラメータを、誤差行列を用いてぼかす方法である。Helix までしか作らないシミュレータでも、全体で測定点がいくつになるか、１点あたりの位置分解能がどれくらいあるか、等の情報があれば、解析的な式によって、ヒットから Helix Track を再構成したときにどのくらいの統計誤差が混入するかを計算できる[1]。Helix の５つのパラメータ（通常 $d_\rho, \phi_o, \kappa, d_z, \tan\lambda$ で表される）をベクトル $\mathbf{a}$ で表し、その誤差行列を $E_{\mathbf{a}}$ で表すことにする。測定におけるカイ２乗は、

$$\chi^2 = \frac{1}{2}\Delta\mathbf{a}^T \cdot E_{\mathbf{a}}^{-1} \cdot \Delta\mathbf{a}. \tag{2.1}$$

で表せるが、$E_{\mathbf{a}}$ は対称行列であるので対格化できる。これを $E_{\mathbf{b}}$ と表す。

$$E_{\mathbf{b}}^{-1} = O^T \cdot E_{\mathbf{a}} \cdot O = \begin{pmatrix} 1/\sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & 1/\sigma_n^2 \end{pmatrix}. \tag{2.2}$$

これを用いれば、χ^2 は、

$$\chi^2 = \frac{1}{2}\Delta\mathbf{b}^T \cdot E_{\mathbf{b}}^{-1} \cdot \Delta\mathbf{b}, \tag{2.3}$$

[1] したがって、この方法では、統計誤差以外の誤差がトラックの再構成に及ぼす影響を見積もることはできない。

となる。ここで、$\Delta\mathbf{b}$ は、

$$\Delta\mathbf{b} = O^T \cdot \Delta\mathbf{a}. \tag{2.4}$$

である。

ところで、$\Delta\mathbf{b}$ の成分は互いに独立であるから、これらの成分は独立にガウス関数で振ることができて、

$$\Delta \mathrm{b}_i = \sigma_i \cdot (\text{Gaussian random number with unit width}) \tag{2.5}$$

と書ける。式 (2.4) と (2.5) から、$\Delta\mathbf{a}$ が計算できる。したがって、Helix パラメータは $\Delta\mathbf{a}$ を用いて次のようにぼかすことができる。

$$\begin{aligned} \mathbf{a} &= \mathbf{a}_{\mathrm{true}} + \Delta\mathbf{a} \\ &= \mathbf{a}_{\mathrm{true}} + O \cdot \Delta\mathbf{b} \end{aligned} \tag{2.6}$$

注目すべきは、このようにパラメトライズすることによって、$\mathbf{a}$ の成分同士の相関を考慮することが出来る点である。

この方法は、検出器の基本パラメータ（CDC の内筒、外筒の大きさ、１測定点あたりの位置分解能など）が分かっていれば、誤差行列を計算するだけで Tracking の誤差を見積もることが出来るので、逆にこれらの基本パラメータを変化させたときに、どれだけ Tracking の精度が変化するかを見積もる作業に適している。したがって、検出器の基本仕様を設定する際に用いられることが多い。

第２の方法は、Helix パラメータをぼかす作業を、もう少し経験的手法によって行うものである。Helix パラメータを振る部分に、解析的な式から見積もった誤差ではなく、フルシミュレータによって見積もられた誤差を代入する。フルシミュレータでは、統計誤差だけでなく、Track Finding、Track Fitting の際に混入する誤差も見積もることができるので、このとき代入される誤差の値は、統計誤差と系統誤差、アルゴリズムからくる誤差を全て合わせた値である。第１の方法が一般に検出器の性能を良く見積もりすぎであるのに対し、この第２の方法ではより実験に近い形の誤差が考慮されている上、フルシミュレータよりは計算速度が圧倒的に早いことから、主に物理計算ではこの第２の方法がとられる。

JLC のクイック シミュレータは、この両者の折衷案ともいうべき形をしており、基本的には第１の方法により誤差を計算するが、その誤差に後からより実際的な誤差を加えることができるようになっている。次節に述べるフルシミュレータからの誤差を適切に加味すれば、第２の方法で作られたシミュレータと同じ性能を発揮することができる。図 2.2 に、JLC のクイック シミュレータによる $e^+e^- \to ZH$（重心系 350GeV）の Event Display を示す。

2.2.2 フルシミュレータ

クイック シミュレータが検出器の基本仕様を設定したり、物理計算を行う際に用いられるのに対し、フルシミュレータの使命はほとんど検出器の具体的な開発と、実際の実験における誤差の見積もりに集中している。まず、フルシミュレータは、検出器の全ての物質（構造）を、可能な限り実物に近い形でインストールしたものでなければならない。これによって、次の３つの特性が顕著になる。

1. 粒子と物質との多重散乱による誤差を正確に見積もることができる。

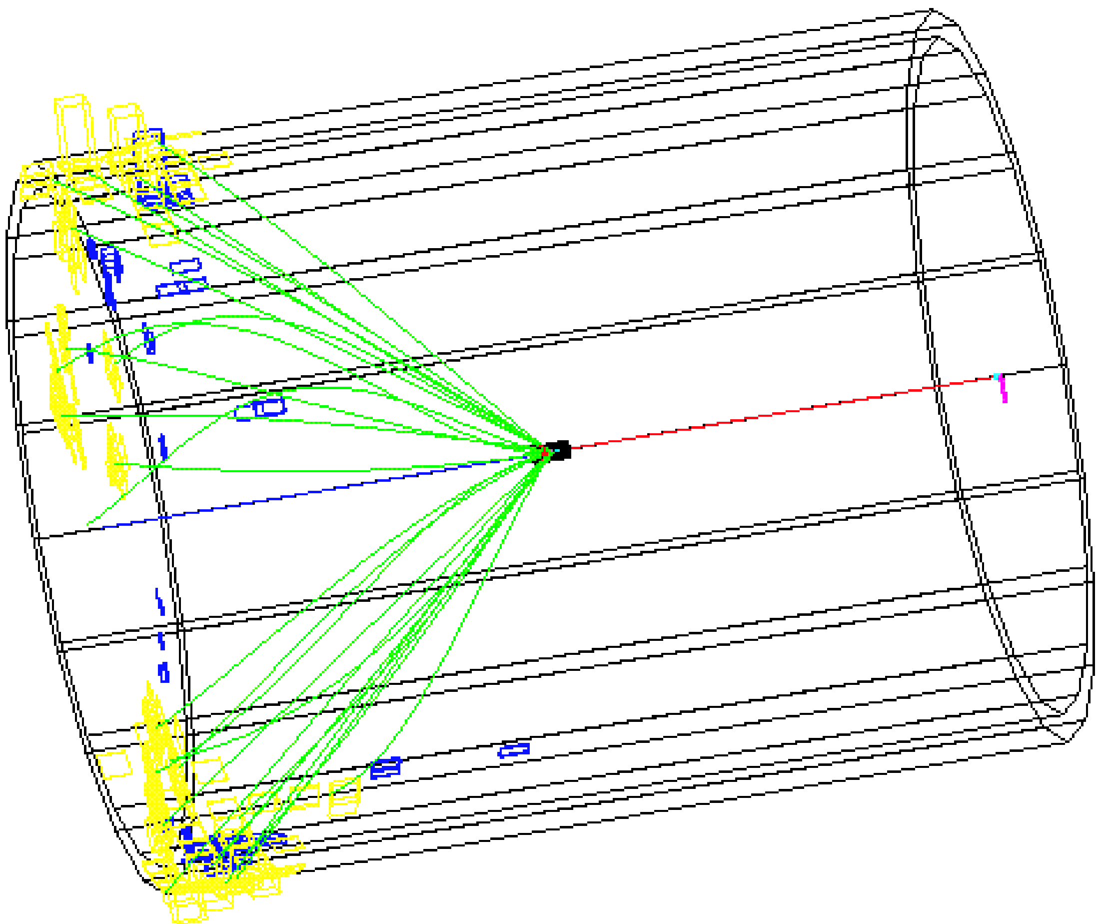


Figure 2.2: $e^+e^- \to ZH$ の 2 jet event。ヒッグスが 2 つの b クォークに、Z が 2 つのニュートリノに崩壊した場合。

2. 検出の方法によって検出効率が落ちる部分が、検出器全体に及ぼす影響を見積もることができ、具体的な検出器の構造やパラメータについての検討が可能になる。

3. 実際の信号検出の手続きを踏んでおり、実際の信号とほとんど同じ形のシミュレーションアウトプットが得られるため、イベント再構成のアルゴリズム検討が可能になる。

まず 1. は、クイック シミュレータの不足を補う意味で重要である。クイック シミュレータにも多重散乱の効果は入っているが前述の通り Helix パラメータに対しぼかしを入れる、といった手法なので、物質の違い（例えば CDC の内筒と内側のガスなど）によって局部的に多重散乱が大きくなったりする効果は見積もることができない。したがって、検出に関係のない部分も、物質のあるところは全てインストールすることが重要である。

2. は、例えば CDC の Cell における位置分解能の位置依存性などをあらかじめテスト実験で求めておき、この値をフルシミュレータに代入することによって、CDC 全体でどのくらいの分解能が得られるかについての情報が得られるという意味である。重要なことは、この解は一通りではなく、Cell の配置によって変わり得るということである。巨大な検出器全体について、もっともよい運動量分解能が得られる Cell の配置を試行錯誤できる場は、フルシミュレータをおいて他にはない。

3. は、イベントを再構成する際に混入する誤差を見積もり、イベント再構成のアルゴリズムを発展させる意味で非常に重要である。多重散乱による誤差が、物質と粒子の相互作用に起因する検出アルゴリズムに無関係の誤差であるとすれば、イベント再構成に於ける誤差は、検出アルゴリズムに直結した誤差である。したがって、これを完全になくすことはできないが、可能な限り減らす努力は行わなければならない。このシミュレータで培ったアルゴリズムは、そのまま実際の実験の解析プログラムに移植される。

2.3 本研究の位置付け

本研究は現在開発中のフルシミュレータ（JUPITER&Satellites）の特性 3 の部分にあるイベント再構成に必要な、トラックフィッティングやバーテックスフィッティングに Kalman Filter を応用することを目的としたものである。具体的には、Kalman Filter の基本アルゴリズムを含んだ基本クラスを作成し、その基本クラスを継承してトラックフィッティングやバーテックスフィッティングを行うクラスを実装する。また、そのトラックフィッティングも様々な検出器に対応させるために、トラックフィッティングの基本クラスを作成し、各検出器毎に詳細を変えて実装できるようにした。

このように Kalman Filter によるイベント再構成プログラムを用いることで、トラックパラメータの測定精度を向上させることができ、より詳細な R&D を行うことができるようになる。

Chapter 3

JUPITERとSatellitesの枠組み

3.1 シミュレータの構造と役割分担

このシミュレータはまず以下を目標として作られた。CDC の 3T デザインにおける飛跡検出器のフルシミュレーションを行うと同時に、他の検出器グループが、自由にそれぞれの検出器部分をインストールできる仕様にする。また、オブジェクト指向言語の特徴を生かし、部品を入れ替えたり、検出器のデザインパラメータを変更したりといった、検出器のデザインの変更が簡単に行えるようにする。（デザインパラメータチューニングのため）。更に、イベント再構成段階で混入する誤差を正確に見積もるため、最終的にはほぼ実験データ解析用のプログラムと同じ行程をシミュレートできる仕様にする。また、３ T デザインが提案された原因ともなった、ビームラインからのバックグラウンドが検出器に与える影響を見積もるため、加速器の衝突点周辺の構造をもシミュレータに組み込み、バックグラウンド Study を加速器と統合して行えるようにする。

以上を踏まえて設計したシミュレータのおおまかな構造と役割分担を図 3.1 に示す。主に検出器物質と粒子の反応部分のプログラムを軽くするため、Monte-Carlo Exact Hit （より一般的には Monte-Carlo Truth と呼ぶ）を出力するところで一度区切る仕様になっている。Monte-Carlo Truth を生成する部分 (JUPITER) は Geant4 をベースに開発されている。本研究ではイベント再構成部分 (Satellites) について、実験で使われている Test Chamber 用解析プログラムを土台にして、ROOT/JSF による 1Cell に 5 本のワイヤーを持つ CDC のためのプログラムを開発した。この内、特に Kalman Filter を用いたトラックフィッティングの部分は、本研究で新たにゼロから開発したものなので、章を改めて詳述する。

この章の次節以降では、まずオブジェクト指向の概念について簡単に説明した後、フレームワーク「JUPITER」「Satellites」「URANUS」について述べる。

3.2 オブジェクト指向プログラミングの概念

従来の FORTRAN や C などの手続き型プログラミング言語を用いたプログラミングでは、解決しようとする問題に現れる様々な物の状態を、基本的には組み込み型変数（例えば整数型、単または倍精度実数型、単または倍精度複素数型、文字型など）のレベルで表現し、また、それらの物がどういう振る舞いをするか

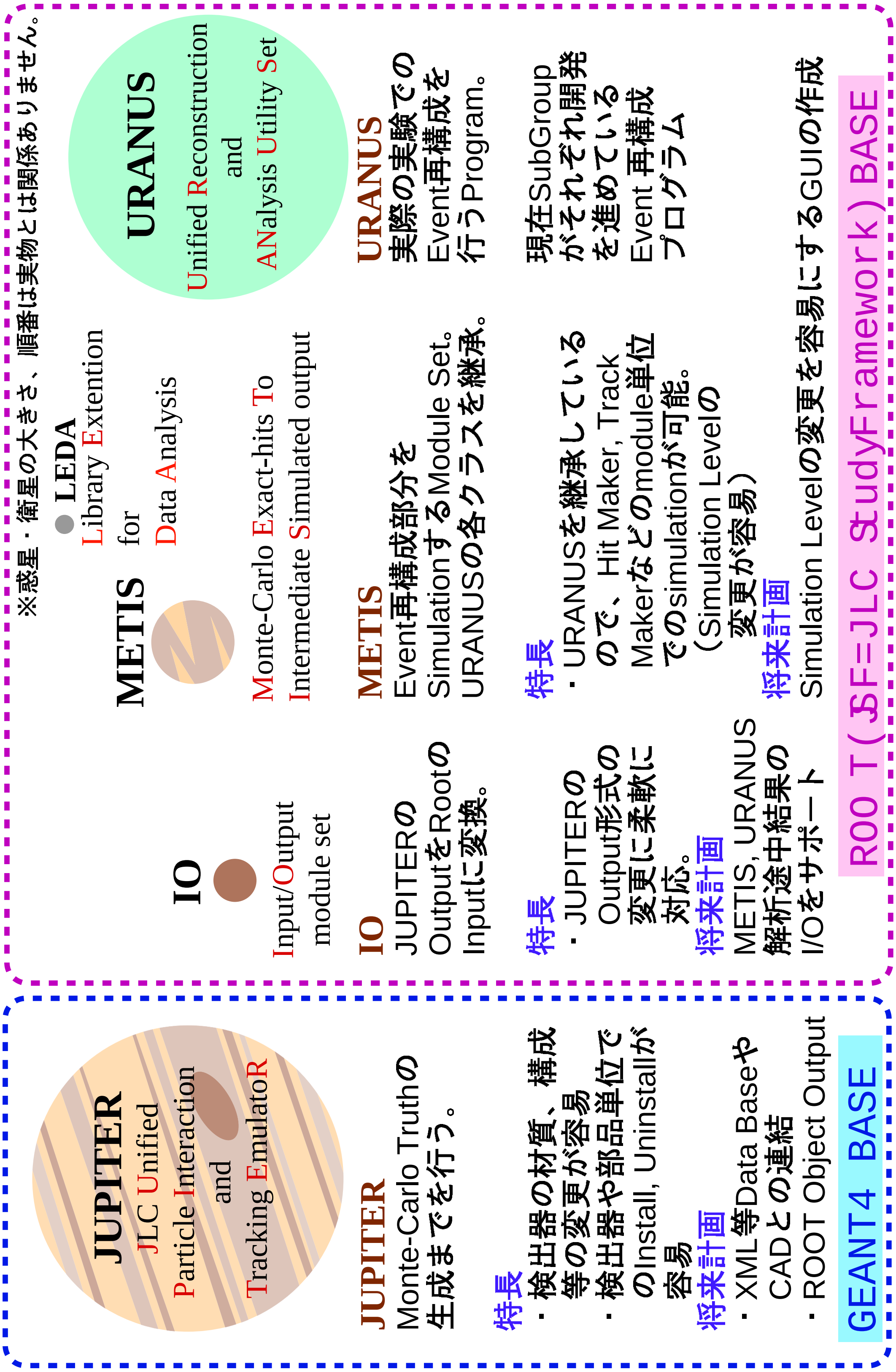


Figure 3.1: JUPITER、Satellites、URANUS の機能

はその物とは切り離された形で実装することが一般的であった。

C++ などのオブジェクト指向プログラミング言語を用いたプログラミングの本質は、問題に現れる物あるいは概念（オブジェクト）を、高度に抽象化されたユーザー定義型（そのオブジェクトの状態を表現するデータとそのオブジェクトの振る舞いを表現する演算あるいは関数を一体にしたもの）で自由に表現でき、それらのオブジェクトの細部にとらわれることなく、そのオブジェクトを単位として、より抽象的なレベルで思考、プログラミングできる点にある。細部を考慮しなくてよいと言うことは、細部をユーザから隠蔽することを可能とする。これは、ユーザーに公開した部分に変更を及ぼさない限り、自由に実装を改良したり拡張したりできることを意味する。本論では、ユーザー定義型をクラスと総称し、メモリー上に確保されたクラスのインスタンスをオブジェクトと呼ぶことにする[1]。また、オブジェクトの状態を表すデータ変数（これ自体ユーザ定義型でもよい）をそのクラスのデーターメンバー、オブジェクトの振る舞いを表す関数を、メンバー関数、あるいはメソッドと呼ぶ。

従来の FORTRAN においても、組み込み型にはある意味でオブジェクト指向の面がある。例えば、先に述べた組み込み型である整数、あるいは実数、複素数の間には四則演算が可能である。しかし同じ足し算であっても、整数と整数の和の場合と整数と実数の和の場合では実際には異なることが行われているはずである。これは＋という同じ演算子が、どういう対象に作用するかでその振る舞いを変えること、つまり対象とそれが関与する関数の連結が起きていると見なせる。同様に、同じ sin(x) という関数であっても、x が単精度なのか倍精度なのかで関数の実体は違っているはずであるが、ユーザーはそのことを気遣う必要はない。しかし、ユーザー定義の同名の関数に引数の型に従って異なる振る舞いをさせることは FORTRAN ではできない。ユーザー定義の同名の関数にコンテクストによって異なる振る舞いをさせられること（関数の多重定義の可能性）も、C++ などのオブジェクト指向言語を用いる大きなメリットである。

さらに重要なのは継承の概念である。既存のユーザー定義型（クラス）は、それを継承することで、既に実装されたそのクラスのデーターメンバーやメンバー関数を再度実装することなく受け継ぎ、新たな機能あるいは変更したい機能のみを実装することで容易に拡張できる[2]。これは、複数のクラスに共通する性質を、基底クラス（ベースクラス）にまとめることで、コーディングの量を大幅に軽減することを可能とする。別のクラスを継承したクラスをそのクラスの派生クラスと呼ぶ。

継承で重要な概念に仮想関数の多態性がある。仮想関数とは、ベースクラスに定義されたメンバー関数であって、ベースクラスのオブジェクトとしてのコンテクストで呼ばれる場合においても、その実体（つまり派生クラス）に同じインターフェースを持つメンバー関数が存在する場合、実体の方が呼ばれるものをいう。この場合、派生クラスが何であるかによって、同名の関数が異なる振る舞いをすることになる。これを多態性（ポリモルフィズム）と呼ぶ。特にベースクラスに仮想関数が実装されていない場合、その仮想関数を純粋仮想関数、また、純粋仮想関数を持つようなベースクラスを仮想クラスと呼ぶ。純粋仮想関数を定義することは、それを継承する派生クラスに同名の関数が実装されることを強要することになる。つまり、派生クラスに特定のインターフェースを持ったメンバー関数が存在することを保証することになる。この仕組みを用いると、特定の系によらない共通のアルゴリズムを仮想クラスに実装し、一般化できない特定の系に依存する部分のみを純粋仮想関数として、派生クラスにその実装を義務づけることができる。これは、逆に派生クラスでは、系に依存する純粋仮想関数の実装のみを行えばよいことを意味する。こうして様々

[1]組み込み型もクラスの一種と見ることもできる

[2]もとのクラスのメンバー関数を同名の関数で上書きすることで、新しい名前の関数を付け加えるだけでなく、既にある機能を変更できる点に注意する。

な系に必要最小限の作業で対応できることになるのである。今後、汎用基底クラスの頭文字には T (Type) を付け、仮想クラスには、それに引き続き V (Virtual) を付ける約束とする。

3.3 Monte-Carlo Truth 生成プログラム JUPITER

3.3.1 特長

JUPITER とは、**J**LC **U**nified **P**article **I**nteraction and **T**racking **E**mulato**R** の略である。前述のとおり、物質と粒子の反応部分のシミュレーションを Geant4 に依っているが、Geant4 は近年宇宙線分野、医学から地雷発見などの平和利用まで広く使用されるプログラムに発展しており、Geant3 に比べより高エネルギー実験色の薄い汎用プログラムになった。従って、個々の事例に応用する際、特に JLC のような大型プロジェクトで使用する際には、よりユーザーが使用しやすいように、アプリケーションを開発する必要がある。

JUPITER では、JLC の測定器パラメータがまだ完全に決定していない点、今後のバックグラウンドの研究によっては、大幅なデザイン変更があり得ることなどから、次の 3 点の特性を要求された。

1. 複数の研究グループが同時に開発可能であること。
2. 可能な限り、実器のデザインに近付けるため、検出器のコーディングそのものが簡単であること。
3. 検出器デザインの変更が簡単であること。特定の測定器や、測定器の一部のインストール／アンインストール、置き換えを含む。

これらの要求を満たすために JUPITER は以下のようになっている。

複数のサブグループによる同時開発

JLC には、バーテックス、Intermediate Tracker、CDC、カロリーメータ、Muon chamber の測定器開発グループがあり、さらにビーム衝突点グループ、Solenoid Magnet などの測定器以外の研究グループがある。これらのサブグループが同時にシミュレータの開発を行えるようにするためには、これらのグループが担当する部分を完全に独立になるように分けてしまうのが良い（各検出器のモジュール化)。

図 3.2 は、これをもっとも単純明快な形で実現したものである。各サブグループには、それぞれのディレクトリ（ワークスペース）を割り当て、自分のディレクトリの中のファイル以外の変更は、原則として行えないようにした。これにより、各サブグループは、いつでも好きなときに自分の担当する検出器のアップグレードを行うことが出来る。アップグレードは、ただディレクトリごと新しいものに置き換えるだけでよい。

シミュレータの管理者は、main ディレクトリと kern（kernel の略。意味するところは木星の核である）ディレクトリの管理を行い、各サブグループに物質を置いてよいスペースを配分する。kern ディレクトリには、Geant4 を走らせる為のクラス（Event ,Run, Physics に関係するクラス）、視覚化のためのクラスなどが置かれており、この kern クラスだけで、実験室がひとつ置かれているだけのミニマムセットを作成し、イベントを走らせることができる。

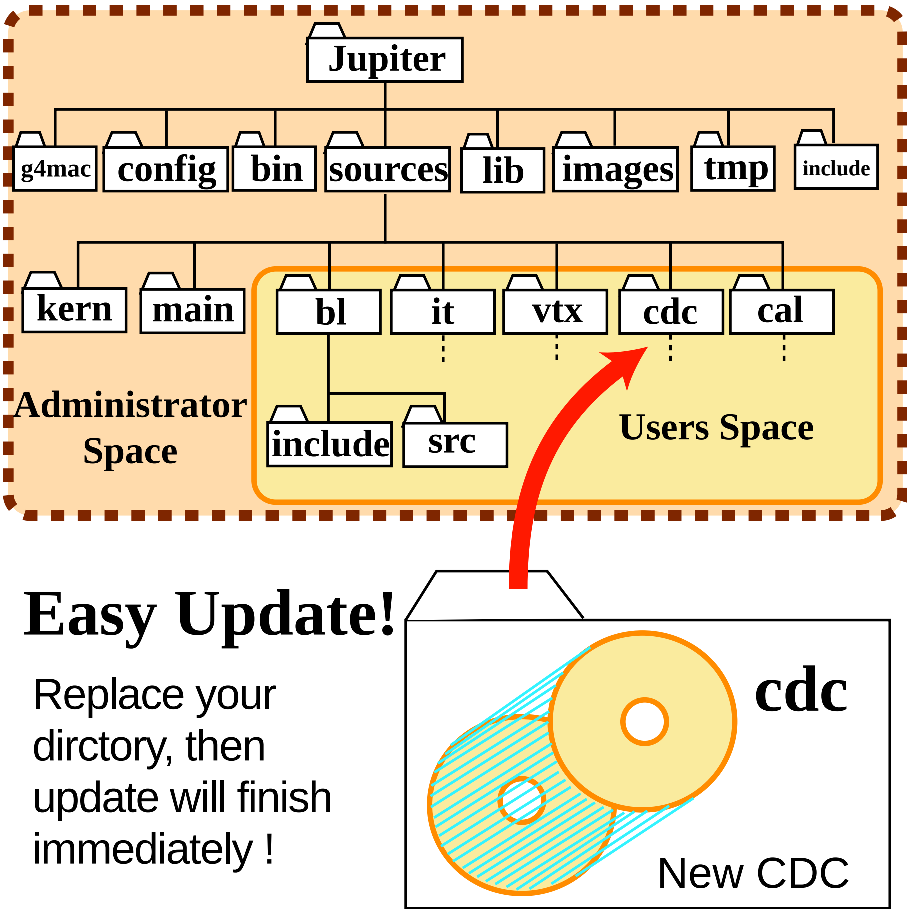


Figure 3.2: JUPITER のディレクトリ構造

プログラムコーディングの簡易さ

一般に、このようなシミュレータを開発するのは、実験家であってプログラマーではない（既に実験開始が決まっている大型プロジェクトを除く）。実験家は多忙であり、時間もコストもかかるビーム実験をいかにして効率よく行うかを知る為にシミュレータを作成し、シミュレーションを行うのであるから、シミュレーションを作成する場面で時間がかかってしまっては、その魅力は半減してしまう。

まず第１に要求されることは、検出器の構造を細部まで作り込む作業の回数を可能な限り減らすことである。同じ構造を持つものを組み立てるのはただ１度でよく、あとはその形をもつオブジェクトを大量に量産すればよい。これはまさしく、ユーザー定義型を提供する C++のもっとも得意とするところであり、この特性をいかに生かすかが、アプリケーション構築のキーポイントになる。

第２には、マニュアルを熟読しなくても、そこそこのコーディングが行えるだけのプログラミングコンセプトを提供することである。勿論、細かいシミュレーションを行うにはマニュアルは不可欠であるが、時間がない実験家が、単純な検出器部品の構造をコーディングするのに、分厚いマニュアルを読破しなければならないようでは使いにくいことこの上ない。Geant4 は大変優れたプログラムであるが、汎用に開発されているが故に、非常に細かい機能に分けてクラスが定義されており、ひとつひとつを眺めてみてもそれが構成する具体的な物体、概念等（オブジェクトと呼ぶ）が見えにくい。そこで、JUPITER では、プログラミングそのものが、なるべく実際の実験における検出器の組立、インストールなどと同じ感覚で行えることを目標としている。実際の検出器部品をそのままオブジェクトとみなしてコーディングすれば、あとは実際の実験で行う作業を表す関数を呼べばよいように設計されている。

検出器デザインの柔軟性

Geant4 には、検出器のジオメトリを記述するための様々な道具が揃っている。2.2 節で述べたように、フルシミュレータには、検出に関係ない大変複雑な形をした部分も詳細に入れることが必要になるため、このような部分は、一般に、CAD 等のアプリケーションを用いて記述したものを Geant4 のオブジェクトに変換する形がとられる。

