Master thesis
Fast Magnetic Field Query Algorithm for the ALICE O^2 Project

Shuto Yamasaki

M164026
Hiroshima University, Graduate School of Science,
Department of Physical Science, Quark Physics Laboratory

Supervisor: Prof. Toru Sugitate
Chief Examiner: Prof. Toru Sugitate
Vice-Examiner: Associate Prof. Tsunefumi Mizuno

February 28, 2018

Abstract

ALICE experiment studies strongly interacting matters in the LHC energy by analyzing unique physics in heavy ion collisions. The large amount of data (1TB/s) that will be collected to measure the rare physics events in ALICE Upgrade project makes record every collision nearly impossible. To address this issue, O^2 (Online-Offline) project was planned to build a new event collecting and processing infrastructure.

Though it is expected to reduce the amount of data to be transferred and recorded by selectively recording only the physical events excluding the background by reconstruction of particle tracks. The conventional data collection program has a issue of speed to calculate the magnetic field applied to the particles inside the magnet.

In this thesis, we proposed a faster implementation for the space solenoid electromagnet applied, and for the space dipole magnet applied, respectively in ALICE detector. As a result, the solenoid region achieved 8.2 times speedup and the dipole region achieved 4.0 times speedup.

Conventional implementation has used corrected measured fields as its sources. It has divided the detector space in order to balance speed and precision. It has used approximation algorithm which can approximate any function in arbitrary precision.

In the proposed implementation, the solenoid region is approximating by at most 3rd order polynomial and by using simplified space division scheme to reduced search time.

In the dipole region, the approximation polynomial forms are restored from the array of coefficients given to the algorithm of the conventional implementation, and it is modified and pre-compiled to minimize the computation cost, thereby speeding up without decreasing much precision. Segment search in dipole region was done conventionally with binary search. We replaced it with time complexity $O(1)$ algorithm at the cost of memory to gain speed.

We compared error and speed between the conventional and the proposed implementations. Comparison was conducted to confirm practicality. Proposed implementation will be used in both data collection and simulation for the ALICE experiment in the future.

Contents

1 Introduction **5**
- 1.1 Quark Gluon Plasma (QGP) . . . 6
- 1.2 Physics program with the upgraded ALICE detector . . . 8
- 1.3 Technical background . . . 11
- 1.4 Implementation difficulties . . . 15
- 1.5 Purpose . . . 16

2 Magnetic field calculation inside ALICE detector **16**
- 2.1 Source measurement . . . 16
- 2.2 Overview . . . 17

3 Methods **18**
- 3.1 Solenoid Region . . . 18
  - 3.1.1 Approximation with linear regression model . . . 18
  - 3.1.2 Space division toward improved precision . . . 19
- 3.2 Dipole Region . . . 21
  - 3.2.1 Loop and Branch elimination . . . 21
  - 3.2.2 Extra multiplication reduction and Fused Multiply Add . 22
  - 3.2.3 Constant time complexity $O(1)$ segment search . . . 23
  - 3.2.4 Quick segment search: Preparation . . . 27

4 Results **29**
- 4.1 Result of Solenoid Region . . . 29
- 4.2 Result of Dipole Region . . . 30

5 Discussions **30**
- 5.1 Trade-off between polynomials calculation speed and precision 30
- 5.2 Comparison of regression or approximation methods other than multiple linear regression . . . 32
- 5.3 Memory usage . . . 32

6 Conclusion **34**

References 35

A Definition of the Chebyshev polynomials T_n 36

B Indices of precision evaluation 37

C Programs 37

List of Figures

1.1 Schematic generation mechanism of the QGP. Quarks are confined by strong force in room tempreature. Quarks are deconfined in high tempreature and high density environment and undergo phase transition to QGP phase (fig.1.2 top.) 7
1.2 QCD phase diagram. 8
1.3 The ALICE detector after upgrade (Run3) 9
1.4 Data flow and processing pipeline of the O^2 system. ITS/TPC track findings are performed online (synchronously) at the Event Processing Nodes (EPNs). Track finding is also performed in the Muon CHember (MCH) and Muon Forward Tracker (MFT) detector. [12] 12
1.5 Possible hardware implementation of the logical data flow. The EPNs compress 500GB/s input data into 90GB/s.[12] . . 13
1.6 Particle tracks, projected on a plane perpendicular to beam axis (Z), of pp collisions measured with TPC detector. Red lines are physics tracks and black lines are background tracks. Nearly half tracks are unnecessary because these are come from gas molecules in the detector and electrons in the beam pipe. In the field of L3 solenoid magnet, some particles with about $50MeV/c$ of momentum are moving circularly. 14
2.1 Schematic flow of the existing B-field calculation. Given a position (x, y, z) in the detector volume, searches corresponding subvolume (segment) first. Then calculate B-field with the Chebyshev polynomials using the coefficients from the segment struct. The last used segment is cached for speed because it is likely near points are queried continuously in track finding. 17
3.1 Partitioning of the solenoid volume into chambers. (i) One of the chambers. (ii) The volume is divided by the detector boundary radius and quadrants. (iii) The volume is divided by the sign of the Z axis. 20
3.2 B_x of conventional implementation near $Z = 0$ $(-25 \leq Z \leq 30)$ on the X-Y plane, plots of 5cm step. The field is flipped near $Z = 0$. 21

3.3 B_y of conventional implementation near $Z = 0$ ($-25 \leq Z \leq 30$) on the X-Y plane, plots of 5cm step. The field is flipped near $Z = 0$. Additionally, ϕ -dependency is remarkable at $Z = -5, 0, 5$. 22
3.4 Code snippet of a generated Bz function. (shortened to 6 digits for visibility) 23
3.5 Comparison of assemblies w/ and w/o Fused Multiply Add (FMA) intrinsics. Left: w/o FMA, right: w/ FMA. Instructions which starts with 'fma' computes $a = ax + b$ in 5 cycles of latency. On the other hand, mulss and addss takes 5, 3 cycles of latency respectively, the former is 1.6 times faster in simple calculation. Used compiler is Clang-900.0.39.2 on Intel Core i7 Haswell Refresh. 24
3.6 Split dipole region $-1760 \leq Z \leq -558.3$ into 89 segments by maximum and minimum Z values of all the chambers. Call a segment with minimum Z as $zid = 0$, and maximum Z as $zid = 88$. A crossing line is placed to show possibility of linear relationship between Z positions and zids. The gap between the line and the segments implies Interpolation search works less efficiently. 26
3.7 Schematic figure of Quick Segment Search algorithm. The points are just an example and nothing to do with real data. Prepare evenly spaced slices (orange arithmetic sequence) which contain only one or zero Z value of a chamber border (blue point) in advance. The id of one of evenly spaced slices is immediately calculated from given z (e.g. 6). Every slice knows a z position of the blue point and a zid of the segment which is smaller than the blue point. The zid of larger side is always (the zid of smaller side + 1). Then by comparing z and z of the blue point, a zid for queried z is obtained (e.g. 3). Thus, zid is used to lookup the other value (data of X,Y axis or an index of a polynomial). 27
3.8 Parametrized part of dipole magnet applied region. Colored cubes are the chambers of dipole field parametrization from AliMagF class. This parametrization contains 1482 chambers. 28
4.1 Positional dependency of mean precision per magnetic field component in solenoid region. 30
5.1 Trade-off between polynomials calculation speed and precision 31

List of Tables

1 Comparison of the physics reach, minimum accessible p_T and relative statistical uncertainty, for selected observables between the approved (current) scenario and the proposed upgrade scenario. [3] 10
2 Estimated signal per event (S/ev), signal-to-background ratio (S/B) and number of background candidates per event (B'/ev) for central Pb-Pb collisions at $\sqrt{s_{\mathrm{NN}}} = 5.5$ TeV[12]. These values are derived in the Conceptual Design Report and in the Technical Design Report for the Inner Tracking System upgrade documents[5], where the geometrical conditions are also described. 12
3 Data reduction plan for the Pb-Pb collision 50kHz[12]. 13
4 Consumed CPU time for single query of one random point. . . 29
5 Importance rank obtained with Recursive Feature Elimination technique. 31
6 Fitting results of regression methods available in scikit-learn (Python) library sorted on coefficient of determination. '-' indicates NODATA. Percentages in the table is ratio against target precision $10^{-3} \times B_z = 5e-03$; less than 100% means the fitting successfully reached target precision. 33
7 Memory usage comparison between the conventional and the proposed implementation. Both 2 kilo Gauss and 5 kilo Gauss cases are shown. The values are for reference. See text for detail. 34

List of Algorithms

1 Prepopulated Quick Segment Search (search part). See text for details. 27
2 Prepopulated Quick Segment Search (construction part). See text for details. 28
3 Mathematica reimplementation of the AliMagF class. It is used to extract polynomials from existing parametrizations. 38
4 Mathematica C-source generator of the B-field parametrization in HornerForm. 39

1 Introduction

ALICE experiments are studying matters that interact with strong force by searching for physics peculiar to heavy ion collisions at the energy of LHC accelerator. In the prediction of lattice Quantum ChromoDynamics (lQCD),

quarks are released from confinement due to strong interaction in a high temperature and high energy density environment as in the early universe, and a new state of matter Quark Gluon Plasma (QGP) is generated. Studies on the specific viscosity and transport coefficient of QGP reproduced by energy heavy ion collision and the mechanism of formation of quark of the QGP derived from QGP are progressing. If heavy quark interacts on medium, its mass and flavor are hardly changed, so QGP in the ALICE upgrade plan, by raising the frequency of heavy ion collision to 50 kHz by 2021, it is considered that $D^0 \to K^-\pi^+$, $\Lambda_c^+ \to pK^-\pi^+$, and other rare physical events. In the collection of scarce physical events scheduled to be measured in ALICE experiments, it is difficult to record everything because data of 1 TB/s or more is generated. Therefore, the O^2(Online-Offline) plan was drafted for the purpose of building a new event collection and processing infrastructure. It is expected to reduce the amount of data to be transferred and recorded by selectively recording only the physical events excluding the background by reconstruction of particle tracks, but with the conventional data collection program, To 5% was spent in the calculation of the magnetic field applied to the particles inside the magnet, which was regarded as a problem. If the calculation speed can not catch up with the occurrence of data, there is a possibility that it is necessary to reduce the collected data and respond. Therefore, speeding up reconfiguration has become important. The current implementation of the magnetic field calculation above has already been undertaken by a general-purpose optimization approach, and further speeding up method is not trivial. In this paper, we propose a method that can speed up while maintaining the precision of the current implementation to a certain extent (or perfectly), without relying on this special speed hardware, And compare the speed with the current implementation.

