Development and software implementation of optimal algorithms for event reconstruction, evaluation of the quality of event reconstruction and simulation of detector components in the BM@N experiment

NEMNYUGIN S.A., DRIUK A.V., IUFRIAKOVA A.A., KAKHANOVSKAYA N.E., MASHITSIN K.I., MERTS S.P., ROUDNEV V.A.

SAINT PETERSBURG STATE UNIVERSITY

JOINT INSTITUTE OF NUCLEAR RESEARCH

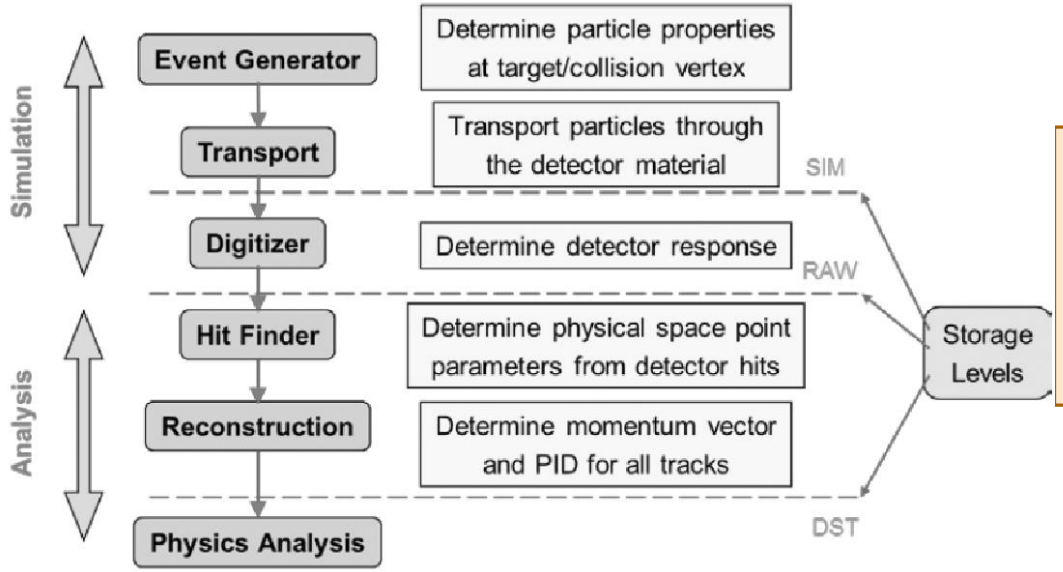
Outline

- Performance study and bottlenecks of the BmnRoot framework.
- Global tracks reconstruction in the BM@N experiment.
- Finding of primary and secondary vertices.
- Quality assurance in the BmnRoot.
- Machine learning methods of particles identification.
- Adding triggers description in the geometry of SRC.

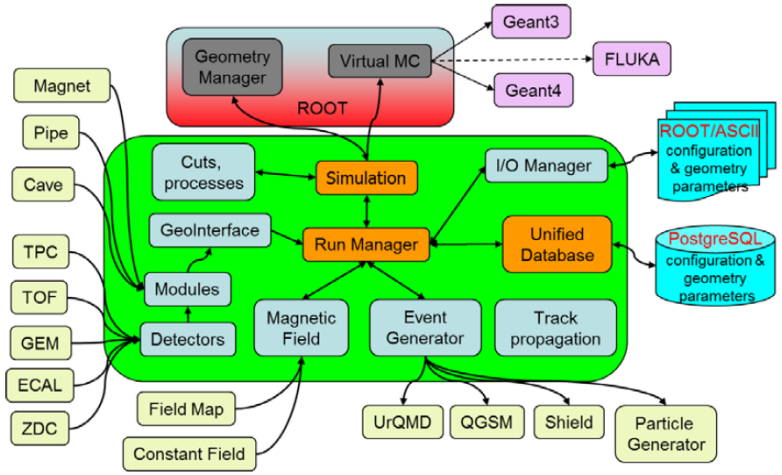
Work is supported by Russian Foundation for Basic Research grant 18-02-40104 mega

Performance study and bottlenecks of the BmnRoot framework

- ✓ Complex structure (simulation/analysis) with a lot of modules, hundreds of thousands lines of code.
- ✓ Simulation modules include: setup configuration and geometry tools, Monte-Carlo event generators (BOX, DQGSM, UrQMD, SHIELD), Virtual Monte-Carlo interface, transport codes (Geant3, Geant4, Fluka), magnetic field mappers, digitizers etc.
- ✓ Sampling size in the Monte Carlo simulation must be 10^5 – 10^6 events and more. Event generators are based on realistic physical models and are time-consuming. Simulation performance should be improved.



Reasonable Monte-Carlo simulation needs for large sampling size.
Realistic simulations are time-consuming.



BmnRoot simulation modules

✓ Dynamic hotspot analysis of the BmnRoot simulation modules is most reasonable (various possible execution paths => performance metrics are parametrized by input data).

✓ Hotspots localization is performed.

Testbench

CPU: Intel Xeon E-2136 @ 4.5GHz Turbo (6C 2xHT, L3 Cache 8MB)
RAM: 32GB 2666MHz DDR4
OS: Ubuntu 16.04.6 LTS

Testcase

Simulation with DQGSM generator 1000-5000 events.

Function / Call Stack	CPU Time ▼	Module
TRandom::Gaus	145.867s	libMathCore.so.6.16.00
DeadZoneOfStripLayer::IsInside	137.187s	libGem.so.0
__cos_fma	98.153s	libm.so.6
TRandom3::Rndm	80.814s	libMathCore.so.6.16.00
__sin_fma	77.566s	libm.so.6
deflate	74.185s	libz.so.1
FairMCApplication::Stepping	67.573s	libBase.so.18.2.0
BmnGemStripModule::AddRealPointF	46.391s	libGem.so.0
BmnGemStripLayer::ConvertPointToSt	43.867s	libGem.so.0
std::map<std::pair<int, int>, int, std::les	41.465s	libBmnData.so.0
StripCluster::AddStrip	41.270s	libGem.so.0
BmnGemStripLayer::IsPointInsideStrip	31.419s	libGem.so.0
BmnGemStripLayer::ConvertNormalPo	27.492s	libGem.so.0
std::map<std::pair<int, int>, int, std::les	20.336s	libBmnData.so.0
std::operator<<int, int>	18.805s	libBmnData.so.0
BmnGemStripLayer::IsPointInsideDea	18.630s	libGem.so.0
BmnNewFieldMap::FieldInterpolate	18.493s	libBmnField.so.0
std::_Rb_tree_iterator<std::pair<int cor	16.267s	libBase.so.18.2.0
TArrayF::At	14.695s	libBmnField.so.0
BmnNewFieldMap::IsInside	13.857s	libBmnField.so.0
std::map<int, bool, std::less<int>, std::a	12.740s	libBmnData.so.0
BmnFieldMap::Interpolate	12.701s	libBmnField.so.0

Hotspots of the BmnRoot/FairRoot simulation modules

✓ Performance optimization of simulation modules = parallelization of time-consuming hotspots.

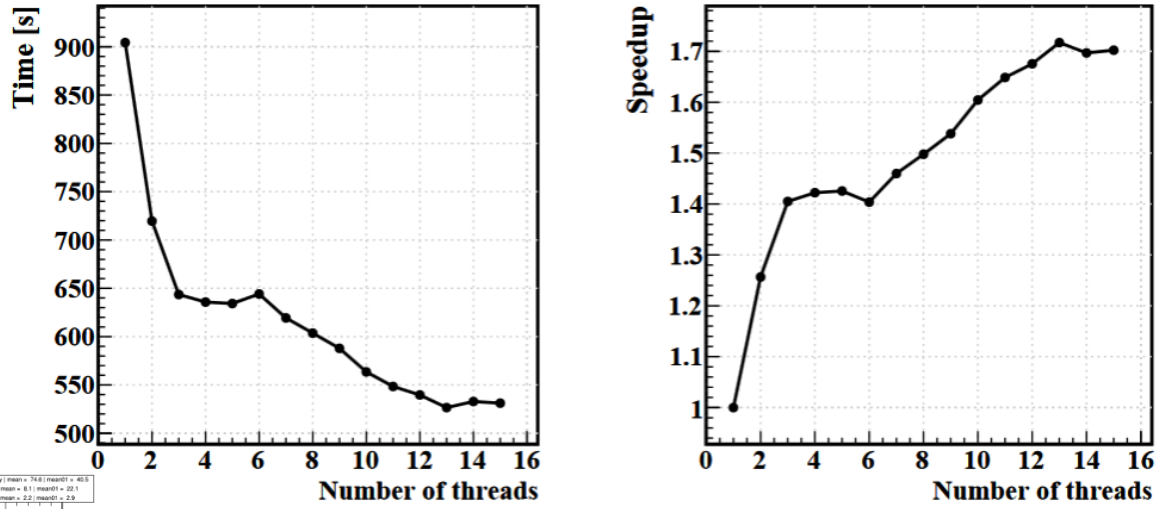
✓ Tests of scalability.

✓ Tests of correctness and scalability of optimized code (Quality Assurance).

