Analysis framework

V. Riabov

Current situation

- Many ongoing physics feasibility studies: trigger efficiency and event centrality, particle spectra and correlations, collective flow, photons and (di)electrons, heavy-flavor signals, etc.
- Analyzers use different codes and approaches to access and process data and produce final results

- Starting from 2022, we are moving to analysis of big MC data samples (~ 20-50M events each) comparable in size to those expected with the first beams → test of our capabilities, tuning of analysis and calibration procedures
- The previous approach of data handling will not be efficient (may not be possible) with the data sets

- Proposal is to move to a centralized Analysis Framework:
  - ✓ all analysis codes are saved (archived) in the MpdRoot → easier sharing of codes and methods
  - ✓ all analyses codes have a similar structure → easier reading of codes, cross checks
  - ✓ all analyses use the same global variables for centrality, T0, z-vertex, reaction plane, n-sigma matching for tracks to external detectors, etc. (input from TaskForces) → consistent approach
  - ✓ analyses are easily grouped in a train, analyses are run simultaneously with a single access to data for all of them → reduced number of input/output operations for disks and databases, easier organization of data storage (tape-to-disk-to-recyclebin cycles)

Analysis Framework

- Analysis manager reads event into memory and calls wagons one-by-one to modify and/or analyze data:

Analysis manager | wagon #1 | wagon #2 | wagon #3 | wagon #4 | wagon #5 | wagon #6

- Example:
  - ✓ Wagon #1 – centrality analyzer – returns values of centrality (TPC, FHCAL, E_T), to be used by all other wagons in the train
  - ✓ Wagon #2 – T_0 analyzer – returns value of T_0 (FFD, TOF), to be used by all wagons in the train
  - ✓ Wagon #3 – recalibrator – redefines some DST variables that need recalibration after production
  - ✓ ………………………..
  - ✓ Wagon #4 – physics analysis 1
  - ✓ Wagon #5 – physics analysis 2
  - ✓ Wagon #6 – physics analysis 3

Realization of Analysis Framework

- Example codes are available in MpdRoot @ mpdroot/physics, originally committed by D. Peresunko:

  MpdAnalysisEvent.cxx MpdAnalysisManager.cxx MpdAnalysisTask.cxx
  MpdAnalysisEvent.h MpdAnalysisManager.h MpdAnalysisTask.h

- Class **MpdAnalysisManager** requires list of input files, list of branches to be used for analysis and list of tasks (wagons) to process. In the end, **MpdAnalysisManager** takes care of writing output objects for each task (wagon)

- Tasks, which are called by **MpdAnalysisManager** should be derived from **MpdAnalysisTask** and have several methods implemented:
  - ✓ void **UserInit()**; // Users should prepare objects to fill (histograms, trees, etc.)
  - ✓ void **ProcessEvent(MpdAnalysisEvent &event)** ; // method is called for each event, data are provided by container **MpdAnalysisEvent**
  - ✓ void **Finish()**; //method is called when scan in finished but class data are not written yet

- Class **MpdAnalysisEvent** contains references to all branched in the DST file for this event and may contain extra global variables of interest (centrality, event plane, T_0, n-sigma matching variables for tracks, etc.)

Example

- Code is available in the MpdRoot
- Classes **MpdAnalysisEvent**, **MpdAnalysisTask**, **MpdAnalysisManager** are defined in mpdroot/physics
- Analysis tasks(wagons) are defined in:
  - ✓ mpdroot/physics/evCentrality – MpdCentralityAll task, event centrality (centralityTPC in **MpdAnalysisEvent**)
  - ✓ mpdroot/physics/photons – MpdConvPi0 task, pi0 with ECAL-ECAL, ECAL-PCM, PCM-PCM
  - ✓ mpdroot/physics/pairKK – MpdPairKKtask, phi->KK using centrality from MpdCentralityAll wagon