1.1 Quark Gluon Plasma (QGP)

A nucleus is a ball of protons and neutrons, generally referred to as nucleons, which is consisted of three quarks. These quarks are bound by gluons so that they cannot be taken out from a hadron at ordinary environment (fig. 1.1). This is called confinement. Quantum Chromodynamics (QCD) calculations suggests strongly interacting particles (hadrons) should cause phase transition at extreme environment that is high temperature (150-200MeV) and high density (> 1GeV/fm^3). In the new state of matter, quarks and gluons move freely without confinement. We call the hot and dense medium as Quark-Gluon Plasma (QGP). Some predictions says critical temperature for phase transition is around 200 MeV ($\sim 2 \times 10^{12} K$) and the QGP existed at $t = 10^6$ to 10^5 seconds from the Big Bang. Thus, study of the characteristics of the QGP lead to our knowledge of the early universe. Quark-Gluon Plasma is also important for understanding mass generation mechanism of hadrons. The mass of the quark constituting the nucleon is estimated to be $\sim 20 \times 10^{-30}$ kg, and there is a big difference between triple the quark

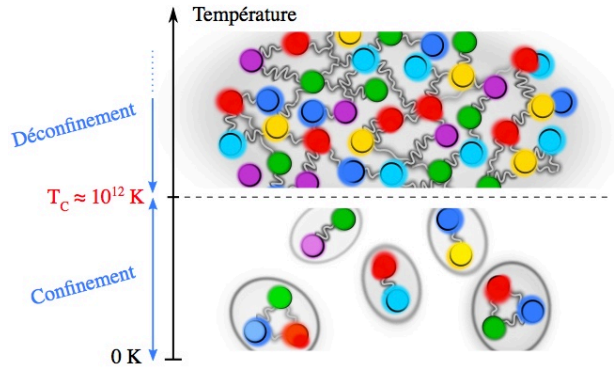

Figure 1.1: Schematic generation mechanism of the QGP. Quarks are confined by strong force in room tempreature. Quarks are deconfined in high tempreature and high density environment and undergo phase transition to QGP phase (fig.1.2 top.)

mass and proton mass 1700×10^{-30} kg. Most of our mass is thought to be generated by "spontaneous break of chiral symmetry" caused by the strong interaction of QCD, and QGP is a mass gain mechanism as a field where chiral symmetry is also restored It will bring important insights to.

High energy heavy ion collisions are used to generate Quark Gluon Plasma in laboratory and to investigate its properties. In this way, heavy ion like Pb are collided in high speed so that a hot and dense matter is created there. Such kind of QGP generation experiments has undergone since 1980s: an experiment using the BEVALAC accelerator of Lawrence Berkeley Laboratory in the United States, the AGS accelerator of Brookhaven National Laboratory (BNL) of the United States, and the SPS accelerator of the European Common Nuclear Research Organization (CERN). In the past, BNL-AGS gold beams with 10 GeV per nucleus, CERN-SPS with 200 GeV sulfur beams per nucleus and 160 GeV per nucleon beam fixed target experiments have been conducted. The world's first hadron collider BNL-RHIC accelerator has started since 2000, and the European CERN-LHC accelerator has started to operate since 2009. Verification of the QGP with 10 to 100 times the collision energy was done until now. The experimental goals are below.

- Discovery of the QCD phase diagram
- Elucidation of properties of hot and dense QCD multi-body system
- Precise verification of QCD in non-perturbative regions
- Elucidation of mass acquisition mechanism of quarks
- Understanding the confinement mechanism in hadron

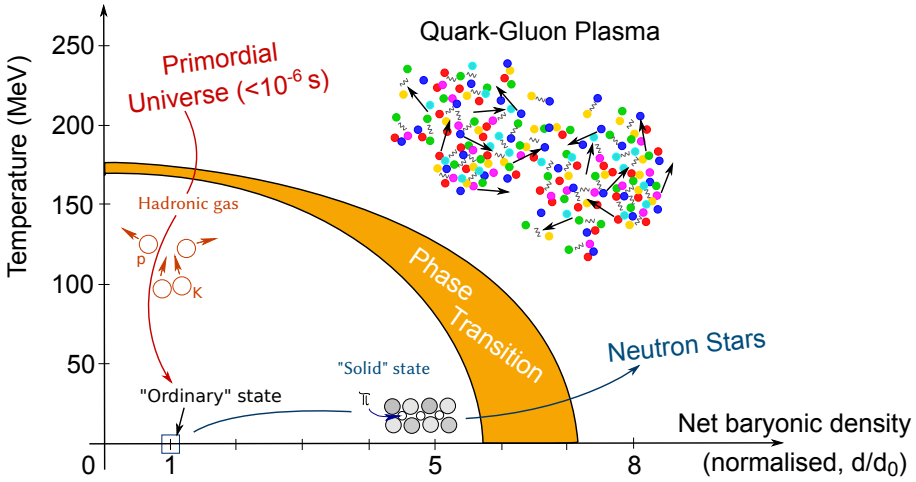

Figure 1.2: QCD phase diagram.

- Understanding the QCD phase transition happened in the early universe after the Big Bang

Several evidence of the QGP was found as a result of the previous experiments (Run 1 and Run 2). Representative results include discovery of high transverse momentum (high-p_{T}) hadron yield suppression (aka. jet quenching), azimuthal anisotropy and flow, thermal photon observation.

Currently it goes beyond the phase of discovery of QGP and researches on physical properties of QGP are in progress. In the future, the following topics are to be understood.

- Temperature dependency of QGP quantities (Specific viscosity, transportation coefficients, and so on)
- QGP response (behavior and propagation of energy lost by jet)
- Initial condition of collisions, early thermalization mechanism
- Hadron mass acquisition mechanism
- QCD phase structure

For above advanced study, more accurate measurements and new measurements that were impossible precisely are indispensable.

1.2 Physics program with the upgraded ALICE detector

LHC will be upgraded by 2022 and develop into high brightness LHC with increased collision frequency (HL - LHC). To further study the physical properties of QGP in this second generation LHC, we focus on rare probes of

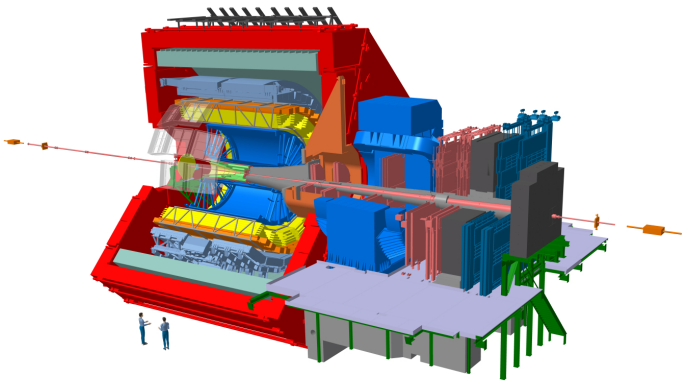

Figure 1.3: The ALICE detector after upgrade (Run3)

heavy ion physics in experiments after Long Shutdown 2 (LS 2) (Run 3, Run 4). The research subject is the coupling and the hadronization process of QGP and the medium, and it consists of heavy flavor particles, quartzone state (probe of temperature of QGP), real photon, virtual photon, jet including heavy quark and jet with high lateral momentum , Correlation between jet and other probes, measurement of lepton pair, etc. For example, in order to enable these high precision measurements, the LHC-ALICE experiment is planning to construct a large-scale new measuring instrument. Comparison of measurement precision before and after upgrade [3] is shown in the table 1.

The upgraded ALICE detector will target for an integrated luminosity of $10nb^{-1}$ at full magnetic field ($B = 0.5$ T) in the ALICE solenoid and $3nb^{-1}$ with reduced field ($B = 0.2$ T) for Pb-Pb collisions. The requirement for an integrated luminosity of $13nb^{-1}$ is motivated mainly by the following performance figures as described in the ALICE upgrade document[12].

- Heavy flavour measurements
  - The nuclear modification factor R_{AA} and elliptic flow v_2 of strange-charmed mesons (D_s) down to a transverse momentum p_T of at least 2 GeV/c with a statistical precision better than 10% for both observables. This will allow a precise comparison of strange and non-strange charm meson dynamics;
  - $\Lambda_c$baryon R_{AA} and v_2 down to 2 GeV/c and 3 GeV/c, respectively, with a precision of about 20% and baryon/meson ratio for charm (Λ_c/D) down to 2 GeV/c with the same precision. This will allow to address the charm quark hadronization mechanisms at low and intermediate momentum;
  - $R_{AA}$and v_2 of beauty-decay particles via non-prompt D_0, non-prompt J/ ψ and beauty-decay leptons (the two latter at both cen-

Observable	Approved		Upgrade	
	$p_{\mathrm{T}}^{\mathrm{Amin}}$ (GeV/c)	statistical uncertainty	$p_{\mathrm{T}}^{\mathrm{Umin}}$ (GeV/c)	statistical uncertainty
Heavy Flavour				
D meson R_{AA}	1	10 % at $p_{\mathrm{T}}^{\mathrm{Amin}}$	0	0.3 % at $p_{\mathrm{T}}^{\mathrm{Amin}}$
D meson from B decays R_{AA}	3	30 % at $p_{\mathrm{T}}^{\mathrm{Amin}}$	2	1 % at $p_{\mathrm{T}}^{\mathrm{Amin}}$
D meson elliptic flow	1	50 % at $p_{\mathrm{T}}^{\mathrm{Amin}}$	0	2.5 % at $p_{\mathrm{T}}^{\mathrm{Amin}}$
D meson from B elliptic flow	not accessible		2	20 % at $p_{\mathrm{T}}^{\mathrm{Umin}}$
Charm baryon-to-meson ratio	not accessible		2	15 % at $p_{\mathrm{T}}^{\mathrm{Umin}}$
$\mathrm{D_s}$ meson R_{AA}	4	15 % at $p_{\mathrm{T}}^{\mathrm{Amin}}$	1	1 % at $p_{\mathrm{T}}^{\mathrm{Amin}}$
Charmonia				
J/ψ R_{AA} (forward rapidity)	0	1 % at 1 GeV/c	0	0.3 % at 1 GeV/c
J/ψ R_{AA} (mid rapidity)	0	5 % at 1 GeV/c	0	0.5 % at 1 GeV/c
J/ψ elliptic flow	0	15 % at 2 GeV/c	0	5 % at 2 GeV/c
ψ(2S) yield	0	30 %	0	10 %
Dielectrons				
Temperature (intermediate mass)	not accessible			10 %
Elliptic flow	not accessible			10 %
Low-mass spectral function	not accessible		0.3	20 %
Heavy Nuclear States				
Hyper(anti)nuclei ${}^{4}_{\Lambda}\mathrm{H}$ yield	35 %			3.5 %
Hyper(anti)nuclei ${}^{4}_{\Lambda\Lambda}\mathrm{H}$ yield	not accessible			20 %

Table 1: Comparison of the physics reach, minimum accessible p_T and relative statistical uncertainty, for selected observables between the approved (current) scenario and the proposed upgrade scenario. [3]

tral and forward rapidity) down to 1–2 GeV/c with precisions from a few percent to 10%. This will allow a detailed assessment of the b quark transport properties in the medium;

  - B meson fully-reconstructed decays ($B^+ \rightarrow \overline{D^0}\pi^+$) down to 3 GeV/c with a precision of about 10%. This will provide an important direct measurement of beauty production;
  - $\Lambda_b$baryon production for $p_T > 7$GeV/c. This will be a unique measurement in heavy ion collisions and should allow for the determination of the nuclear modification factor of the beauty baryon, which is sensitive to the b quark hadronization mechanism.

