



Framework for MPD data analysis

Ivonne Maldonado* for the MPD Collaboration

*VBLHEP, JINR ivonne.alicia.maldonado@gmail.com

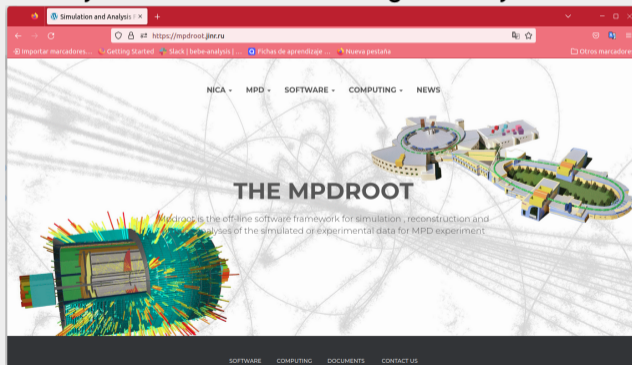
November 9th, 2022

X Collaboration Meeting of the MPD Experiment at the NICA Facility



MpdRoot Framework

It provides a powerful tool for detector performance studies, event simulation, alignment, calibration, visualization, quality assurance and development of algorithms for reconstruction and physics analysis of data of the events registered by the MPD experiment.

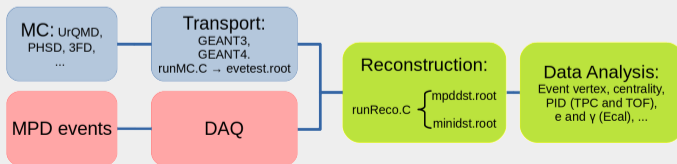


The main page of the mpdroot Framework is <https://mpdroot.jinr.ru>

Chain for Data Analysis: Simulated and real data samples

It is written in C++, it extends the **FairRoot** classes and implement the **FairSoft** packages like ROOT, BOOST and GEANT.

- Interface to MC event generators which model HIC at NICA energies → UrQMD, 3FD, PHSD, PHQMD, QGSM, LAQGSM, DCM-SMM, etc.
- Simulation of the MPD experiment. Particles are propagated through detector material → interact with matter, decay and create additionally particles → GEANT3 and GEANT4.
- Reconstruction of MPD events. Hit reconstruction - Kalman Filter for track reconstruction - Global Tracking - Vertex finding - PID
- Data Analysis. Different analysis require to use the results and recommendations of the different Task Force groups (centrality classes, charged particle identification, electron and photon identification) in order that be consistent and comparable.

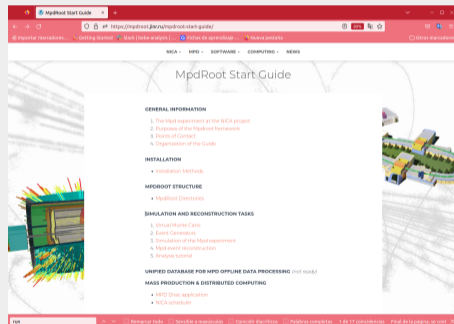


Start Guide

In order to proceed with data analysis, we require training targeting the framework in order to support and consultancy for new users. This is work in progress → Organize and update previous instructions and extend it.

MpdRoot Start Guide

- Easy instructions for Installation
- Description for Transport and Reconstruction macros → updated
- Analysis Tutorial → basic examples for mpddst and minidst files and links to different analysis
- Instructions for mass production within SLURM(Simply Linux utility for Resource Management) or SGE(Sun Grid Engine)



Data Analysis

The available outputs are:

- **mpddst files.** The mpdsim Tree contains the different branches: EventHeader, TpcKalmanTrack, Vertex, FfdHit, TOFHit, TOFMatching, ZdcDigj, MCEventHeader, MCTrack, MPDEvent.
- **minidst files.** The MiniDst Tree contains the different branches: Event, Track, BToFHit, BToFPidTraits, BECalCluster, TrackCovMatrix, FHCalHit, McEvent, McTrack.

Now! How to analyze it?