- How to run:
  - ✓ go to physics/pairKK/macros
  - ✓ run 'root –b –q RunAnalyses.C'

```
GNU nano 2.3.1                    File: RunAnalyses.C

void RunAnalyses(){

  gROOT->LoadMacro("mpdloadlibs.C");
  gROOT->ProcessLine("mpdloadlibs()");

   MpdAnalysisManager man("ManagerAnal") ;
   man.InputFileList("list.txt") ;
   man.ReadBranches("*") ;
   man.SetOutput("histos.root") ;

   MpdCentralityAll pCentr("pCentr","pCentr") ;
   man.AddTask(&pCentr) ;

   MpdConvPi0 pDef("pi0Def","ConvDef") ; //name, parametes file
   man.AddTask(&pDef) ;

   MpdPairKK pKK("pKK","pKK") ;
   man.AddTask(&pKK) ;

   man.Process() ;

}
```

Text file with a list of input DST files

Wagon # 1 - centrality

Wagon # 2 – Pi0 by ECAL and PCM

Wagon # 3 – phi -> KK

Example code

- Code is available in the MpdRoot
- Classes **MpdAnalysisEvent**, **MpdAnalysisTask**, **MpdAnalysisManager** are defined in mpdroot/physics
- Analysis tasks(wagons) are defined in:
  - ✓ mpdroot/physics/evCentrality – defines event centrality (centralityTPC in class **MpdAnalysisEvent**)
  - ✓ mpdroot/physics/photons – runs analysis for pi0 with ECAL-ECAL, ECAL-PCM, PCM-PCM options
  - ✓ mpdroot/physics/pairKK – runs analysis for phi->KK using centrality from evCentrality wagon

- How to run:
  - ✓ go to physics/pairKK/macros
  - ✓ run ‘root –b –q RunAnalyses.C’

```
GNU nano 2.3.1                    File: RunAnalyses.C

void RunAnalyses(){

  gROOT->LoadMacro("mpdloadlibs.C");
  gROOT->ProcessLine("mpdloadlibs()");

   MpdAnalysisManager man("ManagerAnal") ;
   man.InputFileList("list.txt") ;
   man.ReadBranches("*") ;
   man.SetOutput("histos.root") ;

   MpdCentralityAll pCentr("pCentr","pCentr") ;
   man.AddTask(&pCentr) ;

   MpdConvPi0 pDef("pi0Def","ConvDef") ; //name, parametes file
   man.AddTask(&pDef) ;

   MpdPairKK pKK("pKK","pKK") ;
   man.AddTask(&pKK) ;

   man.Process() ;

}
```

wagon name

wagon file name: name.txt – input, name.root - output

Wagon # 1 - centrality

Wagon # 2 – Pi0 by ECAL and PCM

Wagon # 3 – phi -> KK

Example of pairKK wagon

- Input file pKK.txt

```
 GNU nano 2.3.1                         File: pKK.txt

#-------Parameters used for analysis------
# Event selection:
mZvtxCut 100 //  cut on vertex z coordinate
mNhitsCut 10 //  number of hits in TPC tracks used for centrality
# PID cuts:
mPIDsigTPC   2  // dEdx PID parameters
mPIDsigTOF   2  // dEdx PID parameters
mNofHitsCut  10  // minimal number of hits to accept track
mEtaCut      1  // maximal pseudorapidity accepted
mPtminCut   0.05  // minimal pt used in analysis
```

- If input file is not provided then variable definitions are used from mpdroot/physics/pairKK/MpdPairKKParams.h

pairKK wagon: UserInit()

