THE HIGH ENERGY ACCELERATOR RESEARCH ORGANIZATION,
KEK

Muon g–2/EDM

J-PARC Summer Student Program

Novosibirsk State University, NSU
Budker Institute of Nuclear Physics, BINP

Semenov Aleksandr

Track finding with TMVA package for Muon g–2/EDM experiment at J-PARC

Tsukuba — 2019

Abstract

In this work we discuss the new way of the track finding procedure for g–2/EDM experiment at J-PARC. This procedure implements the multivariate classification method on machine learning techniques with TMVA package and ROOT. The data from GEANT4 simulation are used for the training and testing the program. All simulated positron tracks were divided into three groups by the value of the transverse momentum for the better machine learning. Preliminary results of the new track finding method are presented.

Content

Introduction

The purpose of the Muon g–2/EDM experiment at J-PARC is measuring of one of the most baffling quantities in particle physics. In this moment, the value of the muon anomalous magnetic moment factor α_μ is measured with unprecedented precision 0.54 *ppm* by Brookhaven laboratory [1]. The experiment in J-PARC has set a sensitivity goal of $\Delta\alpha_\mu = 0.1\, ppm$. Therefore, to achieve this purpose the experiment needs high detector efficiency and high reconstruction efficiency.

A primary proton beam of 3 GeV kinetic energy hits a 2 cm thick graphite target and produces surface muons ($P \approx 27\ MeV/c$) which are polarized. Then muons are transported via the special channel to the silica aerogel target. Here muons stop and form the muonium atoms ($\mu^+ e^-$). Some part of muoniums are evaporated from the target and ionized by laser excitation producing room-temperature muons (average momentum $P = 2.3\ KeV/c$). Created room-temperature muons are accelerated to a momentum of $300 MeV/c$ by Linac and injected into the storage magnet. The resulting beam has a very low momentum spread ($\Delta P/P = 4 \times 10^{-4}$). The positron detector placed inside the storage magnet detects positron tracks from decay of the stored muon beam. The detector is under development now [2].

After the muon decays, $\mu^+ \to e^+ + \nu_e + \bar{\nu_\mu}$, momentum of the positron is up to 305 MeV/c. In order to detect the positrons in the experiment 40 identical silicon vanes with ($220 \times 400 \times 0.03 mm$) size are used, see Fig. 1. The vanes locate radially around a centre of the detector by 9° between each other. A plane perpendicular to the vane forms the X–Y plane and a axis parallel to an vertical is Z. Also we introduce a cylindrical system, where $r = \sqrt{X^2 + Y^2}$, $\phi = \arctan Y/X$ and Z axis is the same. Therefore, the detector reconstructs three-dimensional hit position.

Track finding is a combinatorial optimisation problem. For the exact solution we should try every various heuristics to combine hits between each other and choose the best option. Therefore, we have identified the following problems:

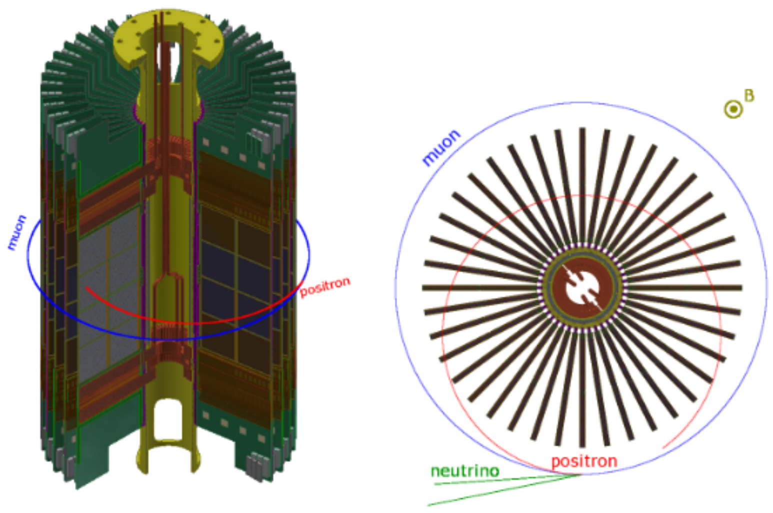


Figure 1 — Scheme diagram of the detector setup.

1. The problem is NP-complete. It means that the computing requirements grow as non-polynomial in the size of the input data, for example, as $N!$ (factorial).
2. The problem is a broken track smoothness due to multiple scattering and energy loss. Moreover, it produces the nonlinear dependencies between values.
3. High rate of muons produces a huge number of hits per event ($5 ns$). The usual number of hits in event is around 10^3 at the maximum. Therefore, classical solutions have to spend a lot of time processing each event.

At the moment, we are using the simulated data with the low muon rate (5 muons per 5 ns) to adjust the program parameters.