- Charmonium measurements

  - $R_{AA}$of J/ψdown to $p_T = 0$ with statistical precision better than 1%, at both central and forward rapidity;
  - $R_{AA}$of $\psi(2S)$ down to $p_T = 0$ with precision of about 10%, at both central and forward rapidity;
  - $v_2$of J/ψ down to $p_T = 0$ with precision of about 0.05 (absolute uncertainty), at both central and forward rapidity;
    these measurements will allow a detailed investigation of the mechanisms of dissociation and regeneration for charmonium states in the deconfined medium.

- Low-mass dileptons

  - the additional sample of 3nb^{-1} with a reduced magnetic field value (0.2 T) in the central barrel is essential for low-mass dielectron analysis to obtain the projected precision of about 10% on the slope of the high-invariant-mass region and of about 10% on the dielectron elliptic flow. This measurement will make it possible to assess the time-evolution of the thermal radiation emitted by the hot medium.

Most of these analysis (with the exception of the exclusive reconstruction of beauty hadron decays) are characterized by a very small signal-to-background ratio. It implies signal candidate events in Pb-Pb collisions are indistinguishable from background events. This means that it is not possible to use some dedicated triggers to select collisions online for offline analysis. Instead, a minimum bias trigger must be used to “record all collisions of Pb-Pb”. A minimum bias trigger is the most basic trigger used to record all the events. Table 2 summarize such branches.

1.3 Technical background

To tackle the data recording challenge, the new computing infrastructures has planned: Online-Offline (O^2) Project. Fig. 1.5 shows O^2 hardware and

Analysis	S/ev [5]	S/B	B'/ev
$D^0 \to K^-\pi^+$	$7.6 \cdot 10^{-3}$	10^{-2}	3.0
$D_s^+ \to K^+K^-\pi^+$	$2.3 \cdot 10^{-3}$	$< 2 \cdot 10^{-3}$	> 4.6
$\Lambda_c^+ \to pK^-\pi^+$	$6.5 \cdot 10^{-4}$	$< 10^{-4}$	> 26
$\Lambda_c^+ \to pK^-\pi^+ (p_\mathrm{T} > 2\mathrm{GeV}/c)$	$3.7 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	7.4

Table 2: Estimated signal per event (S/ev), signal-to-background ratio (S/B) and number of background candidates per event (B'/ev) for central Pb-Pb collisions at $\sqrt{s_{\mathrm{NN}}} = 5.5$ TeV[12]. These values are derived in the Conceptual Design Report and in the Technical Design Report for the Inner Tracking System upgrade documents[5], where the geometrical conditions are also described.

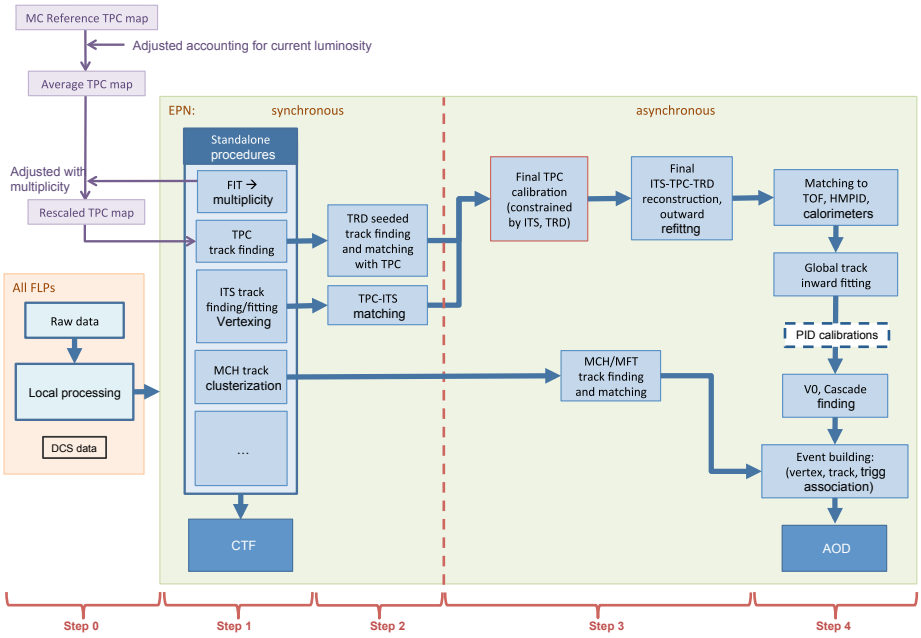

Figure 1.4: Data flow and processing pipeline of the O^2 system. ITS/TPC track findings are performed online (synchronously) at the Event Processing Nodes (EPNs). Track finding is also performed in the Muon CHember (MCH) and Muon Forward Tracker (MFT) detector. [12]

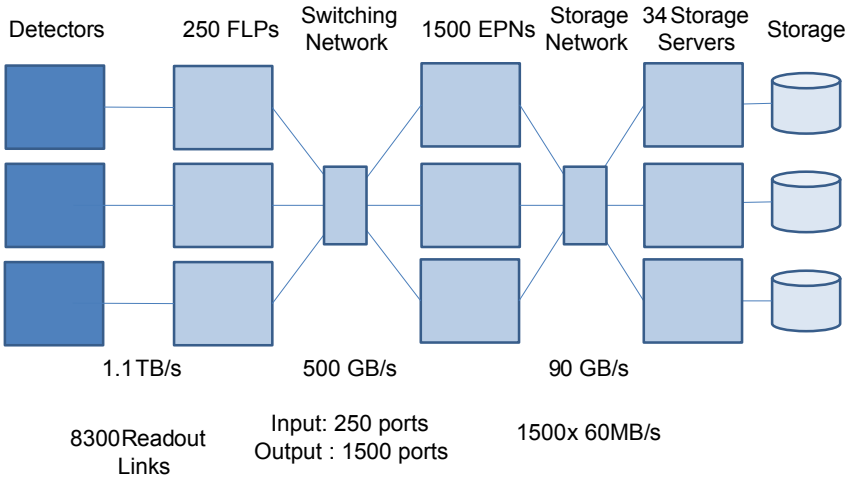

Figure 1.5: Possible hardware implementation of the logical data flow. The EPNs compress 500GB/s input data into 90GB/s.[12]

Detector	Raw rate (GB/s)	Compressed (GB/s)	Reduction (1/x)
TPC	1,000	50	20
ITS	40	26	1.5
TRD	20	3	6.6
TOF	2.5	2	1.25
MCH	2.2	0.7	2.9
MFT	10	5	2
EMCal	4	1	4
PHOS	2	0.5	4
(Total)	1080.7	88.2	12.3

Table 3: Data reduction plan for the Pb-Pb collision 50kHz[12].

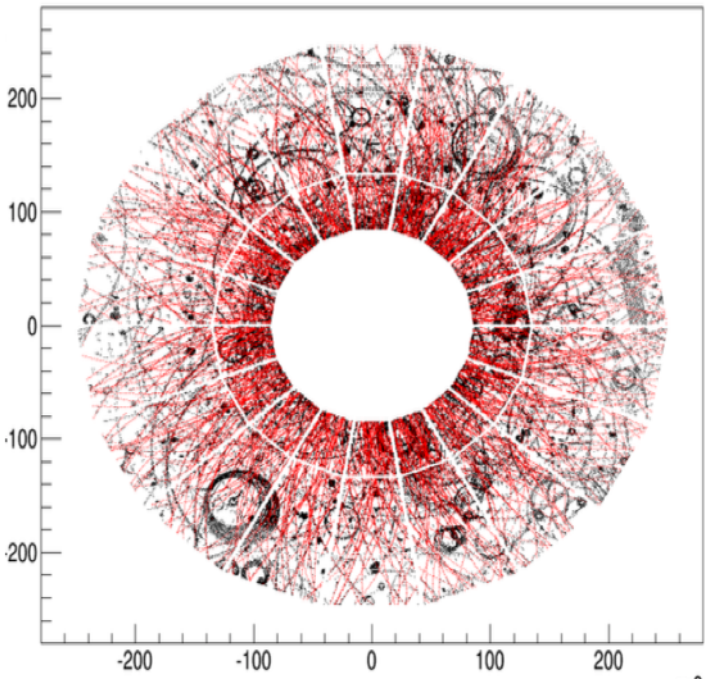

Figure 1.6: Particle tracks, projected on a plane perpendicular to beam axis (Z), of pp collisions measured with TPC detector. Red lines are physics tracks and black lines are background tracks. Nearly half tracks are unnecessary because these are come from gas molecules in the detector and electrons in the beam pipe. In the field of L3 solenoid magnet, some particles with about $50MeV/c$ of momentum are moving circularly.

network communication between them. The data flowing from the detector to the first stage processor (First Level Processor) group reaches 1.1 terabytes per second, which is difficult to record as it is. Therefore, in O^2, background information irrelevant to physics is immediately discarded, and other data is compressed and recorded. The value read from the detector which is the original data is basically discarded. As a result, data amount reduction as shown in the table 3 is scheduled. The original data is organized and compressed from 1.1 TB/s to 90 GB/s. An element which is very important in this data compression process is calculation of particle trajectory. Figure 1.6 lists the particle trajectory plot of the TPC detector located in the solenoid region. Half of that is the background, giving the opportunity to reduce the data volume by half. As a function of the detector software used to obtain the momentum from the curvature of the charged particle for the calculation of the trajectory there is an inquiry of the static magnetic field vector applied inside the detector. Although this function is necessary, it took a little less than 5% to acquire the magnetic field out of the CPU time involved in the reconstruction of the track, which was regarded as a problem. In this paper, the calculation of this magnetic field has been speeded up.

1.4 Implementation difficulties

There are some restrictions when implementing acceleration of magnetic field calculation.

First, since this calculation is performed very frequently, it is required to complete at a speed of 0.1 μs order. Therefore, it is not possible to use inference by a neural network which depends on many tensor products.

Next, the precision of the calculated magnetic field must maintain high precsion of $\pm 10^{-3} B_z$ ($B_z = 2, 5$ kG). Its domain covers a wide range of 11 + 12 m in length and 6 - 10 m in diameter.

This magnetic field is a non-uniform and nonlinear magnetic field based on measured values, and it is difficult to handle with a single equation.

Attempts to obtain precision by directly applying several statistical machine learning methods (this is an analytical approach different from so-called AI) failed.

Therefore, one can think of an approach to lower the amount of calculation by changing the algorithm of existing implementation code. However, as described later, existing implementations have already been considerably optimized, including consideration for caching, so the improvement is not trivial. For example, the binary search method used in existing implementations will be the fastest category as a list search method for real-valued divisions where the distribution is biased.

If coping with software is difficult, one will consider using dedicated hardware such as GPU, FPGA, ASIC. In O^2, we plan to install FPGA in FLP and GPU in EPN. Since track finding is performed by EPN from fig.1.4, there

is a possibility that GPU can be used only when using O^2. However, magnetic field calculation is also required for Monte Carlo simulation outside the ALICE computer room and CERN computer center. Therefore, the existence of special hardware can not be assumed, and it must operate on a commercially available ordinal x86_64 machines.