しかし、検出器の構造は、パラメータチューニング（グローバル、ローカルの両方を含む）のために頻繁に変更される。1 Layer に含まれる Cell の数を変更する度、内部の Sense ワイヤーの配置情報も含めて、いちいち CAD で書き直していたのでは、大変な時間の無駄である。また、検出器を構成する部品は、一般に円柱や箱形で表現できるような単純な形をしたものが多く、必ずしも CAD の助けを借りる必要はない。更に、JLC では、Intermediate Tracker のようにまだ具体的な内部構造が決まっていない検出器も存在する。このような場合に、まず Intermediate Tracker の部分に物質の層をいれておき、具体的なデザインが設定された時点で、層の内部に詳細な構造を作り込めることが重要である。これらのことが簡単に出来るためには、検出器の部品が適切な階層構造を持ち、様々な命令（関数）が階層の終端まで再帰的に呼ばれて、検出器全体に伝わるような形を取るのが好ましい。

これらの全ての要求を C++というプログラミング言語で実現するための解は、サブグループのユーザーが必ず継承すべきベースクラスを作成することである。これらのベースクラスを継承して検出器を作り込んでいくことによって、ユーザーはイベントを走らせる部分を全く意識せずに、実際の検出器を組み立ててインストールする感覚でプログラムを作成できる。次節では、このベースクラスについての詳細を述べる。

検出器コンポーネントの持っているべき属性	クラスの中での表現	
名前	G4String	fName
形、形状	G4VSolid*	fSolid
材質	G4LogecalVolume*	fLV
位置情報	G4VPhysicalVolume	fPV
内包する検出器コンポーネント	J4DtcComponentArray	fDaughters

Table 3.1: 検出器が持っているべき属性とクラス中のデータメンバの対応

3.3.2 ベースクラス構造

JUPITER のベースクラスは、大きく分けて、次の３つの区分に分けられる。

1. 検出器のジオメトリカルな部品（Detector Component）が共通して持つべき特性を集約したベースクラス（J4VComponent、J4VXXXDetectorComponent）
2. 検出器の反応部分（Sensitive Detector）に関する共通特性を集めたクラスと、Monte-Carlo Exact Hit を扱うベースクラス（J4VSensitiveDetector、J4VSD、J4VHit）
3. 物質の定義と管理を行うクラス (J4VMaterialStore、J4XXXMaterialStore）

これらのクラス同士の UML 図を、図 3.3 に示す。なお、頭に J4 がついているのが JUPITER のクラス、G4 がついているものは Geant4 のデフォルトのクラスである。

J4VComponent クラス

J4VComponent は、全ての検出器部品（広くは加速器部分の部品も含まれる）の親となるベースクラスである。J4VComponent クラスは、検出器を構成するそれぞれの部品（コンポーネント）のひな型の原形として働くが、そのひな型は固定された形、材質のコンポーネントを量産できるだけではなく、少しずつ大きさの違うコンポーネントを生成したり、材質の違うコンポーネントを生成したりすることもできる。

J4VComponent クラスは、Geant4 の用語でいう PhysicalVolume １つに対し、１つのオブジェクト[3]を作るよう設計されている。これは、Geant4 で測定器のジオメトリを定義する際に必ず必要な３つのオブジェクト（Solid、LogicalVolume、PhysicalVolume）のうち、もっとも実際の測定器部品に近い特徴を備えているからである[4]。以下に、J4VComponent クラスのデータメンバと、代表的な関数（機能）について述べる。

<u>データメンバと自動ネーミング機能</u>

表 3.1 は、検出器の部品がもっているべき属性と、J4VComponent クラス中での表現の対応を示す。なお、* がついているものは対応するコンポーネント（オブジェクト）へのポインタを所持していることを示す。このうち、名前は、個々のコンポーネントを特定するために大変重要であるが、それゆえに重複した名前をつけると大変なことになる。したがって、JUPITER では全てのコンポーネントの名前は J4VComponent クラスが重複しないようにつける仕様になっている。

[3]オブジェクト指向のオブジェクトだが、ここでは測定器の個々の部品を表すコンポーネントだと思ってほぼさし支えない。

[4]Solid は検出器部品の形のみを定義する。LogicalVolume は、Solid に材質、感応領域であるか否か、絵を描かせたときに表示するか否か、などの情報を合わせたものである。PhysicalVolume は、LogicalVolume にそれが置かれる位置、回転角度などの配置情報を合わせたものである。従って、ビームラインにインストールされた検出器は配置情報を持つので、PhysicalVolume が一番近い。

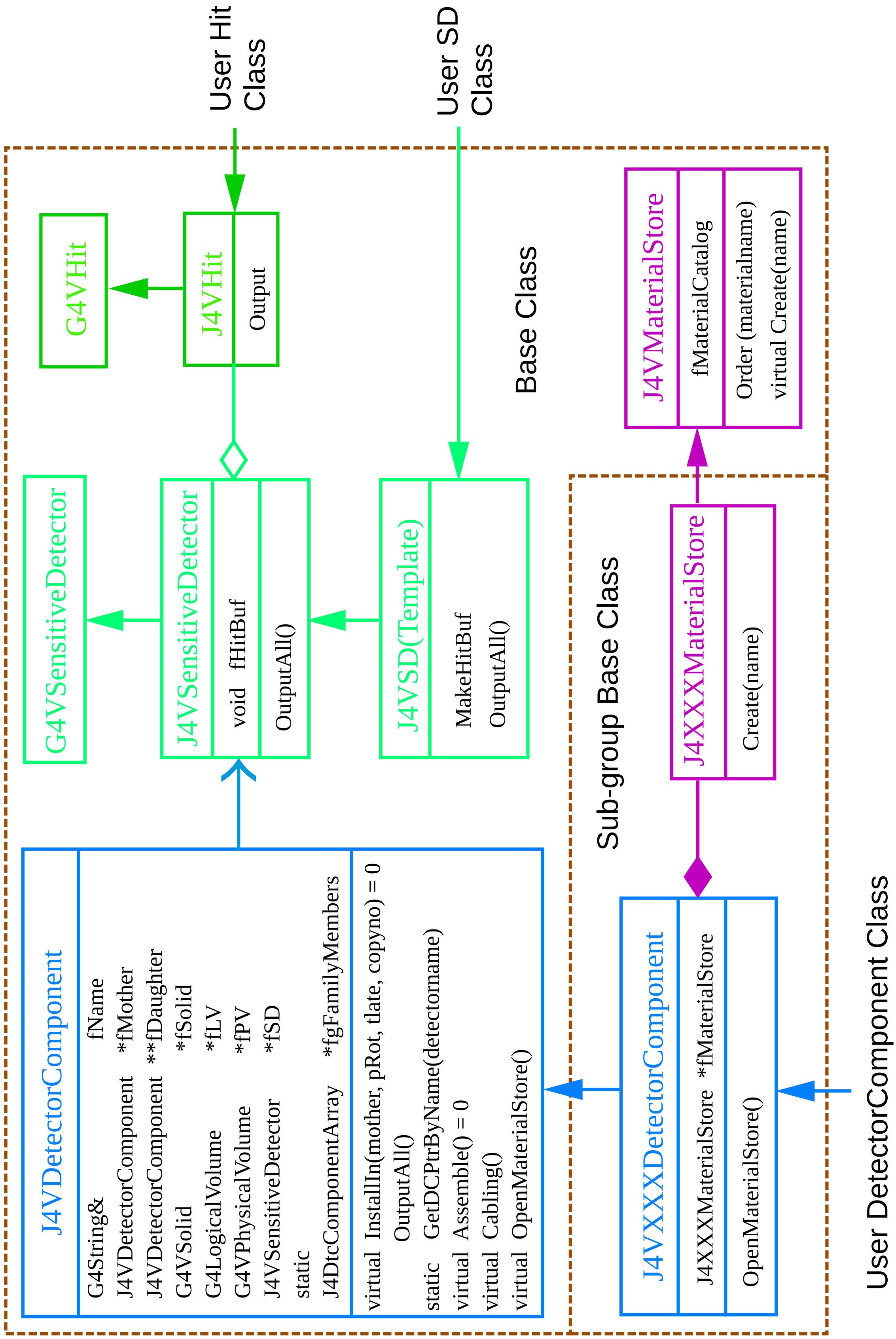


Figure 3.3: JUPITER の Base class

検出器コンポーネントに付加された属性	クラスの中での表現	
自分自身がインストールされる親コンポーネント	J4VComponent	fMother
同じクラスから作られる、形等の違ったコンポーネントの数	G4int	fNbrothers
大きさ等の違ったコンポーネントのうちの何番目か	G4int	fMyID
同じクラスから作られる、同じ形のコンポーネントの数	G4int	fNclones
同じ形のコンポーネントのうちの何番目か	G4int	fCopyNo
測定器全体に含まれるコンポーネントのリスト	J4DtcComponentArray	fgFamiryMembers

Table 3.2: J4VComponent クラスに付加されたデータメンバ

名前を正しくつけるためには、コンポーネントを生成するひな形である DetectorComponent クラスが、自分自身からいくつのオブジェクトが作られるかを知っていなければならない。また、これらの情報をひな形が所持することによって、円筒の分割によって部品を作ったりする場合に作業を簡略化出来る（例：1 Layer 中に N 個詰められた Cell の幅を自動計算させるなど)。

そこで、これらの値をデータメンバに加える。更に、プログラムの操作上検出器部品に付加しておいた方が便利なものを加えて、表 3.2 に示す。ただし、ここでの数とは、同じ階層に属するコンポーネントの数（同じ親コンポーネント中にインストールされる兄弟コンポーネントの数）を示す。

まず、形や材質の違うコンポーネントを考える。例としては、たとえば 1Cell の中に含まれる５本のワイヤーに割り当てられた DriftRegion などが挙げられる。これらは、Sense ワイヤーを内包し、同じ原理でトラックの通過位置を検出するものであるから、同じひな形から作られるべきであるが、Cell は扇形をしているので、５つの DriftRegion は微妙にドリフト領域の大きさが違う（仮に兄弟コンポーネントと呼ぶ)。これらの大きさの変化をパラメトライズするためには、全体でいくつの兄弟コンポーネントがあるかを表す変数と、自分自身がそのうちの何番目かを記憶させる変数とを、コンポーネントのひな形が持っていることが望ましい。このことによって、大きさなどのパラメトライズの部分を完全にひな形クラスの中に閉じ込めることができる。

次に、まったく同じ形、同じ材質のクローンコンポーネントを、配置する場合を考える。単純にコピーを並べる場合には、上と同じ議論でクローンの数と自分が何番目かを表す数があればよい。クローンの数を指定するのは、測定器の部品には全体の何分割かで一つの形が決まるものが多いからである。

更に、特別な場合として、ある単純な形を平面で等分割し、すきまなく並べる場合を考える。この場合には、Geant4 側でレプリカ (G4Replica) というクラスが用意されており、これを用いることで若干プログラム実行時のメモリを減らすことができる。レプリカは、円筒、箱形などを与えられた分割数で等分割して配置するコンポーネントの作り方であるが、実体は分割された１つ分の PhysicalVolume が存在するだけであり、この一つの PhysicalVolume がトラックのやってくる場所に移動して、あたかもたくさんのクローンコンポーネントが存在するかのように働いている。したがって、コンポーネントのひな形が持つべき数は、レプリカに対してはクローンの総数だけでよい。更に、fCopyNo に-1 を入れることによって、生成されたコンポーネントがレプリカによって配置されるものであることを示す[5]。

最後の fgFamilyMembers は、この J4VComponent クラスから作られたオブジェクト全てに対し１つしかない配列であり、J4VComponent から継承して作られた全ての検出器コンポーネント（オブジェクト）へ

[5]Geant4 では、(コピーを含め) 単純に親オブジェクトの中に娘オブジェクトを配置する場合 G4PVPlacement を使うが、Replica と PVPlacement を同じ親オブジェクトの中に同時に配置することを禁じている。したがって、fNclones に入る数は、コピーによるクローンの総数か、レプリカによるクローンの総数かのいずれかである。

のポインタを記憶している。つまり、J4VComponent クラスは、実験室にどんな名前の検出器コンポーネントが存在するかを知っているのである。そこで、名前から逆に個々の検出器コンポーネントへのポインタを辿る GetDtcComponentByName 関数を用意し、検出器のヒットを書き出すか否かをコンポーネント単位で設定したいとき等に使えるようになっている。類似の機能は Geant4 にも存在するが、この関数の利点は戻り値のポインタが J4VComponent 型のポインタであり、J4VComponent にインプリメントされている便利な関数が自在に使える点にある。

Assemble 関数

Assemble 関数は、名前の通り、検出器コンポーネントの組み立てを行う関数である。Geant4 の用語でいえば、LogicalVolume をセットするところまでを行う。検出器コンポーネントは、その内部にまだ内部構造があれば、その内部構造を作り込んだところで形が完成するのであるから、内部にインストールされるコンポーネントを作成[6]するのは、基本的にはこの Assemble 関数の中である。

検出器コンポーネントの組立は、実験ではインストールよりも随分前に行うことが多いが、かならずしもこの点においてまで実験を模倣する必要はなく、インストールの直前で組み立てが仕上がっていればよい。したがって、基本的に Assemble 関数は後に述べる InstallIn 関数の中で呼ばれるべきものであり、外部に公開する必要のないものである。そこで、Assemble 関数はプライベート関数に設定した。これにより、検出器コンポーネントのジオメトリ・材質等についての完全な情報隠蔽が行われ、あるコンポーネントの内部の構造変更がその他のプログラムに影響を及ぼさない仕様になっている。

InstallIn 関数

InstallIn 関数は、自分自身を親コンポーネントにインストールするための関数である。引数に親コンポーネントへのポインタをもらい、主に、その中にどのような形態でインストールされるのかを記述する[7]。引数に位置や回転をとることも出来るので、親コンポーネントの Assemble 関数の中で、位置を指定して娘コンポーネントの InstallIn 関数を読んでやれば、親コンポーネントの望みの位置にインストールすることが可能である。この関数は親コンポーネントから呼ばれるため、当然パブリック関数である。実際のコーディングの一例を、図 3.4 に示す。

J4VXXXDetectorComponent クラス

表題の XXX には、サブグループ名が入る。ユーザーは、各サブグループに一つ、J4VComponent クラスを公開継承してサブグループ独自のベースクラスを作り、全ての検出器コンポーネントは、このサブグループ専用のベースクラスを公開継承しなくてはならない。

J4VXXXDetectorComponent クラスの役目は 3 つある。一つは、名前の中にサブグループ名を入れることである。これにより、もしコンポーネントに他の検出器グループと重複する名前（例えば Layer など）を指定しても、全体では違った名前がつけられることになる。

あとの二つは、各サブグループに一つあればよいクラスを開くという役目である。クラスの中には、複数存在すると混乱を来すものがある。代表的なものが定義を行うクラスであり、検出器パラメータや検出器

[6]プログラム用語で言えば、new 演算子で娘オブジェクトを生成すること。

[7]現実的には、PVPlacement を呼ぶか、Replica を呼ぶかの二者択一である。Geant4 にはもう一つ PVParametrized というオブジェクトの置き方があるが、これは内部構造をもつオブジェクトではうまく作動しない上、実行メモリを気にしなければ PVPlacement で代用出来るので、JUPITER ではサポートしていない。

```
//***************************************************************************
//--------------------------------
// constants (detector parameters)
//--------------------------------

G4String J4CDCDriftRegion::fName("DriftRegion");

//***************************************************************************
//---------------------
// Class Description
//---------------------

        ⋮

//===================================================================
//* Assemble   -------------------------------------------------------

void J4CDCDriftRegion::Assemble()
{
  if(!GetLV())
  {
    // define parameters

    G4double len  =
        ((G4Tubs *)GetMother()->GetLV()->GetSolid())->GetZHalfLength();
    G4double motherRmin =
        ((G4Tubs *)GetMother()->GetLV()->GetSolid())->GetInnerRadius();
    G4double motherRmax =
        ((G4Tubs *)GetMother()->GetLV()->GetSolid())->GetOuterRadius();
    G4double phi =
        ((G4Tubs *)GetMother()->GetLV()->GetSolid())->GetDeltaPhiAngle();

    G4int    nbrothers  = GetNBrothers();
    G4int    myid       = GetMyID();
    G4double thick      = (motherRmax - motherRmin)/(nbrothers + 2);
    G4double rmin       = motherRmin + thick * (myid + 1);
    G4double rmax       = motherRmin + thick * (myid + 2);

    // MakeSolid ----------//
    OrderNewTubs (rmin, rmax, len, phi );

    // MakeLogicalVolume --//
    MakeLVWith(OpenMaterialStore()->Order(_CDCDRIFTREGIONMATERIAL_));

    // SetVisAttribute ----//
    PaintLV(_CDCDriftRegionVisAtt_, G4Color(0.,1.,1.));


    // Install daughter PV //
    // Install Sense Wire  //

    fSenseWire = new J4CDCSenseWire(this);
    fSenseWire->InstallIn(this);

    SetDaughter(fSenseWire);

  }

}
//===================================================================
//* InstallIn  -------------------------------------------------------

void J4CDCDriftRegion::InstallIn(J4VDetectorComponent* mother, G4RotationMatrix* pRot,
                                 G4ThreeVector& tlate )
{
  Assemble();                    // You MUST call Assemble(); at first.
                                 //

  SetPVPlacement(0,0);

  SwitchOn();
}

      ⋮
```

Full name is made from it
(ex. ExpName:CDC:Layer01:Cell:DriftRegion1)

Assemble() is a private method

Making G4Solid

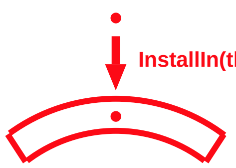
Making G4LogicalVolume

Setting VisAttribute

Making a SenseWire object

Install the SenseWire object into DriftRegion object

Setting the SenseWire object as a daughter of DriftRegion object



InstallIn(mother, pRot, tlate) is a public method

When a mother component calls this function, the DriftRegion object is installed in (0,0,0) point without rotation

Figure 3.4: DetectorComponent クラスのコーディングの一例

の材質を定義するクラスがこれに相当する。前者を ParameterList クラスとし、後者を別に MaterialStore クラスとする。これは、後者がコンポーネントの物質を合成するためのマテリアル工場を持っており、単なるパラメータリスト以上の仕事を行うためである。これらのクラスについては、後に言及する。

勿論、各サブグループ独自に付け加えたい機能があれば、このクラスの中で定義することによって、同じサブグループに属する全ての検出器コンポーネントでその機能が使えるようになる。

J4VSensitiveDetector、J4VSD、J4VHit

これらの 3 つのクラスは互いに連係して働くので、3 つで一つの機能を提供すると考えてさし支えない。

Geant4 では、測定器の感応部分（Sensitive Region）に対し、G4VSensitiveDetector の指定を行う。SensitiveDetector は、LogicalVolume に対して定義され、LogicalVolume の範囲内に粒子が飛び込んで来たときに、何らかのヒット情報を生成し、この Hit 情報を G4VHit クラスのオブジェクトに詰めた上で、バッファに記録する役目を負う。

バッファに記録されたヒット情報は、イベントの終わりに出力される。このとき、書出し命令を行うのは G4EventAction クラスであり、このクラスはシミュレータ全体の動作に関わるクラスなので、当然サブグループユーザの触れない kern ディレクトリの中におさめられている。ここで、G4EventAction にユーザがアクセスしなくてもヒットの書出しを行えるようにしたのが、この 3 つのクラスからなる仕組みである。

まず、G4EventAction の中からは、J4VComponent クラスの OutputAll 関数が呼ばれる。この関数は、全ての検出器コンポーネントを再帰的にスキャンし、SensitiveDetector に指定されたコンポーネントがあれば、そのコンポーネントの SensitiveDetector の OutputAll 関数を呼び出す。SensitiveDetector の OutputAll 関数は、自分が持っているバッファからヒットオブジェクトを一つずつ取り出し、そのヒットオブジェクトの Output 関数を呼ぶ。こうして、ヒットオブジェクトに詰め込まれたヒット情報がアウトプットされる。

この仕組みがうまく働くためには、全てのサブグループの SensitiveDetector は J4VSD クラスを公開継承して作られねばならない。また、Hit クラスも同様に、J4VHit を公開継承して作られる必要がある。

J4VMaterialStore、J4XXXMaterialStore

これらは、検出器の物質を定義し、提供するためのクラスである。J4XXXMaterialStore の XXX の部分には、サブグループ名が入る。

Geant4 では、検出器の物質を自分で定義できる仕様になっている。ガスの混合比を変えたり、特殊合金を作ったりといったことも可能である。検出器のデザインを試行錯誤する段階で、物質を構成する材料の混合比を少しずつ変えてみたり、あるいは温度や圧力を変えてみたりすることは十分あり得る話であるが、その度に管理者に頼んで望みの物質を作ってもらうのは面倒である。

そこで、管理者は基本的な材料カタログを提供するのみとし、各サブグループではカタログにない材料を自分で合成出来る仕様にした。J4VMaterialStore クラスは、仮想関数で Order 関数と Create 関数を持つ。これを継承して J4XXXMaterialStore クラスを作り、Create 関数のみ物質を定義して実装すると、材料の Order を受けたときにまず J4VMaterialStore に行ってカタログの中を探し、そこになければ自分の Create 関数の中を見て物質を合成する、といった作業を行わせることができる。

このしくみは、全体で統一して使用されるべき物質は全体管理者の管理とし、サブグループに対し、他の検出器部分に影響を与えるような勝手な物質定義を許さない、という意味でも重要である。

項目	パラメータ値
CDC 内部	
内筒半径	45cm
外筒半径	156cm
ビーム方向長さ	310cm
材質	アルミニウム
CDC 内部	
ガス	CO_2/isobutane 90:10
総レイヤー数	10 層
レイヤー中のセル数	第 0 層　42 個
	第 1 3 層　63 個
	第 4 6 層　84 個
	第 7 9 層　105 個
セルの半径方向長さ	7cm
セル内のセンスワイヤー数	５本
センスワイヤーの太さ	直径 30μm
センスワイヤー材質	タングステン

Table 3.3: CDC（3T デザイン）のインストールに使用したパラメータ

3.3.3 メインプログラム

さて、各サブグループのユーザーが手を加えることのできるファイルは基本的に各サブグループのディレクトリのみだが、この他に、唯一若干の変更が許されているファイルがある。それがメインプログラム Jupiter.cc である。

メインプログラムでは、どの検出器をインストールするか、どの検出器のスイッチをオンにするか（ヒットの書出しを行うか)、どのようなイベントを発生させるか等を操作することができる。特に Hit の書出しについては、前述の GetDtcComponentByName 関数を用いて、名前から特定のコンポーネントへのポインタを得、ダイレクトにヒットの書出しをオンにしたりオフにしたりすることができる。これは、検出器の一部分が故障して信号を出力できなくなった場合のシミュレーション等を可能にする[8]。

ちなみに Geant4 では、プログラムを走らせている間ならば、同様の作業が可能な仕様になっている。このメインプログラムは、その機能をソースコードの段階で簡単に行えるようにしたものである。

3.3.4 CDC の実装

これらのベースクラスを用いて、実際に CDC の実装を行った。使用したパラメータを表 3.3 に示す。

CDC の検出器コンポーネントの作成、インストールは、全て J4CDCDetectorComponent を継承し、Assemble、Install 関数を使って行った。図 3.5 はこの作業の流れを表したものである。DriftRegion コンポーネントは、Assemble された時点ではまだ配置情報を持っていない（J4VPhysicalVolume へのポインタが null になっている）が、親コンポーネントに InstallIn された時点で位置情報が付加される。図では DriftRegion の作成から始まっているが、図 3.4 のコードにあるように、DriftRegion は SenseWire を new で作成し、中央に配置して作られる。更に Cell は DriftRegion の作成と配置を行い、Layer は Cell の作成と配置を行う。従って、最初に CDC の中に Layer の作成と配置を行うと、そこから再帰的に内部構造が作

[8]ただし、コンポーネント単位（すなわち PhysicalVolume 単位）であるので、Replica でならべたもののスイッチをオフにすると、Replicate された部分の全てが不感領域になってしまうので注意が必要。

り込まれていく。この様子を図 3.6 に示す。コーディングの順番は図の通りだが、実際にプログラムが走ったときにオブジェクトの作られる順番は CDC → Layer → Cell → DriftRegion → SenseWire の順番である。

このようにしてインストールされた CDC と Intermediate Tracker、バーテックスグループによるバーテックスの実装図を、図 3.7 に示す。この図ではカロリーメータにはまだ筒状の物質がインストールされているのみであるが、カロリーメータ、衝突点周りの具体的な構造の実装も開始されている。

3.3.5 Pythia によるヒッグスイベントの生成と JUPITER によるシミュレーション

CDC の構造がインストールされた段階で、JUPITER によるヒッグスイベントのシミュレーションを行った。Geant4 には HEPEvt 形式のインターフェースがあるので、ExampleN04 に付属する pythia_main.f に手を加え、Pythia で重心系 350GeV の $e^+e^- \rightarrow ZH$ イベントを生成、HEPEvt 形式で出力を行った。この出力ファイルを読み、PrimaryGeneratorAction クラス中でイベントをセットしてプログラムを走らせたときのイベントディスプレイが図 3.8 である。Z が 2 つのニュートリノ対に崩壊し、ヒッグスが 2 つの B クォークに崩壊した場合の、典型的な 2 ジェットイベントの図である。第 2.2 節の図 2.2 と比較すると、物質と相互作用してヒットを作っている JUPITER とクイックシミュレータとの違いが明らかである。

3.4 イベント再構成プログラム URANUS とそのシミュレータ Satellites

イベント再構成部分のシミュレータの開発はイベント再構成プログラムの開発にほぼ等しい。JLC のサブグループはこれまでにも多くの R&D 実験を行っており、これらの実験の解析で培われたテクニックは、そのままイベント再構成シミュレータに用いることができる。

CDC グループの場合、現在テスト実験で使っている Baby チェンバーの解析プログラムがあるため、イベント再構成シミュレータ Satellites は、これをもとに作成したイベント再構成プログラム URANUS を継承して作成され、URANUS と自由に連係しながら動かせる形になっている。この節では、これらのプログラムの詳細と連係の方法について述べる。

3.4.1 CDC におけるイベント再構成と URANUS

URANUS は、**U**nited **R**econstruction and **AN**alysis **U**tility **S**et の略である。名前の示す通り、将来的には他のサブグループ検出器と統合して、イベント再構成と解析を受け持つ。JLC では、イベント再構成プログラムのためのフレームワークとして既に JSF が存在するので、JUPITER の場合のように改めて開発しなければならないサブグループ共通のベースクラスはほとんどない[9]。したがって、ここでは CDC 部分の実装に焦点を絞って述べることとする。

CDC における飛跡再構成は、図 3.9 の手順で行われる。

1. FADC データをヒットごとのクラスターに分ける（Clustering)。

[9]シミュレータ部分には、データの形式を統一するためのフレームワークを用意している。実際の実験データのためのフレームワークは、各サブグループのイベント解析プログラムが出揃った段階で整備される予定。現状では、イベント解析に JSF を用いているのは CDC グループのみである。