- mpdroot/physics/pairKK/MpdPairKK.cxx

```
void MpdPairKK::UserInit()
{

   mParams.ReadFromFile(mParamConfig);
   mParams.Print();

   // Prepare histograms etc.
   fOutputList = new TList();
   fOutputList->SetOwner(kTRUE);

   TH1::AddDirectory(kFALSE); // sets a global switch disabling the reference to histos in gROOT and their overwriting

   // General QA
   mhEvents = new TH1F("hEvents", "Number of events", 10, 0., 10.);
   fOutputList->Add(mhEvents);
   mhVertex = new TH1F("hVertex", "Event vertex distribution", 100, -200., 200.);
   fOutputList->Add(mhVertex);
   mhCentrality = new TH1F("hCentrality", "Centrality distribution", 100, 0., 100.);
   fOutputList->Add(mhCentrality);
   mhMultiplicity = new TH1F("hMultiplicity", "Multiplicity distribution", 2000, -0.5, 1999.5);
   fOutputList->Add(mhMultiplicity);

   mInvGen = new TH1F("mInvGen", "mInvGen", 100, 0., 10.);
   fOutputList->Add(mInvGen);

  ................................

   //MC
   if (isMC) {
      mInvTrueNoPID = new TH2F("mInvTrueNoPID", "mInvTrueNoPID", 100, 0., 10., 200, 0.9, 2.0);
      fOutputList->Add(mInvTrueNoPID);
      mInvTrueNoPIDPhi = new TH2F("mInvTrueNoPIDPhi", "mInvTrueNoPIDPhi", 100, 0., 10., 200, 0.9, 2.0);
      fOutputList->Add(mInvTrueNoPIDPhi);
      mInvTrueOnePID = new TH2F("mInvTrueOnePID", "mInvTrueOnePID", 100, 0., 10., 200, 0.9, 2.0);
      fOutputList->Add(mInvTrueOnePID);
      mInvTrueOnePIDPhi = new TH2F("mInvTrueOnePIDPhi", "mInvTrueOnePIDPhi", 100, 0., 10., 200, 0.9, 2.0);
      fOutputList->Add(mInvTrueOnePIDPhi);
      mInvTrueTwoPID = new TH2F("mInvTrueTwoPID", "mInvTrueTwoPID", 100, 0., 10., 200, 0.9, 2.0);
      fOutputList->Add(mInvTrueTwoPID);
      mInvTrueTwoPIDPhi = new TH2F("mInvTrueTwoPIDPhi", "mInvTrueTwoPIDPhi", 100, 0., 10., 200, 0.9, 2.0);
      fOutputList->Add(mInvTrueTwoPIDPhi);
   }

   for (long int i = 0; i < nMixTot; i++) {
     mixedEvents[i] = new TList() ;
   }

}
```

pairKK wagon: ProcessEvent()

- mpdroot/physics/pairKK/MpdPairKK.cxx

```
void MpdPairKK::ProcessEvent(MpdAnalysisEvent &event)
{

   if (!isInitialized) {
      mKF = MpdKalmanFilter::Instance();
      mKHit.SetType(MpdKalmanHit::kFixedR);
      isInitialized = true;
   }

   if (!selectEvent(event)) { //(V)
      return;
   }

   //mEMCClusters  = event.fEMCCluster;
   mKalmanTracks = event.fTPCKalmanTrack;
   if (isMC) {
     mMCTracks = event.fMCTrack;

     for (int i = 0; i < mMCTracks->GetEntriesFast(); i++) {
       MpdMCTrack *pr = (static_cast<MpdMCTrack *>(mMCTracks->At(i)));
       if (pr->GetPdgCode() == 333) {
         if (pr->GetStartX() * pr->GetStartX() + pr->GetStartY() * pr->GetStartY() < 1.) {
           TVector3 momentum;
           pr->GetMomentum(momentum);
           mInvGen->Fill(momentum.Pt());
         }
       }
     }
   }//isMC

   selectPosTrack(event);

   selectNegTrack(event);

   processHistograms(event);
}
```

pairKK wagon: Finish()

- mpdroot/physics/pairKK/MpdPairKK.cxx

```
void MpdPairKK::Finish()
{
   // Post-scan processing not needed
}
```