Simple macro to read files

Create an analysis Task

Plain root tree PicoDst

There are several examples in [mpdroot/macro/physical_analysis](#) folder, however some are not updated

mpdroot/macro/physical_analysis

Additional macros → should be tested with new mpdroot versions, and provide description and instructions for use it

- AnalDilep.C
- AnalXiFull.C
- Anal_L0_best.C
- Chain.C
- di-lep_MassSpec.C
- phys_RoInvMass.C

For example macro: Anal_L0_best.C, Run in interactive mode:

root

.L Anal_L0_best.C++

AnalL0(0,10)

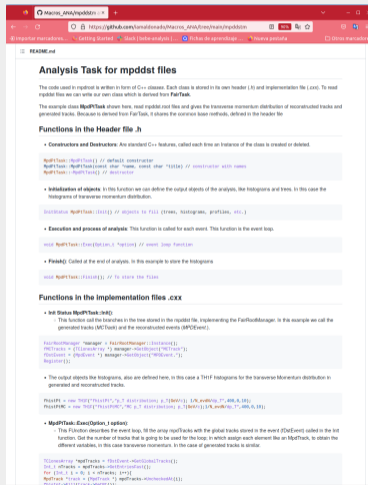
it needs to be updated to run with recent mpdroot version

FairMCTrack → MpdMCTrack

Simple Analysis Task

It requires **dev mpdroot** branch.

- Easy example for mpddst and minidst files, also available in [Analysis Tutorial](#).
- Classes are derived from FairTask, it requires header (.h) and implementation file (.cxx)
- Macro for run the analysis



Analysis Task for mpddst files

The code used in mpdroot is written in form of C++ classes. Each class is stored in its own header (.h) and implementation file (.cxx). To read mpddst files we can write our own class which is derived from FairTask.

The example class `MpdTask` shown here, read mpddst root files and gives the transverse momentum distribution of reconstructed tracks and generated tracks. Because it is derived from FairTask, it shares the common base methods, defined in the header file.

Functions in the Header file .h

- **Constructors and Destructors:** An standard C++ features, called each time an instance of the class is created or deleted.

```
MpdTask:~MpdTask() // default constructor
MpdTask:~MpdTask(const char *name, const char *file) // constructor with name
MpdTask:~MpdTask() // destructor
```

- **Initialization of objects:** In this function we can define the output objects of the analysis, the histograms and trees. In this case the histograms of transverse momentum distribution.

```
void MpdTask::Init() // objects to fill (trees, histograms, profiles, etc.)
```

- **Execution and process of analysis:** This function is called for each event. This function is the event loop.

```
void MpdTask::DoEvent(const T *event) // event loop function
```

- **Finish:** Called at the end of analysis. In this example to store the histograms

```
void MpdTask::Finish() // to store the files
```

Functions in the implementation files .cxx

- **Init Status MpdTask:Init()**
 - This function call the methods in the tree stored in the mpddst file, implementing the FairFileHandler, in this example we call the generated tracks (MCTTrack) and the reconstructed events (MPEvent).

```
FairEventManager *manager = FairEventManager::GetInstance();
MCTTrack *t = (MCTTrack *) manager->GetObject("MCTTrack");
MPEvent *e = (MPEvent *) manager->GetObject("MPEvent");
Register();
```

- The output objects the histograms, also are defined here, in this case a TH1F histograms for the transverse momentum distribution in generated and reconstructed tracks.

```
TH1F * h = new TH1F("h1", "p_T distribution", 5, 0, 5);
TH1F * h2 = new TH1F("h2", "p_T distribution", 5, 0, 5);
```

- **MpdTask:EventOption, 1 option:**
 - This function describes the event loop, fill the array mpdTracks with the global tracks stored in the event (MPEvent) called in the Init function. Get the number of tracks that is going to be used for the loop, in which assign each element like an MpdTrack, to obtain the different variables, in this case transverse momentum. In the case of generated tracks is simple.

```
void MpdTask::MpdTracks = (MPEvent *event) {
    int nTracks = mpdTracks->GetNof();
    for (int i = 0; i < nTracks; i++)
        MpdTrack *track = (MpdTrack *) mpdTracks->GetObject(i);
}
```


Analysis Tasks

Another possibility is the use of MpdAnalysisTask with the MpdAnalysisManager framework in `mpdroot/physics` folder, it requires to be tested on new mpdroot version. Additional modules in same directory are:

- ebye
- femto
- nicafemto
- photons

Plain root tree PicoDst

- Flow analysis uses this format
- Description how to implement it, for example:
`mpdroot/macro/physical_analysis/Flow`

Its advantage is its format, a plain ROOT TREE which allows to use them on any system.

Summary and Final Comments

- MPDroot framework for data analysis is under development.
- Reconstructed Data for analysis is available in two formats: mpddst and minidst files.
- There are different approaches to carry out analysis: Simple macro, analysis task and PicoDst.
- Different groups uses different approaches → They should migrate to same method, format?
- We require to perform different analysis with both (mpddst and minidst) and compare results.
- Several physical macros on mpdroot requires to be updated to run with the most recent mpdroot version.
- We require organized documentation to be easy to use for new users and they can be capable to contribute.

The different ways to analyze data should be integrated in order to make sure that all analyses are performed in a consistent way and can be compared.

GRACIAS