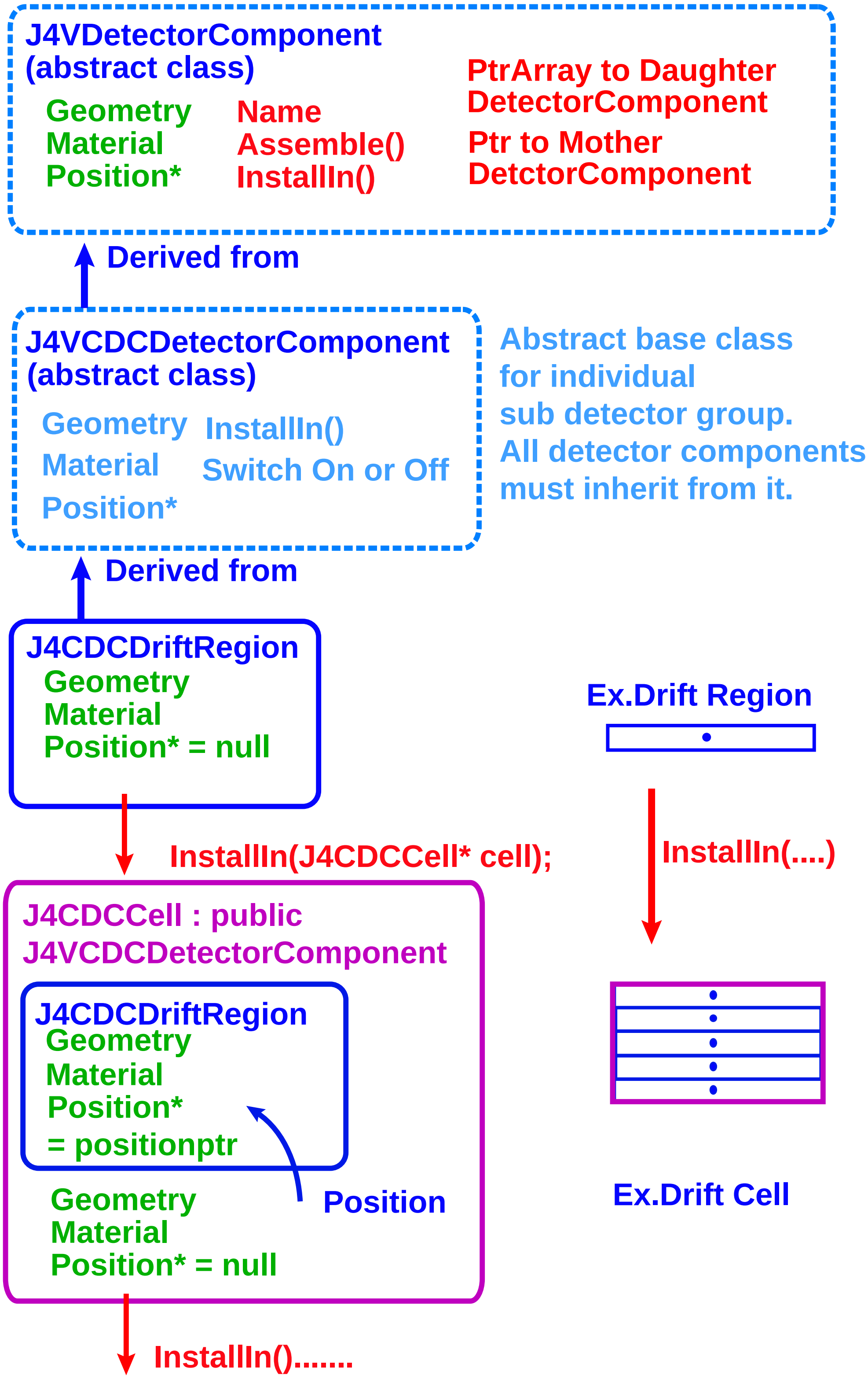


Figure 3.5: 検出器コンポーネント（部品）の作成手順

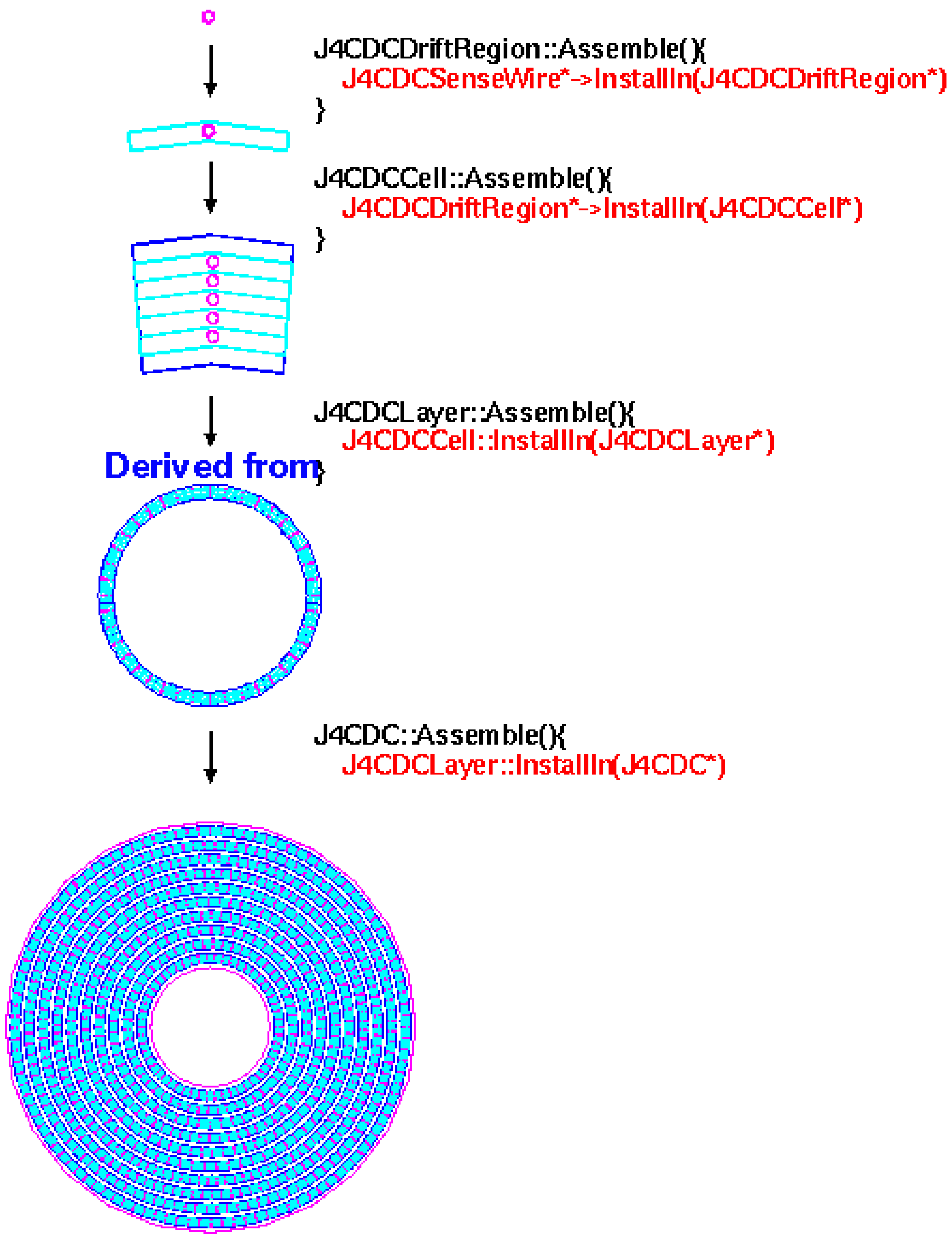


Figure 3.6: CDC のインストール

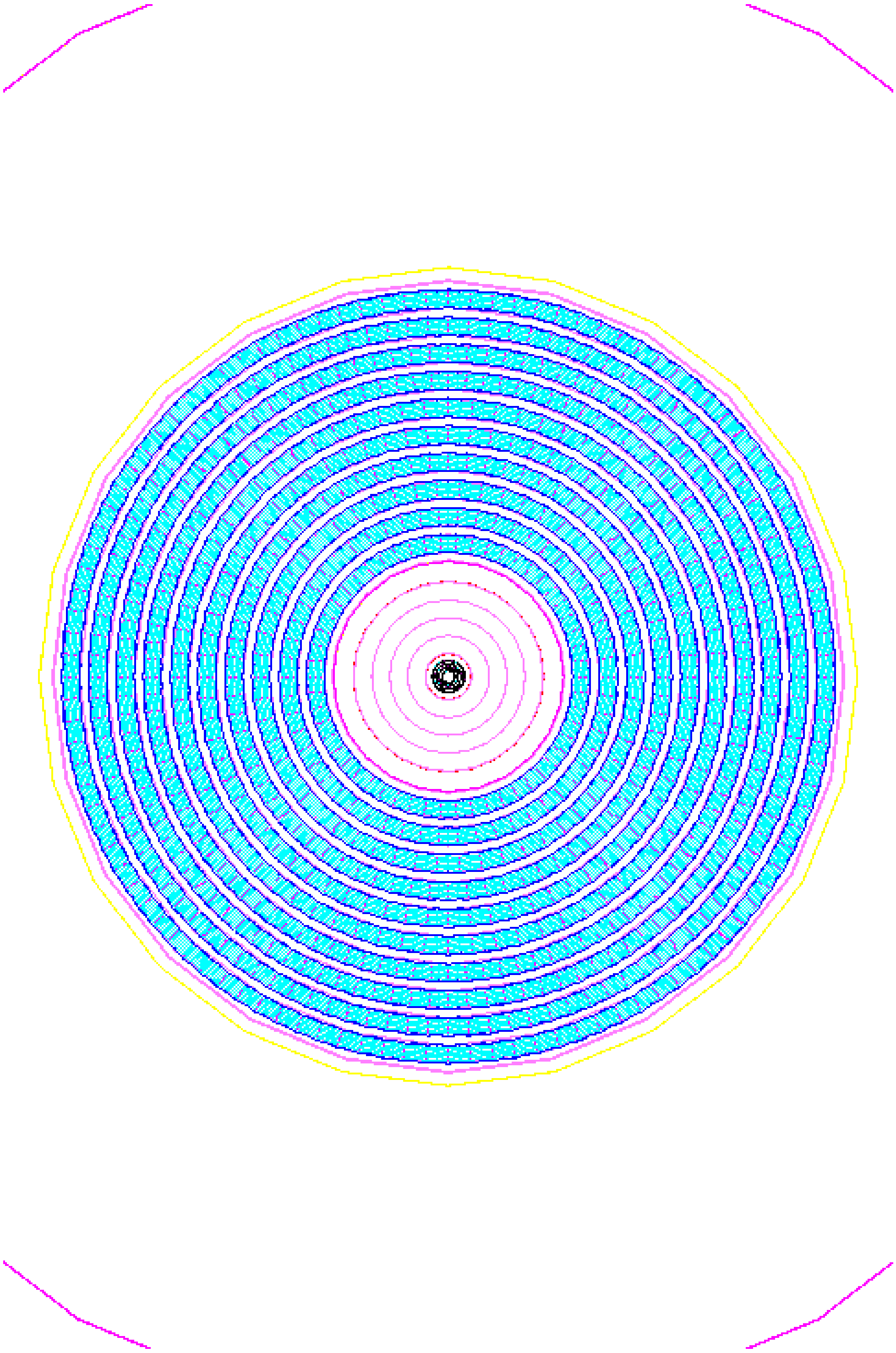


Figure 3.7: バーテックス、Intermediate Tracker と CDC

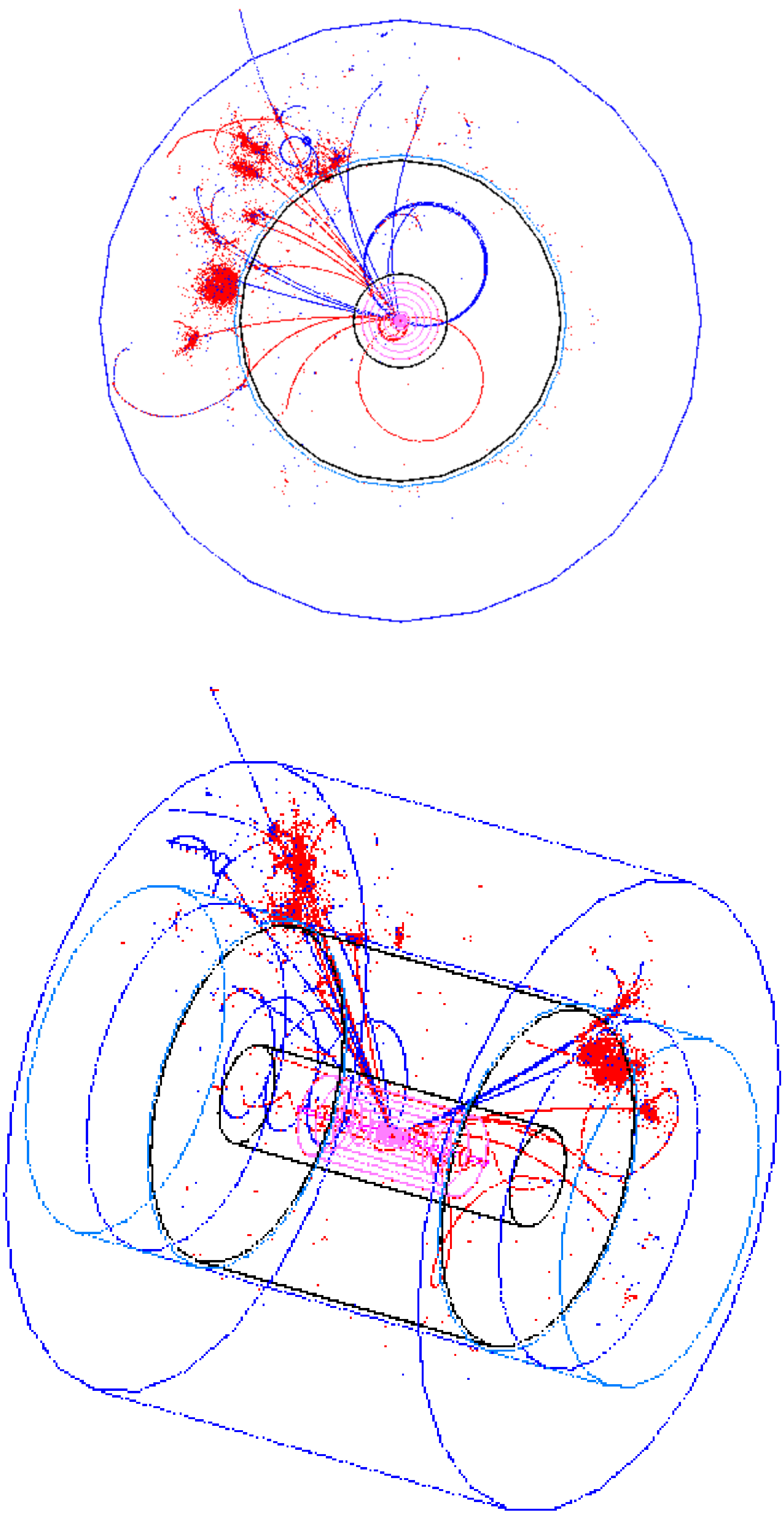


Figure 3.8: $e^+e^- \to ZH$ の反応における 2 Jet event display.

2. クラスターの立ち上がりの時間を読む。FADC の横軸（時間軸）のチャンネル 0 が、トラックがセルを通過した時刻 t_0 に一致していれば、クラスターの立ち上がりの時刻と CO_2/isobutane ガスのドリフト速度を用いて、トラックがセンスワイヤーからどれだけの距離の位置を通ったか（Hit point ）を知ることができる。これを Hit Making という。

3. 得られたヒットを 1 セル分集めて、同じトラック起源と思われるヒットを選びだす。Clustering で間違って生成してしまった Cluster 起源のヒットは、ここでふるいおとされる。これを Track Finding という。

4. Track Finding で選ばれたヒットを Fitting する。

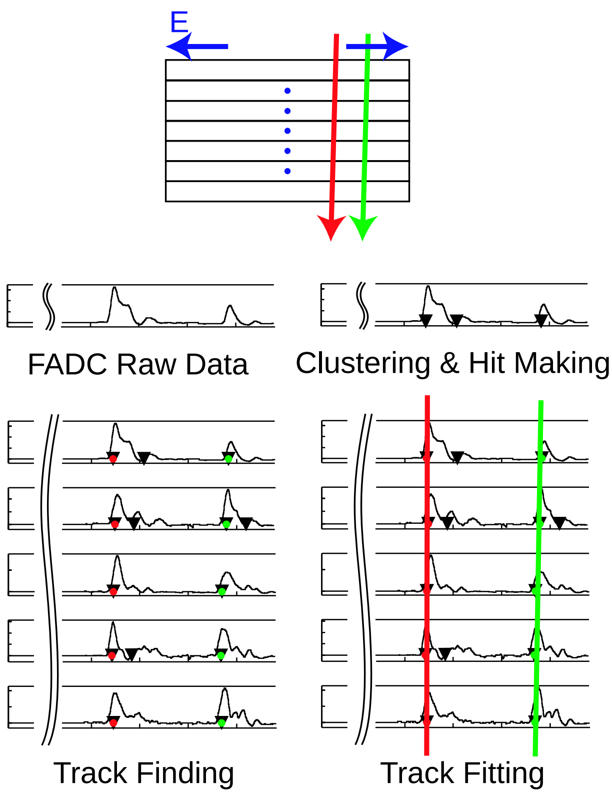


Figure 3.9: 飛跡再構成の手順

これらの作業を行うために、URANUS の CDC パートでは次のモジュールを備えている（そのモジュール内で解析された結果を保存しておくバッファを含む)。

1. CDCUnpacker

2. CDCClusterMaker

3. CDCHitMaker

4. CDCTrackFinder

5. CDCTrackFitter

ここで、CDCUnpacker は、実験によって得られたデータを FADC 1チャンネル分ずつ切り出し、FADC datum オブジェクトに詰める役目を果たすものである。DAQ（オンラインプログラム）の出力を受け取るモジュールでもあるので、DAQ 側のプログラムの変更は全てここで吸収される。その他のモジュールは、それぞれ上で挙げた仕事を受け持っており、直前のモジュールのバッファのデータを解析して、自分のバッファにつめる作業を行っている。この形式の意味するところは、バッファにつまっているデータの形が同じであれば、モジュールを差し換えても問題なく動くということである。

次の節では、この特性を生かしたシミュレータの構造について述べる。

3.4.2 イベント再構成シミュレータ Satellites の全体構造

Satellites とは、木星の衛星の名前を冠したイベント再構成シミュレータのモジュールセットを指す。図 (3.10) は、このシミュレータモジュールセットと JUPITER、URANUS がどのような関係で結ばれているかを示す。

Satellites のうち METIS は、URANUS と同じ構造を持っていることが明らかである。J4CDCHitMaker の上に2か所点線で描かれた空欄は、将来その隣に位置する URANUS のモジュールに対応するシミュレータモジュールがおかれる可能性を示している。この METIS のモジュールに関しては、全てが対応する URANUS のモジュールを継承して作られており、したがって、バッファにつまっているデータの形も同じ顔つきをしている。それゆえ、ある部分までをシミュレーションプログラムで行い、途中から URANUS 解析プログラムに移行するといったシミュレーションレベルの変更が可能である。

このことの利点は、モジュール化されたイベント再構成のそれぞれの過程で、どのように誤差が混入するのかを、モジュールごとに調べられる点である。イベント再構成の過程で混入する誤差を可能な限り減らし、その誤差の大きさと全体のイベント再構成に与える影響を正確に見積もるという点で、URANUS と METIS のセットは分かりやすくかつ使い易い関係にあるといえる。

3.4.3 IO（Input/Output module set）

木星のすぐ下には、ガリレオ衛星の中でももっとも有名な衛星の名前をもつ IO が配置されている。IO は、その名のとおり I/O を司る。第一の役目は、JUPITER のアウトプットをそれぞれの検出器に振り分け、Monte-Carlo Exact Hit を格納する J4VExactHit オブジェクトにつめることである。これにより、JUPITER 側のアウトプット形式の変化に柔軟に対応することができる。現在、IO の仕事はこの1番目の役目に終始している。

一方、将来的には、IO の仕事はもっとも大きく膨らむであろうと考えられる。まずは、シミュレーションの途中経過の書出しと読み込みをサポートすべきであるし、物理解析プログラムとの連結も考えなければならない。JUPITER 本体との接続も、もう少しスマートに行えるべきである[10]。更に言えば、JUPITER 本体の検出器パラメータを IO が管理する可能性もあり得る。この場合には、XML や CAD データとの連係も考えられる。

現在は、仕事量でやや METIS に押され気味であるが、近い将来、ガリレオ衛星の一の名に恥じない働きをする可能性が十分にあるモジュールである。

[10]現在、JUPITER 側から直接 ROOT オブジェクトの形でデータをアウトプットする方針も考えられている。

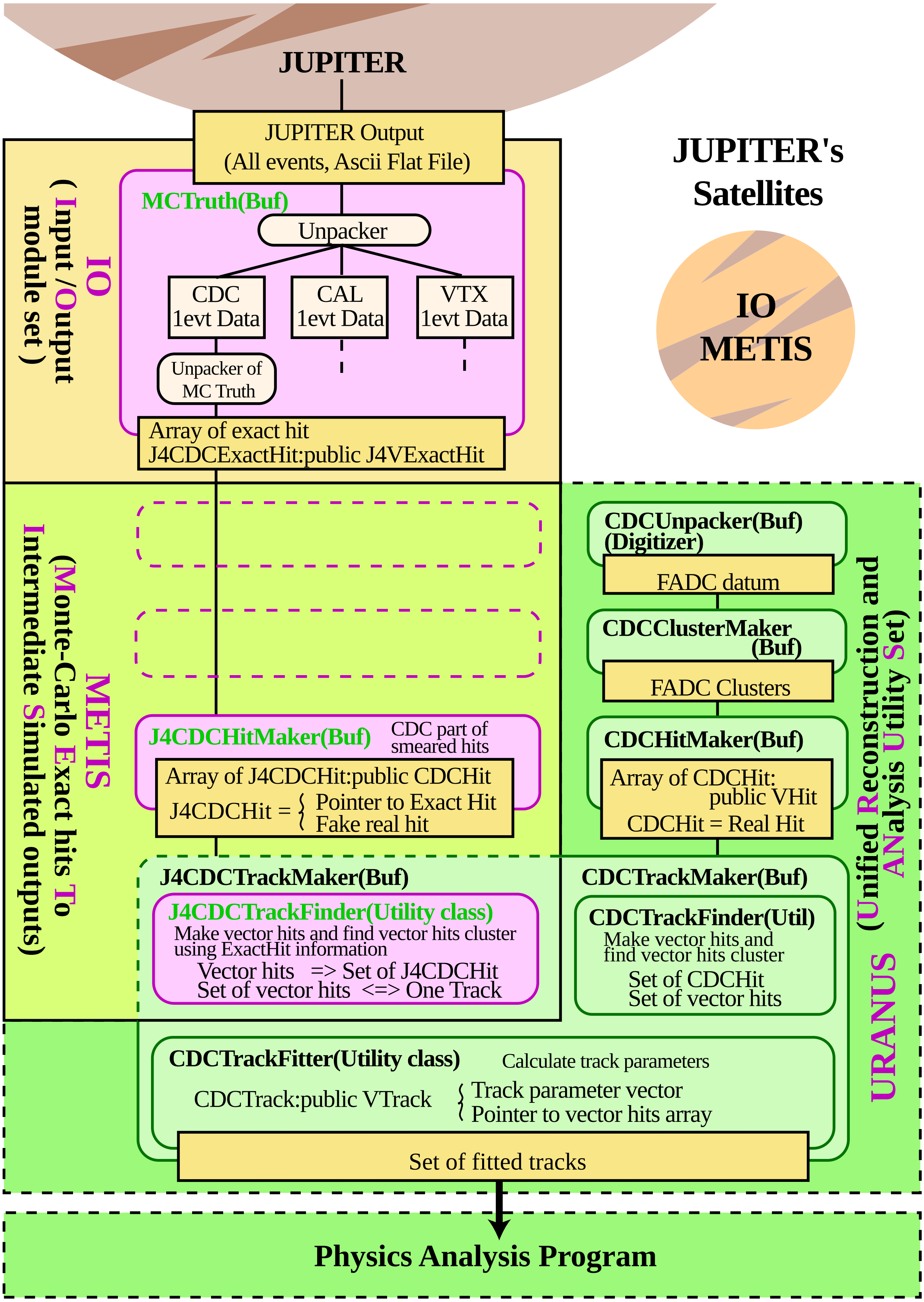


Figure 3.10: Satellites と JUPITER、URANUS との関係

3.4.4 METIS（Monte-Carlo Exacthit To Intermediate Simulated output）

HitMaker

METIS の特徴は、第 3.4.2 節で触れた通り、シミュレーションの途中結果をアウトプットして（Intermediate Simulated output）、その先に通常の解析プログラムをつないで走らせることができる点である。個々のモジュールは、ほとんどが URANUS の対応するモジュールと同じ顔をしているが、シミュレータ独自のメソッドとして、Monte-Carlo Truth に誤差を加えてぼかすメソッドがついている。

HitMaker では、この作業を J4CDCHit クラスの CalculatePosition メソッドの中で行っている。この関数の中では、Monte-Carlo Exact Hit から得られたドリフト距離を、テストチェンバーのビームテスト実験で得られた位置分解能のデータに基づいて、ドリフト距離に依存する幅のガウス関数でぼかしている??。このぼかしたヒットの情報は、同時に Monte-Carlo Exact Hit の Smeared Hit ポインタにも登録され、Exact Hit が辿れる環境では常に Smeared Hit も参照できるようになっている。

TrackFinder

TrackFinder では、シミュレータ独自の方法として、Exact Hit がもっている TrackNumber の情報をみて、トラックごとにヒットを分けていくことができるようになっている（カンニング法）。これは、本当の TrackFinder を Uranus に実装したとき、その性能評価に使用できる。

TrackFitter

Track Fitting の部分では、Fitting アルゴリズムの違いだけが全体の Tracking の精度に関係するので、シミュレーション用のプログラムをわざわざ作成する必要はない。そこで、METIS に組み込む Fitting アルゴリズムとして、簡単な Helix パラメータによる Fitting と、Kalman Filter を応用した Fitting の両方を組み込んである。Kalman Filter を用いた Track Filter については次章で詳述する。

3.4.5 LEDA（Library Extension for Data Analysis）

LEDA はデータ解析を行うさいに使用する汎用ライブラリ群である。今回作成する Kalman Filter もこの LEDA 内に実装され、イベント再構成に使用される。

Chapter 4

Kalman Filterの作成

概要

飛跡検出器による飛跡の再構成は、Hit 点の集合から特定の Track に属する Hit 点を選び出す Track Finding と、そうして選び出された Hit 点を Fit して Track パラメータを決定する Track Fitting に分けられる。ここでは、Track Fitting、すなわち、ある Track に属する n 個の Hit 点の集合が与えられた場合、如何にしてそれらの Hit 点の fit を行い、精度よく Track パラメータ $(d_\rho, \phi_0, \kappa, d_z, \tan\lambda, T_0)$ を決定するかを考察する。

一番単純な方法は、多重散乱（Multiple Scattering）や電離によるエネルギー損失（Energy Loss）がなく、飛跡が完全な Helix を描くと仮定して、最小 2 乗法で fit するものである。しかし、この方法では、1 つの Hit 点の集合に対し、Track パラメータを 1 つしか定義できない。このため、運動量の低い (すなわち多重散乱や電離によるエネルギー損失を無視できない) Track では、途中で比較的大きな多重散乱を受けたり、エネルギー損失により曲率が変化したりすると、その後の Hit 点がビーム衝突点近くの Fitting の精度を悪化させてしまう（図 4.1 - a))。

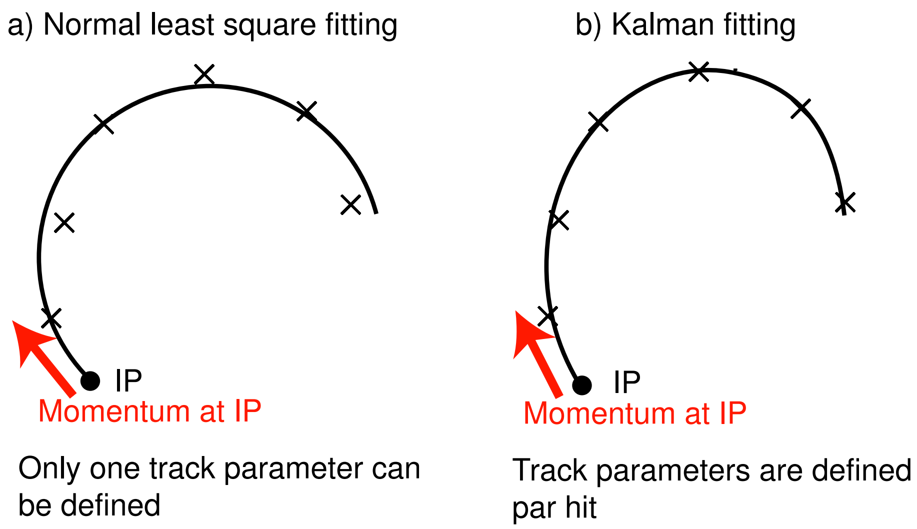


Figure 4.1: Track Fitting の比較

そこで、このような問題点を改善するため、1987 年ごろから、Kalman Filter が用いられはじめた。1960 年に R.E.Kalman が提唱したオリジナルの Kalman Filter [12] を高エネルギー粒子の飛跡再構成に応用したもの [13, 14, 15] で、