Chapter 1. Boosted Decision Tree

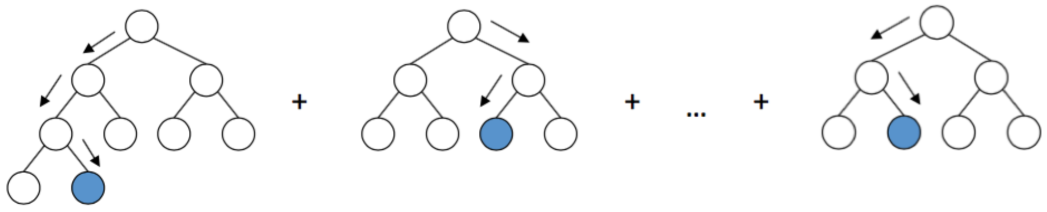
A decision tree is binary tree structured classifier [3]. The tree classifies a parameter set as a signal or background by applying a condition to one single variable at a time, and then to the other variables at each node. Therefore, the phase space is recursively split into many regions with different response based on classification accuracy. Boosting is a method of combining many trees into a strong classifier (forest). The trees are derived from the same training ensemble by reweighting events. Finally single classifier returns the weighted average response of this forest. Boosted decision tree advantages:

- Nonlinear correlations
- Speed of training
- Robustness
- Multidimensionality

Data

$\downarrow$

Boosting



$\downarrow$

Weighted average response

Figure 1.1 — The procedure of the boosting

Chapter 2. Method Description

In this paper we use the simulation data from the GEANT4 [4] with 5 muon decays per 5 ns rate. In the beginning stage we choose the low rate for the parameter adjustment and for the comfortable fast analyses. The increasing of the muon rate is included in the further algorithm improvements. We use the simulation data by two various ways: for the training and for the testing. Every file consists of 15000 events and every event consists of around 200 hits.

2.1 Chain Method

The basic idea of the method is to find for the selected hit next two hits from the same track. The procedure uses the BDT to verify the parameters between three hits. The BDT checks the parameters of the hit triple to decide if the triple belongs to the same track. In other words, the program chooses the triple, which the BDT response is greatest and is greater than the threshold value. For the understanding see Fig. 2.1. Therefore, the method is to form the consecutive segments from three hits and then combine it into the track.

The following variables were picked for the checking by the BDT:

– $\rho_{XY} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$ — distance between two hits;

– $\Delta_\varphi = \varphi_i - \varphi_j$ — difference between azimuthal angles of two hits;

– $|Z_i - Z_j| - |Z_j - Z_k|$ — difference of changing between Z-coordinates of two pairs;

– $|Z_i - Z_j| + |Z_j - Z_k|$ — sum of changing between Z-coordinates of two pairs;

– $\cos\theta_{Z\varphi}$ — cosine between directions in the $Z - \varphi$ plane for pair of the vectors: selected-next and two next hits;

– $\cos\theta_{XY}$ — cosine between directions in the $X - Y$ plane for pair of the vectors: selected-next and two next hits.

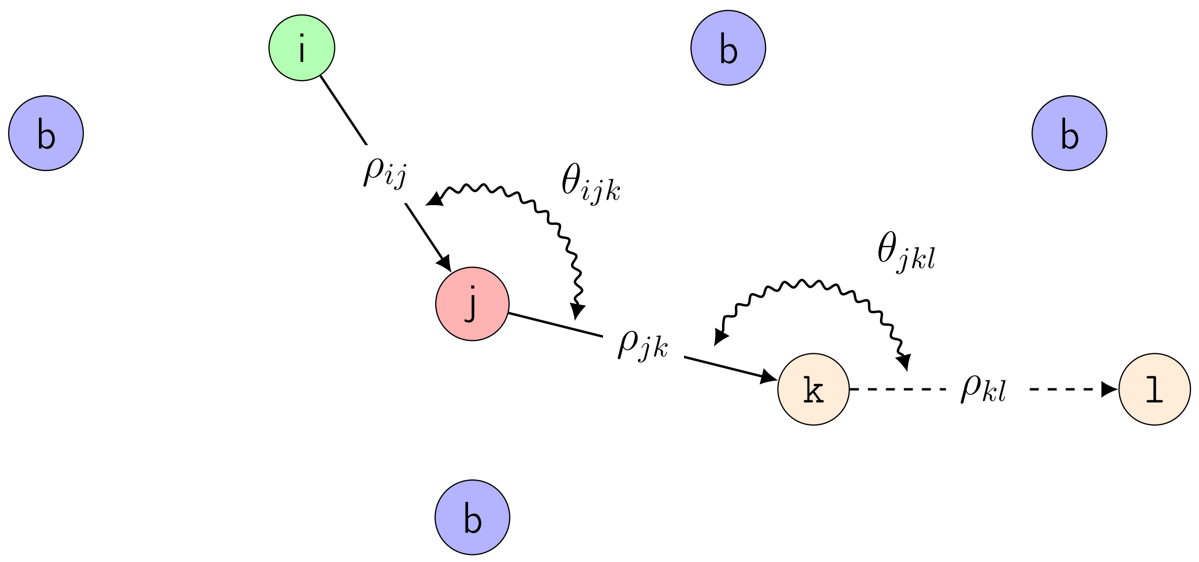


Figure 2.1 — Illustration of the chain method. The circles with names i, j, k are signal hits from the one track in the current step, l is signal in the next step and the blue circles with name b are background hits.

The first two parameters are calculated for two pairs (between selected and next hits and between two next hits). The main idea of this parameter set is to construct a smooth track by consecutive triples of the hits.

The parameters of the signal event for the BDT are defined as calculated parameters for the three consecutive hits produced by one positron. The definition of the background event is more difficult because it can be of many types. It will be discussed in the next Section 2.2. For a more detailed explanation of this method the reader is referred to the Chapter 4.

2.2 Training

After simulating the muon passing through the detector, the final simulated event has a several type of the hits: positron, electron and ghost hits. The ghost hits are electronic noises and usually located near with real hits. Before the training of the BDT, we select the hits from the same track (with the same muon index) and save them as a list. After that the hits produced by electrons are excluded from this list. The ghost hits can not be excluded from the experimental data. Therefore, this type of the hits are used in the training and will be detected in the implementation. Finally, the positron and ghost hits included in the list are sorted by time of creation.