As with the lattice QCD simulation, it may be possible to arrange a large amount of lattice points that have previously calculated the magnetic field in the detector volume. However, the amount of available memory is also limited. The magnetic field calculation is performed in a distributed computing environment and runs in many instances. To cope with other physical calculations, only a few MB of memory can be used for magnetic field calculation. According to rough estimation, at least 70 GB per instance will be required to obtain sufficient precision with the above method.

1.5 Purpose

The purpose of this study is to solve the above problem and to facilitate the measurement of new physical channels by supporting data collection in the coming Run 3. And to document explicitly one of the high-speed implementation which is faster than binary search which is considered the fastest in common sense (this is special case, though).

2 Magnetic field calculation inside ALICE detector

2.1 Source measurement

Analysis of the magnetic field applied by ALICE's L3 solenoid electromagnet was made by Ruben Shahoyan[10] and the analysis of the magnetic field applied by the dipole electromagnet was done by Ruben and Morsch[11]. In the conventional implementation, internally the magnetic field is a set of parameterization (3 dimensional to 3 dimensional mapping) based on Chebyshev polynomial in the cylindrical coordinate system (see appendix for its definition), defined for each divided chember of the magnetic field.

The degree of the polynomial is automatically adjusted so as to guarantee the required precision (the maximum value of the difference from the measured field). The precision varies depending on the location, but it is about $2 \times 10^{-4} \sim 5 \times 10^{-3}$ (solenoid region, in the case of applying 5 k gauss).

Where the data was missing, supplementation by a spline curve is performed. There are two sets of data: 12 kA (low, 2 k Gauss) and 30 kA (high, 5 k Gauss). These dataset are the most accurate data set available.

When referring to conventional (existing) implementation in this paper, it refers to AliMagF class[2] implemented by Ruben and these data sets.

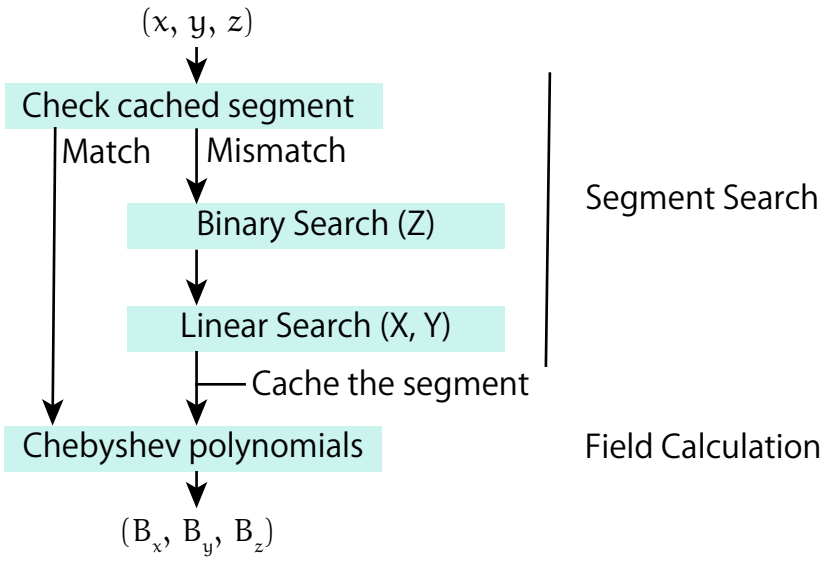

Figure 2.1: Schematic flow of the existing B-field calculation. Given a position (x, y, z) in the detector volume, searches corresponding subvolume (segment) first. Then calculate B-field with the Chebyshev polynomials using the coefficients from the segment struct. The last used segment is cached for speed because it is likely near points are queried continuously in track finding.

2.2 Overview

When a magnetic field inquiry of a certain point inside the detector occurs, the flow until the magnetic field is calculated by the existing implementation is shown in the figure 2.1. The given point (x, y, z) is checked to see if it is contained in the first cached chamber. If it matches, skip search and go to calculation of Chebyshev polynomial. If it does not match, it finds the segment to which z belongs by binary searching the array sorted by the maximum and minimum values (boundary values) of the Z axis of the chamber. Subsequently, an array obtained by sorted the boundary values of the X axis associated with that division is searched by linear search. Likewise, the Y axis is linearly searched to find chambers.

Binary search is considered to be the fastest as a search method for arbitrary length sorted real number list (Z axis boundary values) with unequal distribution. The found chamber is cached for the next time. Thereafter, a Chebyshev polynomial is calculated. Chebyshev polynomials are orthogonal systems, and any function can be expressed when the number of terms is infinite. Unlike the Fourier transform it is very fast because division is not

used in it.

Thus, existing implementations are already optimized for speed.

3 Methods

Bottleneck of conventional method are chamber search and calculation of complete system (Chebyshev polynomial) describing the magnetic field of the chamber. Both took the same degree of time. In the conventional implementation, the solenoid region and the dipole region were calculated in the same way, but in the proposed implementation, the optimization was performed in different ways due to the difference in the gradient of the magnetic field.

3.1 Solenoid Region

To decrease the field query overhead, High Level Trigger (HLT) has used less precise and faster parametrization from before. In this method, only B_z which is the main component of the solenoid region is computed as:

$$bz = c[0] + c[1] * z \ + c[2] * r \ + c[3] * z * z + c[4] * z * r + c[5] * r * r;$$

where c is an array of coefficients, z is distance in the beam line direction in the ALICE local coordinate system with the Interaction Point as origin, and r is the distance from the beam line. As seen above, φ-dependency of B_z and transverse components (B_x, B_y) are ignored. This is not a issue because current HLT reconstruction does not require much precision, but the result cannot be used in final fitting. Therefore, we extended the parametrization of HLT, consider the φ-dependency and transverse components and created an implementation that can be used for the final fitting. The target precision is set to $10^{-3} \times B_z$ for the value returned by the conventional implementation.

3.1.1 Approximation with linear regression model

In the solenoid region, we regard the conventional implementation as a black box and create expressions that reproduce its behavior (aka. supervised learning). Approximate conventional implementation with low order real coefficient polynomial by regarding it as pure function $\boldsymbol{B}(x, y, z)$. The new implementation of the solenoid region is practically a fitting to the conventional implementation. The existing implementation adopted the cylindrical coordinate system (r, ϕ, z) for the solenoid region, so this fitting makes it possible to omit the coordinate system transformation which includes time consuming ϕ calculation. The problem of finding a function that reproduces data is called a regression problem. The linear regression model is the most

basic model. First, the objective variable B_x is represented by a linear sum of coordinates (explanatory variables).

$$B_x = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + ... + w_m x_m = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}$$

Where $w_i \in \mathbb{R}$ is weight, $\boldsymbol{w}$ is a vector $(w_0, w_1, ..., w_m)$ and $\boldsymbol{x}$ is a vector $(x_1, x_2, ..., x_m)$. In normal linear regression, $\boldsymbol{x} = (x, y, z)$, but this time the explanatory variable contains not only the given coordinates x, y, z but also the product of coordinates $(x^2, xy, z^2 x$, etc). This method is called polynomial regression. The number of explanatory variables must be chosen such that sufficient precision is obtained and there is no speed degradation. As a result of fitting and measuring the precision, it was found that at most a third order real coefficient polynomial is sufficient for the gradient of the solenoid region. Therefore, the expression of B_x is given as follows.

$$\begin{aligned} B_x\,(x,\,y,\,z) = {} & a_0 + a_1 x + a_2 y + a_3 z + a_4 xx + a_5 xy + a_6 xz + a_7 yy + a_8 yz + a_9 zz \\ & + a_{10} xxx + a_{11} xxy + a_{12} xxz + a_{13} xyy + a_{14} xyz + a_{15} xzz + a_{16} yyy + a_{17} yyz + a_{18} yzz + a_{19} zzz \end{aligned}$$

Where $\boldsymbol{w} = (a_0, a_1, ..., a_{19})$. To reduce multiplications and additions, transformed as follows. However, do not change $\boldsymbol{x}$.

$$\begin{aligned} B_x\,(x,\,y,\,z) = {} & a_0 + x(a_1 + x(a_4 + x a_{10} + y a_{11} + z a_{12}) + y(a_5 + z a_{14})) \\ & + y(a_2 + y(a_7 + x a_{13} + y a_{16} + z a_{17}) + z(a_8)) \\ & + z(a_3 + z(a_9 + x a_{15} + y a_{18} + z a_{19}) + x(a_6)) \end{aligned}$$

Next, randomly collect sample points from existing implementations N points. Let the pair of explanatory variable vector and B_x corresponding to the i-th point be $(\boldsymbol{x}_i, B_{x,i})$. At this time, care must be taken not to bias the distribution of the sample points. Simply taking sample points in a grid shape has scale dependence. Therefore, it is better to sample them at random. Also, when random numbers are obtained according to the geometry (cylindrical shape) of the detector, it is necessary to check whether points are evenly distributed over the detector volume.

Finally, find the weight $\boldsymbol{w}$ using the method of least squares. For calculation, we used sklearn.linear_model.LinearRegression() of scikit-learn[8]. The approximation precision of this approach is described in the results section.

3.1.2 Space division toward improved precision

When tried to approximated the entire solenoid region by one equation, the precision is not sufficient. So in order to improve it, whole space was divided into chambers as shown in fig. 3.1. The considerations on the division are as follows.

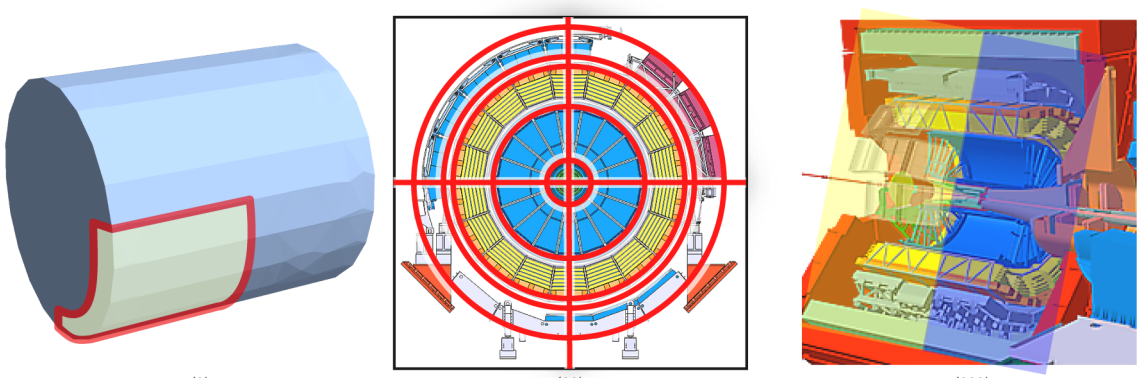

Figure 3.1: Partitioning of the solenoid volume into chambers. (i) One of the chambers. (ii) The volume is divided by the detector boundary radius and quadrants. (iii) The volume is divided by the sign of the Z axis.