- 測定できる量と求めたいパラメータとの関係が近似的に一次の関係にあり、
- 測定量が離散的かつ数が多く、
- それぞれの測定点間の距離が比較的小さく、k 番目の測定点におけるパラメータが、$k-1$ 番目の測定点におけるパラメータを $k-1$ 番目から k 番目の測定点へ外挿したものにほぼ等しいと見なせる場合

に特に有用な fitting 方法である[1]。最小 2 乗法との最大の違いは、測定点が増えるごとに、その点での新たなパラメータを計算しなおすため、測定点ごとにパラメータを変化させられる点にある。これにより、エネルギー損失、クーロン多重散乱、$K \to \mu\nu$ や $\pi \to \mu\nu$ などの崩壊に伴う kink などの様々な process noise の効果を逐次取り入れることができる。その結果、測定精度を向上させることができる。

次の節で、この Kalman Filter の原理について説明する。

4.1 Kalman Filter の原理

測定点 (site) が増加するごとに、逐次直前の測定点でのパラメータ (state) を更新するには、$1 \sim k$ 番目の測定点から得られる、k 番目の測定点におけるパラメータが、$k-1$ 番目までの測定点を用いて計算した $k-1$ 番目の測定点におけるパラメータ と、k 番目の測定点における測定値の関数であるような、漸化式で表される必要がある。これを求めることが目標である。

さて、図 4.2 のように $k-1$ 番目の点に置けるパラメータの真の値と、k 番目の点におけるパラメータの真の値の関係は、次のように書ける。

$$\bar{\boldsymbol{a}}_k = \boldsymbol{f}_{k-1}(\bar{\boldsymbol{a}}_{k-1}) + \boldsymbol{w}_{k-1} \tag{4.1}$$

ここで、$\bar{\boldsymbol{a}}_k$ はパラメータの数を p とすると $p \times 1$ ベクトルになる。上線は真の値であることを示したものであり、下付きの k(または $k-1$) は、k(または $k-1$) 番目の点における、という意味を表す。$\boldsymbol{f}_{k-1}(\bar{\boldsymbol{a}}_{k-1})$ は、ランダムな要因（process noise と呼ばれる）が無視できる場合の、$k-1$ 番目の点におけるパラメータと、k 番目の点におけるパラメータの関係を与え、一般に非線形である。一方 $\boldsymbol{w}_{k-1}$ は、$k-1$ 番目から k 番目にかけて、系のなめらかな進行を阻む process noise に対応し、ここで、$<\boldsymbol{w}_{k-1}>= 0$ である。この process noise の項の共変行列を、

$$\boldsymbol{Q}_{k-1} \equiv <\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T> \tag{4.2}$$

と表すことにする。

実際の測定では、パラメータそのものは測れない場合がある。従って、測定にかかる量とパラメータとの関係を定義しなければならない。測定にかかる量を $\boldsymbol{m}$ とする。

[1]オリジナルの Kalman Filter は、線形系に対するものであり、磁場中の荷電粒子の飛跡再構成のような、非線形の問題に対して応用する場合は、それを適当に線形化して適用することになる。その意味で、正確には Extended Kalman Filter と呼ぶべきであるが、これ以降も、単に Kalman Filter と呼ぶことにする。

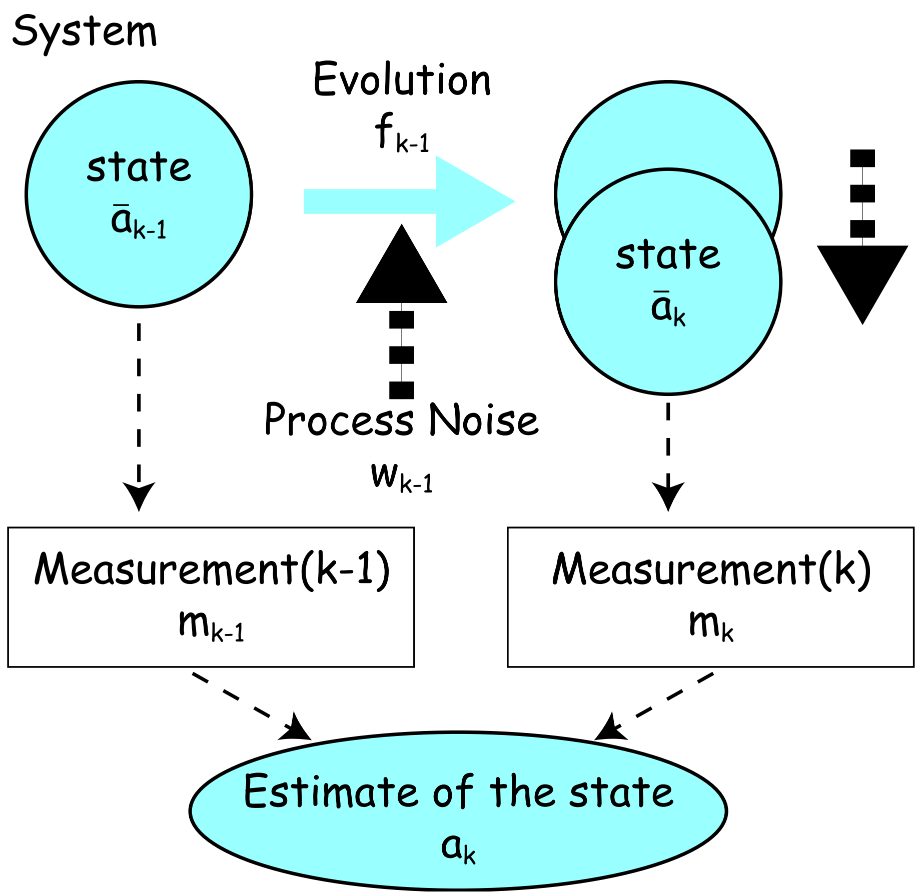


Figure 4.2: Kalman Filter の概要

$\boldsymbol{m}$ の真の値は、パラメータの真の値がわかれば必ず計算できる。しかし実際の測定値には、その上に測定誤差が加わる。そこで、一般に、測定値 $\boldsymbol{m}$ は、次のように表せる。

$$\boldsymbol{m}_k = \boldsymbol{h}_k(\bar{\boldsymbol{a}}_k) + \boldsymbol{\epsilon}_k \tag{4.3}$$

ここで、$\boldsymbol{m}$ は、一般に $n \times 1$ ベクトルであり (位置を表すのであれば $n = 3$ で、x, y, z の 3 次元ベクトル)、$\boldsymbol{h}_k(\boldsymbol{m})$ は測定誤差がない場合にパラメータと測定値の間の関係を与える関数で、これも一般的には非線形である。また、測定誤差（$\boldsymbol{\epsilon}_k$）は、アライメントなどが正しく行われ、系統誤差が無視できるという条件のもとでは、process noise の場合と同様ランダムかつその平均が 0 （$< \boldsymbol{\epsilon}_k >= 0$）であるとみなせるので、共変行列は

$$\boldsymbol{V}_k \equiv (\boldsymbol{G})^{-1} \equiv < \boldsymbol{\epsilon}_k \boldsymbol{\epsilon}_k^T > \tag{4.4}$$

と表せる。

これらを用いて、逐次新しい測定点を加えてパラメータを改善するアルゴリズムを確立するのが目的である。

4.1.1 Prediction：パラメータの予測

$1 \sim k-1$ 番目までの測定情報を用いて、k 番目の測定点におけるパラメータを見積もることを「予測」（prediction）と呼ぶ。この際、(4.1) 式の右辺第 1 項の外挿の部分は予測可能であるが、第 2 項は $k-1$ 番目から k 番目におこるランダムな変動を表しているので、$k-1$ 番目までの情報では予測不可能ということになる。そこで、$k-1$ 番目までの情報を用いた k 点でのパラメータの予測値は、

$$\boldsymbol{a}_k^{k-1} = \boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1}^{k-1}) = \boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1}) \tag{4.5}$$

となる。下付きの k は上と同様「k 番目の測定点における」という意味を表し、上付きの $k-1$ は「$k-1$ 番目までの情報を用いた」という意味を表す。記法の簡便化のため、今後上付き添字と下付き添字が同じ場合は、特に混乱の恐れがなければ上付き添字を省略する。$k-1$ 点におけるパラメータ（$\boldsymbol{a}_{k-1}^{k-1}$）に対する共変行列は、共変行列の定義により

$$\begin{aligned} \boldsymbol{C}_{k-1}^{k-1} &\equiv \boldsymbol{C}_{k-1} \\ &\equiv \left\langle (\boldsymbol{a}_{k-1} - \bar{\boldsymbol{a}}_{k-1})(\boldsymbol{a}_{k-1} - \bar{\boldsymbol{a}}_{k-1})^T \right\rangle \end{aligned} \tag{4.6}$$

で与えられるが、これを用いて、この予測値（$\boldsymbol{a}_k^{k-1}$）に対する共変行列を表すことを考える。再び共変行列の定義により、

$$\boldsymbol{C}_k^{k-1} \equiv \left\langle \left(\boldsymbol{a}_k^{k-1} - \bar{\boldsymbol{a}}_k\right)\left(\boldsymbol{a}_k^{k-1} - \bar{\boldsymbol{a}}_k\right)^T \right\rangle \tag{4.7}$$

$$\begin{aligned} &= \left\langle \left(\boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1}) - \boldsymbol{f}_{k-1}(\bar{\boldsymbol{a}}_k) - \boldsymbol{w}_{k-1}\right)\left(\boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1}) - \boldsymbol{f}_{k-1}(\bar{\boldsymbol{a}}_k) - \boldsymbol{w}_{k-1}\right)^T \right\rangle \\ &\simeq \left\langle \left(\boldsymbol{F}_{k-1}(\boldsymbol{a}_{k-1} - \bar{\boldsymbol{a}}_{k-1}) - \boldsymbol{w}_{k-1}\right)\left(\boldsymbol{F}_{k-1}(\boldsymbol{a}_{k-1} - \bar{\boldsymbol{a}}_{k-1}) - \boldsymbol{w}_{k-1}\right)^T \right\rangle \\ &= \boldsymbol{F}_{k-1}\left\langle (\boldsymbol{a}_{k-1} - \bar{\boldsymbol{a}}_{k-1})(\boldsymbol{a}_{k-1} - \bar{\boldsymbol{a}}_{k-1})^T \right\rangle \boldsymbol{F}_{k-1}^T + \left\langle \boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T \right\rangle + \text{cross terms} \end{aligned} \tag{4.8}$$

が得られる。ここで、

$$\begin{aligned} \boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1}) - \boldsymbol{f}_{k-1}(\bar{\boldsymbol{a}}_k) &\simeq \left(\frac{\partial \boldsymbol{f}_{k-1}}{\partial \boldsymbol{a}_{k-1}}\right)(\boldsymbol{a}_{k-1} - \bar{\boldsymbol{a}}_{k-1}) \\ &= \boldsymbol{F}_{k-1}(\boldsymbol{a}_{k-1} - \bar{\boldsymbol{a}}_{k-1}) \end{aligned}$$

を用いた。点 $k-1$ でのパラメータ のふらつきと、点 $k-1$ から k にかけての process noise は無関係であることから両者の相関に起因する交差項が 0 になることに注意し、式（4.2）、および、式（4.6）を用いると

$$\boldsymbol{C}_k^{k-1} = \boldsymbol{F}_{k-1}\boldsymbol{C}_{k-1}\boldsymbol{F}_{k-1}^T + \boldsymbol{Q}_{k-1} \tag{4.9}$$

を得る。ただし、

$$\boldsymbol{F}_{k-1} \equiv \left(\frac{\partial \boldsymbol{f}_{k-1}}{\partial \boldsymbol{a}_{k-1}}\right) \tag{4.10}$$

である。

4.1.2 Filtering：現在点でのパラメータの最適値の決定

現在点 k において、点 1 から点 k までの全ての測定情報を用いて、現在点 k におけるパラメータの最適値を求めることを filtering と呼ぶ。これを、点 1 から点 $k-1$ までの測定結果から得られたパラメータを点 k における測定結果を加えることにより改善するという形で実現するのが我々のここでの目標である。

式（4.9）の共変行列を用いると、点 1 から点 $k-1$ までの情報から得られる点 k におけるパラメータに対する我々の知識の全ては、次の χ^2 に集約できる：

$$(\chi^2)_k^{k-1} \quad = \quad (\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1})^T \left(\boldsymbol{C}_k^{k-1}\right)^{-1} (\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1})$$

これに、点 k での新たな測定値の情報：

$$(\chi^2)_k^{k,k} \quad = \quad (\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^*))^T \boldsymbol{G} \left(\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^*)\right)$$

を加えると、点 $k-1$ での χ^2 値（χ^2_{k-1}）への差分として、

$$\begin{aligned}
\chi_+^2 \quad &= \quad (\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1})^T \left(\boldsymbol{C}_k^{k-1}\right)^{-1} (\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1}) \\
&\quad + (\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^*))^T \boldsymbol{G} \left(\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^*)\right) \qquad (4.11) \\
&= \quad (\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1})^T \left(\boldsymbol{C}_k^{k-1}\right)^{-1} (\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1}) \\
&\quad + \left(\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1}) - (\boldsymbol{h}_k(\boldsymbol{a}_k^*) - \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1}))\right)^T \boldsymbol{G} \left(\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1}) - (\boldsymbol{h}_k(\boldsymbol{a}_k^*) - \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1}))\right) \\
&\simeq \quad (\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1})^T \left(\boldsymbol{C}_k^{k-1}\right)^{-1} (\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1}) \\
&\quad + \left(\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1}) - \boldsymbol{H}_k(\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1})\right)^T \boldsymbol{G} \left(\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1}) - \boldsymbol{H}_k(\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1})\right) \qquad (4.12)
\end{aligned}$$

が得られる。ここで、$\boldsymbol{a}_k^* - \boldsymbol{a}_k^{k-1}$ が小さいとし、Taylor 展開の一次の項までを残した。ただし、

$$\boldsymbol{H}_k \quad \equiv \quad \left(\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{a}_k^{k-1}}\right) \qquad (4.13)$$

である。

χ^2_{k-1} は、$\boldsymbol{a}_{k-1}$ で最小値をとり、$\boldsymbol{a}_k^*$ の微分に関しては定数である[2]。よって、この差分（χ^2_+）を最小化するような $\boldsymbol{a}_k^*$ が、我々の知りたい k 番目の測定点を加えたとき得られる新しいパラメータ（$\boldsymbol{a}_k^k \equiv \boldsymbol{a}_k$：$k$ 番目の点での k 番目までの測定点を全て使って求めたパラメータ）である。極値条件より χ^2_+ を $\boldsymbol{a}_k^*$ で微分して 0 とおくことにより、ただちに

$$\begin{aligned}
\boldsymbol{a}_k \quad &= \quad \boldsymbol{a}_k^{k-1} + \left[\left(\boldsymbol{C}_k^{k-1}\right)^{-1} + \boldsymbol{H}_k^T \boldsymbol{G}_k \boldsymbol{H}_k\right]^{-1} \boldsymbol{H}_k^T \boldsymbol{G}_k \left(\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1})\right) \\
&= \quad \boldsymbol{a}_k^{k-1} + \boldsymbol{K}_k \left(\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1})\right) \qquad (4.14)
\end{aligned}$$

が得られる。ただし、

$$\boldsymbol{K}_k \quad = \quad \left[\left(\boldsymbol{C}_k^{k-1}\right)^{-1} + \boldsymbol{H}_k^T \boldsymbol{G}_k \boldsymbol{H}_k\right]^{-1} \boldsymbol{H}_k^T \boldsymbol{G}_k \qquad (4.15)$$

とおいた。実用上、逆行列をとる回数はなるべく少ないことが望ましいので、$\boldsymbol{K}_k$ に対する別の表式を求めておく。

$$\begin{aligned}
\boldsymbol{K}_k \left(\boldsymbol{H}_k \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T + (\boldsymbol{G}_k)^{-1}\right) \quad &= \quad \left[\left(\boldsymbol{C}_k^{k-1}\right)^{-1} + \boldsymbol{H}_k^T \boldsymbol{G}_k \boldsymbol{H}_k\right]^{-1} \left(\boldsymbol{H}_k^T \boldsymbol{G}_k \boldsymbol{H}_k \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T + \boldsymbol{H}_k^T\right) \\
&= \quad \left[\left(\boldsymbol{C}_k^{k-1}\right)^{-1} + \boldsymbol{H}_k^T \boldsymbol{G}_k \boldsymbol{H}_k\right]^{-1}
\end{aligned}$$

[2] 全ての測定点を考慮した際に途中の点におけるパラメータがどう改善されるかは後に検討する

$$
\begin{aligned}
& \times\left[\left\{\boldsymbol{H}_k^T \boldsymbol{G}_k \boldsymbol{H}_k+\left(\boldsymbol{C}_k^{k-1}\right)^{-1}-\left(\boldsymbol{C}_k^{k-1}\right)^{-1}\right\} \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T+\boldsymbol{H}_k^T\right] \\
= \quad & \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T
\end{aligned}
$$

よって、式（4.4）に注意すると

$$
\boldsymbol{K}_k \quad = \quad \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T\left(\boldsymbol{V}_k+\boldsymbol{H}_k \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T\right)^{-1} \tag{4.16}
$$

を得る。この、(4.16) 式と (4.14) 式を使うと、新しい k 番目の測定点を加えた時にパラメータ $\boldsymbol{a}_k^{k-1}$ が $\boldsymbol{a}_k \equiv \boldsymbol{a}_k^k$ へと改善されることになる。

このとき、パラメータの誤差行列も同時に改善される。実際、共変行列の定義と、式（4.14）により、

$$
\begin{aligned}
\boldsymbol{C}_k & \equiv \left\langle(\boldsymbol{a}_k-\bar{\boldsymbol{a}}_k)(\boldsymbol{a}_k-\bar{\boldsymbol{a}}_k)^T\right\rangle \\
& = \left\langle\left(\boldsymbol{a}_k^{k-1}+\boldsymbol{K}_k(\boldsymbol{m}_k-\boldsymbol{h}_k(\boldsymbol{a}_k^{k-1}))-\bar{\boldsymbol{a}}_k\right)\left(\boldsymbol{a}_k^{k-1}+\boldsymbol{K}_k(\boldsymbol{m}_k-\boldsymbol{h}_k(\boldsymbol{a}_k^{k-1}))-\bar{\boldsymbol{a}}_k\right)^T\right\rangle \\
& = \left\langle\left[(\boldsymbol{a}_k^{k-1}-\bar{\boldsymbol{a}}_k)+\boldsymbol{K}_k\left\{\boldsymbol{m}_k-\boldsymbol{h}_k(\bar{\boldsymbol{a}}_k)-\left(\boldsymbol{h}_k(\boldsymbol{a}_k^{k-1})-\boldsymbol{h}_k(\bar{\boldsymbol{a}}_k)\right)\right\}\right][\cdots]^T\right\rangle
\end{aligned}
$$

また、これと式（4.3）より、

$$
\begin{aligned}
\boldsymbol{C}_k & = \left\langle\left[(\boldsymbol{a}_k^{k-1}-\bar{\boldsymbol{a}}_k)+\boldsymbol{K}_k\left\{\boldsymbol{\epsilon}_k-\left(\boldsymbol{h}_k(\boldsymbol{a}_k^{k-1})-\boldsymbol{h}_k(\bar{\boldsymbol{a}}_k)\right)\right\}\right][\cdots]^T\right\rangle \\
& \simeq \left\langle\left[(\boldsymbol{a}_k^{k-1}-\bar{\boldsymbol{a}}_k)+\boldsymbol{K}_k\left\{\boldsymbol{\epsilon}_k-\boldsymbol{H}_k(\boldsymbol{a}_k^{k-1}-\bar{\boldsymbol{a}}_k)\right\}\right][\cdots]^T\right\rangle \\
& = \left\langle\left[(1-\boldsymbol{K}_k\boldsymbol{H}_k)(\boldsymbol{a}_k^{k-1}-\bar{\boldsymbol{a}}_k)+\boldsymbol{K}_k\boldsymbol{\epsilon}_k\right][\cdots]^T\right\rangle
\end{aligned}
$$

ただし、ここでも、$\boldsymbol{a}_k^{k-1}-\bar{\boldsymbol{a}}_k$ が小さいとし、式（4.13）を用いて Taylor 展開の一次まで取った。点 k での予測値と真値との差（$\boldsymbol{a}_k^{k-1}-\bar{\boldsymbol{a}}_k$）が、そこでの測定誤差（$\boldsymbol{\epsilon}_k$）とは無関係で相関がないことを考慮すると、交差項は消えて

$$
\begin{aligned}
\boldsymbol{C}_k & = (1-\boldsymbol{K}_k\boldsymbol{H}_k)\left\langle(\boldsymbol{a}_k^{k-1}-\bar{\boldsymbol{a}}_k)(\boldsymbol{a}_k^{k-1}-\bar{\boldsymbol{a}}_k)^T\right\rangle(1-\boldsymbol{K}_k\boldsymbol{H}_k)^T+\boldsymbol{K}_k\left\langle\boldsymbol{\epsilon}_k\boldsymbol{\epsilon}_k^T\right\rangle\boldsymbol{K}_k^T \\
& = (1-\boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_k^{k-1}(1-\boldsymbol{K}_k\boldsymbol{H}_k)^T+\boldsymbol{K}_k\boldsymbol{V}_k\boldsymbol{K}_k^T \qquad (4.17) \\
& = (1-\boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_k^{k-1}-\left[(1-\boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_k^{k-1}\boldsymbol{H}_k^T-\boldsymbol{K}_k\boldsymbol{V}_k\right]\boldsymbol{K}_k^T \\
& = (1-\boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_k^{k-1}-\left[\boldsymbol{C}_k^{k-1}\boldsymbol{H}_k^T-\boldsymbol{K}_k(\boldsymbol{H}_k\boldsymbol{C}_k^{k-1}\boldsymbol{H}_k^T+\boldsymbol{V}_k)\right]\boldsymbol{K}_k^T \\
& = (1-\boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_k^{k-1} \qquad (4.18)
\end{aligned}
$$

となる。ここで、式（4.7)、(4.4)、および（4.16）を用いた。確かに $\boldsymbol{C}_k^{k-1}$ に比較して誤差が小さくなることが分かる。$\boldsymbol{C}_k$ に関する上記の表式は式（4.15）を用いてさらに変形できて

$$
\begin{aligned}
\boldsymbol{C}_k & = (1-\boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_k^{k-1} \\
& = \left(1-\left[\left(\boldsymbol{C}_k^{k-1}\right)^{-1}+\boldsymbol{H}_k^T\boldsymbol{G}_k\boldsymbol{H}_k\right]^{-1}\boldsymbol{H}_k^T\boldsymbol{G}_k\boldsymbol{H}_k\right)\boldsymbol{C}_k^{k-1} \\
& = \left[\left(\boldsymbol{C}_k^{k-1}\right)^{-1}+\boldsymbol{H}_k^T\boldsymbol{G}_k\boldsymbol{H}_k\right]^{-1}\left\{\left(\boldsymbol{C}_k^{k-1}\right)^{-1}+\boldsymbol{H}_k^T\boldsymbol{G}_k\boldsymbol{H}_k-\boldsymbol{H}_k^T\boldsymbol{G}_k\boldsymbol{H}_k\right\}\boldsymbol{C}_k^{k-1} \\
& = \left[\left(\boldsymbol{C}_k^{k-1}\right)^{-1}+\boldsymbol{H}_k^T\boldsymbol{G}_k\boldsymbol{H}_k\right]^{-1} \qquad (4.19)
\end{aligned}
$$

となる。また、これと式（4.15）から、

$$
\boldsymbol{K}_k \quad = \quad \boldsymbol{C}_k\boldsymbol{H}_k^T\boldsymbol{G}_k \tag{4.20}
$$

を得る。

4.1.3 Smoothing：途中の点でのパラメータの最適値の再評価

一般に、Filtering の過程において、ある点（k とする）で得られたパラメータは、そこに至るまでに得られた情報を総合して求めたその時点では最も精度の高いものであるが、最終的に n 個全ての測定情報を得た時点では、点 $k+1$ から点 n までの情報を加えて再評価することでより精度の高いパラメータを求めることができる。これを Smoothing と呼ぶ。明らかに、点 n においては、Filtering の結果がそのまま、Smoothing の結果となる。そこで、Filtering の場合とは逆に、点 n から出発して、点 1 に向かい、逐次、各点でのパラメータを Smoothing する方法を検討する。目標は、既に得られている Filtering の結果と点 $k+1$ での Smoothing の結果を使って点 k における Smoothing の結果を表現することである。

この目的のために、まず、点 $k+1$ における Smoothing を考える。点 $k+1$ における、n 個全ての情報を使ったパラメータの最適値（$\boldsymbol{a}_{k+1}^{n}$）は、点 1 から点 k までの情報を使った点 $k+1$ における予測値（$\boldsymbol{a}_{k+1}^{k}$）と、点 $k+1$ から点 n までの情報を使った点 $k+1$ における逆行 Filter 値（$\boldsymbol{a}_{k+1}^{k+1,n}$）の加重平均で与えられるはずである（今後、上付き添字の $k+1, n$ は、点 $k+1$ から点 n までの情報を使ったものであることを示すものとする）。この加重平均は、次の χ^2 を最小化することに対応する。

$$
\begin{aligned}
\chi^2 &= (\boldsymbol{a}_{k+1}^{n} - \boldsymbol{a}_{k+1}^{k})^T \left(\boldsymbol{C}_{k+1}^{k}\right)^{-1} (\boldsymbol{a}_{k+1}^{n} - \boldsymbol{a}_{k+1}^{k}) \\
&\quad + (\boldsymbol{a}_{k+1}^{n} - \boldsymbol{a}_{k+1}^{k+1,n})^T \left(\boldsymbol{C}_{k+1}^{k+1,n}\right)^{-1} (\boldsymbol{a}_{k+1}^{n} - \boldsymbol{a}_{k+1}^{k+1,n})
\end{aligned}
$$

この χ^2 を $\boldsymbol{a}_{k+1}^{n}$ で微分して 0 と置くことにより、ただちに

$$
\boldsymbol{a}_{k+1}^{n} = \left[\left(\boldsymbol{C}_{k+1}^{k}\right)^{-1} + \left(\boldsymbol{C}_{k+1}^{k+1,n}\right)^{-1}\right]^{-1} \left[\left(\boldsymbol{C}_{k+1}^{k}\right)^{-1} \boldsymbol{a}_{k+1}^{k} + \left(\boldsymbol{C}_{k+1}^{k+1,n}\right)^{-1} \boldsymbol{a}_{k+1}^{k+1,n}\right]
$$

が得られる。また、この時、$\boldsymbol{a}_{k+1}^{n}$ の共変行列の逆行列は

$$
\left(\boldsymbol{C}_{k+1}^{n}\right)^{-1} = \left(\boldsymbol{C}_{k+1}^{k}\right)^{-1} + \left(\boldsymbol{C}_{k+1}^{k+1,n}\right)^{-1}
$$

で与えられる。上の 2 式を、逆に解けば、

$$
\boldsymbol{a}_{k+1}^{k+1,n} = \boldsymbol{a}_{k+1}^{n} + \boldsymbol{C}_{k+1}^{n} \left[\boldsymbol{C}_{k+1}^{k} - \boldsymbol{C}_{k+1}^{n}\right]^{-1} (\boldsymbol{a}_{k+1}^{n} - \boldsymbol{a}_{k+1}^{k}) \tag{4.21}
$$

$$
\left(\boldsymbol{C}_{k+1}^{k+1,n}\right)^{-1} = \left(\boldsymbol{C}_{k+1}^{n}\right)^{-1} - \left(\boldsymbol{C}_{k+1}^{k}\right)^{-1} \tag{4.22}
$$