After the dividing of the hits into tracks, we have identified three types of the tracks by the value of the transverse momentum for the better machine learning: track with large $(200, 300)$ MeV/c, medium $(100, 200)$ MeV/c and small $(0, 100)$ MeV/c transverse momentum, see Fig. 2.2. Since each type of the track has a different trajectory, the mean value of the parameters are varied. Therefore, we train the algorithm for the every track type and produce three types of the weights.

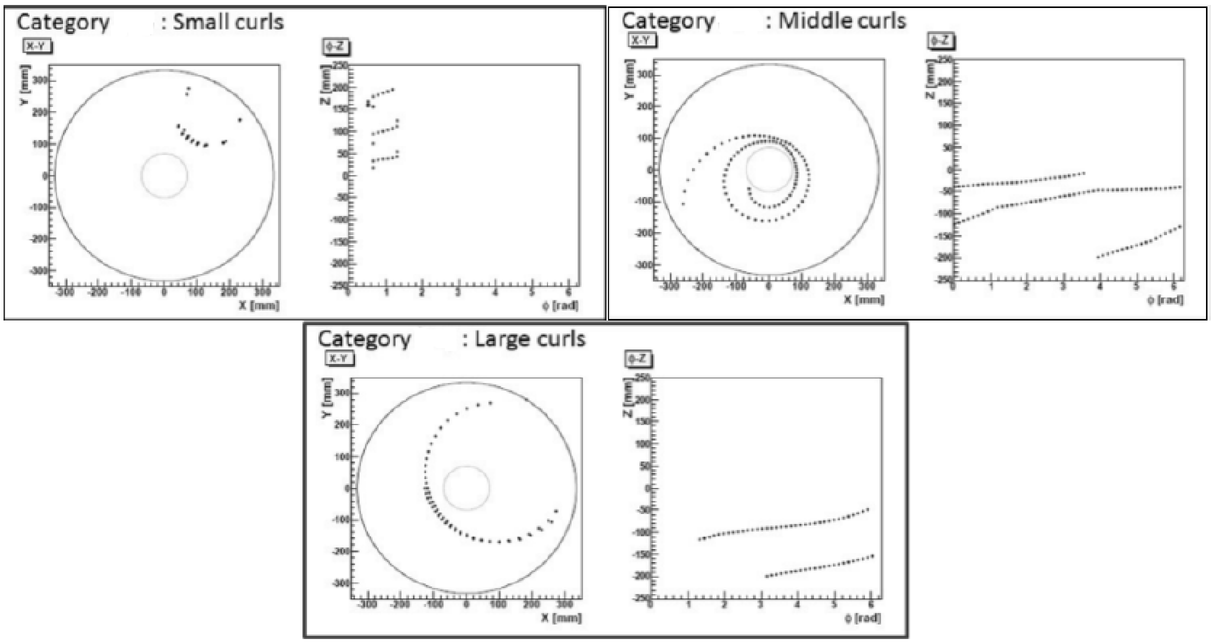


Figure 2.2 — Three types of the positron tracks.

The program selects the only track from the list, detects the value of the track transverse momentum and divides it into the consecutive triples. The signal event has parameters that correspond to three consecutive hits made by the same positron. Also we train the BDT in two directions of the track: in forward and in backward direction of the muon track. Therefore, if the track length is n hits, then number of the triples is equal to $2 \cdot (n - 2)$. As a result, we form the signal event to train the BDT.

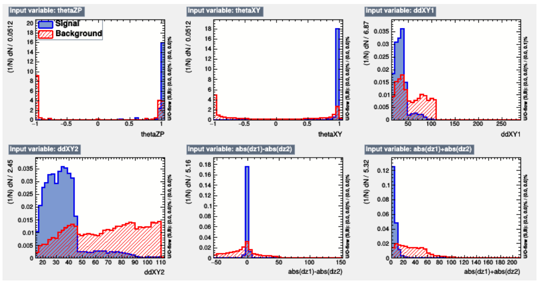
The background event is the event with non-consecutive hits. It means that we can construct a many types of the background events. In this moment, the main three types of the background events were defined for the following sequence of hits:

1. current hit $\rightarrow$ hit from background $\rightarrow$ hit from background

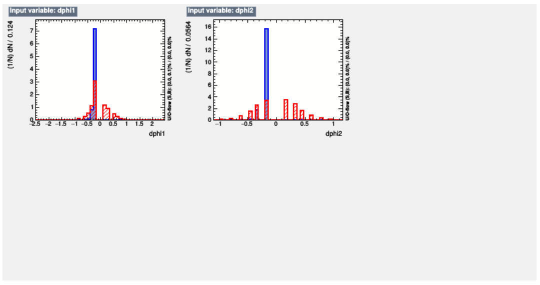
2. current hit → correct next hit → hit from background
3. current hit → other hit from track (not next) → hit from background

For one signal event, the program randomly chooses one background event from this types and trains the BDT. The algorithm forms background events in the current hit vicinity for the better training. It takes the next hit from the current hit vicinity and then takes the third hit from the second hit range.

In Fig. 2.3 the final distribution of the variables for the track with the large momentum is shown as an example. Below the results of the training are attached, see Fig. 2.4, 2.5, 2.6. For learning around 10^6 events were used. Unfortunately, the efficiency of background rejection decreases with the momentum. But the value of this results is more than enough and we apply it. We set the minimum value for the BDT response to 0.6.



a) First page.