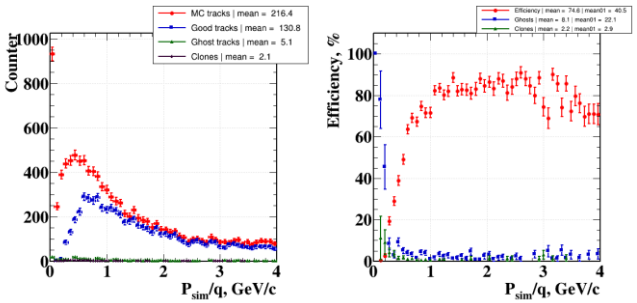
BmnGemStripDigitizer

```
…
FairMCPoint* GemStripPoint;
Int_t NNotPrimaries = 0;
#pragma omp parallel
#pragma omp for schedule(dynamic)
for (UInt_t ipoint = 0; ipoint < fBmnGemStripPointsArray->GetEntriesFast();
ipoint++) {
GemStripPoint = (FairMCPoint*) fBmnGemStripPointsArray->At(ipoint);
…
```

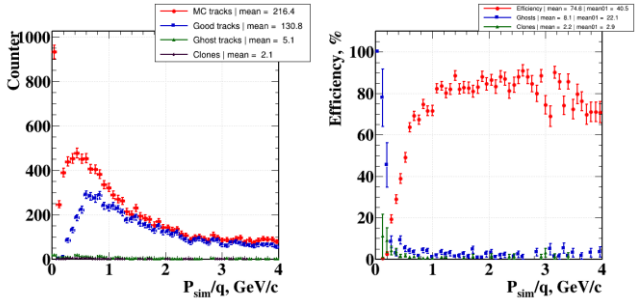
OpenMP parallelization



Scalability of the BmnRoot simulation parallelized with OpenMP



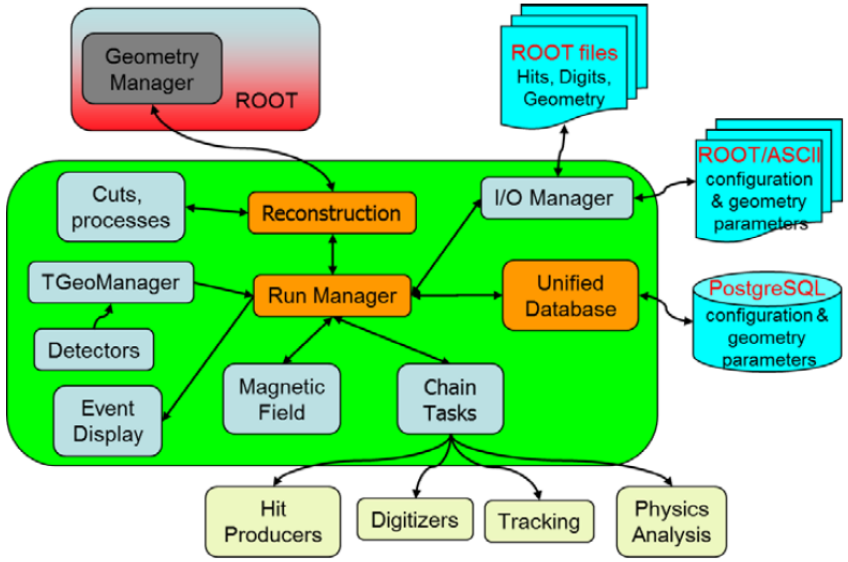
QA - 1 thread



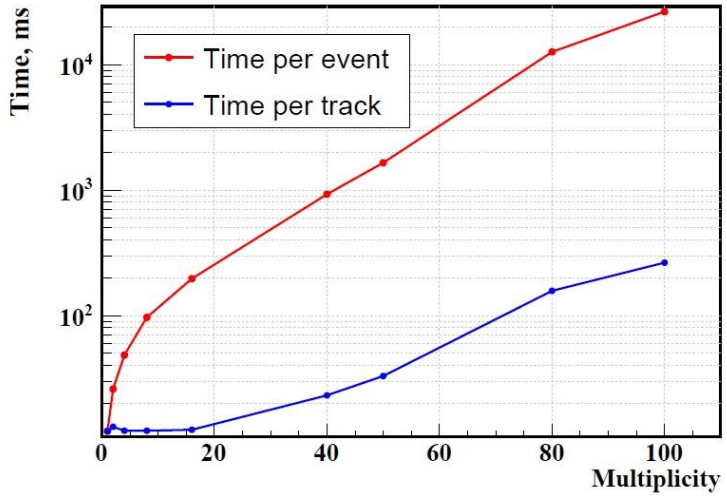
QA - 4 thread

Quality Assurance for simulation

1. Reconstruction BmnRoot modules include: setup configuration and geometry tools, all detector subsystems (GEMs, multiwired proportional chambers, drift chambers, silicon trackers, zero-degree calorimeters, time-of-flight detector, two arms for the SRC experiment etc.), magnetic field accounting, digitizers, hit finders, software realization of matching (local/global) etc.
2. Reconstruction modules are time-consuming.
3. Processing of more than 10^5 – 10^6 events is required. Reconstruction performance should be improved.

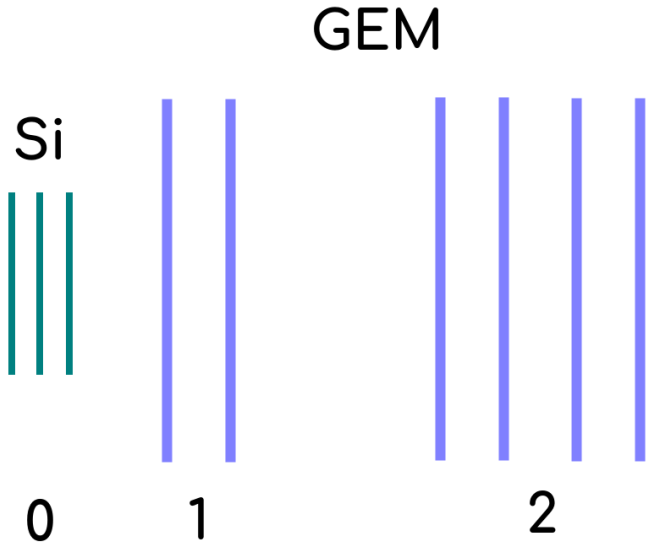


BmnRoot reconstruction modules



BmnRoot reconstruction time vs events multiplicity

BmnRoot reconstruction



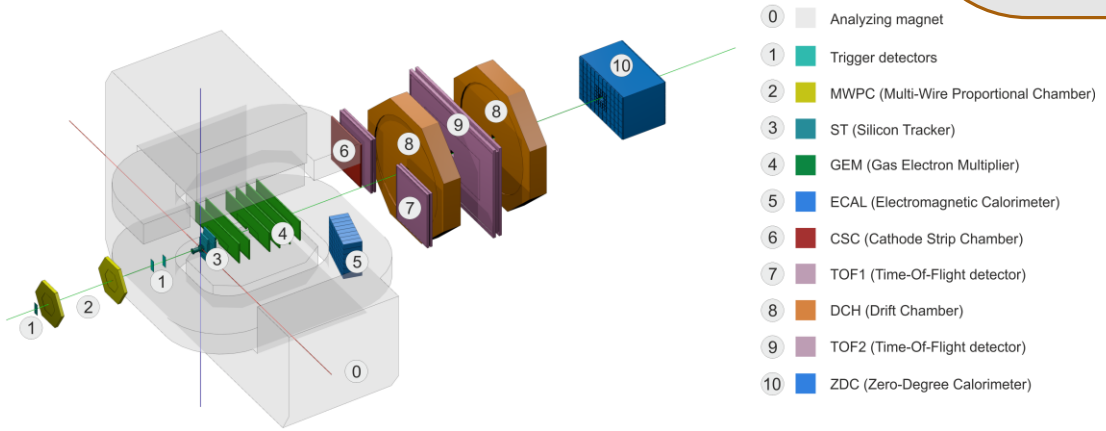
GEM – Gas Electron Multiplier.
SI – Silicon detector.

Track reconstruction algorithm

1. Search for high momentum tracks.
Construct 4-hits candidates and estimate their parameters in zone 2. Propagate each candidate to hits in zone 1 and zone 0 by the Kalman Filter (KF) etc.

2. Search for high momentum tracks with low efficiency.
Construct 3-hits candidates and estimate their parameters in zone 2 for UNUSED hits. Extrapolate each candidate to hits in zone 1 and zone 0 by KF etc.

3. Search for low momentum tracks with inefficiency.
Construct 2-hits candidates in zone 1 for UNUSED hits. Extrapolate each candidate to hits in zone 0 by straight line in ZY plane etc.



BmnInnerTrackingRun7::Exec()

```
…
FindTracks_4of4_OnLastGEMStations();
FindTracks_3of4_OnLastGEMStations();
fNHitsCut = 5;
FindTracks_2of2_OnFirstGEMStationsDownstream();
FindTracks_2of2_OnFirstGEMStationsUpstream();
…
```

Performance problems of the BmnRoot reconstruction modules

Time estimates for reconstruction:
- ✓ **Monte Carlo data 1 sec/event**
- ✓ **Experimental data 6 sec/event**
- ✓ **One file (200 000 event) up to 2 weeks**

Testbench

CPU: Intel Xeon E-2136 @ 4.5GHz (6C 2xHT)

Testcase

Simulated data: DQGSM generator 1000-5000 events.
Experimental data: Run 7 at BM@N, Argon beam, Al target.

Analysis summary

A lot of hotspots belong to the BmnField module – load of the analyzing magnet field:

- 3D Cartesian lattice;
- piecewise linear interpolation between lattice nodes;
- extrapolation outside known values.

Details of CPU Time Consumption
Si+GEM Track Finder: 45%
Global Matching: 21%
Vertex Finder: 19%

HOTSPOTS

Function / Call Stack	CPU Time ▼	Module
clock	66.450s	libc.so.6
BmnKalmanFilter::RK4Order	34.481s	libBmnData.so.0.0.0
BmnNewFieldMap::FieldInterpolate	23.124s	libBmnField.so.0.0.0
TArrayF::At	22.950s	libBmnField.so.0.0.0
inflate	20.673s	libz.so.1
BmnKalmanFilter::TransportC	17.638s	libBmnData.so.0.0.0
BmnNewFieldMap::IsInside	15.992s	libBmnField.so.0.0.0
TArray::BoundsOk	15.566s	libBmnData.so.0.0.0
BmnFieldMap::Interpolate	15.226s	libBmnField.so.0.0.0
std::vector<double, std::allocator<doub	13.862s	libBmnData.so.0.0.0
operator new	12.704s	libstdc++.so.6
std::vector<double, std::allocator<doub	12.222s	libBmnDst.so.0.0.0
BmnKalmanFilter::RK4TrackExtrapolat	11.896s	libBmnData.so.0.0.0
std::vector<double, std::allocator<doub	11.128s	libBmnData.so.0.0.0
TGeoVoxelFinder::GetNextCandidates	10.982s	libGeom.so.6.16
__pow	10.944s	libm.so.6
std::__fill_n_a<double*, unsigned long	10.074s	libBmnData.so.0.0.0
TGeoVoxelFinder::GetCheckList	9.832s	libGeom.so.6.16
__GI_	8.701s	libc.so.6
std::vector<double, std::allocator<doub	8.420s	libBmnData.so.0.0.0
std::vector<BmnLink, std::allocator<Bn	7.352s	libSilicon.so.0.0.0
TGeoNavigator::SearchNode	6.766s	libGeom.so.6.16