pairKK wagon: output

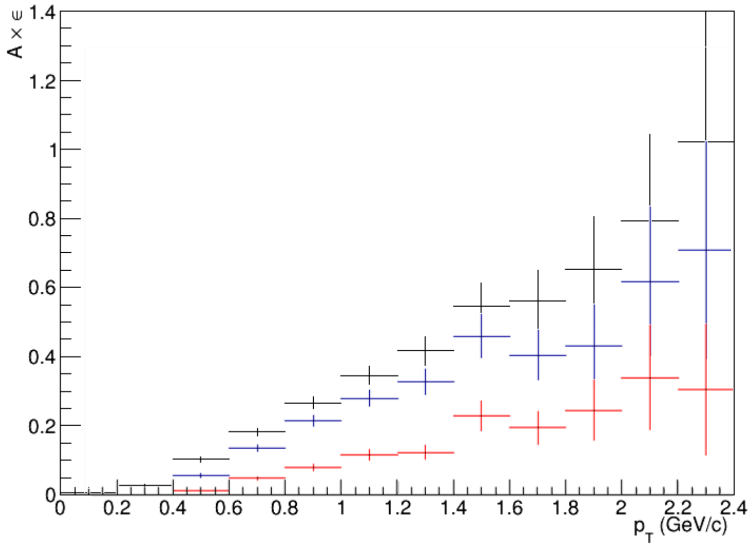
- mpdroot/physics/pairKK/macros/pKK.root

```
root [2] .ls
TFile**         pKK.root
 TFile*         pKK.root
  OBJ: TH2F     mInvTrueTwoPID  mInvTrueTwoPID : 0 at: 043A6BC8
  OBJ: TH2F     mInvTrueTwoPID  mInvTrueTwoPID : 0 at: 044695B0
  OBJ: TH1F     hEvents Number of events : 0 at: 0359AD78
  OBJ: TH1F     hEvents Number of events : 0 at: 03583D80
  KEY: TH1F     hEvents;1       Number of events
  KEY: TH1F     hVertex;1       Event vertex distribution
  KEY: TH1F     hCentrality;1   Centrality distribution
  KEY: TH1F     hMultiplicity;1 Multiplicity distribution
  KEY: TH1F     mInvGen;1       mInvGen
  KEY: TH2F     mInvNoPID;1     mInvNoPID
  KEY: TH2F     mInvOnePID;1    mInvOnePID
  KEY: TH2F     mInvTwoPID;1    mInvTwoPID
  KEY: TH2F     mInvMixNoPID;1  mInvMixNoPID
  KEY: TH2F     mInvMixOnePID;1 mInvMixOnePID
  KEY: TH2F     mInvMixTwoPID;1 mInvMixTwoPID
  KEY: TH2F     mInvTrueNoPID;1 mInvTrueNoPID
  KEY: TH2F     mInvTrueNoPIDPhi;1      mInvTrueNoPIDPhi
  KEY: TH2F     mInvTrueOnePID;1        mInvTrueOnePID
  KEY: TH2F     mInvTrueOnePIDPhi;1     mInvTrueOnePIDPhi
  KEY: TH2F     mInvTrueTwoPID;1        mInvTrueTwoPID
  KEY: TH2F     mInvTrueTwoPIDPhi;1     mInvTrueTwoPIDPhi
root [3]
```

- Contains all objects registered in UserInit()

pairKK wagon: reconstruction efficiecny

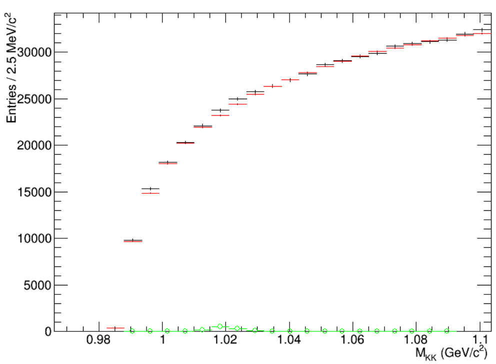
- mpdroot/physics/pairKK/macros/pKK.root
- Receff for phi->KK with **noPID**, **OneKaonPID**, TwoKaonPID:
  - ✓ mInvGen – generated phi mesons, pT-spectrum
  - ✓ mInvTrue**NoPID**Phi – reconstructed phi->KK mesons, M_{inv} vs. p_T
  - ✓ mInvTrue**OnePID**Phi – reconstructed phi->KK mesons, M_{inv} vs. p_T
  - ✓ mInvTrue**TwoPID**Phi – reconstructed phi->KK mesons, M_{inv} vs. p_T