б) Second page.

Figure 2.3 — Distributions of the variables.

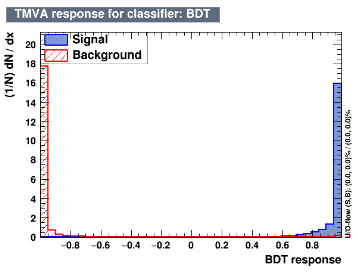


Figure 2.4 — Results of training in the large mode.

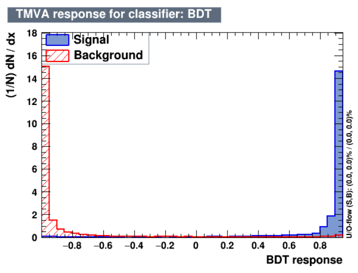


Figure 2.5 — Results of training in the middle mode.

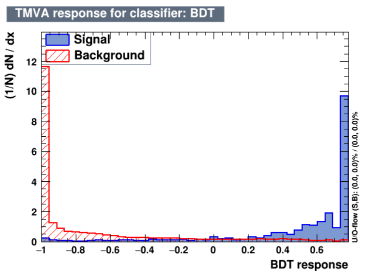


Figure 2.6 — Results of training in the small mode.

We should determine the minimum number of hits for the track candidate. After the hits dividing into the simulated tracks a distribution of the track length is presented in Fig. 3.1 for three different momentum ranges. It became clear that if we take candidates with more than 4 hits around 25% of the muon tracks are lost for the small and middle ranges. It means that the efficiency of the track reconstruction is limited by 75% for the positron with the small momentum (before 100 MeV/c). For this work 4 hits are enough to form the track candidate.

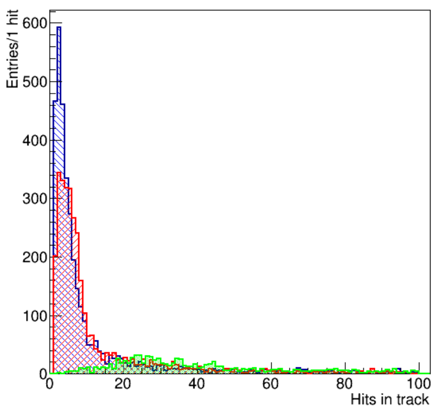


Figure 3.1 — Number of hits in one track for different curves. Green — large, red — medium, blue — small curves.

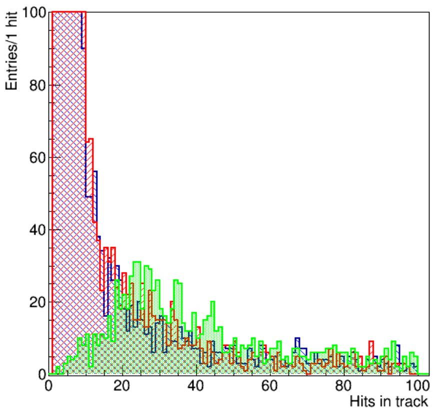


Figure 3.2 — Number of hits in one track for different curves (Zoom). Green — large, red — medium, blue — small curves.

3.1 Progress

An example of the track finding for a single event is presented in Fig. 3.3. Each colour are corresponded to a separate track candidate composed by the program. Muon rate is 5 muons in 5 ns. The algorithm correctly detected tracks. However, it divided one track into three parts (→ 0, 2, 5). To check the quality of the track finding algorithm 1000 events with 5 are processed.

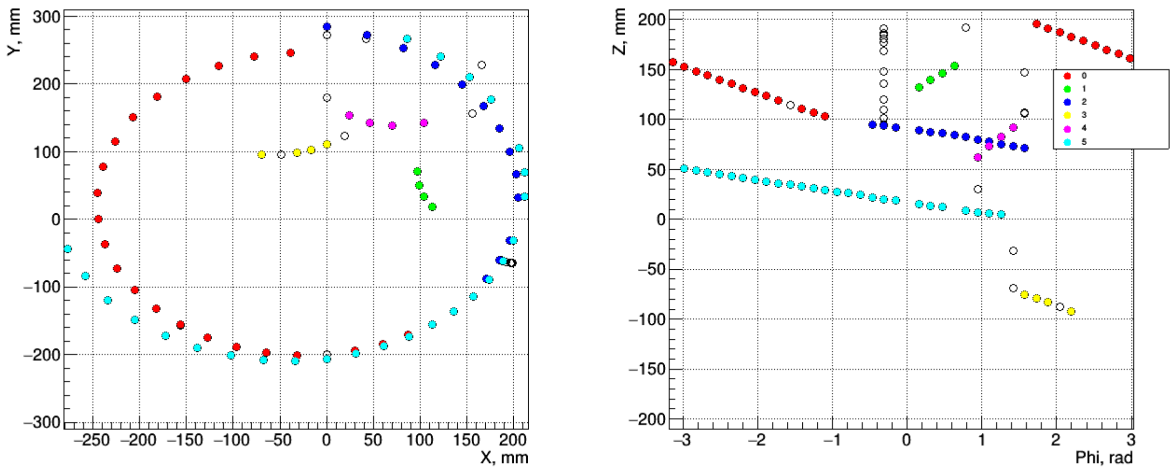


Figure 3.3 — Example of the track finding by Chain method.