Hotspots of the BmnRoot reconstruction modules

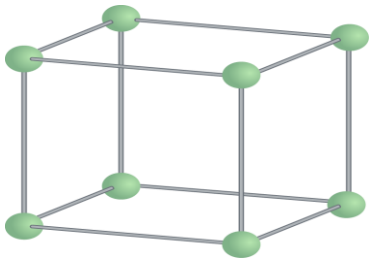
Compiler (GCC) optimization
DEBUG➔RELEASE
(O2 level optimization)

Before optimization
Monte Carlo 1 sec/event
Experimental 6 sec/event
One file (200 000 events)
2 weeks

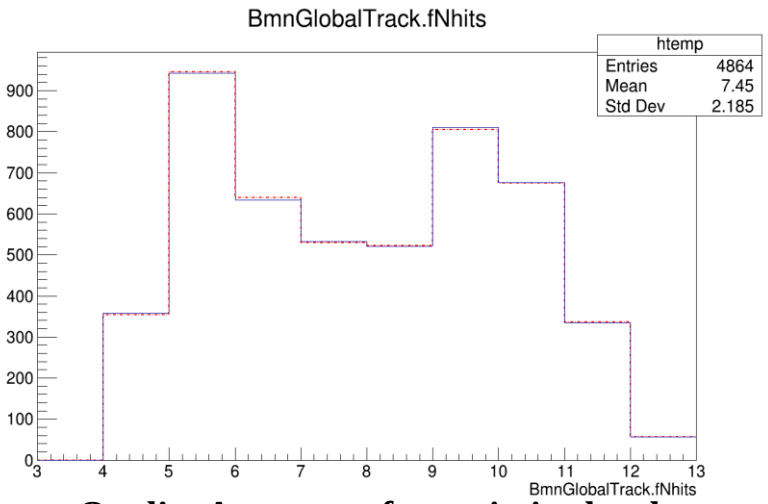
After optimization
Monte Carlo 0.3 sec/event
Experimental 0.7 sec/event
One file (200 000 events)
39 hours

Algorithmic optimization of BmnFieldMap

1. Second hotspot next to the Kalman Filter.
2. Measured values of the analyzing magnet's field are saved at discrete set of 3-dimensional cubic lattice.
3. Proposal for optimization – replace linear-piecewise interpolation by constant-piecewise interpolation. Calculation for 8 vertices of cube elementary cell is not necessary => reduction of number of floating point operations.



$\Delta x = 0.25$ cm
$\Delta y = 0.45$ cm
$\Delta z = 0.17$ cm



Quality Assurance for optimized code.
Histogram for the number of reconstructed hits

```
Double_t BmnNewFieldMap::FieldInterpolate(TArrayF* fcomp, Double_t x, Double_t y, Double_t z) {
    Int_t ix = 0;
    Int_t iy = 0;
    Int_t iz = 0;
    Double_t dx = 0.;
    Double_t dy = 0.;
    Double_t dz = 0.;

    Int_t iix = 0;
    Int_t iiy = 0;
    Int_t iiz = 0;


    if (IsInside(x, y, z, ix, iy, iz, dx, dy, dz)) {

        iix = Int_t(Nint((x - fXmin) / fXstep));
        iiy = Int_t(Nint((y - fYmin) / fYstep));
        iiz = Int_t(Nint((z - fZmin) / fZstep));

        Hh = fcomp->At(iix * fNy * fNz + iiy * fNz + iiz);

       /**fHa[0][0][0] = fcomp->At(ix * fNy * fNz + iy * fNz + iz);
        fHa[1][0][0] = fcomp->At((ix + 1) * fNy * fNz + iy * fNz + iz);
        fHa[0][1][0] = fcomp->At(ix * fNy * fNz + (iy + 1) * fNz + iz);
        fHa[1][1][0] = fcomp->At((ix + 1) * fNy * fNz + (iy + 1) * fNz + iz);
        fHa[0][0][1] = fcomp->At(ix * fNy * fNz + iy * fNz + (iz + 1));
        fHa[1][0][1] = fcomp->At((ix + 1) * fNy * fNz + iy * fNz + (iz + 1));
        fHa[0][1][1] = fcomp->At(ix * fNy * fNz + (iy + 1) * fNz + (iz + 1));
        fHa[1][1][1] = fcomp->At((ix + 1) * fNy * fNz + (iy + 1) * fNz + (iz + 1));**/

        return Interpolate(dx, dy, dz);
    }
    return 0.;
}
```

Optimized code of FieldInterpolate method of BmnNewFieldMap class.

- ✓ Build in Debug mode (compiler optimization switched off) reduced total execution time by 10%.
- ✓ Build in O2 optimization mode reduced execution time by 4%.
- ✓ Execution time of the BmnField is 7% from total reconstruction time.
- ✓ Quality Assurance methods used in BM@N demonstrates very small difference between non-optimized and optimized results.

Track finders OpenMP parallelization

```
BmnInnerTrackingRun7::FindTracks_4of4_OnLastGEMStations() {
const Int_t nxRanges = 8;
const Int_t nyRanges = 5;
...
vector<BmnTrack> candidates;
vector<BmnTrack> sortedCandidates;
Int_t nThreads = THREADS_N;
vector<vector<BmnTrack>> candsThread(nThreads);
clock_t t0 = 0;
Int_t threadNum;
Int_t sH8 = sortedHits[8].size();
#pragma omp parallel if(sH8 > 100)  num_threads(nThreads)
#pragma omp for  // schedule(static,1)
for (Int_t ii = 0; ii < sH8; ++ii) {
BmnHit*  hit8;
hit8 = sortedHits[8].at(ii);
...
```

Reasonable scalability of reconstruction is not yet achieved

Possible reasons of low efficiency:

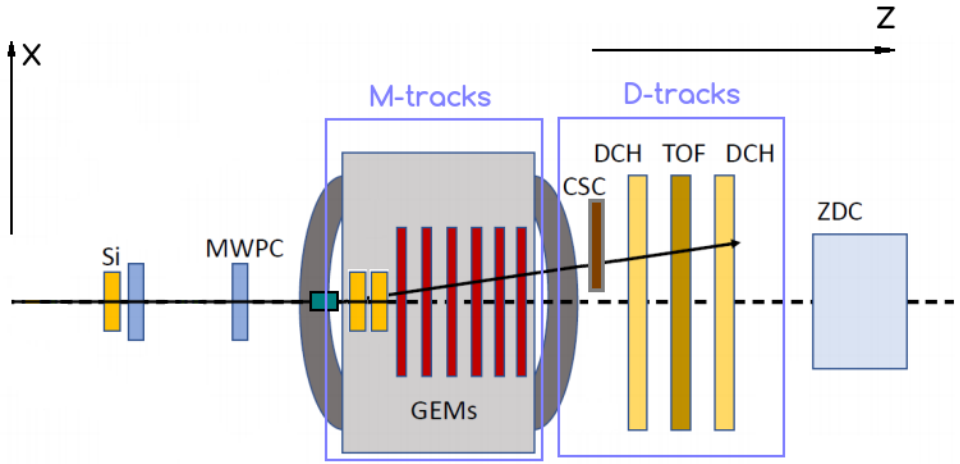
- ✓ In many cases number of loops iterations is zero so efficiency of OpenMP-parallelization is low.
- ✓ Most significant hotspot relates to the Kalman filter, so it should be optimized first.

Perspective approaches to performance optimization

Distributed computing (PROOF).

Hybrid programming (GPGPU accelerators).

Global tracks reconstruction in the BM@N experiment

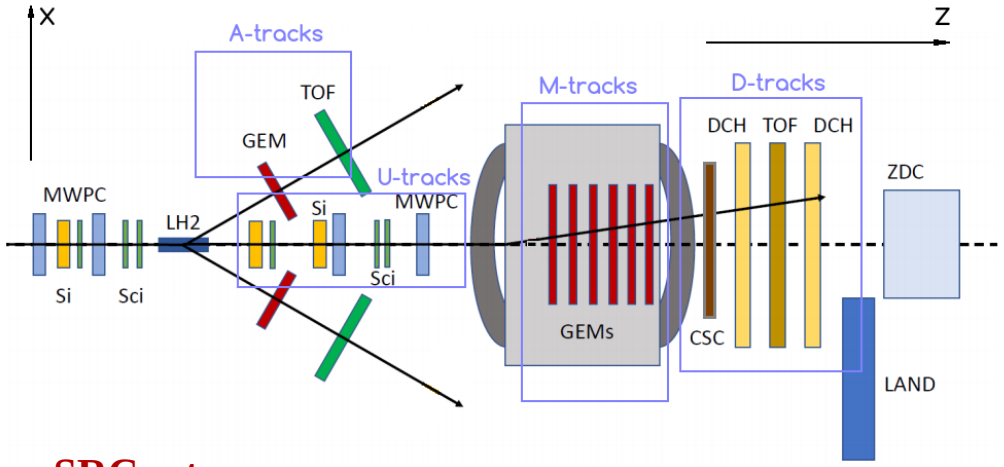


BM@N setup

Detectors inside magnet: 3 Silicon + 6 GEM => M-tracks

Downstream detectors: 1 CSC + 2 TOF + 2 DCH => D-tracks

Merts S. *Global track and vertex reconstruction in SRC and argon beam BM@N experiment.* RFBR Grants for NICA Conference.