となる。点 k における Smoothing も同様に行えて、

$$
\boldsymbol{a}_{k}^{n} = \left[(\boldsymbol{C}_{k})^{-1} + \left(\boldsymbol{C}_{k}^{k+1,n}\right)^{-1}\right]^{-1} \left[(\boldsymbol{C}_{k})^{-1} \boldsymbol{a}_{k} + \left(\boldsymbol{C}_{k}^{k+1,n}\right)^{-1} \boldsymbol{a}_{k}^{k+1,n}\right] \tag{4.23}
$$

$$
\left(\boldsymbol{C}_{k}^{n}\right)^{-1} = (\boldsymbol{C}_{k})^{-1} + \left(\boldsymbol{C}_{k}^{k+1,n}\right)^{-1} \tag{4.24}
$$

となるが、これから式（4.21）と（4.22）を使って、上添字 $k+1, n$ をもつ因子を、上添字 n と k のものに置き換えられれば目的を達したことになる。まず、共変行列の定義により

$$
\boldsymbol{C}_{k}^{k+1,n} \equiv \left\langle (\boldsymbol{a}_{k}^{k+1,n} - \bar{\boldsymbol{a}}_{k})(\boldsymbol{a}_{k}^{k+1,n} - \bar{\boldsymbol{a}}_{k})^T \right\rangle
$$

であるが、これと、式（4.1）から、

$$
\boldsymbol{C}_{k}^{k+1,n} = \left\langle \left(\boldsymbol{f}_{k}^{-1}(\boldsymbol{a}_{k+1}^{k+1,n}) - \boldsymbol{f}_{k}^{-1}(\bar{\boldsymbol{a}}_{k+1} - \boldsymbol{w}_{k})\right) \left(\boldsymbol{f}_{k}^{-1}(\boldsymbol{a}_{k+1}^{k+1,n}) - \boldsymbol{f}_{k}^{-1}(\bar{\boldsymbol{a}}_{k+1} - \boldsymbol{w}_{k})\right)^T \right\rangle
$$

$$
\begin{aligned}
&\simeq \left\langle \left(\boldsymbol{F}_k^{-1}(\boldsymbol{a}_{k+1}^{k+1,n} - \bar{\boldsymbol{a}}_{k+1} + \boldsymbol{w}_k) \right) \left(\boldsymbol{F}_k^{-1}(\boldsymbol{a}_{k+1}^{k+1,n} - \bar{\boldsymbol{a}}_{k+1} + \boldsymbol{w}_k) \right)^T \right\rangle \\
&= \boldsymbol{F}_k^{-1} \left\langle \left((\boldsymbol{a}_{k+1}^{k+1,n} - \bar{\boldsymbol{a}}_{k+1}) + \boldsymbol{w}_k \right) \left((\boldsymbol{a}_{k+1}^{k+1,n} - \bar{\boldsymbol{a}}_{k+1}) + \boldsymbol{w}_k \right)^T \right\rangle \boldsymbol{F}_k^{-1T}
\end{aligned}
$$

を得る。ここで、$\boldsymbol{a}_{k+1}^{k+1,n}$ と $\bar{\boldsymbol{a}}_{k+1} - \boldsymbol{w}_k$ は近く、$\boldsymbol{f}_k^{-1}$ は、Taylor 展開の一次まででよく近似できるとした。点 k から点 $k+1$ にかけての process noise と、点 $k+1$ から点 n の測定点から得たパラメータの真値からのずれとは無関係で相関がないことに注意し、式（4.2）と共変行列の定義を使うと

$$
\boldsymbol{C}_k^{k+1,n} = \boldsymbol{F}_k^{-1} \left(\boldsymbol{C}_{k+1}^{k+1,n} + \boldsymbol{Q}_k \right) \boldsymbol{F}_k^{-1T} \tag{4.25}
$$

となる。$\boldsymbol{C}_{k+1}^{k+1,n}$ は、式（4.22）で表されているので、これで、$\boldsymbol{C}_k^{k+1,n}$ が上添字 n と k のものに置き換えられたことになる。一方、式（4.23）と（4.24）より

$$
\begin{aligned}
\boldsymbol{a}_k^n &= \boldsymbol{C}_k^n \left\{ (\boldsymbol{C}_k^n)^{-1} \boldsymbol{a}_k + \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} (\boldsymbol{a}_k^{k+1,n} - \boldsymbol{a}_k) \right\} \\
&= \boldsymbol{a}_k + \boldsymbol{C}_k^n \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} (\boldsymbol{a}_k^{k+1,n} - \boldsymbol{a}_k) \\
&= \boldsymbol{a}_k + \boldsymbol{C}_k^n \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} \left(\boldsymbol{f}_k^{-1}(\boldsymbol{a}_{k+1}^{k+1,n}) - \boldsymbol{f}_k^{-1}(\boldsymbol{a}_{k+1}^k) \right) \\
&\simeq \boldsymbol{a}_k + \boldsymbol{C}_k^n \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} \boldsymbol{F}_k^{-1}(\boldsymbol{a}_{k+1}^{k+1,n} - \boldsymbol{a}_{k+1}^k)
\end{aligned}
$$

となるが、これから式（4.21）を使って簡単な変形をすると

$$
\boldsymbol{a}_k^n = \boldsymbol{a}_k + \boldsymbol{A}_k(\boldsymbol{a}_{k+1}^n - \boldsymbol{a}_{k+1}^k) \tag{4.26}
$$

$$
\boldsymbol{A}_k \equiv \boldsymbol{C}_k^n \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} \boldsymbol{F}_k^{-1} \boldsymbol{C}_{k+1}^k \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right)^{-1} \tag{4.27}
$$

が得られる。$\boldsymbol{A}_k$ の中の $\boldsymbol{C}_k^n$ は、式（4.24）によって、また、$\boldsymbol{C}_k^{k+1,n}$ は、式（4.25）と（4.22）によって書き換えられる。次にこれを実行する。

$$
\begin{aligned}
\boldsymbol{A}_k &\equiv \boldsymbol{C}_k^n \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} \boldsymbol{F}_k^{-1} \boldsymbol{C}_{k+1}^k \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right)^{-1} \\
&= \left[(\boldsymbol{C}_k)^{-1} + \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} \right]^{-1} \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} \boldsymbol{F}_k^{-1} \boldsymbol{C}_{k+1}^k \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right)^{-1} \\
&= \left[\boldsymbol{C}_k^{k+1,n} \left\{ (\boldsymbol{C}_k)^{-1} + \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} \right\} \right]^{-1} \boldsymbol{F}_k^{-1} \boldsymbol{C}_{k+1}^k \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right)^{-1} \\
&= \left[\left(\boldsymbol{C}_k + \boldsymbol{C}_k^{k+1,n} \right) (\boldsymbol{C}_k)^{-1} \right]^{-1} \boldsymbol{F}_k^{-1} \boldsymbol{C}_{k+1}^k \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right)^{-1} \\
&= \boldsymbol{C}_k \left(\boldsymbol{C}_k + \boldsymbol{C}_k^{k+1,n} \right)^{-1} \boldsymbol{F}_k^{-1} \boldsymbol{C}_{k+1}^k \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right)^{-1}
\end{aligned} \tag{4.28}
$$

一方、式（4.9）と（4.25）から導かれる

$$
\boldsymbol{C}_k + \boldsymbol{C}_k^{k+1,n} = \boldsymbol{F}_k^{-1} \left(\boldsymbol{C}_{k+1}^k + \boldsymbol{C}_{k+1}^{k+1,n} \right) \boldsymbol{F}_k^{-1T}
$$

と式（4.22）を使うと、

$$
\begin{aligned}
\boldsymbol{C}_k + \boldsymbol{C}_k^{k+1,n} &= \boldsymbol{F}_k^{-1} \left(\boldsymbol{C}_{k+1}^k + \boldsymbol{C}_{k+1}^{k+1,n} \right) \boldsymbol{F}_k^{-1T} \\
&= \boldsymbol{F}_k^{-1} \left(\boldsymbol{C}_{k+1}^k + \boldsymbol{C}_{k+1}^k \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right)^{-1} \boldsymbol{C}_{k+1}^n \right) \boldsymbol{F}_k^{-1T} \\
&= \boldsymbol{F}_k^{-1} \boldsymbol{C}_{k+1}^k \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right)^{-1} \boldsymbol{C}_{k+1}^k \boldsymbol{F}_k^{-1T}
\end{aligned} \tag{4.29}
$$

が成立することが分かる。これを式（4.28）に代入すると、ただちに

$$\boldsymbol{A}_k = \boldsymbol{C}_k \boldsymbol{F}_k^T \left(\boldsymbol{C}_k^{k+1} \right)^{-1} \tag{4.30}$$

が導かれる。これで、$\boldsymbol{a}_k^n$ を求める漸化式が得られたことになる。次に $\boldsymbol{a}_k^n$ の共変行列を求めておく。式（4.28）と（4.29）から、

$$\boldsymbol{C}_k^n \left(\boldsymbol{C}_k^{k+1,n} \right)^{-1} = \boldsymbol{C}_k \boldsymbol{F}_k^T \left(\boldsymbol{C}_{k+1}^k \right)^{-1} \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right) \left(\boldsymbol{C}_{k+1}^k \right)^{-1} \boldsymbol{F}_k$$

であるが、この両辺に右から $\boldsymbol{C}_k^{k+1,n}$ をかけ、式（4.29）を使って $\boldsymbol{C}_k^{k+1,n}$ を消去すると

$$\begin{aligned} \boldsymbol{C}_k^n &= \boldsymbol{C}_k - \boldsymbol{C}_k \boldsymbol{F}_k^T \left(\boldsymbol{C}_{k+1}^k \right)^{-1} \left(\boldsymbol{C}_{k+1}^k - \boldsymbol{C}_{k+1}^n \right) \left(\boldsymbol{C}_{k+1}^k \right)^{-1} \boldsymbol{F}_k \boldsymbol{C}_k \\ &= \boldsymbol{C}_k + \boldsymbol{A}_k \left(\boldsymbol{C}_{k+1}^n - \boldsymbol{C}_{k+1}^k \right) \boldsymbol{A}_k^T \end{aligned} \tag{4.31}$$

が得られる。

4.2 Kalman Filter ベースクラスの作成

前節で得られた漸化式は、問題とする特定の系によらない一般的な形で表されている。Kalman Filter を特定の系に使うためには、その系に対応させる必要がある。その作業は後に詳述するとして、先に共通部分の一般的なアルゴリズムをベースクラスとして実装する。

必要なパラメータと公式

まず、前節で求めた漸化式の中から、必要な数式を選び出す。必要な数式は、パラメータ (state) とその共変行列（誤差行列）に関する漸化式を与える以下である。

$$\begin{aligned} \boldsymbol{a}_k &= \boldsymbol{a}_k^{k-1} + \boldsymbol{K}_k \left(\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1}) \right) \\ \boldsymbol{C}_k &= \left[\left(\boldsymbol{C}_k^{k-1} \right)^{-1} + \boldsymbol{H}_k^T \boldsymbol{G}_k \boldsymbol{H}_k \right]^{-1} = (1 - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{C}_k^{k-1} \\ \boldsymbol{a}_k^n &= \boldsymbol{a}_k + \boldsymbol{A}_k (\boldsymbol{a}_{k+1}^n - \boldsymbol{a}_{k+1}^k) \\ \boldsymbol{A}_k &= \boldsymbol{C}_k \boldsymbol{F}_k^T \left(\boldsymbol{C}_k^{k+1} \right)^{-1} \end{aligned}$$

ただし

$$\left\{ \begin{aligned} \boldsymbol{K}_k &= \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T \left(\boldsymbol{V}_k + \boldsymbol{H}_k \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T \right)^{-1} \\ \boldsymbol{C}_k^{k-1} &= \boldsymbol{F}_{k-1}\ \boldsymbol{C}_{k-1}\ \boldsymbol{F}_{k-1}^T + \boldsymbol{Q}_{k-1} \end{aligned} \right.$$

また

$$\left\{ \begin{aligned} \boldsymbol{F}_{k-1} &\equiv \left(\frac{\partial \boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1})}{\partial \boldsymbol{a}_{k-1}} \right) \\ \boldsymbol{H}_k &\equiv \left(\frac{\partial \boldsymbol{h}_k(\boldsymbol{a}_k^{k-1})}{\partial \boldsymbol{a}_k^{k-1}} \right) \end{aligned} \right. \tag{4.32}$$

$\boldsymbol{a}_k$ は、k 番目までの測定情報を用いて計算した、k 番目の測定点におけるパラメータの最適値、$\boldsymbol{a}_k^{k-1}$ は $k-1$ 番目までの測定情報を用いて計算した、k 番目の測定点におけるパラメータの予測値、$\boldsymbol{C}_k$ と $\boldsymbol{C}_k^{k-1}$ は

対応するパラメータの誤差行列を表す。また、$\boldsymbol{a}_k^n$ は、逆向きに k 番目までの測定情報を用いて計算した、k 番目の測定点におけるパラメータの Smoothing された最適値、$\boldsymbol{A}_k$ は対応するパラメータの誤差行列を表す。すでに述べたように $\boldsymbol{m}_k$ は測定ベクトル、$\boldsymbol{h}_k(\boldsymbol{a}_k^{k-1})$ は、$\boldsymbol{a}_k^{k-1}$ を用いて計算した、測定誤差がない場合の k 番目の測定ベクトルの予測値である。一方、$\boldsymbol{f}_{k-1}$ は、process noise が無視できる場合に、$k-1$ 番目の測定点におけるパラメータと k 番目の測定点におけるパラメータの関係を与える式であり、ここでは、$k-1$ 番目までの測定情報を用いて計算した最適なパラメータ $\boldsymbol{a}_{k-1}$ を引数にとっている。process noise を無視すれば、$\bar{\boldsymbol{a}}_k = \boldsymbol{f}_{k-1}(\bar{\boldsymbol{a}}_{k-1})$ の関係がある。ここで、$\bar{\boldsymbol{a}}_k$ 等の上線は真の値であることを示す。また、$\boldsymbol{Q}_k$、$\boldsymbol{V}_k$ はそれぞれ process noise、測定誤差に関する共変行列、$\boldsymbol{G}_k$ は $\boldsymbol{V}_k$ の逆行列である。

これにより、$\boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1})$ のパラメータに対する偏微分 $\boldsymbol{F}_{k-1}$ と、$\boldsymbol{H}_k$（測定方程式のパラメータ微分)、$\boldsymbol{Q_{k-1}}$ および $\boldsymbol{V}_k$ の具体的な値を代入してやれば、新しい測定点 $\boldsymbol{m}_k$ が加わるたびに、パラメータ $\boldsymbol{a}_k$ が更新できることになる。しかし、これらの具体的な表式は問題にする特定の系によるものなので、ベースクラスの仮想関数を継承した派生クラスで実装することになる。

構造

図 4.3 は、作成した Kalman Filter のベースクラス群のクラス図である。このベースクラスライブラリは LEDA に組込まれ、以降はこれを継承した派生クラスで仮想関数を問題とする特定の系に対応させた形で実装することによって、簡単に Kalman Filter を使った解析を Satellites 内で行うことができる。以下は各ベースクラスの主な関数群とその役割である。

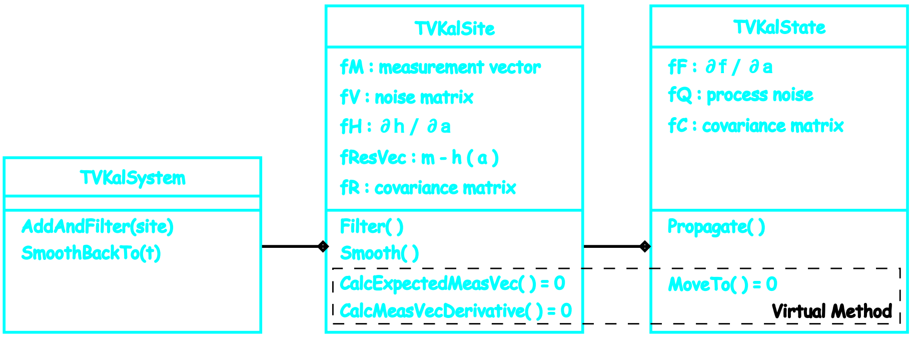


Figure 4.3: Kalman Filter Base Class Library

TVKalSystem クラス（TObjArray を継承：Site の配列）

TObjArray を継承しているので、それ自体が配列の属性を持ち、各測定点 (Site) での測定情報を保持する。異なる Site 間の関係を扱うことのできる問題とする系を代表する基底クラスである。

AddAndFilter(Site)
: 新しい Site の Filtering を行い、結果が許容できれば Site の配列に加える。

SmoothBackTo(k)
: n 番目の Site から逆向きに k 番目の Site まで Smoothing を行う。

TVKalSite クラス（TObjArray を継承：State の配列）

これもそれ自体が配列属性を持ち、異なる段階での状態 (State) の推定値 (Predicted , Filtered , Smoothed) を保持する。データメンバとして測定ベクトルや、その誤差行列を持ち、対応する測定点での測定情報、状態の推定値などをまとめた Site を代表する基底クラスである。

CalcExpectedMeasVec($\boldsymbol{a}_{k-1}$), CalcMeasVecDerivative($\boldsymbol{a}_{k-1}$)
: 状態ベクトルを引数に取り、対応する測定ベクトル $\boldsymbol{h}_k(\boldsymbol{a}_k^{k-1})$、及びその状態ベクトルに関する微分 $\boldsymbol{H}_k$ を計算する。これらは問題にする系によるため、純粋仮想関数である。実装は派生クラスで行われる。

Filter( )
: Site は predicted state、測定ベクトルの情報を持つのでその測定ベクトルの Filter を自分自身の中で閉じて実行できる。このメンバ関数は Filtering を行い、この測定点の適否を判断するとともに、$\boldsymbol{a}_k$, $\boldsymbol{C}_k$ を求める。

Smooth($\boldsymbol{a}_{k+1}^n$)
: Smoothing を行うには、次の点の Smoothing の結果が必要となる。このメンバ関数はこれを引数として受け取り Smoothing を行い、$\boldsymbol{a}_k^n$, $\boldsymbol{A}_k$ を求める。

TVKalState クラス（TKalMatrix クラスを継承：状態ベクトルを表す）

MoveTo(Site)
: 引数で与えられた次の Site(k) へ状態を移動させ、$\boldsymbol{a}_k^{k-1}(=\boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1}))$, $\boldsymbol{F}_{k-1}$, $\boldsymbol{Q_{k-1}}$ を計算する。これらは問題にする系によるため、純粋仮想関数である。実装は派生クラスで行われる。

Propagate(Site)
: 内部で MoveTo(site) を呼び、引数で与えられた次の Site(k) へ状態を移動させ、そこでの $\boldsymbol{C}_k^{k-1}$ を計算する。(k-1) から (k) へ移動する際に必要な計算で、問題の詳細（仮想関数部分）によらないアルゴリズムを実装する。

4.3 飛跡再構成プログラムへの組込み

Kalman Filter を問題にする特定の系に応用するためには、4.2 節で作成したベースクラスの純粋仮想関数を特定の系に対応させた形で実装する必要がある。ここでは Kalman Filter をトラックフィッティングに応用することを目的とする。図 4.4 は、トラックフィッティングに必要なクラス群である。このトラックフィッティングクラスライブラリは、前節の Kalman Filter のベースクラスライブラリを継承し、特定の測定器には依存しないが、トラックフィッティングに機能を限定した際に共通化できるアルゴリズム部分を実装するベースクラス群である。これに対しジオメトリクラスライブラリは、Kalman Filter とは独立 (Kalman Filter ライブラリに依存しない) 測定器の形状を記述するためのトラック、円筒、つつみ形などの幾何学的オブジェクトとそのベースクラス群である。以下にトラックフィッティングクラスライブラリとジオメトリクラスライブラリの詳細を述べる。

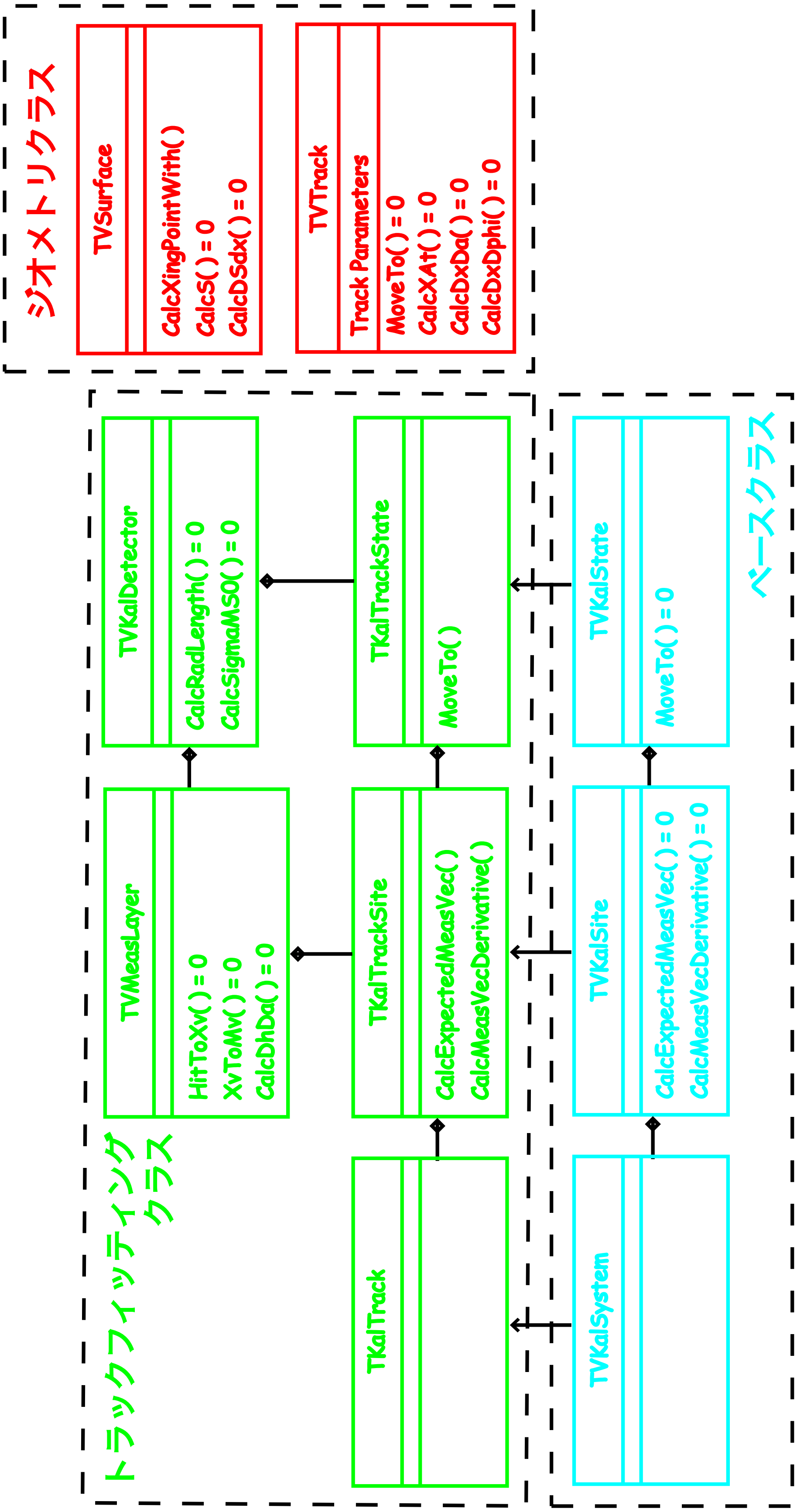


Figure 4.4: Kalman Filter Track Fitting Class Library

4.3.1 トラックフィッティングクラスライブラリの実装

既に述べた通り、Kalman Filter をトラックフィッティングに応用するためには、ベースクラスの TVKalSystem クラス、TVKalSite クラス、TVKalState クラスの派生クラスを作成し、純粋仮想関数を対応させた形で実装する必要がある。

TKalTrack クラス (TVKalSystem の派生クラス)

トラックは測定点（ヒット点）の集合とみなせるので、Site すなわち測定点の配列である TVKalSystem を継承したオブジェクトとして定義するのが適当である。TVKalSystem クラスには純粋仮想関数が存在しないので、その派生クラスである TKalTrack には新たに実装すべきメンバ関数はない。

TKalTrackSite クラス (TVKalSite の派生クラス)

トラックフィッティングの場合には状態ベクトルはトラックパラメータ、測定ベクトルは各ヒット点での読み出し座標になる。一方実装すべきベースクラスの純粋仮想関数は、CalcExpectedMeasVec( ) と CalcMeasVecDerivative( ) である。先に述べたように CalcExpectedMeasVec( ) は $\boldsymbol{h}_k(\boldsymbol{a}_k^{k-1})$ を、CalcMeasVecDerivative( ) はそのトラックパラメータ微分 $\boldsymbol{H}_k$ を計算するメソッドである。

JLC-CDC を含む様々な検出器では、荷電粒子の通過位置は、飛跡に沿って連続的に測定されるのではなく、離散的に測定される。このときの測定箇所の集合が作る仮想的な面を測定面と呼ぶことにする。トラックパラメータ $\boldsymbol{a}$ で決まるトラックの、k 番目の測定点に対応する測定面 $S_k(\boldsymbol{x}) = 0$ での測定値 $\boldsymbol{h}_k(\boldsymbol{a})$ を予想するためには、

- トラックパラメータ $\boldsymbol{a}$ から測定面 $S_k(\boldsymbol{x}) = 0$ との交点 $\boldsymbol{x}_k(\boldsymbol{a})$ を求める方程式

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{x}_k(\boldsymbol{a}) \\ &= \boldsymbol{x}\left(\phi_k(\boldsymbol{a}), \boldsymbol{a}\right) \end{aligned} \tag{4.33}$$

- トラックと測定面との交点 $\boldsymbol{x}_k(\boldsymbol{a})$ を測定ベクトルに変換する方程式

$$\boldsymbol{m}_k = \boldsymbol{m}_k(\boldsymbol{x}_k) \tag{4.34}$$

が必要になる。これらは、使用する飛跡や測定面の幾何学的形状によるものなので、飛跡や測定面の情報を持つジオメトリクラスを作り、そちらに記述することにする。これらのクラスについては後に詳述する。それらを使うことにより、トラックパラメータ $\boldsymbol{a}$ に対応する測定誤差がないときの測定ベクトルが計算でき

$$\begin{aligned} \boldsymbol{m}_k &= \boldsymbol{m}_k\left(\boldsymbol{x}_k(\boldsymbol{a})\right) \\ &= \boldsymbol{m}\left(\boldsymbol{x}\left(\phi_k(\boldsymbol{a}), \boldsymbol{a}\right)\right) \\ &= \boldsymbol{h}_k(\boldsymbol{a}) \end{aligned} \tag{4.35}$$

と表すことができる。これを測定方程式と呼ぶ。ただし、

$$\boldsymbol{x} = \boldsymbol{x}\left(\phi, \boldsymbol{a}\right) = \begin{pmatrix} x\left(\phi, \boldsymbol{a}\right) \\ y\left(\phi, \boldsymbol{a}\right) \\ z\left(\phi, \boldsymbol{a}\right) \end{pmatrix} \tag{4.36}$$

は $\boldsymbol{a}$ をトラックパラメータ、ϕ を飛跡に沿った粒子の位置を示す媒介変数とするトラック（飛跡）の方程式で、局所的には磁場の有無に対応してヘリックスあるいは直線となる。トラックのパラメータ表示については後述する。CalcExpectedMeasVec( ) メソッドにはこのアルゴリズムが実装してある。

次にこの測定方程式をトラックパラメータで微分する。一般に、$\boldsymbol{h}_k$ のトラックパラメータ微分 $\boldsymbol{H}_k \equiv \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{a}}$ は次のように書ける。

$$\boldsymbol{H}_k \equiv \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{a}} = \left(\frac{\partial \boldsymbol{m}_k}{\partial \boldsymbol{x}}\right)\left(\frac{\partial \boldsymbol{x}\ (\phi_k(\boldsymbol{a}), \boldsymbol{a})}{\partial \boldsymbol{a}}\right) \tag{4.37}$$