The efficiency of Chain method is defined by the following way: $\varepsilon = \frac{N_{det}}{N_{full}}$, where N_{det} — the number of the reconstructed tracks from the different muons in the transverse momentum range with size 10 MeV/c and N_{full} — the full number of the simulated muons in the transverse momentum range with size 10 MeV/c, see Fig. 3.4. The statistical deviation is $\delta\varepsilon = \sqrt{\frac{\varepsilon\cdot(1-\varepsilon)}{N_{full}}}$.

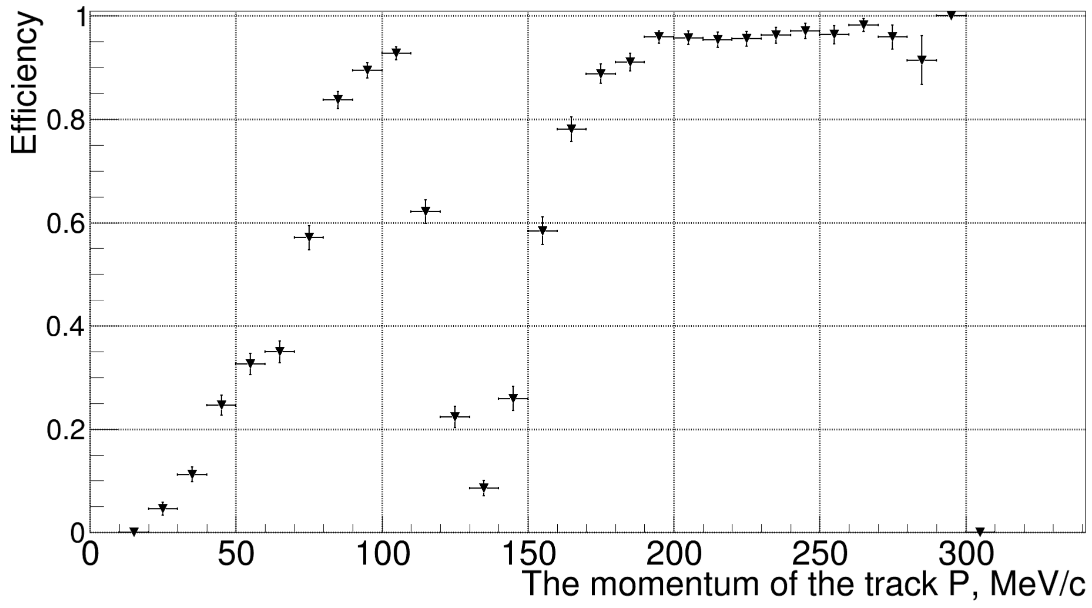


Figure 3.4 — Dependence of the efficiency of the track finding from the track transverse momentum.

Also we control the purity of the tracks: $\eta = \frac{N_{e^+}}{L}$, where N_{e^+} — the number of the positron hits produced by the main muon (most popular in the track candidate) and L — full length of the track. Then the purity is averaged by each track candidate in the transverse momentum range with size 10 MeV/c: $\widetilde{\eta} = \frac{\sum \eta_i}{n}$, where n — the number of the track candidates in the corresponding range and η_i — the purity of the track in the same range. The statistical deviation of the purity is $\delta\widetilde{\eta} = \frac{\sigma_\eta}{\sqrt{n}}$, $\sigma_\eta = \sqrt{\frac{1}{n-1}\sum\limits^{n}(\eta_i - \widetilde{\eta})^2}$ — standard deviation. The resulting dependence of the purity from the transverse momentum is shown in Fig. 3.5.

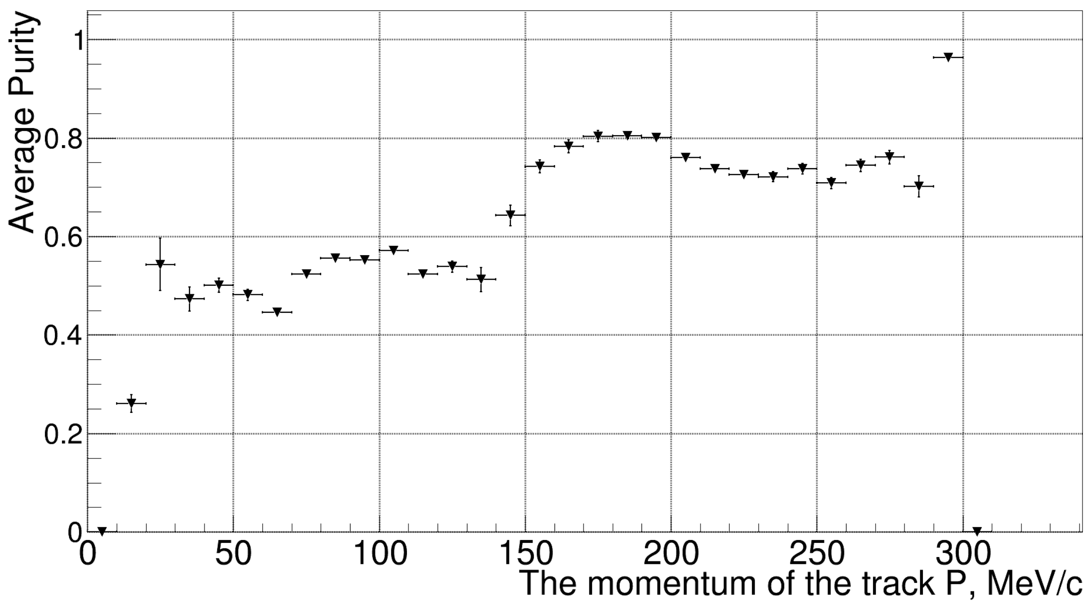


Figure 3.5 — Dependence of the track purity from the track transverse momentum.