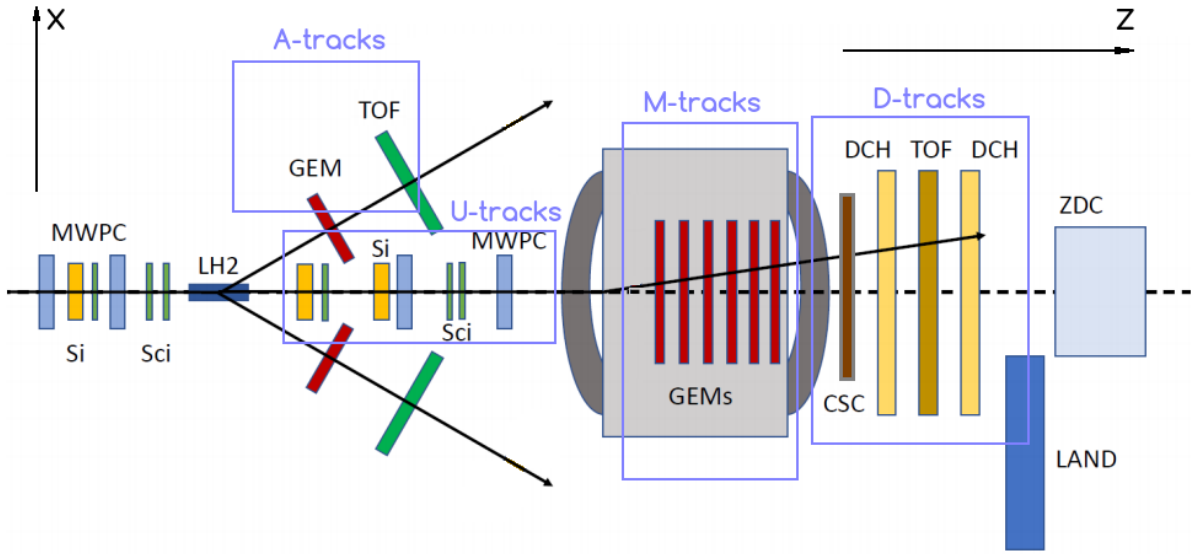
SRC setup

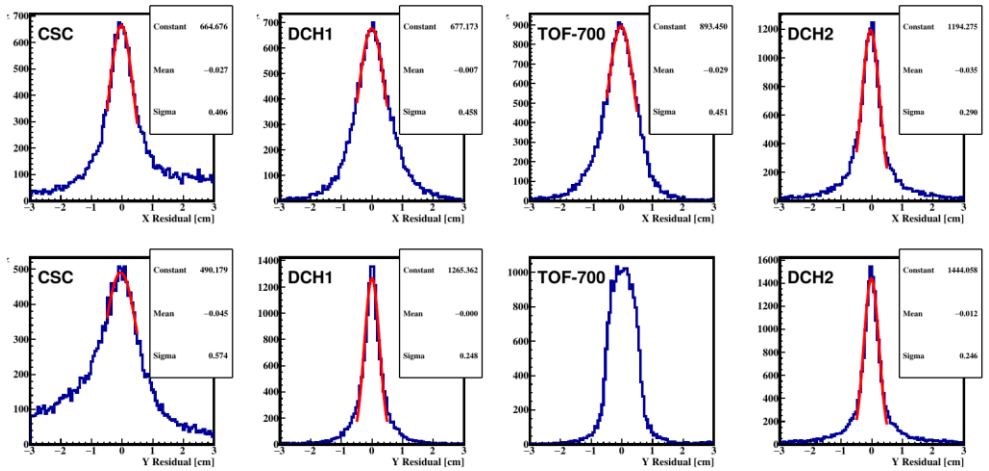
Upstream detectors: 3 Silicon + 2 MWPC => U-tracks

Detectors inside magnet: 6 GEM => M-tracks

Downstream detectors: 1 CSC + 1 TOF + 2 DCH => D-tracks

Detectors in arms: 2 GEM + 2 TOF => A-tracks





Residuals for SRC setup

Algorithm of global reconstruction

Step 1. Alignment

1. Extrapolate M-tracks to plane with hits.
2. Create track-to-hit (all-to-all) connections.
3. Calculate and fit residuals μ_x, μ_y, σ_x, σ_y.
4. Shift all hits by μ_x, μ_y.

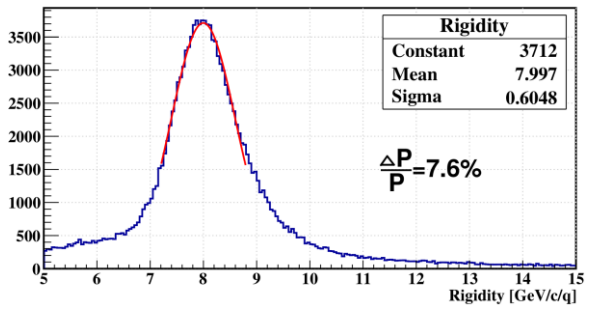
Step 2. Matching

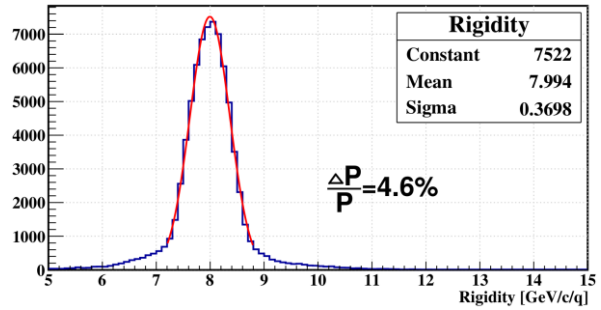
1. Propagate each track to plane with hits.
2. Find the nearest hit in $\pm 3\sigma_x$ and $\pm 3\sigma_y$.
3. Update track parameters by connected hit information: track length, momentum etc.

Benefits of global tracking

Momentum resolution

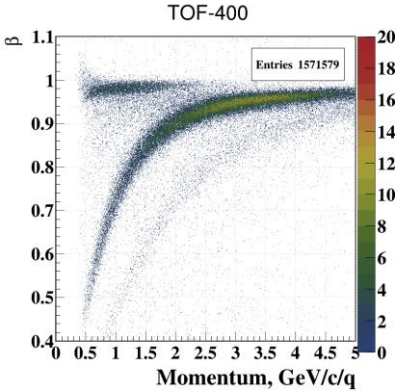
U-tracks - input angle to the magnetic field region
D-tracks - output angle from the magnetic field region
M-tracks - integral of the magnetic field along the trajectory
Momentum $\frac{p}{q} = \frac{0.3 \cdot \int Bdl}{\alpha_{out} - \alpha_{in}}$





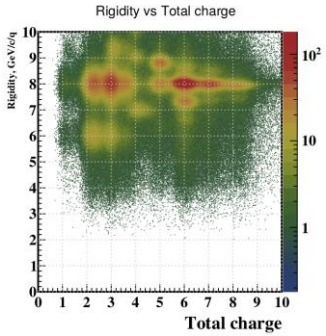
Particles identification

Quality test of global tracking - particle identification plot (correlation particle momenta/velocity).
10^6 reconstructed events of Ar+Pb collisions.
Bunches for π-mesons, protons and deuterons are seen.



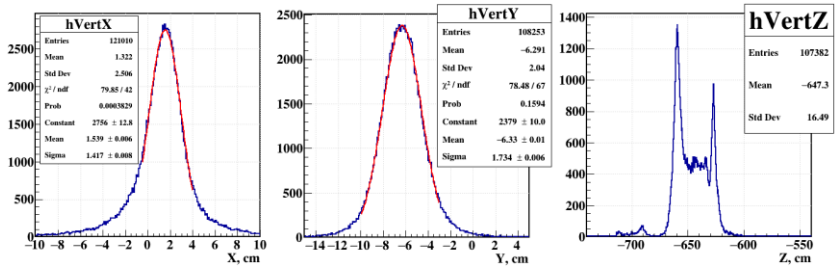
Fragments identification (SRC)

Amplitudes of BC triggers => total charge of event
Momentum + total charge => fragment identification



Primary vertex finder

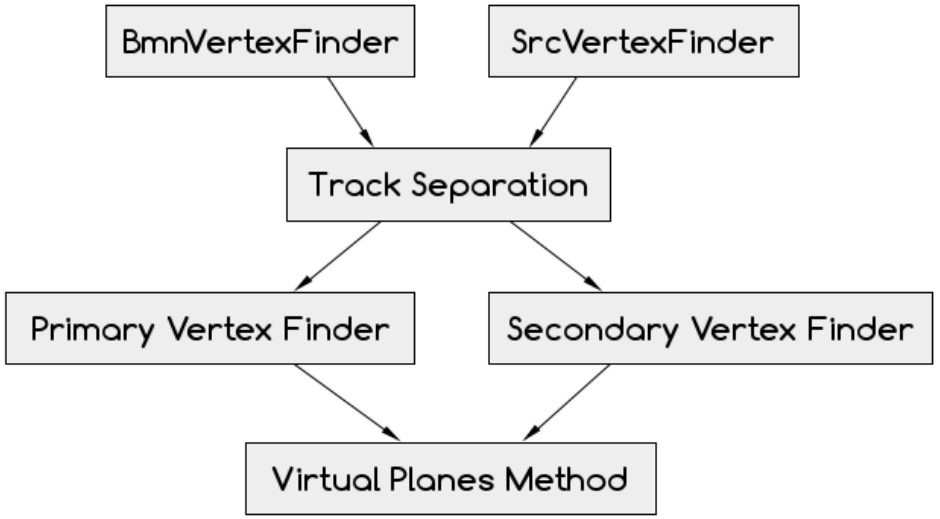
Without A-tracks resolution of primary vertex in Z was approx. 20 cm
Global approach displays structure of target



The global tracking task is integrated into reconstruction chain and implemented into BmnRoot software

Finding of primary and secondary vertices

- ✓ Primary vertex finder is a part of standard event reconstruction chain.
- ✓ Secondary vertices are necessary for decays analysis.



Vertex finder algorithm

Track separation

Extrapolate all tracks to initial approximation of primary vertex position.
Check if track is in "beam region" or not and mark track with corresponding flag.