- For R axis
  - Divided along the detectors' material boundary taking advantage of the fact that the detectors in the barrel are centered around the point at $R = 0$. The gradient is greatly different inside and outside of detector boundaries.
  - More specifically, the volume was divided into five regions: ITS detector $(R \leq 80\,\text{cm})$, TPC detector $(80 < R \leq 250)$, TOF detector $(250 < R \leq 400)$, a gap which is just outside of the TOF detector $(400 < R \leq 423)$, the volume where there are calorimeters such as PHOS, EMCal, DCal detector $(423 < R \leq 500)$.
  - $r = \sqrt{x^2 + y^2}$ is used for branching depending on R, but by squaring the above constants beforehand, extra calculation of square root can be reduced by comparing it with $x^2 + y^2$.
- For ϕ axis
  - In some cases ϕ dependency is large like fig. 3.3, and approximation precision improves by dividing in ϕ axis direction.
  - Simplified conditional branching by splitting just at 90 degrees.
- For Z axis
  - B_x, B_y has symmetry around $Z = 0$ as in fig. 3.2,3.3, therefore, as with the ϕ axis, splitting with $Z = 0$ improves precision.

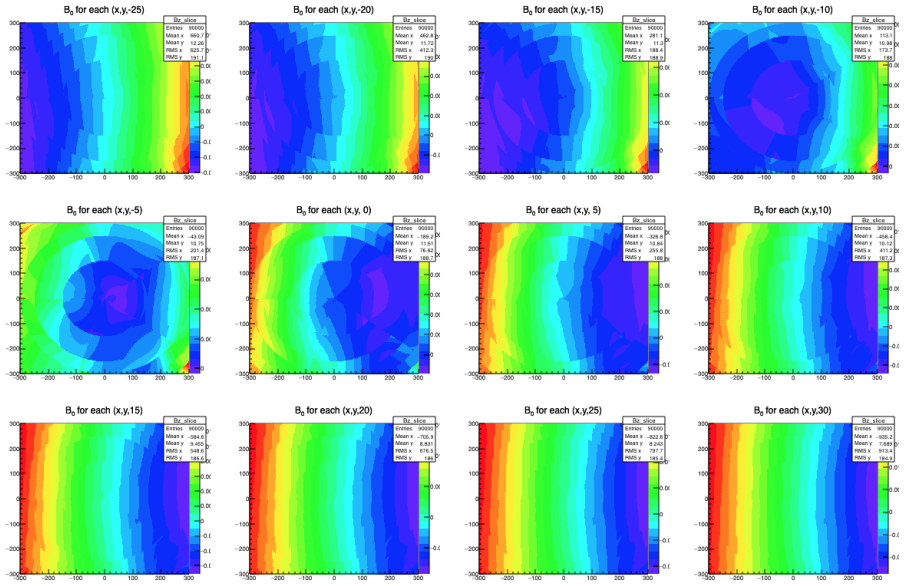

Figure 3.2: B_x of conventional implementation near $Z = 0$ $(-25 \leq Z \leq 30)$ on the X-Y plane, plots of 5cm step. The field is flipped near $Z = 0$.

- $-550 \leq z\,(\mathrm{cm}) \leq 550$ to divide 22 at $50\,\mathrm{cm}$ intervals to ensure sufficient precision in the wide region.
- Since it is divided at regular intervals, an index of the array can be specified with one division, and a penalty (pipeline stall in CPU) due to failure of conditional branch does not occur.

The entire solenoid region is divided into $5 \times 4 \times 22 = 440$ chambers. Since a simple division method is used, it is possible to identify the chamber which contains the point to be queried quickly. 100,000 points were sampled and fitted for each chamber.

3.2 Dipole Region

Unlike the solenoid region, the gradient of the dipole region is remarkably large, and if fitted in the same way, the number of divisions becomes very large. Therefore, we did not dare fit in the dipole region, but speed up by extracting polynomial from existing parametrization and modifying it.

3.2.1 Loop and Branch elimination

The Chebyshev polynomial calculation was implemented with a code having a triple loop and a branch within the loop. Since branch prediction is mainly

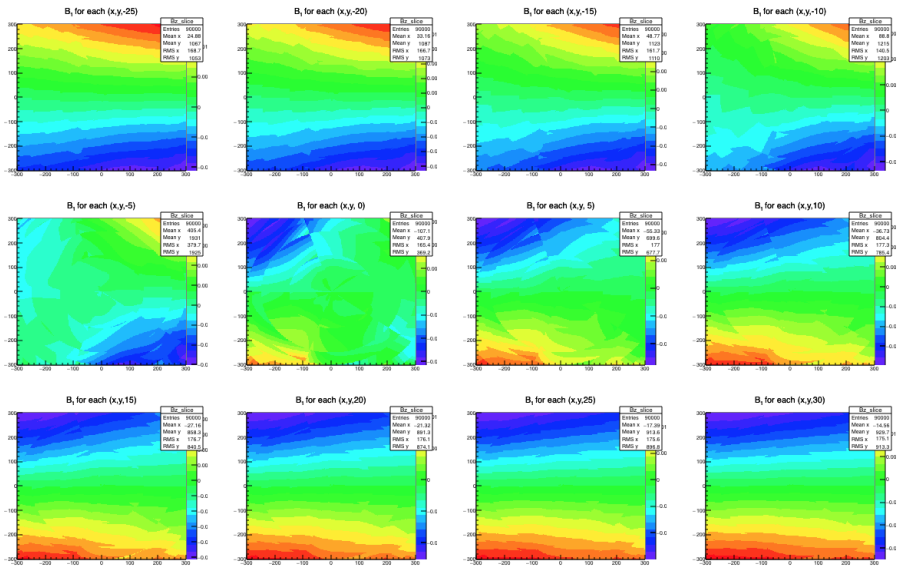

Figure 3.3: B_y of conventional implementation near $Z = 0$ $(-25 \leq Z \leq 30)$ on the X-Y plane, plots of 5cm step. The field is flipped near $Z = 0$. Additionally, ϕ -dependency is remarkable at $Z = -5,\ 0,\ 5$.

performed based on the previous results that passed through the branch, if the for loop satisfies the termination condition, the branch prediction fails and the penalty is received. Therefore, it is considered that speed can be increased by removing the loop. Other branches should be removed as much as possible.

Parametrizations of the magnetic field using the conventional implementation is stored in the ROOT file format. From this file one can write parametrization in plain text format. In order to make it easier to handle, we created software that parses this text and converts it into portable JSON text format. Furthermore, using Mathematica, parametrizations using the Chebyshev polynomials in the dipole region are extracted from the converted JSON file, and obtained the polynomials consisting of just x, y, z with no loop or branch by simplification for each chamber and components B_x, B_y, B_z.

3.2.2 Extra multiplication reduction and Fused Multiply Add

Then converted the obtained polynomials into C code and made it compilable beforehand (Fig.3.4). For the magnetic field query, two function is requried. One is to calculate all of $(B_x,\ B_y,\ B_z)$ and another calculate only B_z. The calculation of $(B_x,\ B_y,\ B_z)$ and B_z was separately defined so as not to include extra jump instructions in the generated code. Figure 3.4 shows one $B_z\,(x, y, z)$ function of a chamber describing a magnetic field of 5k gauss.

```
float dip5k1bz(const float p[3]) {
        const float x = p[0], y = p[1], z = p[2];
        return 5484.90f+-0.00734f*y+x*(-0.01580f+0.00008f*y+-0.00004f*z+
                x*(0.00008f+1.67927e-7f*x+-1.06067e-6f*y+3.12794e-7f*z))+
                z*(40.0776f+z*(0.10987f+z*(0.00013f+6.10621e-8f*z)));
};
```

Figure 3.4: Code snippet of a generated Bz function. (shortened to 6 digits for visibility)

This function takes the Cartesian coordinates of the point to query the magnetic field as an argument and returns B_z at that point.

A polynomial in the Figure3.4 was made from the expanded form by the Horner's method. Horner's method is like following.

$$1 + 2x + 3x^2 + 4x^3 \to 1 + x(2 + x(3 + 4x))$$

As shown above, the variable x is extracted so as to reduce the number of multiplications. Horner's method has been proved to have the smallest number of multiplications when used for polynomials[7]. Also, the number of additions is also minimum[6]. In addition, recent CPUs have dedicated instructions for calculating $ax+b$ in Horner format. In the Intel Haswell architecture and after, this Fused Multiply Add (FMA) instructions was extended to handle floating point numbers only from integers. O^2 computers have not been purchased yet, so there is no problem that old CPUs mix in online processing. For non-compliant CPUs used in places other than O^2, they are compiled to use normal AVX instructions or similar instructions (using the -march compiler option). Figure 3.5 shows the difference in assembly depending on the presence or absence of FMA. Instructions starting with fma could calculate $a = ax + b$ with 5 cycles of latency. For mulss and addss has 5 and3 cycles of latency respectively, it takes 8 cycles in total, 1.6 times faster with fma.

Since the proposed implementation of the dipole region requires compiling in advance, it can be said that it is an implementation specialized for calculation speed, discarding versatility.

3.2.3 Constant time complexity $O(1)$ segment search

The computational complexity of the binary search of figure 2.1 is $O(\log n)$ (n is the number of Z segments to be searched + 1 = 90), but about 7 array accesses ($90 < 2^7$) and three cache miss opportunities (if the number is single precision floating point number and the cache line is 64 bytes). Penalties due to cache miss have been improved year by year, but still there is a big difference in the time taken when L1 cache hit and main memory is

```
_dip5k1bz:
  pushq %rbp
  movq %rsp, %rbp
  movss (%rdi), %xmm0
  movss 4(%rdi), %xmm3
  movss 8(%rdi), %xmm1
  movss 3297658(%rip), %xmm2
  mulss %xmm3, %xmm2
  addss 3297650(%rip), %xmm2
  movss 3297646(%rip), %xmm4
  mulss %xmm3, %xmm4
  addss 3297638(%rip), %xmm4
  movss 3297634(%rip), %xmm5
  mulss %xmm1, %xmm5
  addss %xmm4, %xmm5
  movss 3297622(%rip), %xmm4
  mulss %xmm0, %xmm4
  addss 3297614(%rip), %xmm4
  mulss 3297610(%rip), %xmm3
  addss %xmm4, %xmm3
  movss 3297602(%rip), %xmm4
  mulss %xmm1, %xmm4
  addss %xmm3, %xmm4
  mulss %xmm0, %xmm4
  addss %xmm5, %xmm4
  mulss %xmm0, %xmm4
  addss %xmm2, %xmm4
  movss 3297574(%rip), %xmm0
  mulss %xmm1, %xmm0
  addss 3297566(%rip), %xmm0
  mulss %xmm1, %xmm0
  addss 3297558(%rip), %xmm0
  mulss %xmm1, %xmm0
  addss 3297550(%rip), %xmm0
  mulss %xmm1, %xmm0
  addss %xmm4, %xmm0
  popq %rbp
  retq
```

```
_dip5k1bz:
  pushq %rbp
  movq %rsp, %rbp
  vmovss (%rdi), %xmm0
  vmovss 4(%rdi), %xmm1
  vmovss 8(%rdi), %xmm2
  vmovss 2538974(%rip), %xmm3
  vfmadd213ss
2538969(%rip), %xmm1, %xmm3
  vmovss 2538965(%rip), %xmm4
  vfmadd213ss
2538960(%rip), %xmm1, %xmm4
  vfmadd231ss
2538955(%rip), %xmm2, %xmm4
  vmovss 2538951(%rip), %xmm5
  vfmadd213ss
2538946(%rip), %xmm0, %xmm5
  vfmadd231ss
2538941(%rip), %xmm1, %xmm5
  vfmadd231ss
2538936(%rip), %xmm2, %xmm5
  vfmadd213ss
%xmm4, %xmm0, %xmm5
  vfmadd213ss
%xmm3, %xmm0, %xmm5
  vmovss 2538922(%rip), %xmm0
  vfmadd213ss
2538917(%rip), %xmm2, %xmm0
  vfmadd213ss
2538912(%rip), %xmm2, %xmm0
  vfmadd213ss
2538907(%rip), %xmm2, %xmm0
  vfmadd213ss
%xmm5, %xmm2, %xmm0
  popq %rbp
  retq
  nopw %cs:(%rax,%rax)
```