The multiplicity of the track means that how often one muon track is divided into M parts, where M is an average multiplicity. The multiplicity is defined for the only muon index by the following way: $m_{\mu_i} = N_{rec}$, where N_{rec} — the full number of the reconstructed tracks with the same muon index. Then the multiplicity is averaged by the each muon (μ_i) with the transverse momentum in the corresponding range with size 10 MeV/c: $M = \frac{\sum m_{\mu_i}}{N_\mu}$, where N_μ — the full number of the detected muons in the corresponding momentum range. The statistical deviation is $\delta = \frac{\sigma_M}{\sqrt{n}}$, where σ_M — standard deviation of the multiplicity. The resulting dependence of the purity from the transverse momentum is shown in Fig. 3.6.

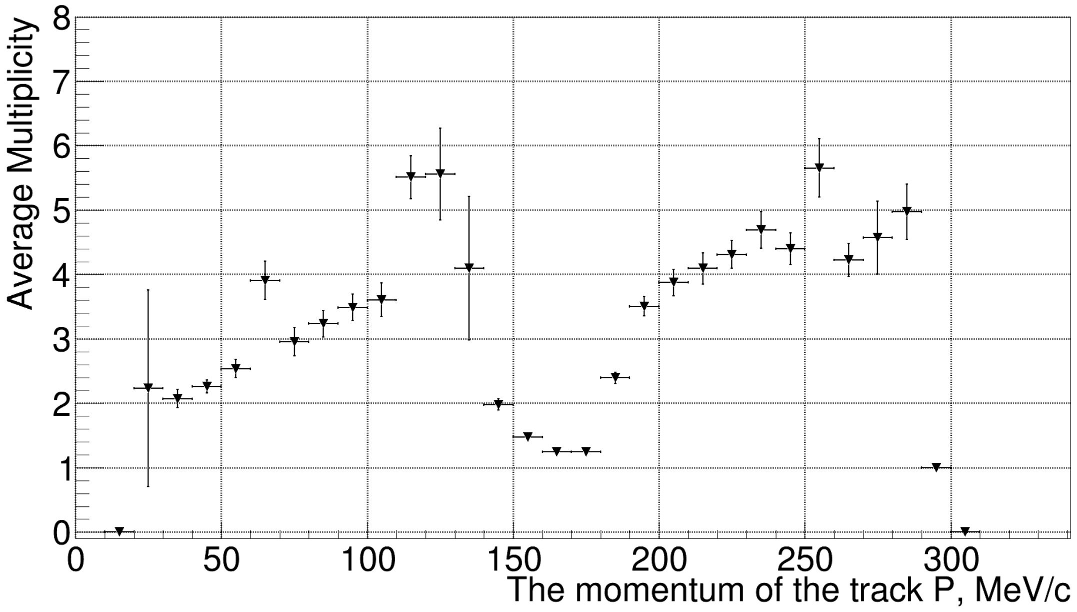


Figure 3.6 — Dependence of the track multiplicity from the track transverse momentum.

For monitoring the performance of the algorithm the dependence of the time performance from the number of hits in event are presented in Fig. 3.7. The value of the time performance is approximately increasing as a $A * N^2 \cdot \ln(N)$, where N — number of hits in event and $A \approx 1.3 \cdot 10^{-5}$ — proportional constant.

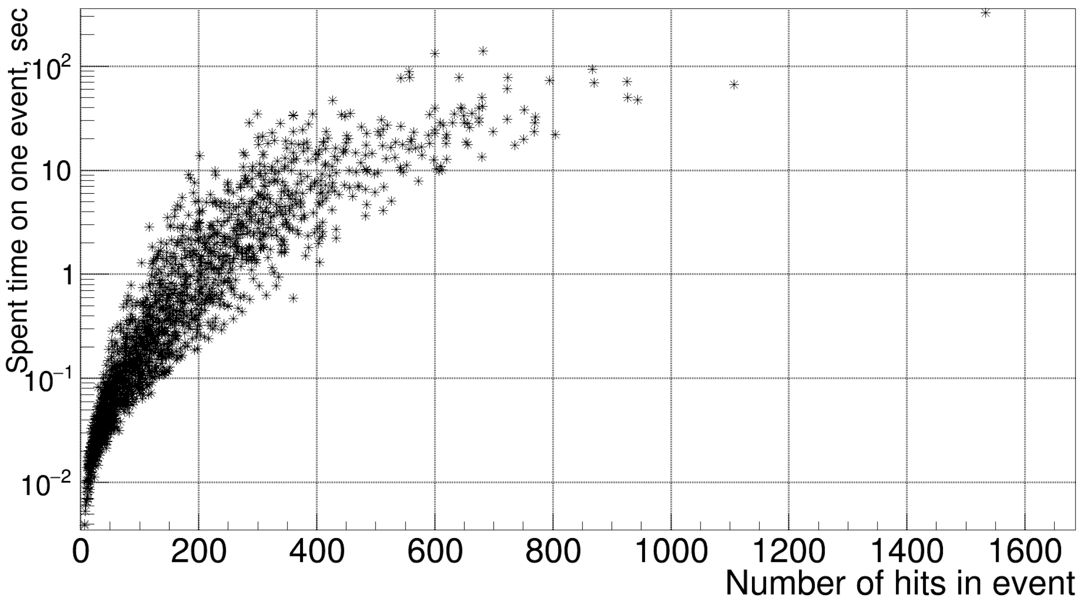


Figure 3.7 — Dependence of the algorithm time performance from the number of hits in event.