Primary vertex finder

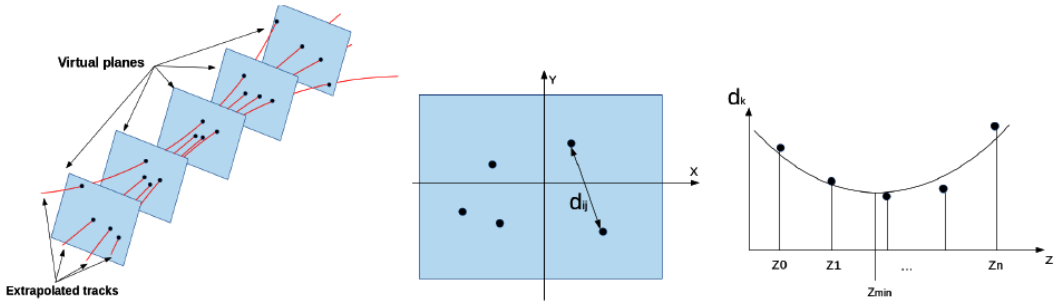
If there are less than 2 tracks marked as primary, return.
Reconstruct primary vertex for tracks marked as primary by virtual planes Method.
If Z position of found vertex is out of range (R), mark tracks as secondary and return.
Extrapolate tracks belonging this vertex to found Z_{PV} and calculate means for X and Y distributions (X_{PV} and Y_{PV})

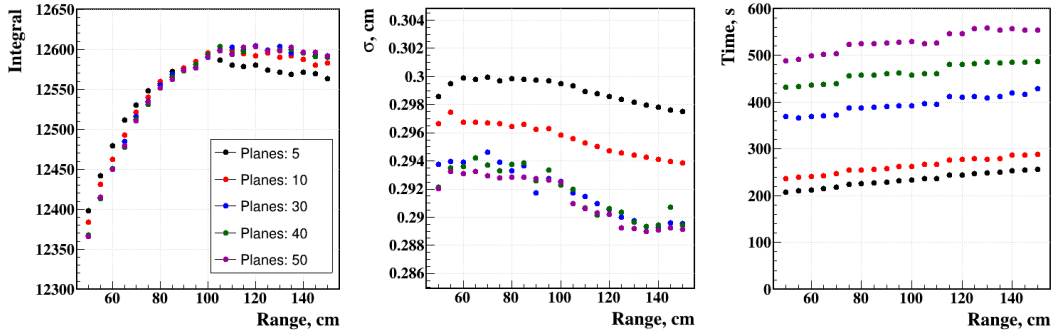
Secondary vertex finder

If there are less than 2 tracks marked as secondary, return
Reconstruct secondary vertex for tracks marked as secondary by virtual planes method in wide range
Extrapolate tracks belonging this vertex to found Z_{SV} and calculate mean for X and Y distributions (X_{SV} and Y_{SV})

Virtual planes method

1. Extrapolate reconstructed tracks to set of $\{z_k\}_0^{N_{planes}}$ by Kalman Filter around initial estimation: $Z_v^{init} - R < z_k < Z_v^{init} + R$
2. Calculate distance between each pair of points on plane k: $d_{ij}^k = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
3. Calculate mean distance for each plane: $d^k = \sum d_{ij}^k / N_{pairs}$
4. Fit $d^k(z_k)$ by parabolic function and find z_{min}
5. Reduce R by factor speed: $R = R/speed$
6. Repeat 1-5 until required accuracy is achieved





Optimization of the algorithm

Input parameters

Range to search primary vertex in (Range).
Number of virtual planes (Planes).
Range reduction rate (Speed).

Control parameters

Number of found vertexes in -3cm < z < 3cm (Integral).
Width of Gaussian fit.
Work time (Time).

Main idea

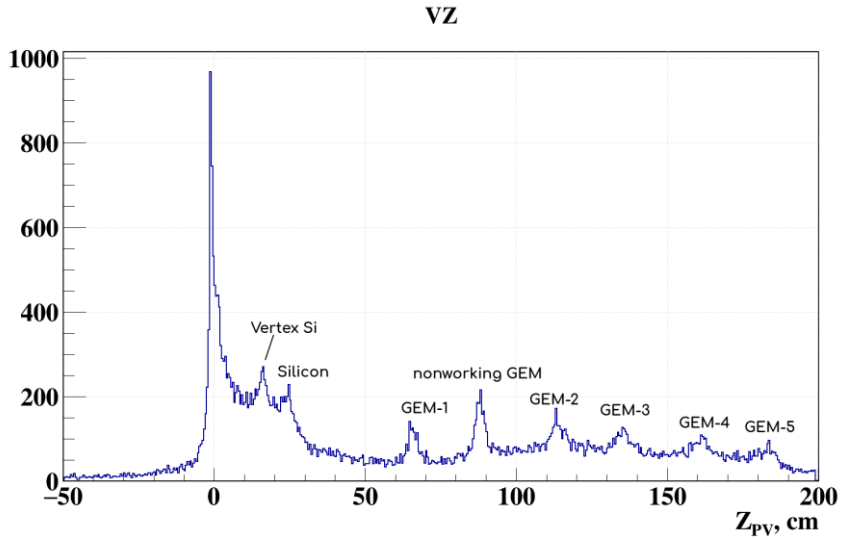
Scan algorithm over input parameters to maximize Integral and minimize and Time.

Output parameters dependencies on number of virtual planes and search range for range reducing speed 1.5
Sample: 10^6 events of Ar+Sn (BD > 3)

BM@N

Targets: C, Al, Cu, Sn, Pb.

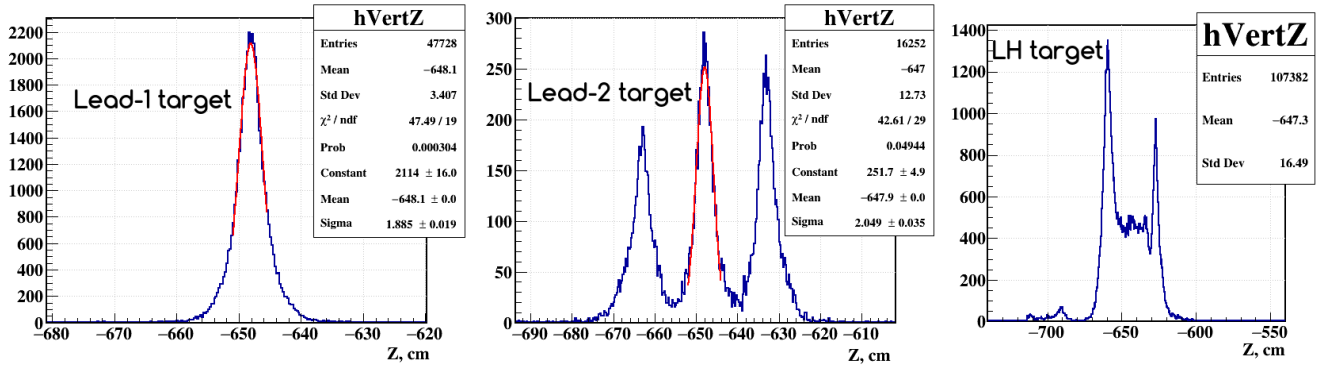
Z distribution of reconstructed vertices for Ar+Sn (BD > 3).



SRC

Targets:

- one lead plane for calibration
- three lead planes for calibration
- liquid hydrogen barrel as a physics target



Quality assurance subsystem of the BmnRoot

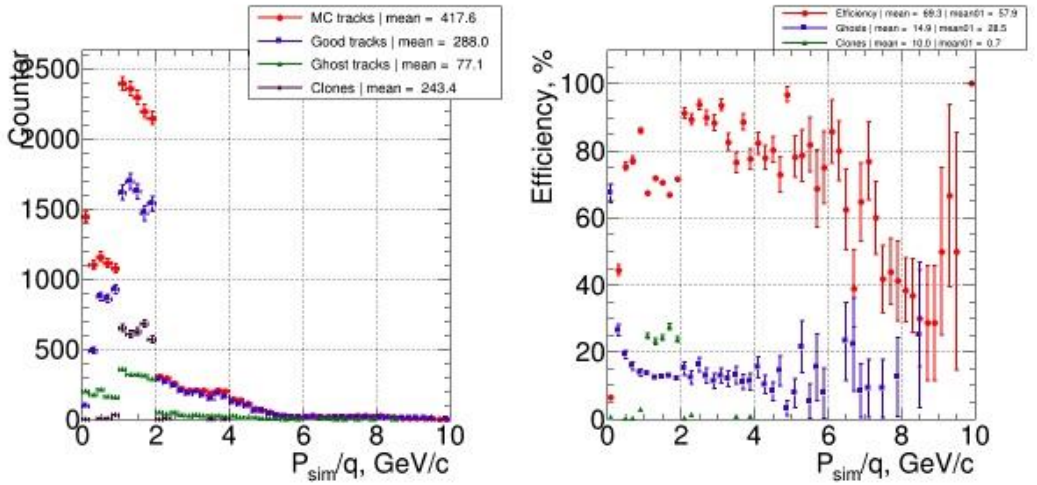
Drawbacks of the previous version of QA subsystem of the BmnRoot

1. Difficult to extract the necessary information.
2. Can't change kinds of axes (for ex. linear to logarithmic scale).
3. Zooming of histograms absent.

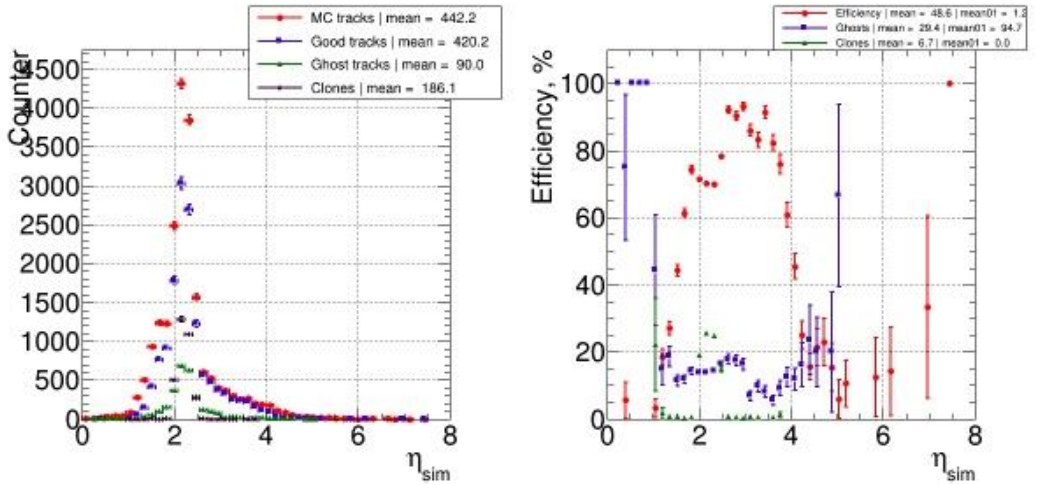
Benefits of the JSROOT:

1. Does not require installation
2. WebUI for users
3. Supports almost all ROOT graphical objects
4. Wide range of tools

tracking_qa: Distribution of MC-tracks, reco-tracks, fakes and clones vs P_sim per event in wide momentum range



tracking_qa: Distribution of MC-tracks, reco-tracks, fakes and clones vs Pseudorapidity per event