Figure 3.5: Comparison of assemblies w/ and w/o Fused Multiply Add (FMA) intrinsics. Left: w/o FMA, right: w/ FMA. Instructions which starts with 'fma' computes $a = ax + b$ in 5 cycles of latency. On the other hand, mulss and addss takes 5, 3 cycles of latency respectively, the former is 1.6 times faster in simple calculation. Used compiler is Clang-900.0.39.2 on Intel Core i7 Haswell Refresh.

referenced, improving cache misses may affect performance[4]. It is considered that the Intel Haswell architecture is about 40 times faster in this case. Therefore, it is necessary not only to reduce the computation complexity, but also to make an access such that main memory access is reduced or cache prefetch succeeds. For linear search, there are few data to be searched and if it is in the cache, the order of the computation time may be better than the binary search despite $O(n)$.

Interpolation search[9] is a search algorithm that may be faster than binary search if the data to be searched does not fit in the cache line. In Interpolation search, a linear search is performed by determining the indexs to start searching by linear expression, expecting that the values are distributed linearly in the sorted real number array. This is similar to how to find a specific person from the phone book. Interpolation search completes in $O(\log \log n)$ steps on average on uniformly distributed arrays on average[9]. However, if the data is skewed, it is substantially the same as the linear search and it becomes the worst calculation complexity $O(n)$. Fig.3.6 is a plot of the variation in values in order to investigate whether Interpolation search can be used for searching Z section. Dipole region $-1760 \leq Z \leq -558.3$ is divided into 89 segments without duplication using the Z coordinate maximum value minimum value of the chambers. Let's call $zid = 0$ for the segment with the smallest z and $zid = 88$ for the segment with the biggest z. The straight line in the figure tries to associate the Z position with zid by a linear equation, and the gap between this straight line and the segment indicates the possibility that the interpolation search does not operate efficiently.

The proposed implementation is a new algorithm that corrects the latter part of this interpolation search so that the search always ends at one branch. Since there is no loop, both the average computational complexity and the worst computational complexity are $O(1)$. Such ideas have been known at least within ALICE for a long time but have not been clarified. We call this method "Prepopulated Quick Segment Search". The fast segment search consists of a slow build part that creates a table from the data to be searched and a fast search part that uses a table. A schematic diagram of the search part is shown in figures 3.7 and pseudocode in the algorithm1. In the figure, since it is a one-dimensional search method, the Z axis is taken as an example. The black dot z is the z coordinate of the point where the magnetic field is desired to be acquired. Black line segments represent divisions. The distance of the minimum value z_{min} and the maximum value z_{max} $width = z_{max} - z_{min}$, the number of divisions $number\,of\,divisions$, The difference between the smallest element of the sequence and the origin is characterized by $offset = z_{min}$. Indices starting from 0 are given to slices created by this arithmetic progression. The number of divisions is determined beforehand so that the maximum value of only one Z axis value (boundary value of segments) of the end of the chamber is included. You can follow the procedure later. The first step in the search is to identify this index from z (the first line of pseudocode). In the example shown in

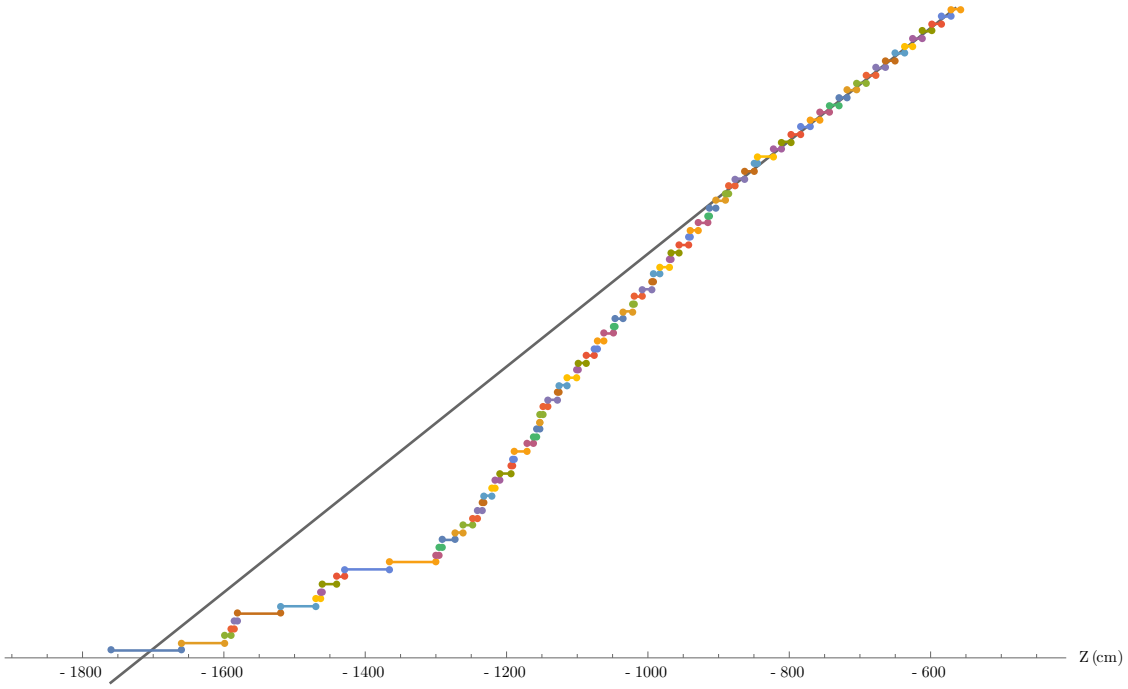

Figure 3.6: Split dipole region $-1760 \leq Z \leq -558.3$ into 89 segments by maximum and minimum Z values of all the chambers. Call a segment with minimum Z as $zid = 0$, and maximum Z as $zid = 88$. A crossing line is placed to show possibility of linear relationship between Z positions and zids. The gap between the line and the segments implies Interpolation search works less efficiently.

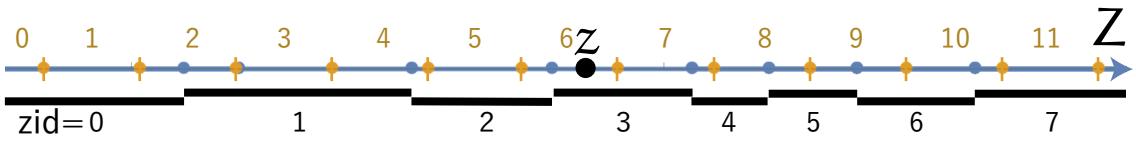

Figure 3.7: Schematic figure of Quick Segment Search algorithm. The points are just an example and nothing to do with real data. Prepare evenly spaced slices (orange arithmetic sequence) which contain only one or zero Z value of a chamber border (blue point) in advance. The id of one of evenly spaced slices is immediately calculated from given z (e.g. 6). Every slice knows a z position of the blue point and a zid of the segment which is smaller than the blue point. The zid of larger side is always (the zid of smaller side + 1). Then by comparing z and z of the blue point, a zid for queried z is obtained (e.g. 3). Thus, zid is used to lookup the other value (data of X,Y axis or an index of a polynomial).

Algorithm 1 Prepopulated Quick Segment Search (search part). See text for details.

$index \leftarrow \lfloor (z - offset) * (number\,of\,divisions/width) \rfloor$.
if $index > number\,of\,divisions$ then $search\,failed$.
$slice \leftarrow slices[index]$.
$id \leftarrow former\,zid\,of\,slice +$(if $z < segment\,end\,position\,of\,slice$ then 0 else 1) .
return id.

the figure, $index = 6$. $0 \leq index < number\,of\,divisions$, continue, else the search is unsuccessful because it is out of range. By the work in the construction part, an array $slices$ is prepared in which slice boundary values $segment\,end\,position$ and the set of zid ($former\,zid$) of the division smaller than the boundary value are stored for all the slices. If there is no boundary value in the slice, $segment\,end\,position = +\infty$ is stored. On the other hand, since the divisions are sorted, the zid of the division larger than the boundary value is always $zid + 1$. On the fourth line of the pseudo code, we use this property to obtain the zid of the segment containing z. In the example shown in the figure, $zid = 3$. You can obtain a search table for other axes by subtracting zid obtained in this way with another sequence. As with the Z axis, search in the X axis and Y axis direction can be performed, and finally the number of the chambers can be obtained, and an approximation function pointer is found.

3.2.4 Quick segment search: Preparation

Describe the construction part of the high-speed segment search algorithm. Pseudocode is shown in the algorithm2.

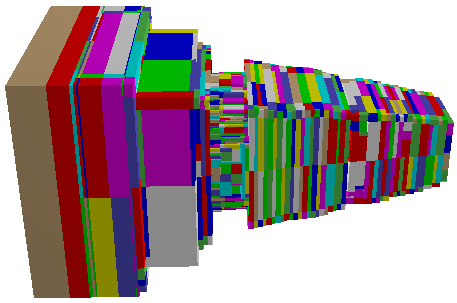

Figure 3.8: Parametrized part of dipole magnet applied region. Colored cubes are the chambers of dipole field parametrization from AliMagF class. This parametrization contains 1482 chambers.

Algorithm 2 Prepopulated Quick Segment Search (construction part). See text for details.

```
n ← 1
while(true){
        maxSegments ← 0
        lastzid ← 0
        for(i ← 0; i < n; ++i){
                slice start ← i * (width/n) + offset
                slice end ← (i + 1) * (width/n) + offset
                nSegs ← 0
                slices[i] ← (lastzid, +inf)
                for(j ← 0; j < segends.length; ++j)
                        if(slice start < segends[j] < slice end)
                                nSegs ++
                                slices[i] ← (j, segends[j])
                                lastzid ← j
                        }
                }
                maxSegments ← max(maxSegments, nSegs)
        }
        if(maxSegments == 1)return(n, slices)
        n ++
}
```

μs/call	Conventional	Proposed	Conventional/Proposed
Solenoid region	0.43 (exact)	0.053 (approx.)	8.2
Dipole region	0.61 (exact)	0.15 (exact.)	4.0

Table 4: Consumed CPU time for single query of one random point.