3.2 Failures

At the moment we have identified the following failures: high multiplicity and paradoxically high efficiency in low momentum range.

The high multiplicity connected with the several reasons. The first reason is that the track with the large momentum has a part outside the detector. Therefore the detector loses the part of the track and can not connect two consecutive hits into triple. In Fig. 3.3 the tracks number 5 and 2 are example of this effect. The second reason is the smoothness breaking of the track. In Fig. 3.3 the tracks with numbers 2 and 0 presents the second effect.

The paradoxically high efficiency in low momentum range is explained by forming of fake tracks. In Fig. 3.5 the average purity is about 50% before 100 MeV/c.

In this chapter the track finding algorithm will be explained in details. This chapter is divided into two parts. In the first section we explain the BDT parameters. The second section provides a step-by-step implementation.

4.1 Training and testing

The following parameters set for the BDT are presented in Tab. 4.1, 4.2. The other parameters are default [3].

Table 4.1

The BDT parameters

Name	Volume	Description
NTrees	200	Number of trees in the forest
MaxDepth	4	Max depth of the decision tree allowed
MinNodeSize	5%	Minmum percentage of training events required in a leaf node
nCuts	5	Number of grid points in variable range used in finding optimal cut in node splitting
BoostType	Grad	Boosting type for the trees in the forest
Shrinkage	1	Learning rate for GradBoost algorithm
UseNvars	4	Size of the subset of variables used with RandomisedTree option

Table 4.2

The BDT parameters

Name	Volume	Description
SeparationType	SDivSqrtSPlusB	Separation criterion for node splitting
NodePurityLimit	0.75	In boosting/pruning, nodes with purity > NodePurityLimit are signal; background otherwise

4.2 Implementation

Before reading the step-by-step instruction look at scheme of the algorithm which is presented in Fig. 4.1.

1. Select the hit with the highest absolute value of the Z variable.
2. Check all possible triples with the current hit and select the triple with the greatest response.
3. If the algorithm does not find the triple (every triple has a response which is less than threshold) it changes direction of search from forward to backward and repeat previous action.
4. Check all possible triples with the new two hits (found in the previous action) and select the better third hit candidate. And so continuing on.
5. If the algorithm does not find the good candidate (every response is less than cut) it changes the direction of the search from forward to backward and repeat the previous action for the second and first hits in this track candidate. It will find the another end of the track.
6. When the algorithm finished to find the both ends of the track candidate it remove every hit of this track from full hit list for the current event.
7. Repeat from the first action until the hit list is empty.

8. If the track length is more than 4 hits it will be saved and another hits are saved together in the hit list.
9. Change the file with weights for the other momentum range and repeat all actions.