System modification tools

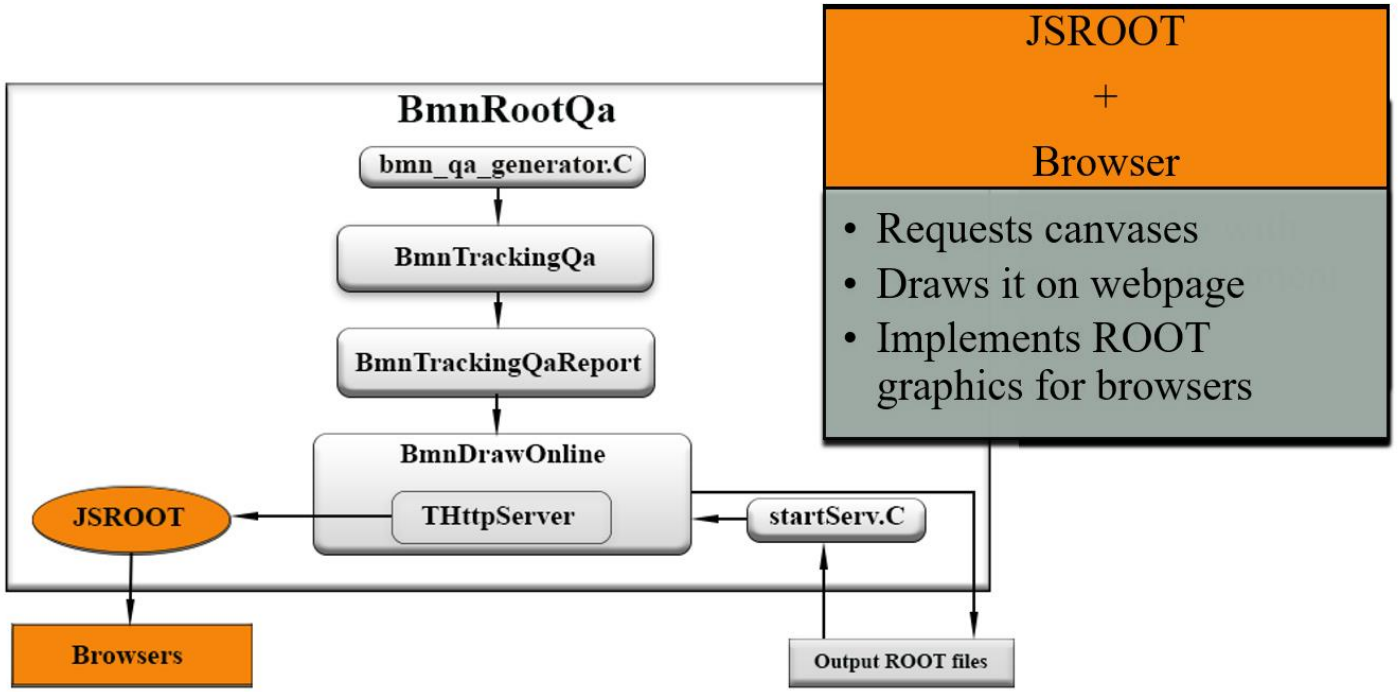
THttpServer class

- Registers objects on server
- Provides http access to them

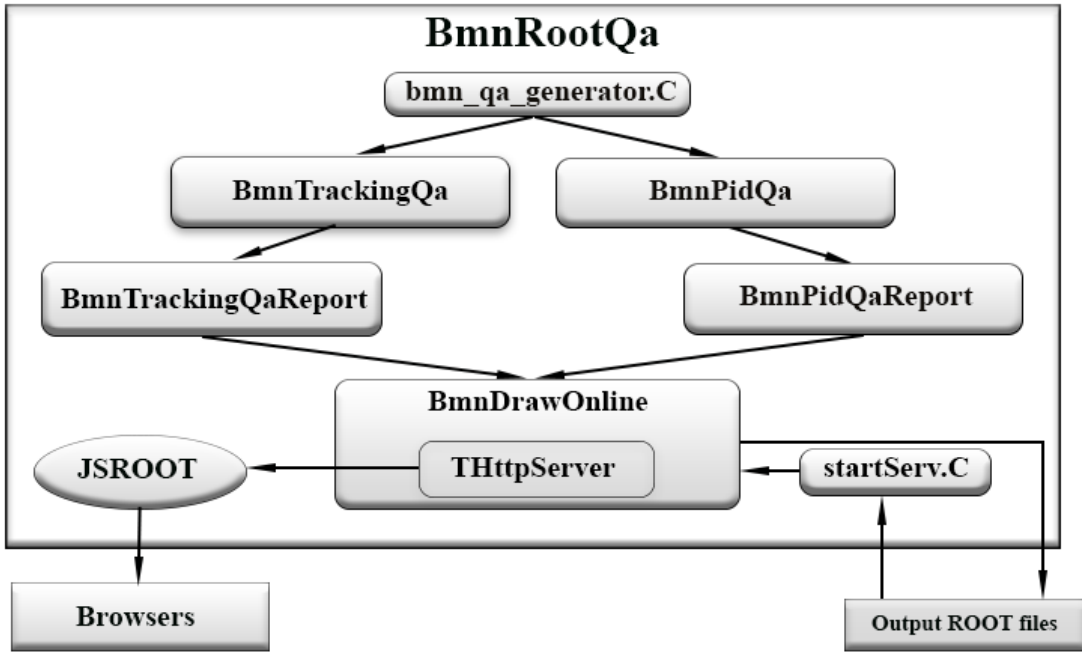
JSROOT library

- Receives object from the server
- Implements interactive ROOT-like graphics in web browsers

Algorithm description



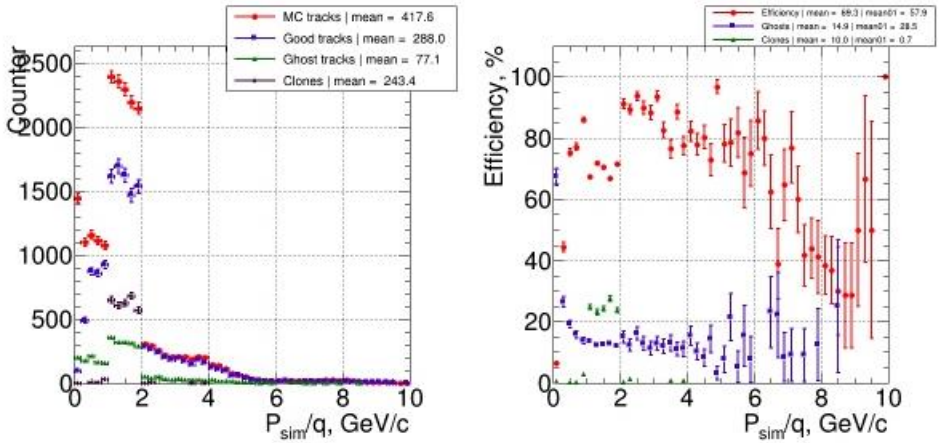
New version is extendable



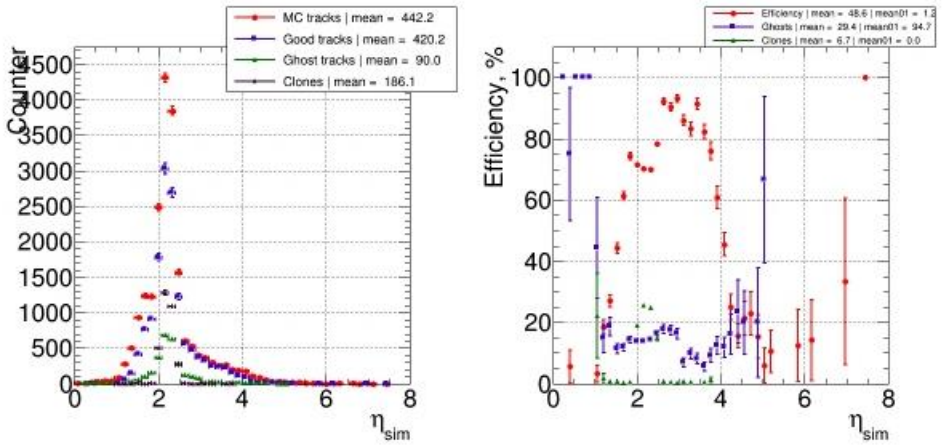
What the modified QA system can

Previous version of the QA system

tracking_qa: Distribution of MC-tracks, reco-tracks, fakes and clones vs P_sim per event in wide momentum range



tracking_qa: Distribution of MC-tracks, reco-tracks, fakes and clones vs Pseudorapidity per event

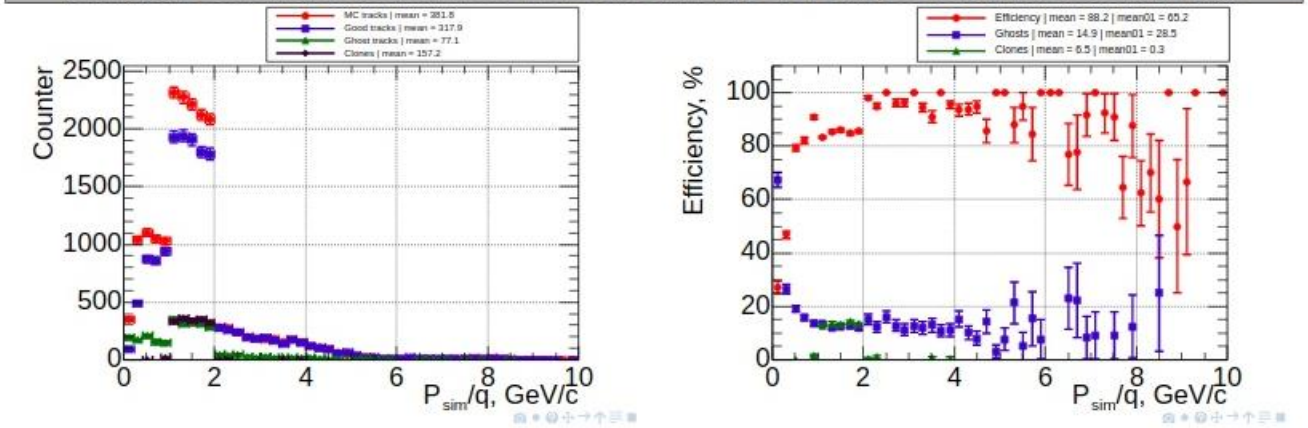


JSROOT

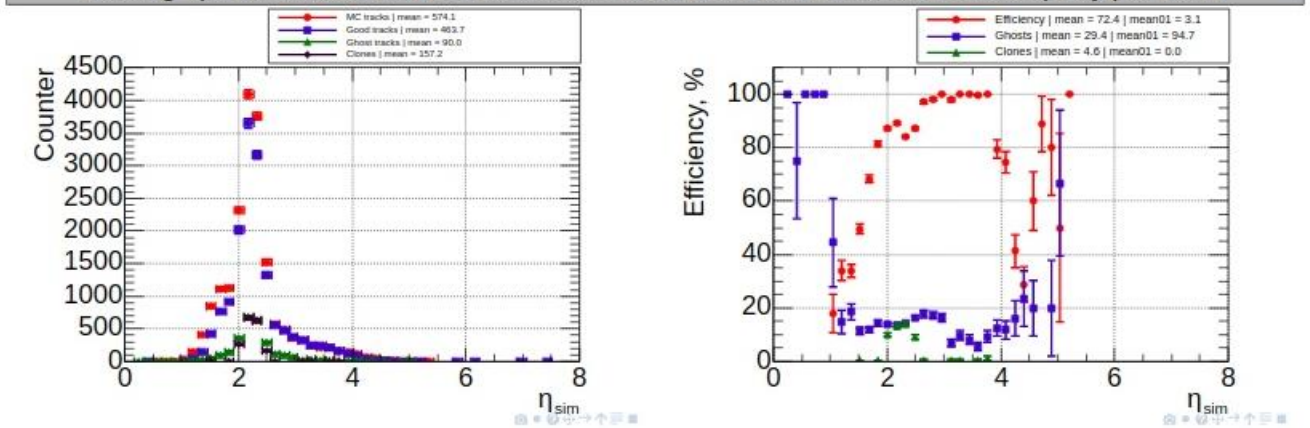
THttpServer

Updated QA system

tracking_qa: Distribution of MC-tracks, reco-tracks, fakes and clones vs P_sim per event in wide momentum range