The basic idea is to increase n from 1 in order until the maximum value of the number of segment boundaries contained in the slice becomes 1 when divided into n.

Since this preliminary preparation takes a relatively long time, it is suitable for inquiries about data whose data to be queried is rarely changed. In this way, the implementation of the dipole region discards versatility and specializes in the search speed.

4 Results

Improvement of speed by proposed implementation is shown in the table 4. A speedup of 8.2 times in the solenoid region and 4.0 times in the dipole region was observed. For the compiler, Clang-900.0.39.2 was used. The environment is Intel Core i7 Haswell Refresh.

4.1 Result of Solenoid Region

The magnetic field approximation precision of the solenoid region is shown in Fig.4.1. Focusing on the center column (Z dependency) of this figure, the error exceeds 0.005 kG of the target precision in the region of$|Z| > 240$ cm. In this region, the gradient is large and it is difficult to approximate with a three-dimensional polynomial. Due to this influence, some errors (φ, Z dependence) exceeding target precision are also seen. Fortunately, this Z range region is outside the acceptance region, so it has no direct effect on physics. There is a possibility of small influence on background particles, but it is not very important that there is high precision when considering them. Therefore, the proposed implementation seems to have sufficient precision for use in real time reconstruction and simulation. Therefore, we decided to allow errors on $|Z| > 250$ cm. One can fall back to existing implementations whenever more accurate results are needed, and you can also examine the effect of errors by comparing the results using existing implementations and those using suggested implementations.

By approximation based on simple linear regression and simple splitting, precision that is not problematic in practical use could be secured.

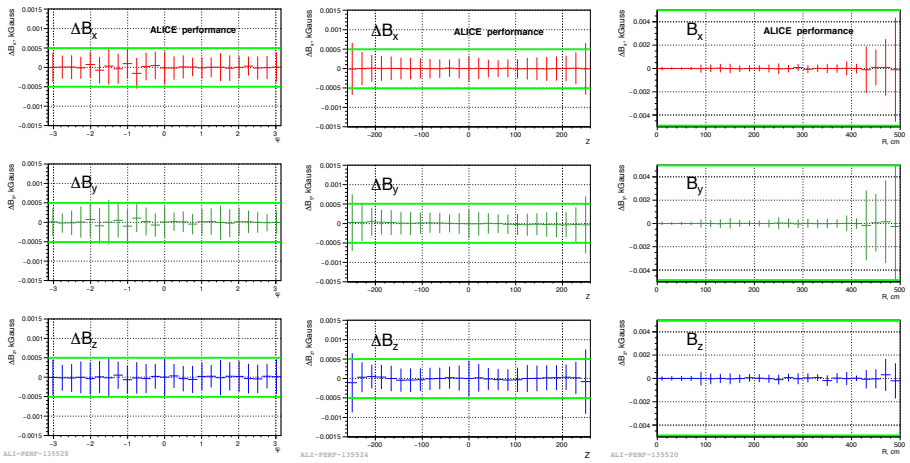

Figure 4.1: Positional dependency of mean precision per magnetic field component in solenoid region.

4.2 Result of Dipole Region

Since the calculation of the dipole region is implemented so as to be compatible with the conventional implementation including the behavior at chamber interface, there should be no error.

The proposed implementation using loop and multiplication elimination, precompilation and fast search algorithm was not as fast as the solenoid region using approximation, but since it was able to secure the same precision as in the conventional implementation, we used the conventional implementation in all use cases It can safely be replaced.

5 Discussions

5.1 Trade-off between polynomials calculation speed and precision

Consider the trade-off between precision and speed by reducing the number of polynomial terms. As the general tendency, the average approximation precision is higher (lower) as the number of terms of the approximate polynomial is larger (smaller), and the time taken for calculation is increased (decreased). Therefore, Recursive Feature Elimination was used to extract terms contributing to high precision. The ranking of importance of terms by RFE is shown up to the 4th order term tab 5. RFE is performed in the following procedure.

1	z	p	18	yyz	rrpp
2	y	pp	19	zzz	zrp
3	x	ppp	20	xyzz	zrpp
4	zz	pppp	21	xxyy	zzr
5	yy	z	22	xxxz	rrr
6	xx	r	23	xxxx	zzpp
7	xy	zp	24	yyyy	rrrp
8	xz	zpp	25	xzzz	zzz
9	yz	zppp	26	yzzz	zrrp
10	xyz	rp	27	yyyz	zzzp
11	xzz	rpp	28	xyyy	zrr
12	yzz	rppp	29	xyyz	zzrp
13	yyy	zz	30	xxyz	rrrr
14	xyy	zr	31	xxxy	zzzr
15	xxy	rr	32	yyzz	zzrr
16	xxx	rrp	33	zzzz	zrrr
17	xxz	zzp	34	xxzz	zzzz

Table 5: Importance rank obtained with Recursive Feature Elimination technique.

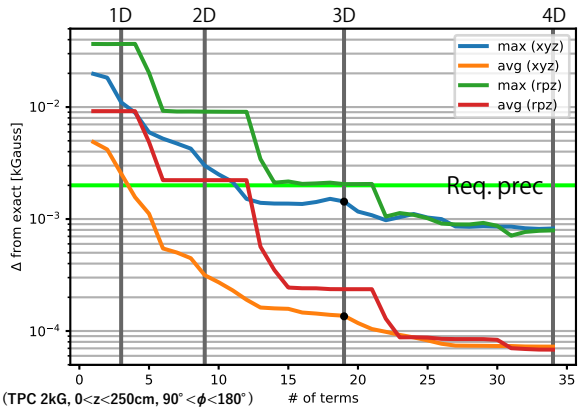

Figure 5.1: Trade-off between polynomials calculation speed and precision

1. Create a model starting from all features (terms) and delete features with the lowest importance (absolute value of coefficient)

2. Create the model again and repeat deletion of the feature quantity with a small importance until there is no item

In the table, r and p represent the radius and the azimuth angle ϕ in cylindrical coordinates.

The relationship between the number of terms used and the maximum deviation / average deviation sorted in this order is shown in 5.1.

The departure from the conventional implementation decreases in the Cartesian coordinate system due to the increase in order. With the precision of cylindrical coordinate display, it is inferior to Cartesian coordinate display up to item 23.

Considering the above-mentioned trade-off between precision and speed, we made technical decisions to adopt polynomials of at most 3 in the Cartesian coordinate system.

5.2 Comparison of regression or approximation methods other than multiple linear regression

What other approaches to efficiently satisfy the required precision? The table 6 is the result of using Python's scikit-learn package which contains many regression algorithms. The four regression algorithms reached the target precision with respect to the maximum deviation amount. The most systematic was the linear regression and the Bayesian ridge regression. Since the fitting speed was higher in the linear regression, the Bayesian ridge regression algorithm was not selected. We tried mini (DFP method), conjugate gradient method, Levenberg-Marquardt method, BFGS method, SVM, decision tree, original algorithm etc. as a method not limited to the scikit-learn regression algorithm, but sufficient precision was obtained I did not use it or because I did not finish the calculation.

5.3 Memory usage

You should not allocate more memory to calculate the magnetic field. The table 7 is a comparison of memory consumption of the conventional implementation and the proposed implementation. Root [0] new AliMagF () // Finish class loading Similarly, 2492 usage measurement was made using GetProcInfo () of ROOT, but it should be regarded as a reference value because the blur is large for each measurement. Conventional implementation loads two parameterizations besides solenoid and dipole, so it is reasonable to compare with half of the values in the table.

In addition, although the dipole region of the proposed implementation is implemented by dynamic loading of shared libraries, a reference count is made with global variables, and when all instances are destroyed, unloading

	coeff. of det. R^2	Max deviation	Mean deviation	RMS
LinearRegression()	0.99988	3.13e-03 (62.6%)	2.57e-04 (5.1%)	3.84e-04 (7.7%)
BayesianRidge()	0.99988	3.13e-03 (62.6%)	2.57e-04 (5.1%)	3.84e-04 (7.7%)
Lasso(alpha = 0.1)	0.99987	3.00e-03 (60.0%)	2.85e-04 (5.7%)	4.11e-04 (8.2%)
TheilSenRegressor()	0.99985	3.34e-03 (66.7%)	2.69e-04 (5.4%)	4.35e-04 (8.7%)
ElasticNetCV(cv=5)	0.97536	1.97e-02 (395%)	4.59e-03 (91.7%)	5.57e-03 (111%)
OrthogonalMatchingPursuit()	0.81910	3.61e-02 (721%)	1.30e-02 (260%)	1.51e-02 (301.8%)
LassoLars(alpha=.1)	0	1.07e-01 (2140%)	2.95e-02 (591%)	3.55e-02 (709.5%)
Lars()	-171.50	3.03e+00 (60630%)	3.56e-01 (7119%)	-
HuberRegressor()	-795.85	3.91e+00 (78169%)	7.12e-01 (14237%)	-
RidgeCV(alphas=[0.1,1,10])	-4603	8.10e+00 (1.6e5%)	1.99e+00 (39804%)	-
PassiveAggressiveRegressor()	-726315	1.20e+02 (2.4e6%)	2.02e+01 (4.4e5%)	-
SGDRegressor()	-5.7e48	3.34e+23 (6.7e27%)	6.40e+22 (1.2e27%)	-

Table 6: Fitting results of regression methods available in scikit-learn (Python) library sorted on coefficient of determination. '-' indicates NODATA. Percentages in the table is ratio against target precision $10^{-3} \times B_z = 5e-03$; less than 100% means the fitting successfully reached target precision.

param\impl (KB)	AliMagF (conv.)	AliMagFast (prop.)
5kGauss	38,724	4,724
2kGauss	33,104	6,208

Table 7: Memory usage comparison between the conventional and the proposed implementation. Both 2 kilo Gauss and 5 kilo Gauss cases are shown. The values are for reference. See text for detail.

is performed and the memory is freed It has become. Therefore, the memory consumption of the suggested implementation should be larger than the size of the shared library to be dynamically loaded, but the difference of the value returned by GetProcInfo () only increased or decreased by around 1000 KB. Therefore, the size of the shared library is listed as a reference value in the right column of the table 7. Considering the above, looking at the table, there is a possibility that the memory consumption of the proposed implementation is less than half that of the conventional implementation.

6 Conclusion

In the heavy ion collision physics, the discovery of the quark gluon plasma is now an important subject of study for its physical properties. For rare physical events scheduled to be measured by the ALICE experimental advancement plan to reach unexplored physics, the use of online triggers is not efficient for the low signal / background ratio. Therefore, it is necessary to collect all data with lead-lead collision. However, since data of 1 TB/s or more is generated from the detector, it is difficult to record everything. For data compression, it is required to calculate the momentum of the particles from the magnetic field inside the detectors, reconstruct their track, and eliminate the background tracks. In the conventional data acquisition program, ~ 5% of the reconstruction time was spent in calculating the magnetic field applied to the particles inside the magnet.