ただし

$$\frac{\partial \boldsymbol{x}\ (\phi_k(\boldsymbol{a}), \boldsymbol{a})}{\partial \boldsymbol{a}} = \frac{\partial \boldsymbol{x}}{\partial \phi_k}\frac{\partial \phi_k}{\partial \boldsymbol{a}} + \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{a}} \tag{4.38}$$

このうち、位置座標 $\boldsymbol{x}$ の ϕ 偏微分、トラックパラメータ偏微分は飛跡の方程式で決まり、ジオメトリクラスの TVTrack クラスから得られる。一方、測定ベクトル $\boldsymbol{m}_k$ の位置座標微分は測定面の幾何学的形状とその上での読み出し座標の定義に依存し、TVMeasLayer クラスから得られる。ϕ のトラックパラメータ微分は、$\frac{\partial S}{\partial \boldsymbol{x}}$ が測定面の形状から、また $\frac{\partial \boldsymbol{x}}{\partial \phi}$ 及び $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{a}}$ がトラック（飛跡）の方程式から決まることに注意すると、測定面とトラックの交点の方程式

$$S_k\ (\boldsymbol{x}(\phi_k, \boldsymbol{a})) = 0 \tag{4.39}$$

をトラックパラメータで微分した式

$$0 = \frac{\partial S_k}{\partial \boldsymbol{a}} = \frac{\partial S_k}{\partial \boldsymbol{x}}\left(\frac{\partial \boldsymbol{x}}{\partial \phi_k}\frac{\partial \phi_k}{\partial \boldsymbol{a}} + \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{a}}\right) \tag{4.40}$$

から、一次方程式の解として必ず得られる。

$$\frac{\partial \phi_k}{\partial \boldsymbol{a}} = -\left(\frac{\partial S_k}{\partial \boldsymbol{x}}\right)\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{a}}\right)\Bigg/\left(\frac{\partial S_k}{\partial \boldsymbol{x}}\right)\left(\frac{\partial \boldsymbol{x}}{\partial \phi_k}\right) \tag{4.41}$$

CalcMeasVecDerivative( ) メソッドにはこのアルゴリズムが実装してある。

TKalTrackState クラス (TVKalState の派生クラス)

この派生クラスでは、TVKalState クラスの純粋仮想関数 MoveTo(site) を実装する。そのためには、$\boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1})$、$F_{k-1}$、$Q_{k-1}$ を定義する必要がある。$\boldsymbol{f}_{k-1}(\boldsymbol{a}_{k-1})$ は、$k-1$ 番目の測定点におけるトラックパラメータの Filter 値 $\boldsymbol{a}_{k-1}$ を、k 番目の測定点におけるトラックパラメータ $\boldsymbol{a}_k$ へ外挿する関数、F_{k-1} は、そのトラックパラメータ微分である。これらの関数は、一般に、磁場の有無などの外的状況や、トラックのパラメータ表示の仕方に依存する。そのため、ジオメトリクラスで定義されている TVTrack クラスの MoveTo($\boldsymbol{x}$) 関数を呼ぶことで一般化している。Q_{k-1} はプロセスノイズの誤差行列で、測定点 $k-1$ と測定点 k のみでは決まらず、その間にある物質の分布に依存する。現在のバージョンでは、TVKalDetector クラスの CalcSigmaMS0(from , to) メソッドを呼んで多重散乱角の標準偏差 (σ^0_{MS}) をもらい、次式で計算している。

$$Q_{k-1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1+\tan^2\lambda & 0 & 0 & 0 \\ 0 & 0 & (\kappa\tan\lambda)^2 & 0 & \kappa\tan\lambda(1+\tan^2\lambda) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \kappa\tan\lambda(1+\tan^2\lambda) & 0 & (1+\tan^2\lambda)^2 \end{pmatrix} \tag{4.42}$$

この式は、多重散乱が測定点 $k-1$ と測定点 k の間で、幅 σ^0_{MS} のガウス分布にしたがって一度だけ起こるとみなせるとする近似式である[3]。

TVMeasLayer クラス

測定器は様々な形状を持つ測定面の集合とみなすことができる。各測定面の特性（形状、および読み出し座標の性質）は TVMeasLayer クラスとジオメトリクラスライブラリに属する TVSurface クラスを継承した幾何学的形状クラス (例えば円筒、つつみ形など) の多重継承クラスで定義される。TVMeasLayer は $\boldsymbol{x}_k$ と $\boldsymbol{m}_k$ の関係を与えるインタフェースを規定するための仮想クラスである。ジオメトリクラスライブラリでトラックと面の交点などの公式が定義されているので、派生クラスでは測定面とトラックの交点 $\boldsymbol{x}_k(\boldsymbol{a})$ と $\boldsymbol{m}_k$ の関係 (式 4.34 に対応) を定義するだけでよい。以下は TVMeasLayer クラスの主な関数群とその役割である。

XvToMv($\boldsymbol{x}$)
: トラックと測定面との交点 $\boldsymbol{x}$ を測定ベクトル $\boldsymbol{m}$ に変換する。式 4.34 に対応する純粋仮想関数。

HitToXv($\boldsymbol{m}$)
: 測定ベクトル $\boldsymbol{m}$ をトラックと測定面との交点 $\boldsymbol{x}$ に変換する。式 4.34 に対応する純粋仮想関数。

CalcDhDa($\boldsymbol{x}$, $\partial\boldsymbol{x}\left(\phi(\boldsymbol{a}),\boldsymbol{a}\right)/\partial\boldsymbol{a}$, H)
: トラックと測定面の交点 $\boldsymbol{x}$ を引数として、$\boldsymbol{x}$ を測定ベクトルに変換する方程式 (式 4.34 あるいは XvToMv($\boldsymbol{x}$)) の $\boldsymbol{x}$ 偏微分を計算する。これと引数で受け取った、$\partial\boldsymbol{x}\left(\phi(\boldsymbol{a}),\boldsymbol{a}\right)/\partial\boldsymbol{a}$ を用いて、測定ベクトルのトラックパラメータ偏微分を計算し、H に入れて返す。純粋仮想関数。

TVKalDetector クラス (TObjArray を継承)

TVSurface と TVMeasLayer を多重継承した測定面を表す個々の測定器に特化した派生クラスを配列要素とする測定器系を表す仮想クラスで、測定面と測定面の間の関係、その間の物質に関する情報を提供するメソッドのインターフェースを規定する。測定面 (TVMeasLayer クラス) を TVKalDetector クラスに Add していくことで、一般には異なる形状の測定面を持つ測定器の集合を一体に扱える。

CalcRadLength(from,to)
: TVKalDetector の配列要素である測定面の from から to までの Radiation Length を計算する。純粋仮想関数。

CalcSigmaMS0(from,to)
: TVKalDetector の配列要素である測定面の from から to までの多重散乱を計算する。純粋仮想関数。

[3]測定点 $k-1$ と測定点 k の間の物質量が少なく、距離も小さい時に有効な近似である。

4.3.2 ジオメトリクラスライブラリ

最後にジオメトリクラスライブラリに定義されているクラスの主な関数群とその役割と、実際の幾何学的オブジェクトについて述べる。

TVTrack クラス

トラックパラメータ $\boldsymbol{a}$ と、飛跡に沿った粒子の位置を示す媒介変数 ϕ との関係を与える関数のインターフェースを規定する仮想クラス。

MoveTo($\boldsymbol{x}$)
: 基準点 (pivot：後述) を $\boldsymbol{x}$ に移動するメソッド。$k-1$ 番目の測定点におけるトラックパラメータの値 $\boldsymbol{a}_{k-1}$ を、k 番目の測定点におけるトラックパラメータ $\boldsymbol{a}_k$ へ外挿する際に用いる。純粋仮想関数。

CalcXAt(ϕ)
: ϕ を引数に取り、トラック上の粒子の位置座標（飛跡の方程式）を計算する。純粋仮想関数。

CalcDxDa(ϕ)
: ϕ を引数に取り、位置座標（飛跡の方程式）のトラックパラメータ偏微分を計算する。純粋仮想関数。

CalcDxDphi(ϕ)
: ϕ を引数に取り、位置座標（飛跡の方程式）の ϕ 偏微分を計算する。飛跡の接線ベクトルを計算することに対応する。純粋仮想関数。

TVSurface クラス

測定面の幾何学的形状を定義するクラスが共通して持つべきインターフェースを規定する仮想クラス。

CalcS($\boldsymbol{x}$)
: 位置座標 $\boldsymbol{x}$ を引数に取り、面の方程式の左辺 ($S(\boldsymbol{x})$) を求める。純粋仮想関数。

CalcDSDx($\boldsymbol{x}$)
: 位置座標 $\boldsymbol{x}$ を引数に取り、面の方程式の左辺の位置偏微分 ($\frac{\partial S_k}{\partial \boldsymbol{x}}$) を計算する。純粋仮想関数。

CalcXingPointWith($\boldsymbol{a}$)
: 与えられたトラックパラメータに対してニュートン法で $S_k\ (\boldsymbol{x}(\phi, \boldsymbol{a})) = 0$ を解き、トラックと面の交点を与える ϕ 及び交点の位置ベクトルを求める。

$$0 \simeq S\ (\boldsymbol{x}(\phi_n, \boldsymbol{a})) \simeq S\ (\boldsymbol{x}(\phi_{n-1}, \boldsymbol{a})) + (\phi_n - \phi_{n-1}) \left(\frac{\partial S}{\partial \boldsymbol{x}}\right)_{n-1} \left(\frac{\partial \boldsymbol{x}}{\partial \phi}\right)_{n-1}$$

$$\phi_n = \phi_{n-1} - S\ (\boldsymbol{x}(\phi_{n-1}, \boldsymbol{a})) \bigg/ \left(\frac{\partial S}{\partial \boldsymbol{x}}\right)_{n-1} \left(\frac{\partial \boldsymbol{x}}{\partial \phi}\right)_{n-1} \tag{4.43}$$

THelicalTrack クラス (TVTrack クラスの派生クラス)

THelicalTrack クラスは TVTrack を継承し、その純粋仮想関数を Helix トラックの幾何学的性質に従って実装するものである。そのためにはまず、トラックパラメータ ($\boldsymbol{a}$) を定義しなければならない。一般に Helix を表すパラメータの選び方はいろいろあるが、特定の測定点におけるトラックのふるまいを問題にする場合には、その測定点に対応する適当な基準点 (pivot と呼ばれる）をとり、それを基準として決まる次の 5 つのパラメータを用いるのがよい [11]。

$$\begin{cases} d_\rho & : \ z \text{ 軸 (磁場と平行な軸) に垂直な平面 } (x-y \text{ 平面）での、Helix と基準点の間の距離} \\ \phi_0 & : \ \text{Helix の中心点に対する、基準点の方位角} \\ \kappa & : \ \text{荷電粒子の電荷 Q を横運動量 } (x-y \text{ 平面に射影した運動量})Pt \text{ で割ったもの} \\ d_z & : \ \text{Helix と基準点の間の } z \text{ 軸方向の距離} \\ \tan\lambda & : \ \text{Helix のビーム軸に垂直な面からの角度 (Dip Angle)} \end{cases} \tag{4.44}$$

今後、この 5 つのパラメータをまとめて 5 成分の列ベクトル $\boldsymbol{a} = (d_\rho, \phi_0, \kappa, d_z, \tan\lambda)^T$ で表すことにする。これを用いると、Helix 上の任意の点 $\boldsymbol{x} = (x, y, z)^T$ を表す式（飛跡の方程式）は次のようになる。

<u>CalcXAt(ϕ)</u>

$$\boldsymbol{x}\ (\phi, \boldsymbol{a}) = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_0 & + & d_\rho \cos\phi_0 & + & \frac{\alpha}{\kappa}\,(\cos\phi_0 - \cos(\phi_0+\phi)) \\ y_0 & + & d_\rho \sin\phi_0 & + & \frac{\alpha}{\kappa}\,(\sin\phi_0 - \sin(\phi_0+\phi)) \\ z_0 & + & d_z & - & \frac{\alpha}{\kappa}\tan\lambda\cdot\phi \end{pmatrix} \tag{4.45}$$

ただし、$\boldsymbol{x}_0 = (x_0, y_0, z_0)^T$ を基準点の座標とした。ρ は

$$\begin{aligned} \kappa &= Q/Pt \\ \rho &= \alpha/\kappa \end{aligned} \tag{4.46}$$

で表される（荷電粒子の電荷による）符号付きの Helix の半径であり、α は磁場一定の場合には $1/cB$（c は光速度、B は磁場）に等しい。ϕ は、基準点から Helix 上の点への、Helix の中心に対する偏向角である。

図 4.5 は、これらのパラメータを粒子の電荷が正の場合と負の場合に分けて表したものである。注意すべき点は、ϕ_0 の定義が電荷が正の場合と負の場合では異なることである。実際に飛跡再構成を行う際には、勿論、そのトラックの粒子の電荷はわからない。したがって、トラックが曲がる方向を見て電荷の正負を決めることになるが、高い運動量のトラックでは曲率が非常に小さくなるため、フィッティングの過程で曲率が正負の境を超える場合がある。このとき、曲率が正（負）から負（正）に変化する際に、Helix の中心はトラックを挟んで反対側に移動してしまい、基準点の方位角はその瞬間に π だけずれてしまう。このような不連続を避けるため、ϕ の定義は κ の正負によってあらかじめ別に分けられている。

また、4.45 式を用いて飛跡の方程式の ϕ 偏微分を与える関数は

<u>CalcDxDphi(ϕ)</u>

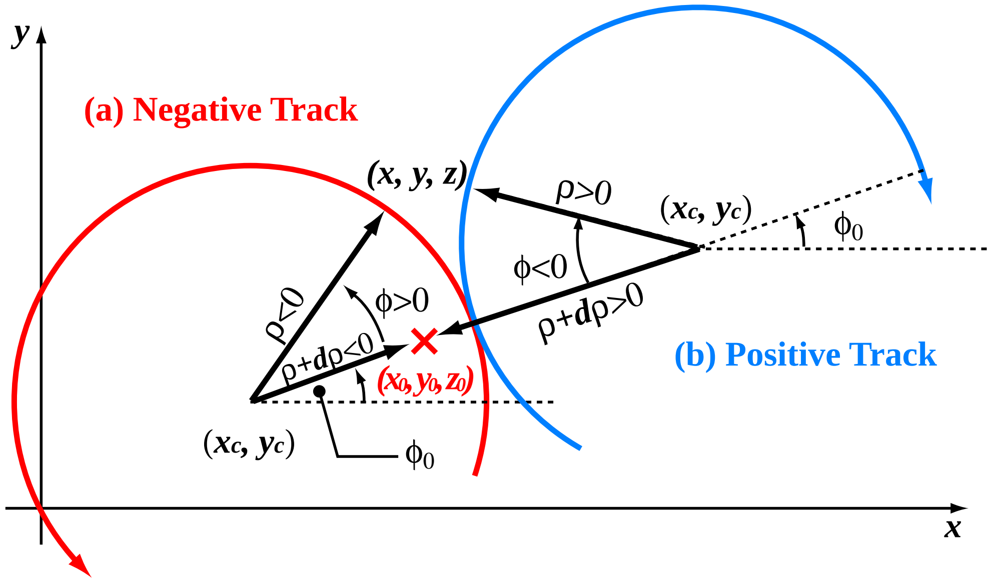


Figure 4.5: トラックパラメータ

$$\frac{\partial \boldsymbol{x}\ (\phi, \boldsymbol{a})}{\partial \boldsymbol{\phi}} = \begin{pmatrix} \dfrac{\partial x}{\partial \phi} \\ \dfrac{\partial y}{\partial \phi} \\ \dfrac{\partial z}{\partial \phi} \end{pmatrix} = \begin{pmatrix} \dfrac{\alpha}{\kappa} \sin(\phi_0 + \phi) \\ -\dfrac{\alpha}{\kappa} \cos(\phi_0 + \phi) \\ -\dfrac{\alpha}{\kappa} \tan \lambda \end{pmatrix} \tag{4.47}$$

である。飛跡の方程式のトラックパラメータ偏微分 $\frac{\partial \boldsymbol{x}\ (\phi, \boldsymbol{a})}{\partial \boldsymbol{a}}$ は

CalcDxDa(ϕ)

$$\frac{\partial \boldsymbol{x}\ (\phi, \boldsymbol{a})}{\partial \boldsymbol{a}} = \begin{pmatrix} \frac{\partial x}{\partial \boldsymbol{a}} \\ \frac{\partial y}{\partial \boldsymbol{a}} \\ \frac{\partial z}{\partial \boldsymbol{a}} \end{pmatrix} \tag{4.48}$$

ただし

$$
\frac{\partial x}{\partial \boldsymbol{a}} = \begin{pmatrix} \frac{\partial x}{\partial d_\rho} \\ \frac{\partial x}{\partial \phi_0} \\ \frac{\partial x}{\partial \kappa} \\ \frac{\partial x}{\partial d_z} \\ \frac{\partial x}{\partial \tan\lambda} \end{pmatrix}^T = \begin{pmatrix} \cos\phi_0 \\ -(d_\rho + \frac{\alpha}{\kappa})\sin\phi_0 + \frac{\alpha}{\kappa}\sin(\phi_0+\phi) \\ -\frac{\alpha}{\kappa^2}(\cos\phi_0 - \cos(\phi_0+\phi)) \\ 0 \\ 0 \end{pmatrix}^T
$$

$$
\frac{\partial y}{\partial \boldsymbol{a}} = \begin{pmatrix} \frac{\partial y}{\partial d_\rho} \\ \frac{\partial y}{\partial \phi_0} \\ \frac{\partial y}{\partial \kappa} \\ \frac{\partial y}{\partial d_z} \\ \frac{\partial y}{\partial \tan\lambda} \end{pmatrix}^T = \begin{pmatrix} \sin\phi_0 \\ (d_\rho + \frac{\alpha}{\kappa})\cos\phi_0 - \frac{\alpha}{\kappa}\cos(\phi_0+\phi) \\ -\frac{\alpha}{\kappa^2}(\sin\phi_0 - \sin(\phi_0+\phi)) \\ 0 \\ 0 \end{pmatrix}^T \tag{4.49}
$$

$$
\frac{\partial z}{\partial \phi} = \begin{pmatrix} \frac{\partial z}{\partial d_\rho} \\ \frac{\partial z}{\partial \phi_0} \\ \frac{\partial z}{\partial \kappa} \\ \frac{\partial y}{\partial d_z} \\ \frac{\partial z}{\partial \tan\lambda} \end{pmatrix}^T = \begin{pmatrix} 0 \\ 0 \\ \frac{\alpha}{\kappa^2}\phi\tan\lambda \\ 1 \\ -\frac{\alpha}{\kappa}\phi \end{pmatrix}^T
$$

と表せる。

THype クラス (TVSurface クラスの派生クラス)

THype クラスは TVSurface の規定する純粋仮想関数に対して一葉双曲面の持つ幾何学的性質を実装した派生クラスである。測定面 S の方程式には、一葉双曲面の方程式

<u>CalcS($\boldsymbol{x}$)</u>

$$
S(\boldsymbol{x}) = x^2 + y^2 - r_{z=0}^2 - z^2\tan^2 A = 0 \tag{4.50}
$$

の左辺が実装されている。ここで、$r_{z=0}$ は一葉双曲面の $z=0$ 平面における半径、A はワイヤーのステレオ角である[4]。従って、CalcDSDx($\boldsymbol{x}$) としては

[4]アクシャルレイヤーでは、$\tan A = 0$ になる。

<u>CalcDSDx($\boldsymbol{x}$)</u>

$$\frac{\partial S}{\partial \boldsymbol{x}} = \begin{pmatrix} \dfrac{\partial S}{\partial x} \\ \dfrac{\partial S}{\partial y} \\ \dfrac{\partial S}{\partial z} \end{pmatrix}^T = \begin{pmatrix} 2x \\ \\ 2y \\ \\ -2z\tan^2 A \end{pmatrix}^T \tag{4.51}$$

を返す。

4.4 Kaman Filter によるトラックフィッティングの使い方

この章で実装されたトラックフィッティング・ライブラリも LEDA に組込まれる。以降は TVMeasLayer クラス、TVKalDetector クラスを問題にする検出器に対応させた形で実装した派生クラスを用意するだけで、Kalman Filter によるトラックフィッティングを簡単に使用することができるようになる。

Chapter 5

Kalman Filterによる飛跡再構成プログラムの評価

概要

4 章で作成した Kalman Filter を用いた飛跡再構成プログラムの評価を行い、それを利用してタイムスタンピング性能を研究する。まず、タイムスタンピング能力の必要性を述べる。次に、簡易シミュレータを用いて CDC と最近開発研究が始まった TPC のタイムスタンピング能力の比較を行う。次に、Kalman Filter を用いた飛跡再構成プログラムを Satellites に実装し、JUPITER でモンテカルロフルシミュレートしたデータを実際に解析する。

5.1 タイムスタンピング能力の必要性

JLC では、1 バンチトレイン 268.9nsec の間に、1.4nsec 間隔で 192 個のビームバンチが衝突する。268.9nsec という時間は、電子のドリフト時間（最大 4.5cm をドリフトする場合には 6.7μs）よりも短いので、1 バンチトレイン内で起こったバックグラウンドイベントのトラックは、全てドリフト領域に蓄積されていくことになる。

図 5.1 は、検出器にバンチ分離能がないとした場合に、起こり得る現象を示したものである。注目するイベントのジェットに、バックグラウンドのミニジェットが重なって見える。計算では、検出器にバンチ分離能が全くなかった場合、192 バンチで平均 20 個以上のバックグラウンドミニジェット事象が重なって見えてしまう。この場合にもヒッグス粒子の質量分布を得ることは不可能ではないが、ミニジェットの混入によりヒッグスに対する不変質量分解能が悪化してしまう。そこで電子陽電子衝突加速器の重要な特性であるクリーンな反応終状態の長所を活かすために、バンチを分離するためのタイムスタンピング能力が重要となる。

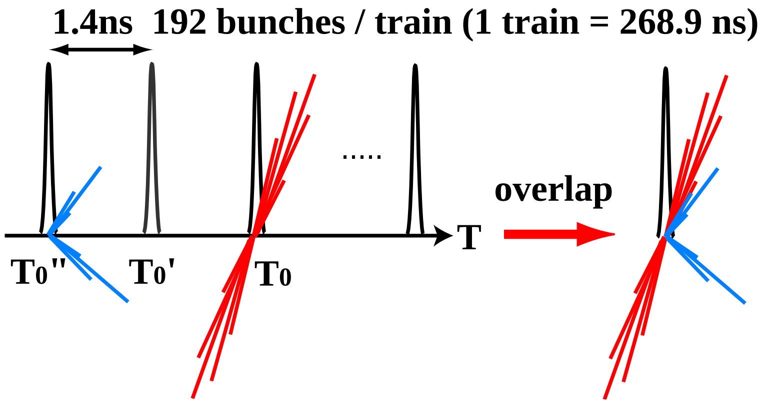


Figure 5.1: バンチ分離ができなかった場合のイベントの見え方。

5.1.1 CDC のタイムスタンピング

CDC のタイムスタンピングは、図 5.2 のように、スーパーレイヤー毎に互い違いになったセル構造で行う。正しい T_0 のトラック以外はスーパーレイヤーの境界でトラックが Δx ずれてしまう[1]。このずれはガス中での電子のドリフト速度 (v_{drift}) と T_0 のずれ (ΔT_0) の関数として次式で与えられる。

$$\Delta x = 2v_{drift} \times \Delta T_0 \tag{5.1}$$

一方、Δx はワイヤーあたりの位置分解能 (σ_{xy}) をヒット点の数 (ワイヤー本数:n) の平方根で割った値の 2 倍の分解能で測定できると考えられるので、CDC で予想される T_0 分解能は

$$\sigma_{\Delta T_0} \simeq \frac{\sigma_{xy}}{v_{drift}\sqrt{n}} \tag{5.2}$$

で与えられる。これから、$\sigma_{xy} = 85\ \mu$ m、$v_{drift} = 0.7$ cm/μs、n = 51 とすると $\sigma_{\Delta T_0} \simeq 1.7$ ns になる。

5.1.2 TPC のタイムスタンピング

TPC のように、検出器単体でのタイムスタンピング能力がない場合には、TPC と内側の検出器の間でうまく飛跡がつながるかどうかで時間分解能を得ることになる (図 5.4)。この場合には、判断材料となるトラックの接続点が 1 点しかないので、その 1 点での位置分解能がどれだけ期待できるかが焦点になる。正しい T_0 のトラック以外は、TPC と内側の検出器の間でトラックが Δz ずれてしまう。

$$\Delta z = v_{drift} \times \Delta T_0 \tag{5.3}$$

TPC で予想される T_0 分解能は、外装誤差を評価すると

[1] しかしステレオレイヤーの場合、図 5.3 のように異なったセルの間のトラックの不連続面がなくなり、なめらかにトラックがつながってしまうことがある。後に見るように、このためステレオレイヤーを含めた場合の T_0 分解能は、ここで解析的に評価した値よりも悪くなる。

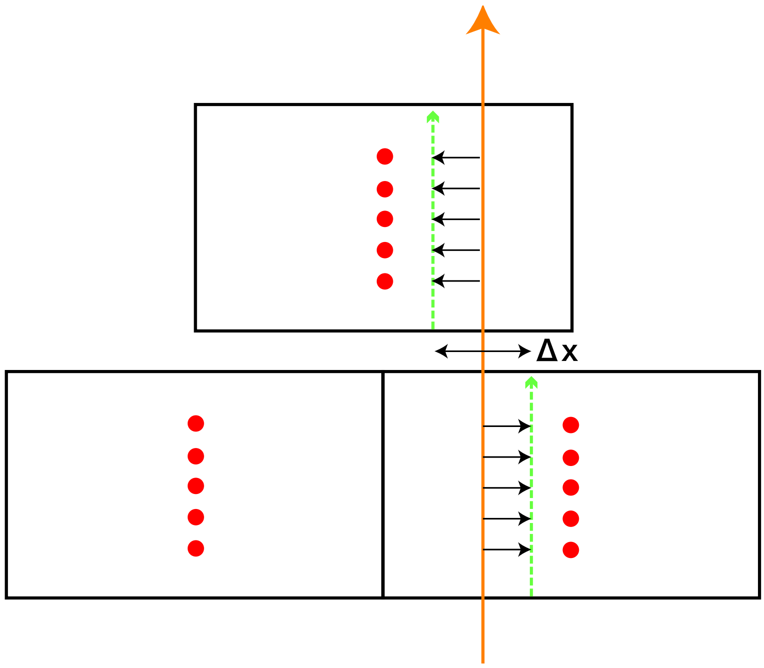


Figure 5.2: 互い違いになったセル構造により、間違った T_0 のトラックはなめらかにつながらない。

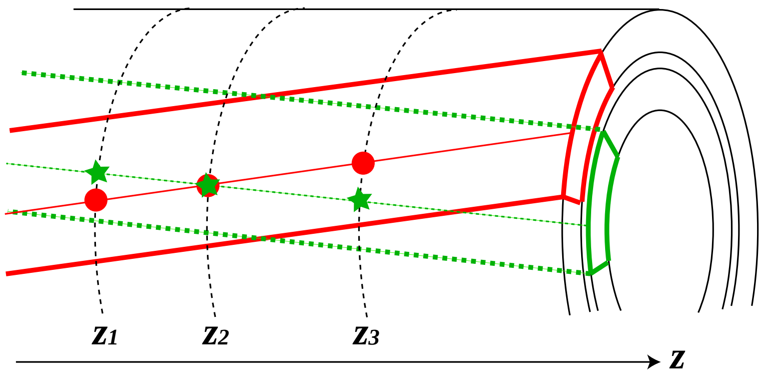


Figure 5.3: 互いにステレオ角をもつ 2 つのレイヤー間におけるヒット点の関係。セル中の直線は、等ドリフト距離を表す。同じドリフト距離を持つヒットでも、z_1、z_3 では $x-y$ 平面上で異なった ϕ を持つように見えるが、z_2 では同じ ϕ の上にあるように見える。異なったセルの間のトラックの不連続点も、同じ原理で、z を動かせばなめらかにつながる場所がみつかる。