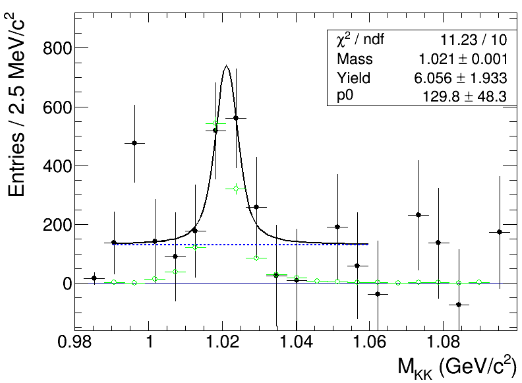


pairKK wagon: Minv spectra

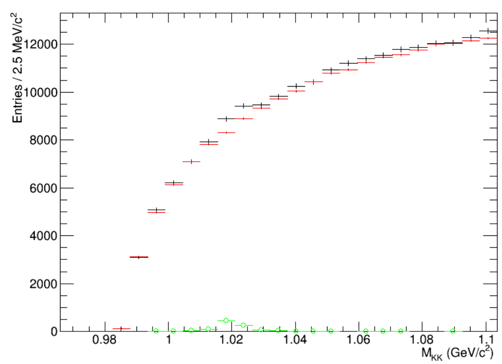
- mpdroot/physics/pairKK/macros/pKK.root
- Minv distributions:
  - ✓ mInv**NoPID,** mInv**OnePID,** mInv**TwoPID,** – foreground M_{inv} vs. p_T distributions
  - ✓ mInvMix**NoPID,** mInvMix**OnePID,** mInvMix**TwoPID,** – mixed-event M_{inv} vs. p_T distributions

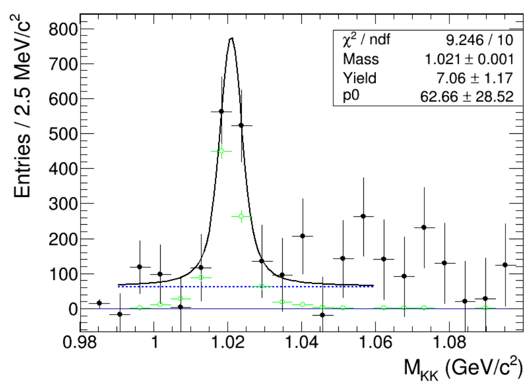
NoPID



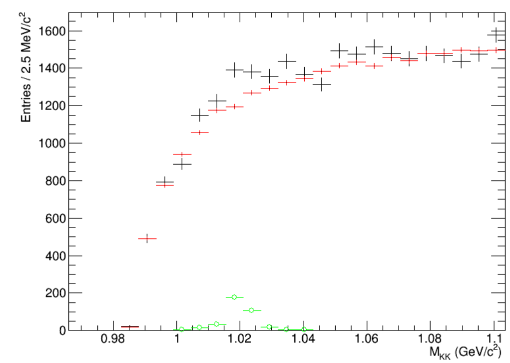


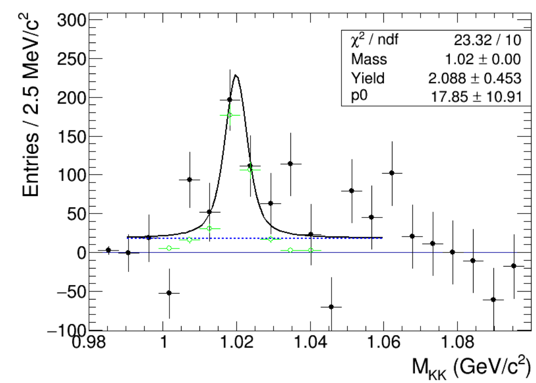
OnePID





TwoPID





Advantages for users

- No risk of losing your code because it's not part of MpdRoot and your hard drive crashes
- Train can contain multiple instances of your analysis, meaning you can run your default configuration and in addition multiple cut variations and other cross-checks. The success rate of the jobs is EXACTLY the same for each of your configurations, as they are all handled in parallel. This means that the all the configurations handle the exact same set of data
- If train is run by a Conductor then there is no headaches with running the jobs (no need to stay up all night to resubmit jobs, no need to think of user quota, etc.)
- No risk of running with wrong global variables (centrality, T0, reaction plane, etc.)
- No need to understand how global variables are calculated to get the correct results
- …….

Conclusions

- Please have a look at the examples and let us know if you have any problems with the approach
- If this approach works for most of us then we will stick to it as a default option of running analyses