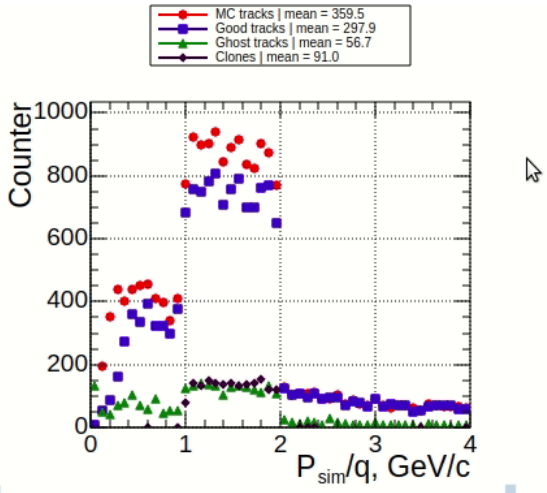
tracking_qa: Distribution of MC-tracks, reco-tracks, fakes and clones vs Pseudorapidity per event



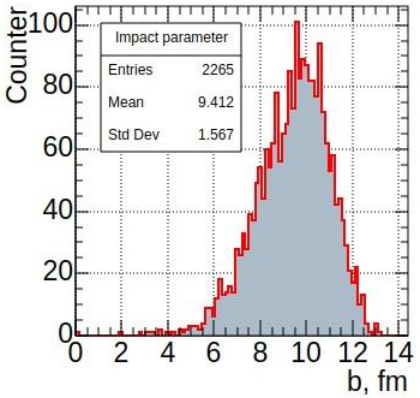
Context menu

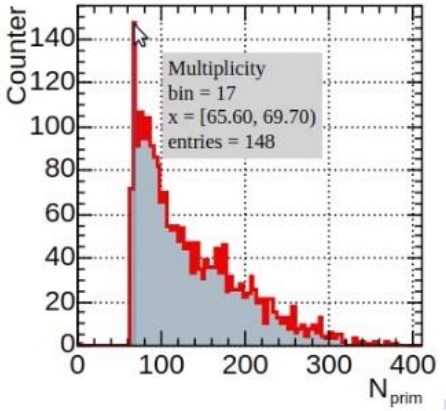
Context menu

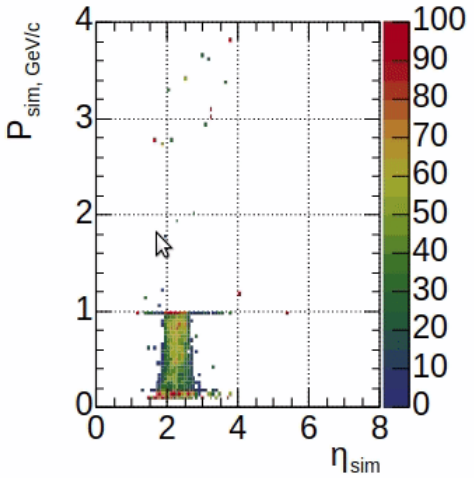
- Can be called for all datasets in a histogram
- Calls the statbox
- Set up a custom scale
- Configures options for displaying histograms

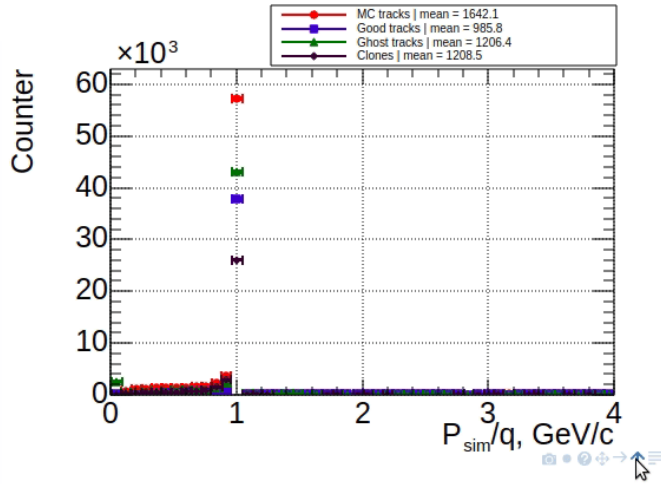


Statbox, Getting a value, Zooming, Lin/log scale

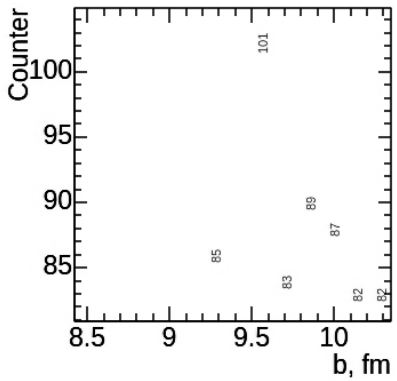


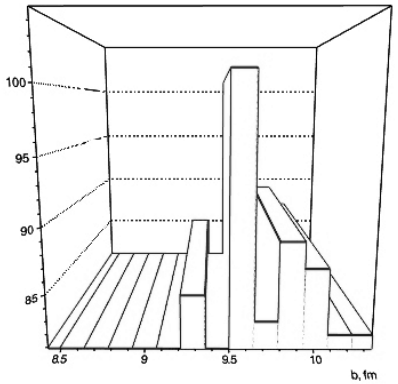






Various kinds of data representation





Machine learning methods of particles identification

Summary

Tracks geometry and time-of-flight data +standard neural network-based ML technique based on a single hidden layer feed-forward fully connected neural network.

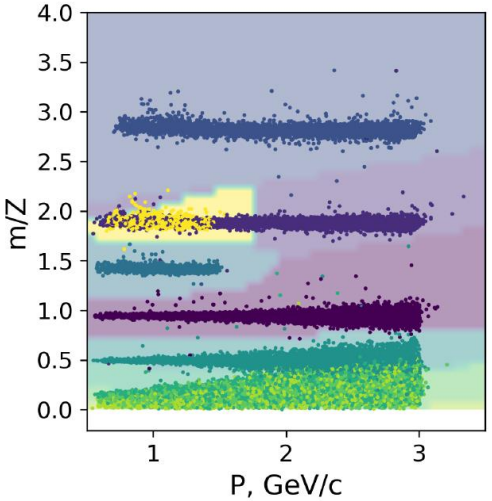
Input data for the neural network: rigidity + mass to charge ratio which are extracted from the particle pass through detector. Data simulated by the BmnRoot framework are processed to mimic some statistical properties of the experimental datasets. Datasets of approximately 10^5 particles have been used for neural network training and testing. Fake tracks have been excluded. Half of the dataset was used for training and the rest one for tests of the neural network. As the loss function the categorical cross-entropy has been chosen.

The light particle trajectories in the given momentum range are close to straight lines so the evaluation of the momentum from the trajectory curvature leads to high variance and errors of estimated values. It can be seen on fig. that light particle data merge into a single cluster. It seems that reliable identification of light particles only on the base of time-of-flight analysis is not possible.

For heavy particles such as p, He or t nuclei separation of particles with small difference of the mass to charge ratio is a problem. It leads to over-contamination - up to 4000%. All particles with rigidity lower than 1.5 GeV/c/q are identified as same kind of particles.

ML methods may be used also to improve simulation of particles transport through BM@N modules. Development of such methods for simulation of avalanches in GEMs is now in progress.

Machine learning based TOF charged particle identification at BM@N detector of NICA collider V A Roudnev *et al* 2020 *J. Phys.: Conf. Ser.* **1479** 012043



Results of simulation and regions highlighted by the classificator

Efficiency of identification and contamination are basic metrics of a neural network.

Identification efficiency for pi-mesons has maximum about 80% at low momentum and falls off as the particle momentum grows. The degree of contamination rises with momentum.

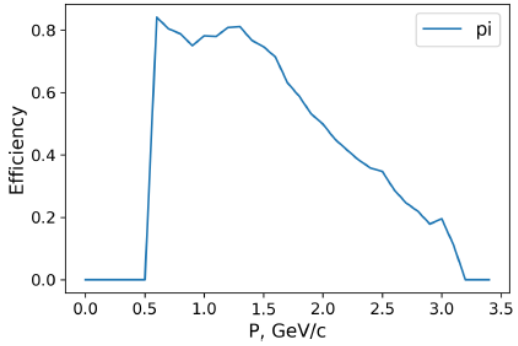
$$E = \frac{N_{id}^{+}}{N_{data}},$$

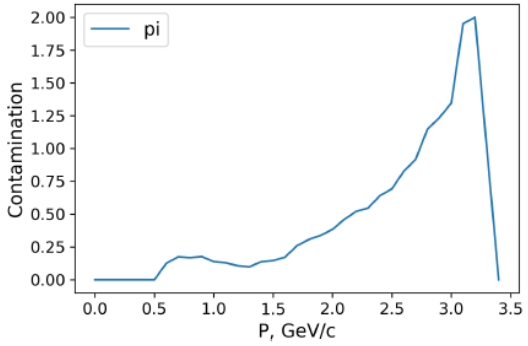
where N_{id}^{+} is the number of correctly identified particles of the given class, N_{data} is the number of particles of the given class in the data set.

$$C = \frac{N_{id}^{-}}{N_{predict}},$$

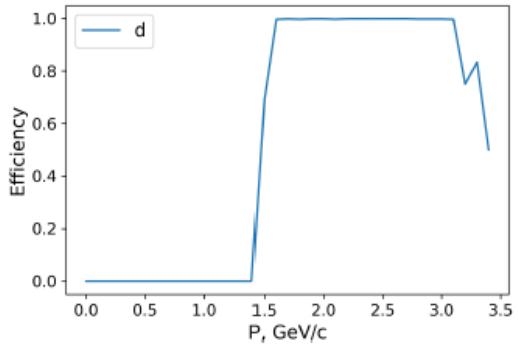
where N_{id}^{-} is the number of the given class particles that were identified incorrectly, and $N_{predict}$ is the number of particles that were identified as belonging to the given class.