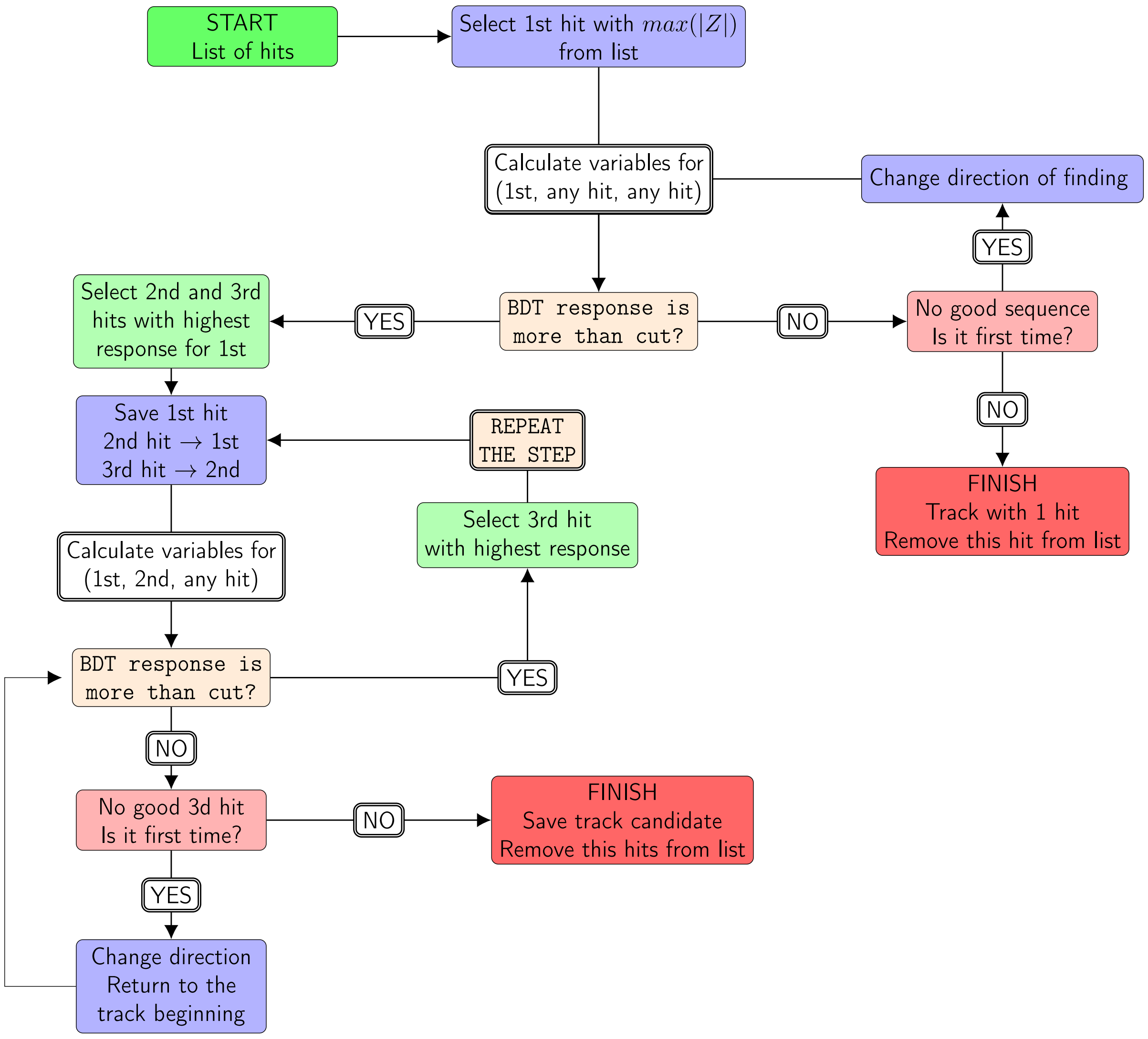

START
List of hits
Select 1st hit with $max(|Z|)$ from list
Calculate variables for (1st, any hit, any hit)
Change direction of finding
YES
BDT response is more than cut?
YES
Select 2nd and 3rd hits with highest response for 1st
NO
No good sequence Is it first time?
NO
FINISH
Track with 1 hit
Remove this hit from list
Save 1st hit
2nd hit $\rightarrow$ 1st
3rd hit $\rightarrow$ 2nd
REPEAT THE STEP
Select 3rd hit with highest response
Calculate variables for (1st, 2nd, any hit)
BDT response is more than cut?
YES
NO
No good 3d hit Is it first time?
NO
FINISH
Save track candidate
Remove this hits from list
YES
Change direction Return to the track beginning

Figure 4.1 — Illustration of Chain Method algorithm.

The number of the action loops grows up as a $N^2 \cdot \log N$, where N is the number of hits in a single event.

To solve the the failures mentioned in Sec. 3.2 we suggest two possible solutions: Welding Module and Self-Learning.

5.1 Welding module

Once the track candidates are collected, the following parameters of these tracks are determined:

– Center and radius of the track in XY plane

– Velocity along Z axis

After that, all the hits of the event are checked for belonging to each candidate track. If some candidates have common hits they will be combined into one track. If a hit from the candidate does not fit for the final track, it will not be included. The module has to reduce the multiplicity and increases the purity of the tracks.

5.2 Self-Learning

The algorithm can use the output files to complete the definitions of the signal and background events. The correct and incorrect consequences of the saved hits can be used in the new learning of the BDT. After a few iterations the balance between events will be achieved. The illustration of this module is presented in Fig. 5.1. As a result, the procedure has to reduce the multiplicity and increase the purity of the tracks due to better detection of the signal and background hits.

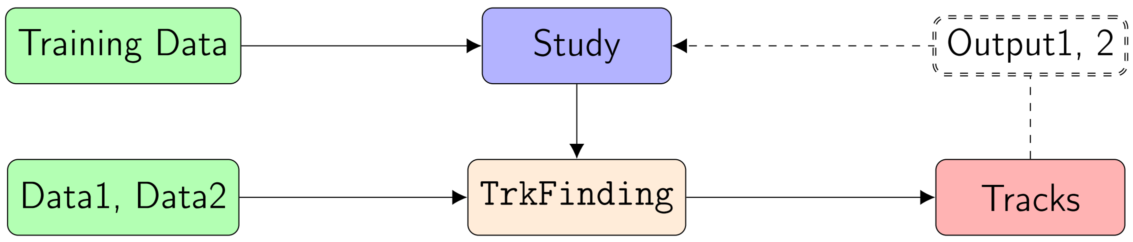


Figure 5.1 — Illustration of Self-Learning.

Conclusion

In the current work we suggest and develop the track finding method based in the Machine Learning techniques. According to the results of the papers we can say that the method works stably and fast.

- The new algorithm based on the Machine Learning technique for the track finding are developed
- The preliminary results have been obtained
- The reconstruction efficiency exceeds 95% at the track momentum value of more than 200 MeV/c.
- The algoritm shows good computational performance with KEK CC
- Further improvements of the algorithm are required
- Tests with the high rate and 1 muon per event are necessary

In conclusion, we think that the current method has a good prospect and has to be improved in the future.

References

1. Measurement of the Negative Muon Anomalous Magnetic Moment to 0.7 ppm / G. Bennett [et al.]. — 2004.
2. A New Approach for Measuring the Muon Anomalous Magnetic Moment and Electric Dipole Moment / M. Abe [и др.] // PTEP. — 2019. — Т. 2019, № 5. — С. 053C02. — DOI: 10.1093/ptep/ptz030. — arXiv: 1901.03047 `[physics.ins-det]`.
3. TMVA 4.2.0 – Toolkit for Multivariate Data Analysis with ROOT / A. Hoecker [et al.]. — 2017.
4. Geant4—a simulation toolkit / S. Agostinelli [et al.] // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2003. — Vol. 506, no. 3. — P. 250–303.

Figure list

Table list