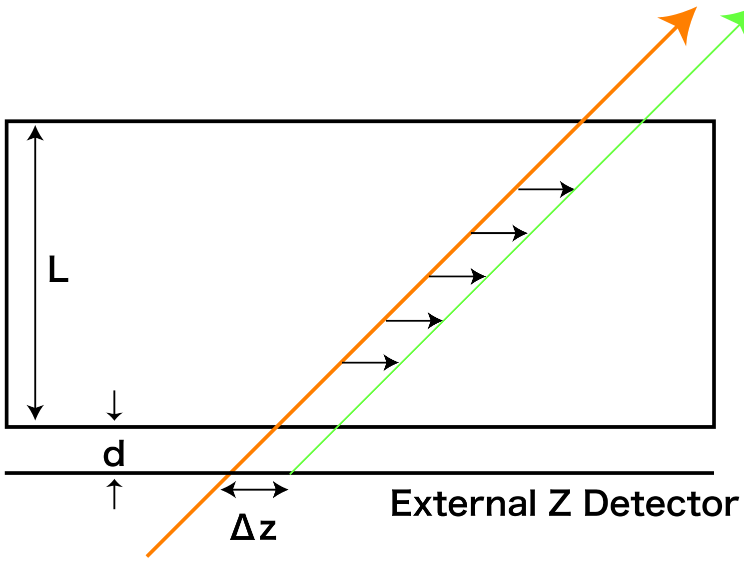


Figure 5.4: TPC には単独でのタイムスタンピング能力がないので、TPC の内側の External Z Detector と飛跡がなめらかにつながるかどうかでタイムスタンピングを行う。

$$\sigma_{\Delta T_0} \simeq \frac{2\sigma_z}{v_{drift}\sqrt{n}} \left[1 + 3\left(\frac{d}{L}\right) + 3\left(\frac{d}{L}\right)^2 \right]^{-\frac{1}{2}} \tag{5.4}$$

で与えられるが、これは $\frac{d}{L} \ll 1$ ならば次式に帰着する。

$$\sigma_{\Delta T_0} \simeq \frac{2\sigma_z}{v_{drift}\sqrt{n}} \tag{5.5}$$

(5.2) 式に比して factor2 異なることに注意する。k の式から、$\sigma_z = 500\ \mu$ m、$v_{drift} = 5$ cm/μs、n = 120 とすると内側検出器の位置分解能が無視できるとして、$\sigma_{\Delta T_0} \simeq 2.0$ ns になる。

5.2 簡易シミュレータによるトラックフィッティングプログラムの試検

タイムスタンピング性能の評価の前に、前章で作成したトラックフィッティングクラスライブラリの動作をテストする。そこで、まずトラックフィッティングクラスライブラリを用い、必要な派生クラスを CDC に対応させて実装する。次にそれをテストするために、多重散乱のみを考慮した簡易シミュレータを作成し、イベントを解析する。

5.2.1 CDC クラスの実装

トラックフィッティングライブラリを使用するためには、TVMeasLayer クラスの純粋仮想関数を派生クラスに実装し、ジオメトリを決めるパラメータ (測定面の半径 (z=0)、ステレオ角、セルの幅、z 方向の半長) を定義する必要がある。そこでまず、TVMeasLayer クラスと THype クラスを継承した CDCLayer クラス

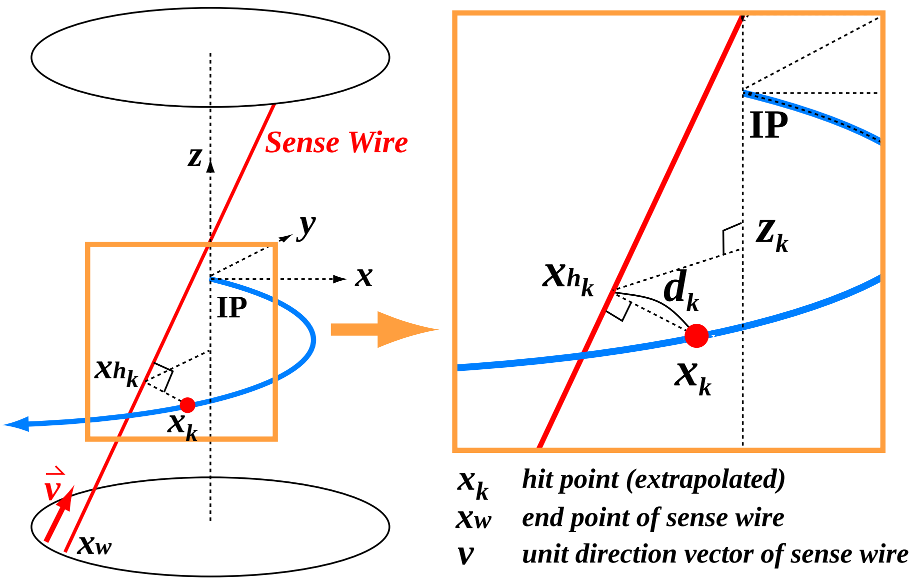


Figure 5.5: CDC の測定ベクトルの定義

を作り、トラックと測定面との交点 $\boldsymbol{x}$ を測定ベクトル m に変換する XvToMv($\boldsymbol{x}$)、測定ベクトル m をトラックと測定面との交点 $\boldsymbol{x}$ に変換する HitToXv($\boldsymbol{x}$)、測定方程式のトラックパラメータ微分 $\boldsymbol{H}_k$ を計算する CalcDhDa($\boldsymbol{x}$, $\partial\boldsymbol{x}$ $(\phi(\boldsymbol{a}),\boldsymbol{a})$ $/\partial\boldsymbol{a}$, H) の 3 つの純粋仮想関数を実装する。

CDC の場合、測定される量は測定面に沿って測られたドリフト距離と、電荷分配法によるセンスワイヤー方向の位置（z 座標と同等）である。ここでは、近似的に、トラックと測定面との交点の位置からセンスワイヤーに対して下ろした垂線の長さをドリフト距離として定義する[2]。トラックと測定面との交点 $\boldsymbol{x}_k(\boldsymbol{a})$ から求まるドリフト距離と z 座標はそれぞれ次のように書ける（図 5.5)。

$$\boldsymbol{h}_k(\boldsymbol{a}) = \begin{pmatrix} d_k(\boldsymbol{a}) \\ z_k(\boldsymbol{a}) \end{pmatrix} = \begin{pmatrix} \sqrt{(\boldsymbol{x}_k(\boldsymbol{a}) - \boldsymbol{x}_{wk})^2 - \{(\boldsymbol{x}_k(\boldsymbol{a}) - \boldsymbol{x}_{wk}) \cdot \boldsymbol{v}_k\}^2} \\ \{\boldsymbol{x}_{wk} + (\boldsymbol{x}_k(\boldsymbol{a}) - \boldsymbol{x}_{wk}) \cdot \boldsymbol{v}_k \boldsymbol{v}_k\} \cdot \boldsymbol{e}_z \end{pmatrix} \tag{5.6}$$

ただし、k 番目のセンスワイヤーの端点のひとつを $\boldsymbol{x}_{wk}$、ワイヤー方向の単位ベクトルを $\boldsymbol{v}_k$、z 軸方向の単位ベクトルを $\boldsymbol{e}_z$ とした。ゆえに、$\boldsymbol{h}_k$ のトラックパラメータ微分 $\boldsymbol{H}_k \equiv \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{a}}$ は次のように書ける。

$$\boldsymbol{H}_k \equiv \begin{pmatrix} \frac{\partial d}{\partial \boldsymbol{a}} \\ \frac{\partial z}{\partial \boldsymbol{a}} \end{pmatrix} = \begin{pmatrix} \frac{\partial d}{\partial \boldsymbol{x}} \\ \frac{\partial z}{\partial \boldsymbol{x}} \end{pmatrix} \left(\frac{\partial \boldsymbol{x}}{\partial \phi_k} \frac{\partial \phi_k}{\partial \boldsymbol{a}} + \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{a}} \right) \tag{5.7}$$

ここで、$\frac{\partial d}{\partial \boldsymbol{x}}$、$\frac{\partial z}{\partial \boldsymbol{x}}$ は

$$\begin{pmatrix} \frac{\partial d}{\partial \boldsymbol{x}} \\ \frac{\partial z}{\partial \boldsymbol{x}} \end{pmatrix} = \begin{pmatrix} \frac{(1-\boldsymbol{v}_k\boldsymbol{v}_k)(\boldsymbol{x}_k(\boldsymbol{a})-\boldsymbol{x}_{wk})}{\sqrt{(\boldsymbol{x}_k(\boldsymbol{a})-\boldsymbol{x}_{wk})^2-\{(\boldsymbol{x}_k(\boldsymbol{a})-\boldsymbol{x}_{wk})\cdot\boldsymbol{v}_k\}^2}} \\ \boldsymbol{v}_k\boldsymbol{v}_k \cdot \boldsymbol{e}_z \end{pmatrix} \tag{5.8}$$

[2] より正確には、垂線の長さではなく測定面に沿った距離や、ローレンツ角を考慮した長さにすべきである。

で 与えられ、また $\left(\frac{\partial \boldsymbol{x}}{\partial \phi_k}\frac{\partial \phi_k}{\partial \boldsymbol{a}}+\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{a}}\right)$ は飛跡 (ヘリックス) の方程式で決まり、既に式 (4.48) に与えてある。

以上を実装した CDCLayer クラスに、ジオメトリを決めるパラメータをレイヤー毎に設定する。さらに、TVKalDetector クラスを継承した CDCDetector クラスに、測定面間の多重散乱を計算する CalcRadLength( ) と CalcSigmaMS0( ) を実装する。CDCDetector クラスに CDCLayer クラスを Add することで、簡単に CDC のインストールができる。

シミュレーション結果

簡易シミュレータを用いて、100GeV のミューオンを磁場 3T 中に置かれた CDC に 5000 イベント打ち込んで得たデータの解析を行った。アクシャルレイヤーのみで構成された CDC と、アクシャルレイヤーとステレオレイヤーが混在する CDC について、χ^2 分布、confedence level に注目する。

図 5.6 は上が Axial レイヤーのみ、下が Axial レイヤーと Stereo レイヤーが混在した時の χ^2 分布である。測定点が真値のまわりに分布する場合、χ^2 の平均値は自由度の値に一致し、標準偏差 (RMS) は $\sqrt{2n}$ となる。自由度が十分大きければ χ^2 分布自体もガウス分布に近づく。CDC の測定点は 10(レイヤー数) × 5(1 レイヤー当たりのワイヤー数) で 50 点だが、Kalman Filter を行うためには注目する測定点の前の情報が必要となるので、一番内側の測定点の前に、分解能が事実上無限に悪いとみなせるダミー測定点を設けている。そのため測定点は 51 点になり、このそれぞれに 2 変数 (ドリフト距離と z) の自由度が存在するため、全部で 102 となるが、フィッティングの際にトラックパラメータとして 6 つの成分が拘束されるので、その分を差し引いて最終的な自由度は 96 になる。フィッティングした値は Axial only が 95.7、Axial+Stereo が 95.8 といずれも計算値とほぼ同等である。また RMS もほぼ理論値 $\sqrt{2n} = 13.9$ を再現している。これはフィットが期待通り動作していることを示している。

一方、図 5.7 の confidence level 分布を見ると、共に 0 付近にピークが見られる。confidence level 分布は正しいフィッティングが行われていれば一様分布となるはずなので、0 付近のピークは全イベントの 1%未満で実用上問題ないとはいえ、フィットが正しく行われない場合があることを示している。これは、Kalman Filter の初期でトラックパラメータの誤差が大きいときに、間違った L/R を取ってしまうことが原因と考えられる。ここで L/R と呼んでいるのは、トラックがセンスワイヤーの左側を通ったのか右側を通ったのかの区別のことである[3]。

5.2.2 TPC クラスの実装

CDC と同様に、TPCLayer クラスを実装する。TPC の測定面は円筒形であるので、TPCLayer は TVMeasLayer クラスと TCylinder クラスを継承して作成する。この場合は、測定される量は z 軸方向のドリフト距離と、基準となるパッドの中心からの ϕ 方向の距離である。z 軸方向のドリフト距離は z の正負に従って+z 側の端から測定したものか、-z 側から測定したものか区別する必要がある点に注意する。トラックと測定面との交点 $\boldsymbol{x}_k(\boldsymbol{a})$ から求まるドリフト距離と ϕ 方向の距離はそれぞれ次のように書ける（図 5.8)。

$$\boldsymbol{h}_k(\boldsymbol{a}) = \begin{pmatrix} R_k \cdot \phi_k(\boldsymbol{a}) \\ d_k(\boldsymbol{a}) \end{pmatrix} = \begin{pmatrix} R \cdot \tan^{-1} \frac{y_v}{x_v} \\ L/2 \mp z_v \end{pmatrix} \tag{5.9}$$

[3]CDC はドリフト時間を測定するので、粒子がワイヤー近くを通った場合には L/R の判定に不定性がありうる。

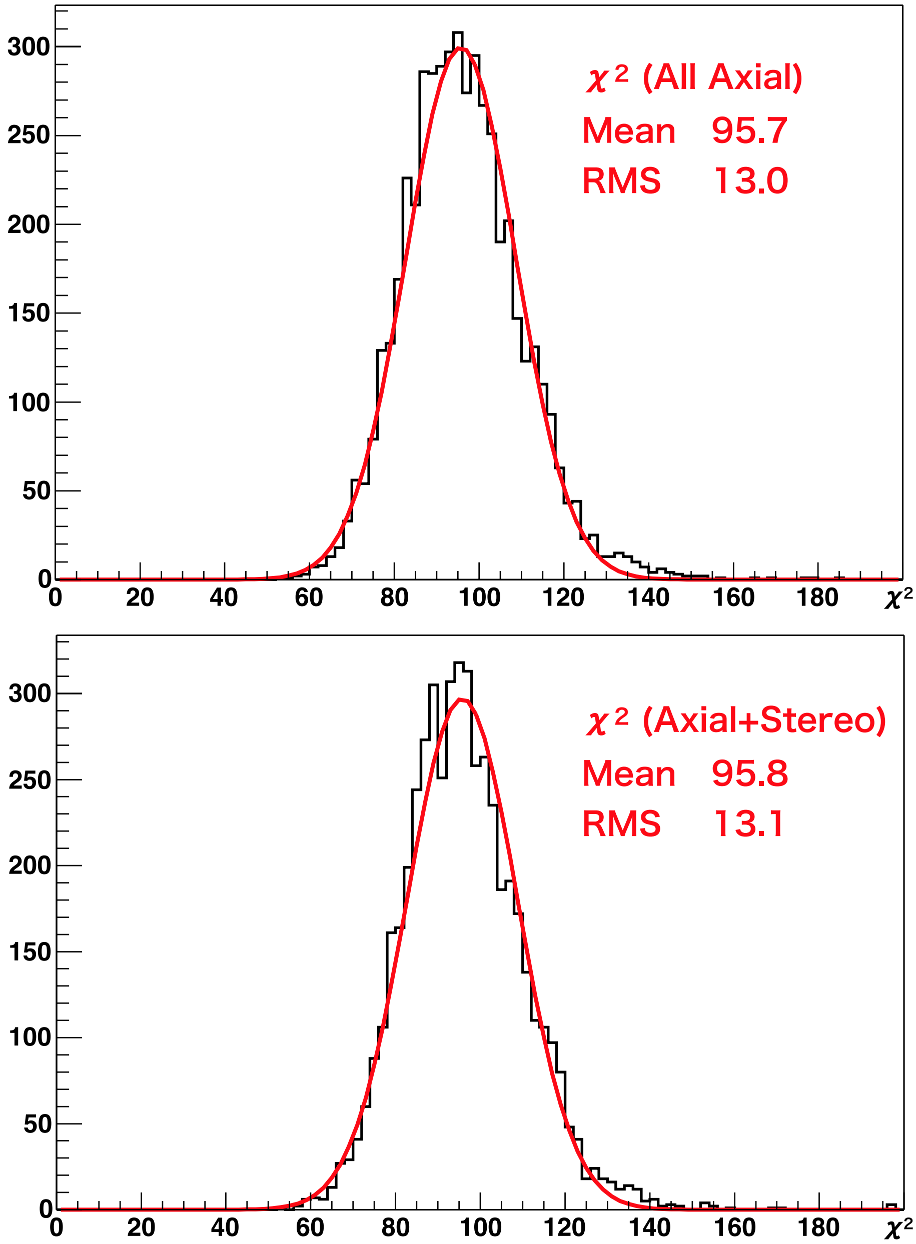


Figure 5.6: CDC の χ^2 分布

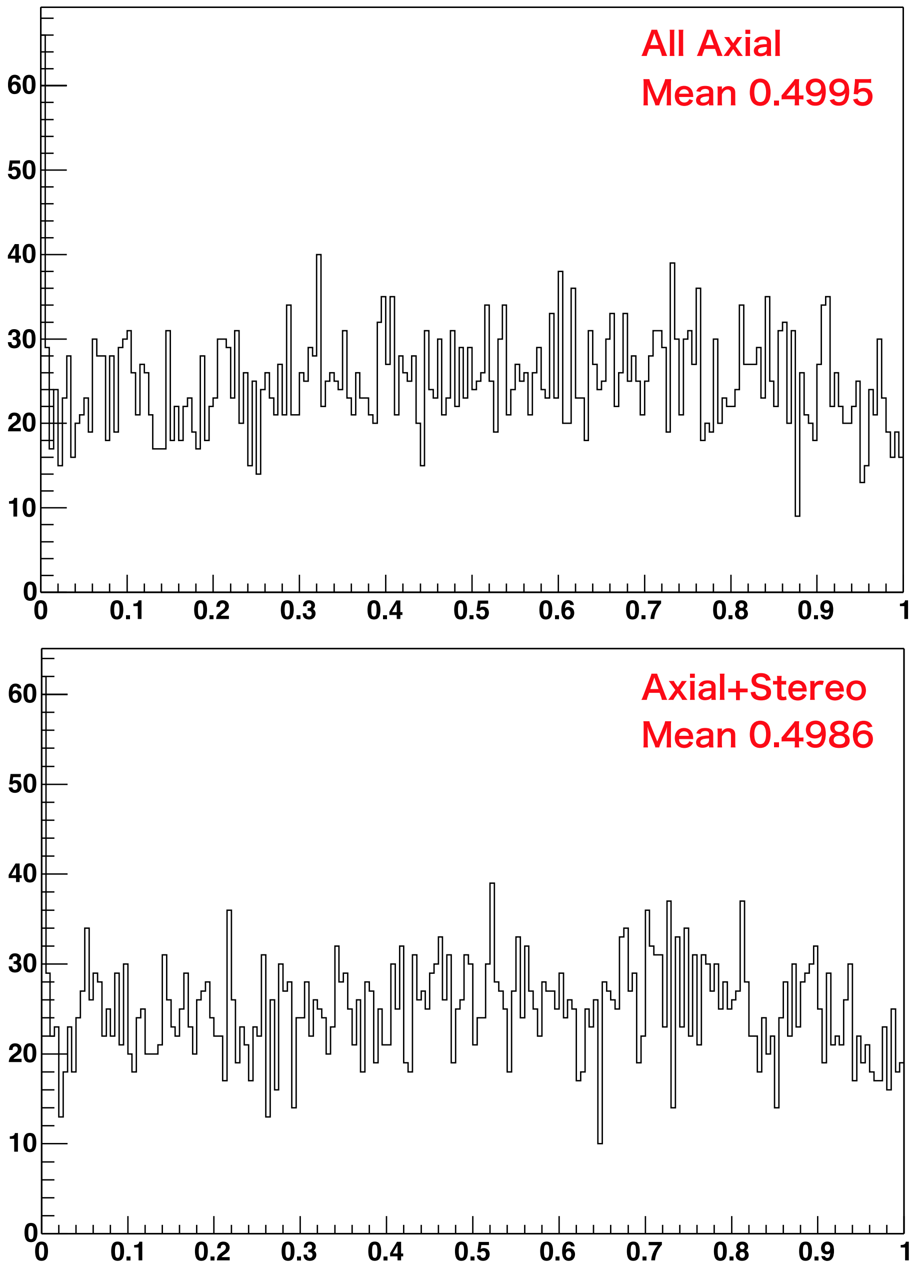


Figure 5.7: CDC の confidence level 分布

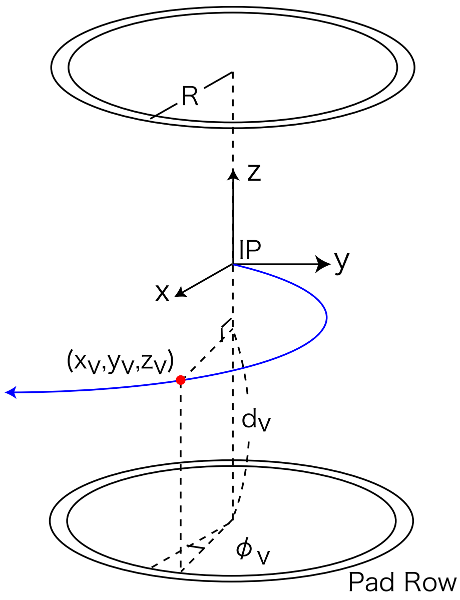


Figure 5.8: TPC の測定ベクトルの定義

従って、$\boldsymbol{h}_k$ のトラックパラメータ微分 $\boldsymbol{H}_k \equiv \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{a}}$ は次のように書ける。

$$\boldsymbol{H}_k \equiv \begin{pmatrix} \dfrac{\partial(R\cdot\phi_v)}{\partial \boldsymbol{a}} \\ \dfrac{\partial d_v}{\partial \boldsymbol{a}} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial(R\cdot\phi_v)}{\partial \boldsymbol{x}} \\ \dfrac{\partial d_v}{\partial \boldsymbol{x}} \end{pmatrix} \left(\frac{\partial \boldsymbol{x}}{\partial \phi_k}\frac{\partial \phi_k}{\partial \boldsymbol{a}} + \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{a}} \right) \tag{5.10}$$

ただし

$$\begin{pmatrix} \frac{\partial(R\cdot\phi_v)}{\partial \boldsymbol{a}} \\ \frac{\partial d_v}{\partial \boldsymbol{a}} \end{pmatrix} = \begin{pmatrix} \frac{-y_v}{R} & \frac{x_v}{R} & 0 \\ 0 & 0 & \mp 1 \end{pmatrix} \tag{5.11}$$

ここで符号 $(\mp)$ は、ヒット点の z 座標の $\pm$ に対応する。

シミュレーション結果

CDC と同様に簡易シミュレータを用いて、100GeV のミューオンを磁場 4T 中に 5000 イベント打ち込んで得たデータの解析を行う。

図 5.9 は対応する χ^2 分布である。TPC の測定点はダミーを含めて 120 点、このそれぞれに 2 変数 (ドリフト距離と ϕ) の自由度が存在するため、全部で 240 となるが、フィッティングの際にトラックパラメータとして 6 つの成分が拘束されるので、その分を差し引いて最終的な自由度は 234 になる。フィッティングした値は 233.9 と計算値と誤差の範囲で一致している。また、RMS も誤差の範囲で理論値 $\sqrt{2n} = 21.6$ を再現している。

図 5.10 の confidence level 分布を見ると、CDC にみられた 0 付近のピークがなくなっていることがわかる。これは TPC には L/R といった概念がないため、CDC のような問題 (L/R を間違える) がないためである。TPC の場合にはフィットが完全に正しく動作していることがわかる。

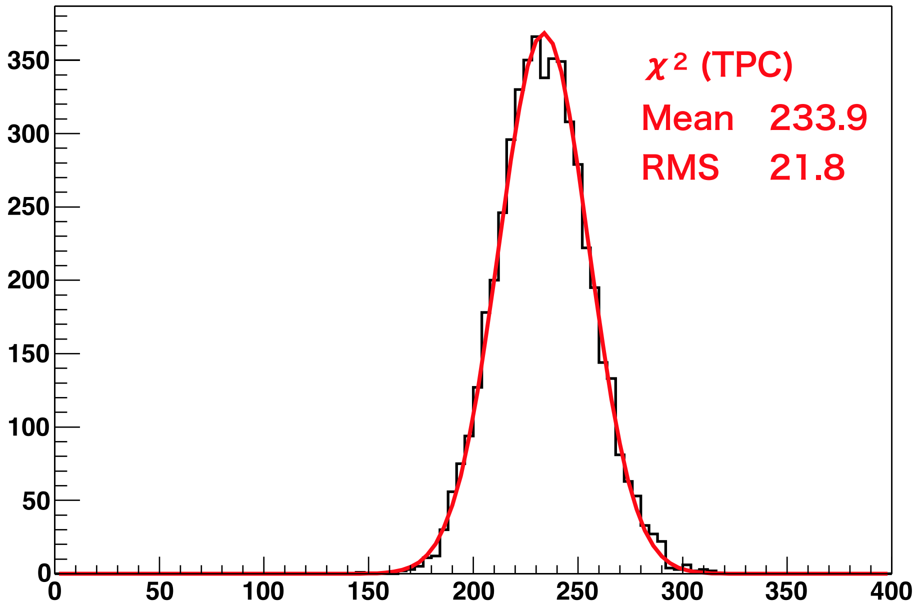


Figure 5.9: TPC の χ^2 分布

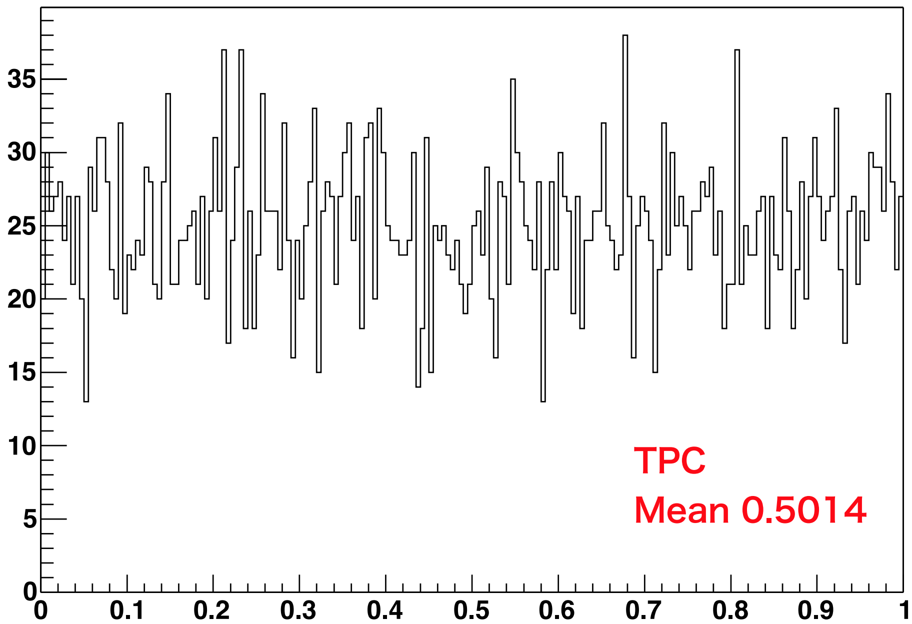


Figure 5.10: TPC の confidence level 分布

5.3 簡易シミュレータによる CDC と TPC のタイムスタンピング能力の比較

5.3.1 CDC、TPC の T_0 分解能

CDC、TPC 共に Kalman Filter によるフィッティングプログラムが充分正しく動作していることが分かったので、ここではこれらを用いて CDC と TPC のタイムスタンピング性能の比較を行う。まず、CDC と TPC について T_0 を 7ns ずつ人工的にずらした 100GeV ミューオンイベントのフィッティングを行った。図 5.11、5.12 のように、入力した T_0 を正しく再現しタイムスタンピングしているのがわかる。CDC(All Axial) と TPC の T_0 分解能は、(5.2)、(5.5) 式で解析的に得られた T_0 分解能と誤差の範囲で一致している。CDC(Axial+Stereo) の T_0 分解能は Dip Angle を動かすことで不連続をなくすことができるため、解析値より悪化したものと考えられる。