For each space applied by solenoid electromagnet and dipole electromagnet, we proposed a faster implementation than before. In the solenoid region, approximation by a polynomial of at most 3rd order is performed, and by simplifying the division method of the space, the time spent on the search can be greatly shortened. The solenoid region is divided into $5 \times 4 \times 22 = 440$, and the precision around $B_z \times 10^{-3}$ is ensured, especially in the center where precision is required.

In the dipole region, approximation polynomials that record magnetic fields recorded using conventional implementation are extracted, modified and precompiled so as to minimize the number of multiplications and additions, thereby speeding up without decreasing precision. The search of the divided space in the dipole region was conventionally binary search and linear search combined, but by replacing the algorithm with the time com-

plexity $O(1)$ at a price of memory, the speed was increased. In this algorithm, in order to associate a one-dimensional position with a segment, the interval between the maximum value and the minimum value is divided into slices. The number of divisions is chosen so that the number of the boundaries in each division is at most 1 and the number is minimum. Such divided slices hold the boundary position of the segment and the id of the section smaller than the boundary. As a result, the division is definitely found by dividing once, lookup of the array, and comparing the position to be inquired with the boundary value. A high-speed segment search is performed for each of the Z axis, the X axis, and the Y axis, so that a polynomial corresponding to the queried point can be obtained.

For both solenoid and dipole regions, practicality was confirmed by comparing the measurement of the deviation from the conventional implementation and the speed. The proposed implementation will be used both for data collection and physical simulation of future ALICE experiments.

References

[1] P. L. Chebyshev. Théorie des mécanismes connus sous le nom de parallélogrammes. Mémoires des Savants étrangers présentés à l'Académie de Saint-Pétersbourg, vol. 7:539–586, 1854.

[2] R.Shahoyan (ALICE collaboration). https://github.com/alisw/AliRoot/blob/master/STEER/STEERBa

[3] B Abelev et al (The ALICE Collaboration). Upgrade of the ALICE experiment: Letter of intent. Journal of Physics G: Nuclear and Particle Physics, 41(8):087001, August 2014.

[4] Intel Corporation. Intel® 64 and IA-32 Architectures Optimization Reference Manual, December 2017.

[5] Musa, L (CERN). Conceptual Design Report for the Upgrade of the ALICE ITS. Technical Report CERN-LHCC-2012-005. LHCC-G-159, CERN, Geneva, Mar 2012.

[6] A. Ostrowski. On two problems in abstract algebra connected with horner's rule. In Studies in Mathematics and Mechanics presented to Richard von Mises, pages 40–48. Academic Press, 1954.

[7] V Ya Pan. "METHODS OF COMPUTING VALUES OF POLYNOMIALS". Russ Math Surv, 21(1):105–136, feb 1966.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[9] Yehoshua Perl, Alon Itai, and Haim Avni. Interpolation search—a log logN search. Communications of the ACM, 21(7):550–553, jul 1978.

[10] R.Shahoyan. Summary of the l3 magnet field analysis. ALICE Internal Note ALICE-INT-2007-012 v.1, CERN, 2007.

[11] A.Morsch R.Shahoyan. Summary of the alice dipole magnet field analysis. ALICE Internal Note ALICE-INT-2008-019 version 1.0, CERN, 2008.

[12] The ALICE Collaboration. Technical Design Report for the Upgrade of the Online-Offline Computing System. Technical Report CERN-LHCC-2015-006, ALICE-TDR-019, CERN, 2015.

A Definition of the Chebyshev polynomials T_n

The Chebyshev polynomials[1] of the first kind are defined by the recurrence relation:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) \end{aligned}$$

Here is a example of the Chebyshev polynomials of the first kind.

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1 \\ T_5(x) &= 16x^5 - 20x^3 + 5x \\ T_6(x) &= 32x^6 - 48x^4 + 18x^2 - 1 \\ T_7(x) &= 64x^7 - 112x^5 + 56x^3 - 7x \\ T_8(x) &= 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1 \\ T_9(x) &= 256x^9 - 576x^7 + 432x^5 - 120x^3 + 9x \\ T_{10}(x) &= 512x^{10} - 1280x^8 + 1120x^6 - 400x^4 + 50x^2 - 1 \\ T_{11}(x) &= 1024x^{11} - 2816x^9 + 2816x^7 - 1232x^5 + 220x^3 - 11x \end{aligned}$$

The Chebyshev polynomials T_n are polynomials with the largest possible leading coefficient, but subject to the condition that their absolute value on the interval $[-1,1]$ is bounded by 1. The interpolation built with the Chebyshev polynomial minimizes the problem of Runge's phenomenon and provides an approximation that is close to the polynomial of best approximation to a continuous function under $n \to \infty$.

B Indices of precision evaluation

Several indices of precision evaluation are considered where a field calculated by the conventional implementation at given point p_j is $A_{i,j}(i = x, y, z)$ and approximate field $B_{i,j}$. The definitions are:

- Mean deviation

$$\langle \Delta B_i \rangle = \frac{1}{N} \sum_{j=1}^{N} |B_{i,j} - A_{i,j}|$$

- Root Mean Square (RMS)

$$RMS(\Delta B_i) = \sqrt{\frac{1}{N} \sum_{j=1}^{N} (B_{i,j} - A_{i,j})^2}$$

- Maximum deviation

$$\max(\Delta B) = \max_{j \in Samples} |B_{i,j} - A_{i,j}|$$

Compared above and target precision $B_z \times 10^{-3}$(=0.002 or 0.005kG)

C Programs

Algorithm 3 Mathematica reimplementation of the AliMagF class. It is used to extract polynomials from existing parametrizations.

```
(* Cheb: A piece of scalar parametrization *)
Cheb[p_, {r_, ϕ_, z_}] :=
  Sum[Sum[Sum[-p[[j, k, l]] ChebyshevT[l - 1, z], {l, 1, Length[p[[j, k]]]}] ChebyshevT[k - 1, ϕ], {k, 1, Length[p[[j]]]}]
    ChebyshevT[j - 1, r], {j, 1, Length[p]}];

(* Inbox: Point (r,φ,z) ∈ b or not *)
InBox[b_] := And @@ Thread[Between[{r, ϕ, z}, MapThread[List, b]]];

(* Cheb3D: A pair of 3 params (Bx,By,Bz) and its interpolation region *)
Cheb3D[block_] :=
  {Cheb[block[["interpolationOutputs", #, "chebyshevPolynomialCoeffs"]],
      RescalingTransform[block[["interpolationRegion"]] // Transpose, {{-1, 1}, {-1, 1}, {-1, 1}}][
       {r, ϕ, z}]], InBox[block[["interpolationRegion"]]]} & /@ {1, 2, 3};

(* AliMagWrapCheb: Create a 3D-field parametrization *)
AliMagWrapCheb[cheb_] := Piecewise /@ (Cheb3D /@ cheb // Transpose);

(* AliMagF: Naive reimplementation of AliMagF#Field() *)
AliMagF[measurement_, param_] :=
  If[param == "dipoleParams", # /. {r → x, ϕ → y} &,
    TransformedField["Cylindrical" → "Cartesian", #, {r, ϕ, Null} → {x, y, z}] &] /@
   AliMagWrapCheb[Import[measurement, "RawJSON"][[2, param, All, 2]]];

(* Usage *)
sol = AliMagF["Sol30_Dip6_Hole.json", "solenoidParams"];
sol /. {x → 0, y → 0, z → 0}

dip = AliMagF["Sol30_Dip6_Hole.json", "dipoleParams"];
dip /. {x → 0, y → 0, z → -1000}
```

Algorithm 4 Mathematica C-source generator of the B-field parametrization in HornerForm.

```
(* Cheb: A piece of scalar parametrization *)
Cheb[p_, {r_, ϕ_, z_}] :=
  Sum[ChebyshevT[j - 1, r] Sum[ChebyshevT[k - 1, ϕ] Sum[-p[[j, k, l]] ChebyshevT[l - 1, z], {l, Length[p[[j,k]]]}], {k, Length[p[[j]]]}], {j, Length[p]}];

(* Cheb3D: A pair of 3 params (Bx,By,Bz) and its interpolation region *)
Cheb3D[block_] :=
  {Cheb[block[["interpolationOutputs", #, "chebyshevPolynomialCoeffs"]],
      RescalingTransform[(block[["interpolationRegion"]] // Transpose), {{-1, 1}, {-1, 1}, {-1, 1}}][
       {x, y, z}]], block[["interpolationRegion"]]} & /@ {1, 2, 3};

(* ExpandPower: Replace Power[y,2] to (y*y) *)
ExpandPower[e_] := StringReplace[ToString[CForm[
    e //. Power[y_, n_] :> StringJoin["(", Riffle[Table[SymbolName[y], n], "*"], ")"]
  ]], {" " → "", "\"" → ""}]; (* remove unneeded characters *)

(* CHornerForm: Encode a polynomial to C-like efficient form *)
CHornerForm[e_] := ExpandPower[HornerForm[e]];

fieldTemplate = StringTemplate[
   "void ``(const float p[3], float b[3]) { const float x = p[0], y = p[1], z = p[2];
     b[0] = ``; b[1] = ``; b[2] = ``bz(p); };"];
bzTemplate = StringTemplate[
   "float ``bz(const float p[3]) { const float x = p[0], y = p[1], z = p[2]; return ``; };"];

(* ChebCodeGen: Write C functions to be used with Field() and GetBz() *)
ChebCodeGen[hornered_, prefix_] := StringRiffle[MapIndexed[
    Module[{funcname = prefix <> ToString[First[#2] - 1]},
      bzTemplate[funcname, #1[[2, 3]]] <> "\n"
       <> fieldTemplate[funcname, #1[[2, 1]], #1[[2, 2]], funcname]] &,
    hornered], "\n\n"];

(* PrepareFastDipoleData: Take a jsonified AliMagWrapCheb data and write dipole parametrizations *)
PrepareFastDipoleData[jsonFilePath_, exportFilePath_, prefix_] :=
  Module[{data = Import[jsonFilePath, "RawJSON"]},
   Module[{dipole = Cheb3D /@ data[[2, "dipoleParams", All, 2]]},
    Module[
     {hornered = Map[{(*region*)#[[1, 2]], (*Bx,By,Bz*)CHornerForm /@ #[[All, 1]]} &, dipole]},
Export[exportFilePath, ChebCodeGen[hornered, prefix], "String"]
    ]]];

AliRootGit = "alice/AliRoot";
AliPhysicsGit = "alice/AliPhysics";
SourceFile[name_] := FileNameJoin[{$HomeDirectory, AliPhysicsGit, "PWGPP/FieldParam", name}];
DestFile[name_] := FileNameJoin[{$HomeDirectory, AliRootGit, "data/maps", name}];
PrepareFastDipoleData[SourceFile["Sol12_Dip6_Hole.json"], DestFile["dip2k.c"], "dip2k"]
PrepareFastDipoleData[SourceFile["Sol30_Dip6_Hole.json"], DestFile["dip5k.c"], "dip5k"]
```