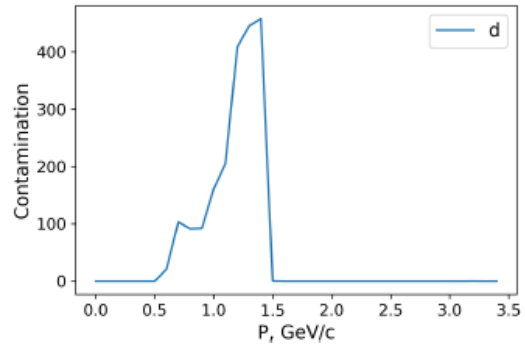
Unlike the light particles, the differentiation between these light nuclei is complicated not by big data inaccuracies only, but also by a very small difference of the mass to charge ratio. This leads to over-contamination (up to 4000 %) of the deuteron channel from α-particles: all the particles with rigidity 1.5 GeV/c/e and lower are identified as α -particles.





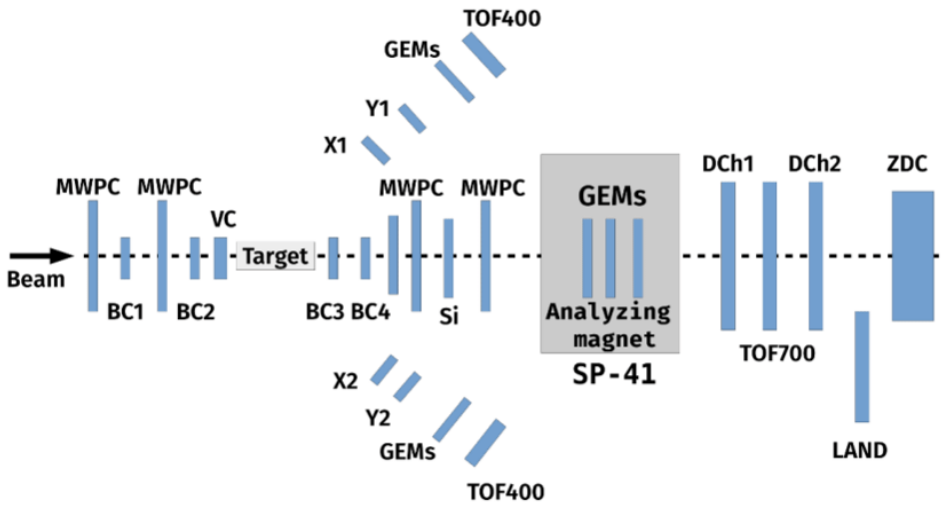
Identification efficiency and contamination for π-mesons





Identification efficiency and contamination for deuterons

Adding triggers description into geometry of SRC



The arm scintillator trigger detectors (X1, Y1, X2, Y2) are important part of the experimental setup. They are used to select events corresponding SRC reaction.

The work consisted of adding geometry of triggers into geometry of the experiment and developing classes for digitization of Monte-Carlo information. When the information from the arm triggers was integrated in reconstruction procedure the experimental and simulation data could be compared correctly, as it provided us with an ability to select the events in which the left or/and the right arm triggers worked.

The SRC (Short Range Correlation) experiment is an extension of the physics program of the BM@N experiment.

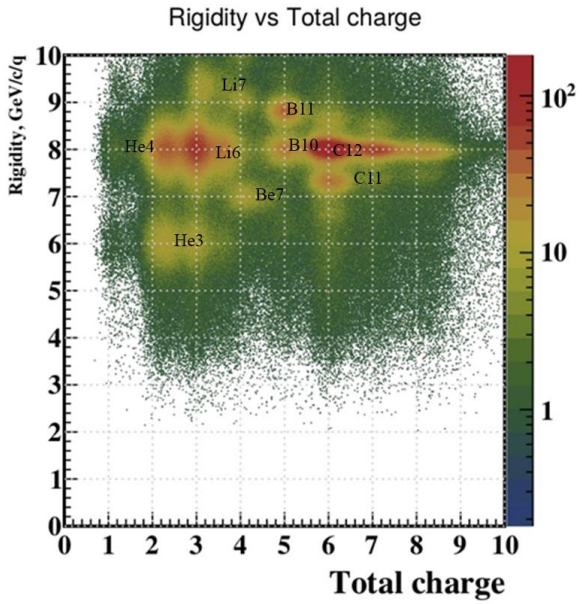
- Multiwire proportional chambers (MWPC).
- Liquid-hydrogen target.
- Beam counters (BC1, BC2, VC, BC3, BC4).
- Scintillation-based trigger detectors in the arms (X1, X2, Y1, Y2).
- Three silicone planes (Si).
- Gas electron multiplier stations (GEMs).
- Drift chamber stations (DCh1, Dch2).
- Time-of-flight detectors (TOF-400 in the arms and TOF-700).
- Zero degree calorimeter.
- Analyzing magnet SP-41.

Carbon beam with energy 4 GeV/c/nucleon + liquid-hydrogen target.

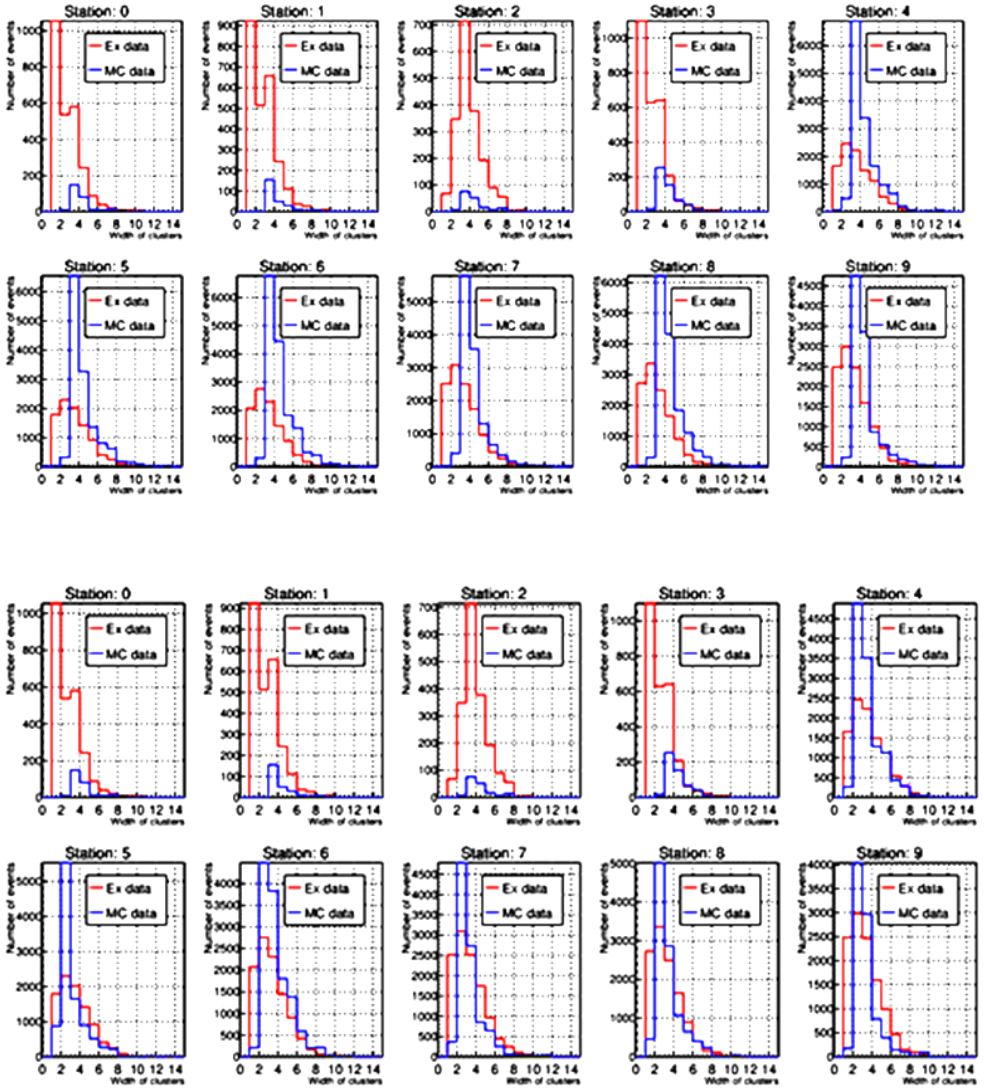
Distributions of momentum of particles, residuals between tracks and hits, cluster widths in GEM stations were compared.

MC data are adopted to the experimental data for the future physical analysis. Clusters of the MC data in GEM stations in magnet (stations 4-9) are wider than clusters of the experimental data.

When the thresholds were set, the widths became almost the same (maximum values corresponds the same value of cluster widths).



With the corrections



Conclusion

Performance study of the BmnRoot framework revealed bottlenecks. Some problems have been resolved.

Algorithm of global tracks reconstruction in the BM@N experiment has been developed and successfully tested. It is included in reconstruction chain of the BM@N. Commited to Github.

Algorithm of finding of primary and secondary vertices based on method of virtual planes is developed and tuned for maximum efficiency. Commited to Github.

Quality assurance subsystem of the BmnRoot framework is modified with significant improvement of its functionality. Commited to Github.

Machine learning methods have been adopted for particles identification and studied for efficiency. Some problems have been revealed.

Triggers description is added in the geometry of SRC. Global tracks reconstruction method have been applied to analysis of fragments.

Thank you!