5.3.2 CDC、TPC のタイムスタンピング能力

CDC と TPC のタイムスタンピング能力を高運動量 (P=100GeV) 及び低運動量 (P=1GeV) のミューオンについて比較した。

図 5.13 は CDC に、高運動量時（P=100GeV）低運動量時（P=1GeV）それぞれで、注目すべきイベント ($T_0 = 0$) が 4500 イベント、バンチが違うバックグラウンドが 3000 イベント混在して打ち込んで得たデータを解析して得た T_0 分解能である。図 5.14 は同様の条件で得た TPC の T_0 分解能である。

CDC では高運動量時（P=100GeV）の T_0 分解能が $\sigma = 2.49 \pm 0.04$ns、低運動量時（P=1GeV）の T_0 分解能が $\sigma = 2.16 \pm 0.03$ns なのに対し、TPC では高運動量時（P=100GeV）の T_0 分解能が $\sigma = 2.19 \pm 0.03$ns、低運動量時（P=1GeV）の T_0 分解能が $\sigma = 3.82 \pm 0.06$ns と低運動量時の T_0 分解能が大幅に悪化することがわかる。これは、CDC が TPC に比べ誤った T_0 に対してトラックが不連続となる境界を多く持っているためと考えられる。タイムスタンピング能力に関していうと、CDC の方が TPC に比べ優れた性能を持っていると結論できる。

5.4 フルシミュレータ (JUPITER&Satellites) による CDC のタイムスタンピング能力の評価

5.4.1 Satellites への組込み

JUPITER のデータを解析するために、Kalman Filter を Satellites へ組込む。5.3 節で述べた通り、各検出器をインストールするためには、図 5.15 のようにそれぞれの測定面に対応した TVMeasLayer クラスの派生クラスに各メンバ関数を実装し、KalDetector クラスにつめる作業を行う。将来的にはバーテックス検出器やカロリメータも実装すれば、検出器間を越えてトラックを連結し精度の高いフィッティングを行えるようになる。

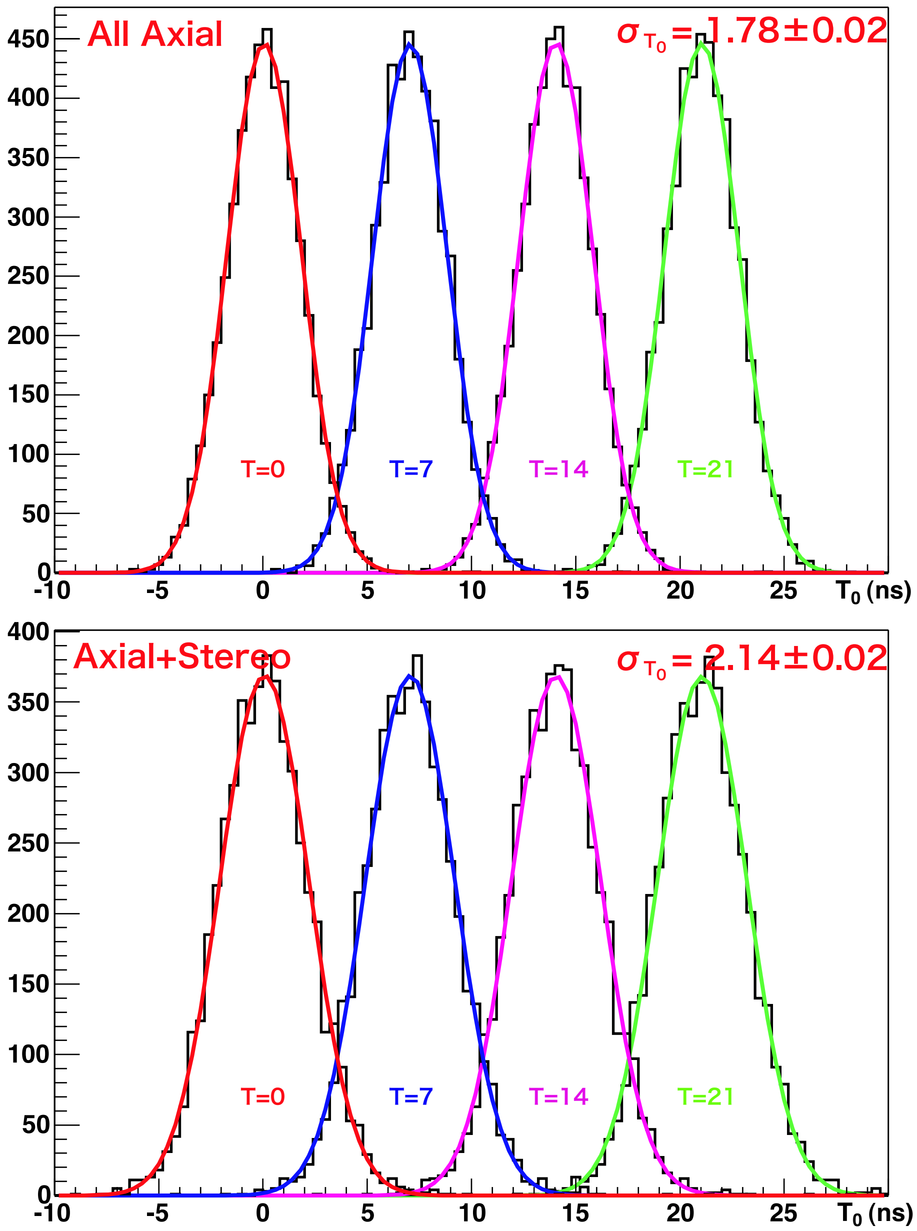


Figure 5.11: CDC の T_0 分解能。入力した T_0(0ns、7ns、14ns、21ns) によって分布がシフトしているのが、見てわかる。

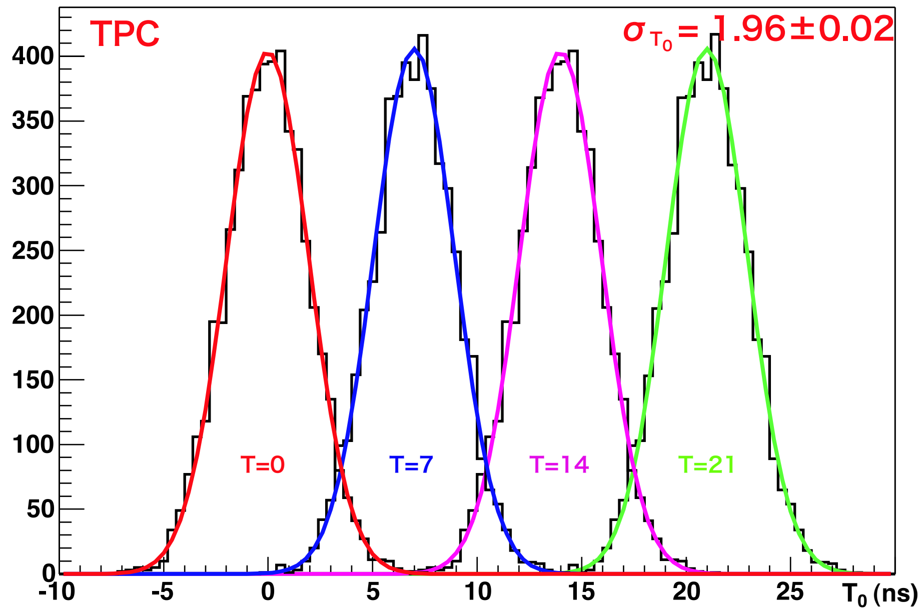


Figure 5.12: TPC の T_0 分解能。CDC と同様に入力した T_0 を正しく再現している。

5.4.2 シミュレーション結果

Kalman Filter を用いたトラックフィッティングを Satellites に実装し、ミューオン 0.5GeV を JUPITER に打ち込んで得たデータを用いて、低エネルギー時における CDC のタイム・スタンピング能力について調べた。1GeV 未満の粒子は CDC 内部を突き抜けることができず、CDC の内側へ戻ってしまう。これらカールしたトラックは Dip Angle(ビーム軸に垂直な面からの角度) によって、通過するレイヤーの数が変化する。そこで、カールしたトラックが通過したレイヤー数が少ないと、、タイムスタンピングを行うのに充分な不連続点が得られず、T_0 分解能が悪化すると期待される。図 5.16 の (a) は横軸に自由度、縦軸に T_0 分解能をとったものである。プロットから見て取れるように自由度 80 付近を境界に T_0 分解能が急激に悪化することが見て取れる。自由度 80 はスーパーレイヤー数にすると 4 層に対応し、これはアキシャルスーパーレイヤーを 2 層通ることに対応する。以前述べたようにステレオレイヤーでは Dip Angle を動かすことで不連続をなくすことができてしまうため、これ以下で急に T_0 分解能が悪化するのだと考えられる。図 5.16 の (b) は自由度 80 以上のトラックだけを用いた T_0 分解能である。その値は $\sigma = 1.74$ と、タイムスタンピングを行うのに十分な値である。このことから自由度 80 以上、つまりスーパーレイヤーを 4 層以上通ったトラックであればタイムスタンピングを行わうのに充分な T_0 分解能が得られると結論できる。

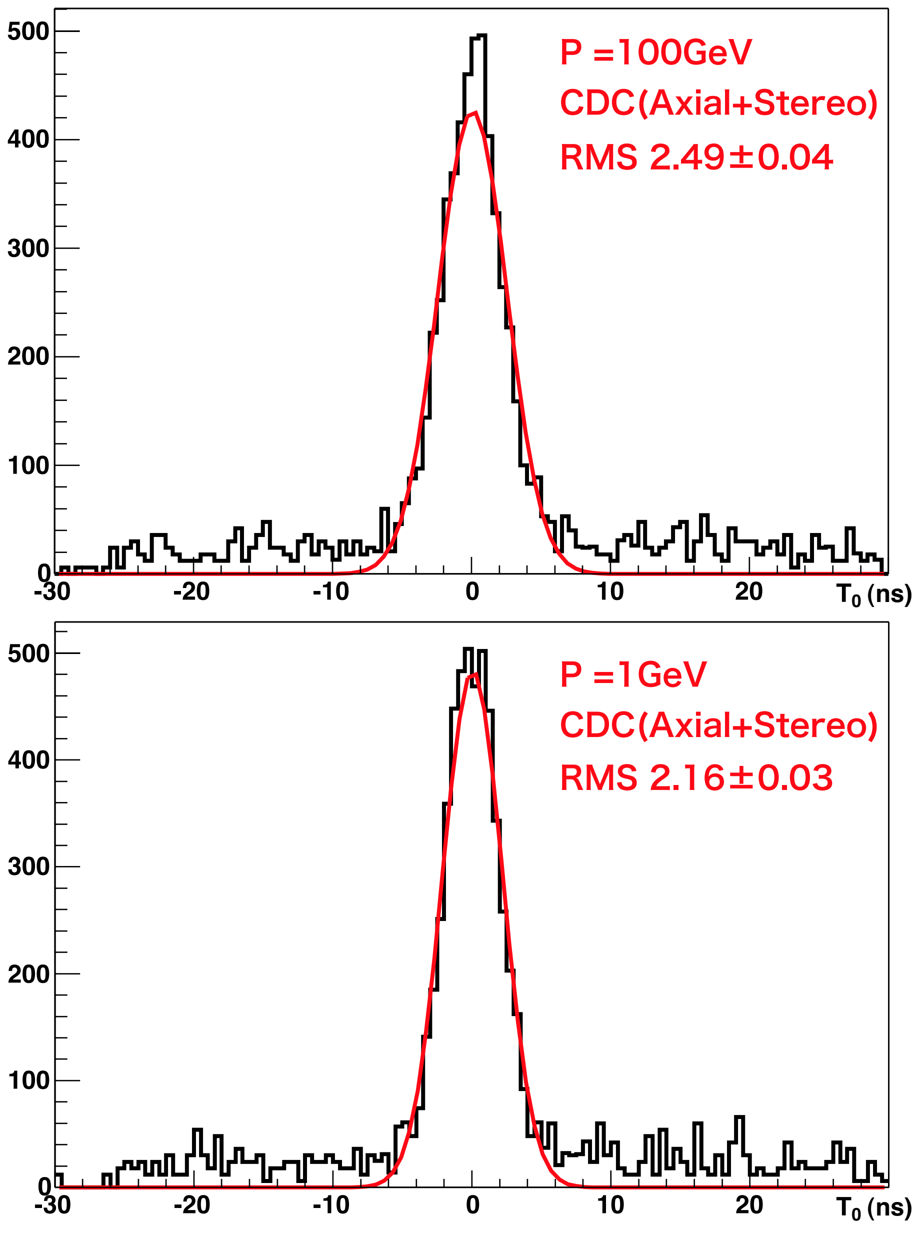


Figure 5.13: CDC の T_0 分解能

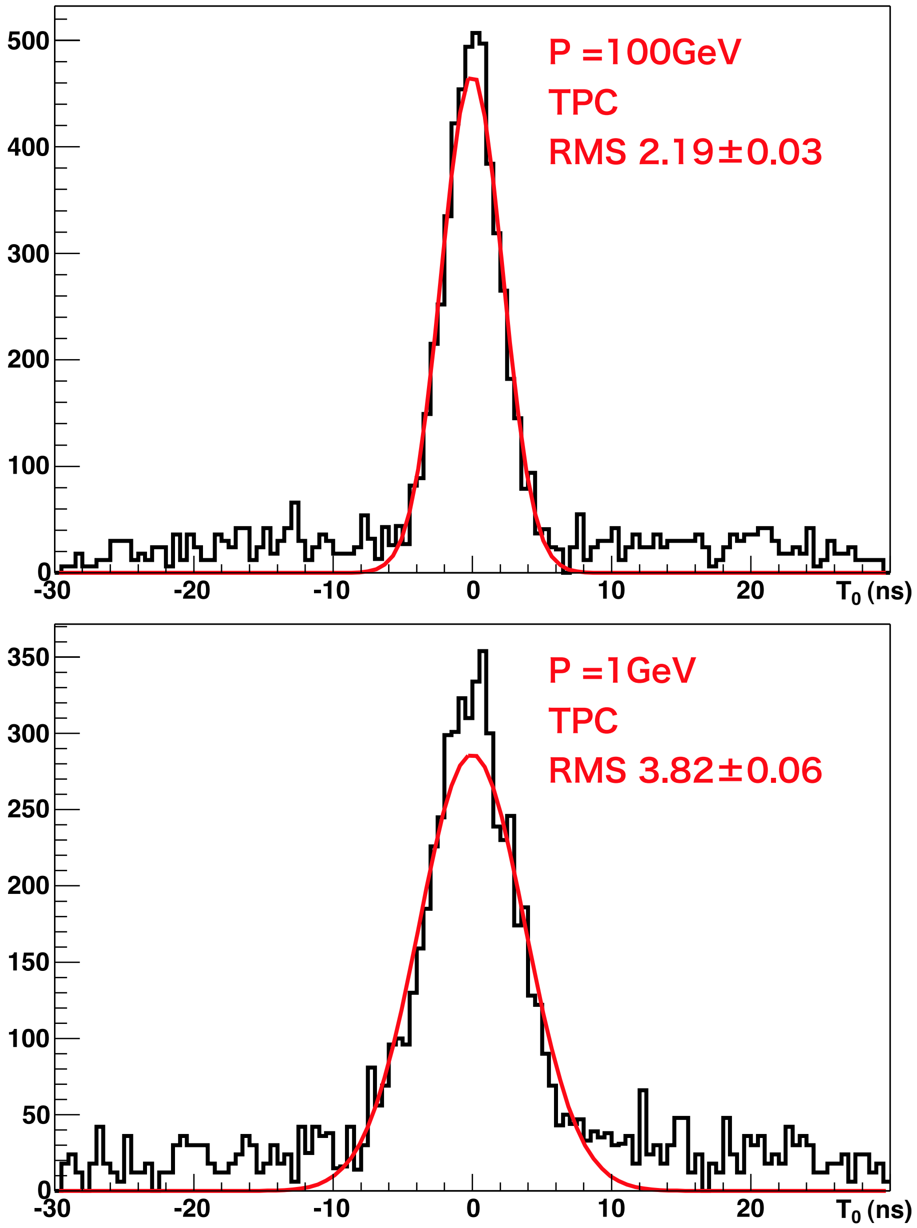


Figure 5.14: TPC の T_0 分解能

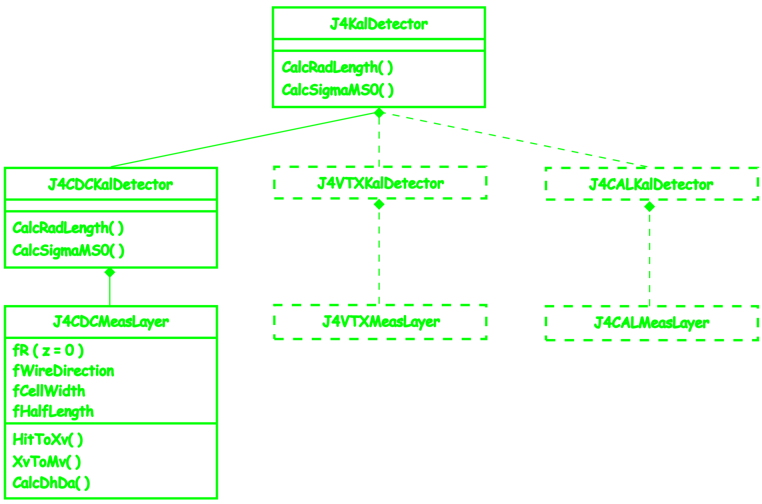


Figure 5.15: satellites のクラス構造

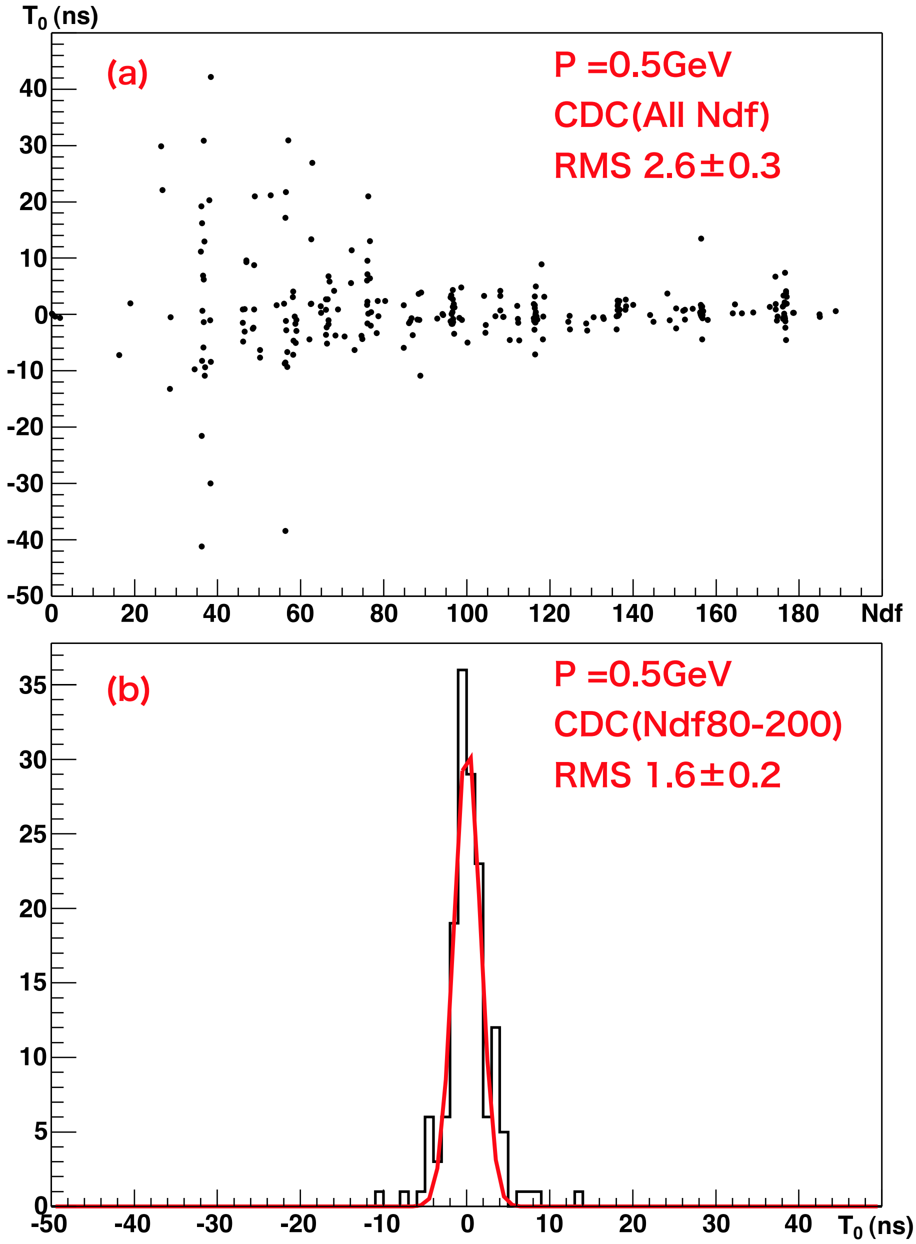


Figure 5.16: P=0.5GeV での CDC の T_0 分解能

Chapter 6

結論

6.1 本研究の成果

JLC-CDC (中央飛跡検出器) は、JLC 計画で期待される物理のあらゆる場面で中心的な役割を担うことになる測定器であり、その基本設計は、物理からの性能要求を適切に反映したものである必要がある。本研究は、JLC-CDC の基本設計を目標として結成された、JLC-CDC グループの開発研究の一環としてスタートした。基本設計は、物理からくる性能要求を測定器パラメータに翻訳する作業と、その測定器パラメータが実現可能である事を実証する作業に大別できる。前者は、主としてシミュレーションを主要な手段とするが、後者においても、実機のプロトタイプ建設前の段階では要素技術のハードウェア開発が主体となるため、全体システムの性能評価は、要素技術のハードウェア開発の結果を踏まえたシミュレーションが本質的である。一方、要素技術のハードウェア開発は、その結果がフィードバックされる結果として、測定器パラメータに影響を与えるため、両者は車の両輪の関係にある。

JLC-CDC グループでは、システム全体としての性能評価をし、基本設計を完成するためフル・シミュレータ (JUPITER+Satellites) の開発を行っている。JUPITER は Geant4 ベース、Satellites は ROOT ベースで、いずれもオブジェクト指向技術を最大限に活用し、C++ 言語を用いて開発中である。本研究ではテストチェンバーデータ解析プログラムを元にして、URANUS 及び Satellites、CDC 解析プログラムの実装を行った。

続いてオブジェクト指向技術を活用し、Kalman Filter の基本アルゴリズムを提供する汎用的なトラックフィッティングクラスライブラリを開発した。その動作をテストするために、トラックフィッティングクラスライブラリを CDC、TPC に対応させる派生クラスを作成した。これを用いてミューオンシングルトラックイベントを解析した。CDC、TPC 共に解析的に求めた χ^2 分布と誤差の範囲で一致した。confidence level 分布は TPC では一様だが、CDC では 0 付近に全体の 1% 未満ではあるが、ピークが見られた。これは、Kalman Filter の初期でトラックパラメータの誤差が大きいときに、間違った L/R を取ってしまうことが原因と考えられる。

CDC、TPC 共に Kalman Filter によるトラックフィッティングが充分正しく動作していることが分かったので、これらを用いてタイムスタンピング性能の比較を行った。CDC(All Axial) 及び TPC の場合は解析的に求めた T_0 分解能と誤差の範囲で一致した。CDC(Axial+Stereo) の場合には解析的な値より悪化し

た。これはステレオレイヤーの場合 z の場所によってセルのスタガー構造がなくなることによると考えられる。また、高運動量 (P=100GeV) 時と低運動量 (P=1GeV) 時を比較すると、CDC に比べ TPC では低運動量の T_0 分解能の悪化が大きかった。これは、CDC の方が TPC に比べ誤った T_0 に対してトラックが不連続となる境界をより多く持っているためと考えられる。タイムスタンピング能力に関していうと、CDC の方が TPC に比べ優れた性能を持っていると結論できる。

JUPITER のデータを解析するために、Kalman Filter を用いたトラックフィッティングを Satellites へ組込んだ。この組込みにより、低運動量で CDC を突き抜けず内側へカールするトラックのフィッティングが可能になった。カールするトラックは Dip Angle によって通過するレイヤー数が変化してしまう。このようなトラックについてはスーパーレイヤーを 4 層以上通ったトラックであればタイムスタンピングを行わうのに充分な T_0 分解能 ($\sigma_{T_0} \lesssim 2ns$) が得られるとわかった。

6.2 今後の方向

今回の Kalman Filter を用いたトラックフィッティングには幾つかの問題点が残っている。前述した通り、多重散乱について今は簡易的に測定面間の物質を一カ所に集め、そこで散乱が一回起こるだけと仮定している。これを、連続媒質で行わなければならない。また、今回の解析はミューオンのシングルトラックで行ったものである。本来の目的は、ジェット中のトラックに対してタイムスタンピングを行うことなので、Kalman Filter をジェット中のトラックに対して用いて性能評価する必要がある。

JUPITER および Satellites の枠組みは、すでに CDC 部分に応用され、また、JUPITER の部分について言えば、バーテックス検出器の部分の実装もほぼ終了している。また、もう一つの重要な検出器要素であるカロリメータの実装も始まった。さらには、これら測定器全体が、加速器起源のバックグラウンドの中でどのように動作するかを評価するため、加速器の最終収束系をもシミュレーションに組み込む計画が進んでいる。本研究は、イベント再構成部分において重要な根幹をなし、当初の目的である JLC-CDC の基本設計の完成にとどまらず、JLC 測定器全体の完成に向けて大きな役割を果たすであろう。

謝辞

本研究は大学院修士課程の 2 年間、JLC のための中央飛跡検出器（JLC-CDC）の開発研究の一環として行ったものです。これまでの間、JLC 物理グループでお世話になりました皆様に、心より感謝致します。

特に、筑波大学浅野研究室の浅野侑三教官、Norik Khalatyan 教官には、数々のご指導を頂きました。高エネルギー加速器研究機構素粒子原子核研究所（KEK）においては、藤井恵介氏、小林誠氏、宮本彰也氏から、研究計画から論文の執筆に至るまで、研究生活全般にわたって様々な局面で貴重なご意見を賜りました。

そして、中野雄生氏をはじめとする浅野研究室の皆様や、保科琴代氏、黒岩洋敏氏をはじめとする東京農工大仁藤研究室の皆様には、研究生活、学生生活共に大変お世話になりました。皆様方の励ましとお力添えがなければ、この研究は完成を見ませんでした。

ここに、再度皆様に深くお礼申し上げます。

Bibliography

[1] R.L. Gluckstern, Nucl. Instr. and Meth. **A24** (1963) 381.
JLC physics group, http://www-jlc.kek.jp/subg/offl/lib/docs/helix_manip.ps.gz.

[2] S. Sudou *et al.*, Nucl. Instrum. Meth. A **383**(1996) 391 .

[3] ACFA Linear Collider Working Group, KEK Report 2001-11, August (2001), 374,
http://www-jlc.kek.jp/subg/offl/jim/index-e.html

[4] http://wwwinfo.cern.ch/asd/geant4/geant4.html

[5] ACFA Linear Collider Working Group, KEK Report 2001-11, August (2001), 360,
http://www-jlc.kek.jp/subg/offl/jsf/index.html

[6] http://root.cern.ch/

[7] http://www-jlc.kek.jp/subg/offl/www-jlcsim/index.html

[8] http://wwwinfo.cern.ch/asd/geant4/G4UsersDocuments/Overview/html/index.html

[9] http://www.thep.lu.se/ torbjorn/Pythia.html

[10] R.Kuboshima, JLC-CDC シミュレータへのチェンバー性能テスト結果の組込み

[11] http://www-jlc.kek.jp/subg/offl/lib/docs/helix_manip/main.html

[12] R. E. Kalman, J. Basic Eng. 82 (1961) 34.

[13] R. Frühwirth, Nucl. Instr. and Meth. **A262** (1987) 444.

[14] E. J. Wolin and L. L. Ho, Nucl. Instr. and Meth. **A219** (1993) 493.

[15] P. Astier *et al.*, Nucl. Instr. and Meth. **A450** (2000) 138.