ROOT and Analysis framework

Sourav Kundu
CERN
Email:sourav.kundu@cern.ch

IV ALICE-India School on Quark-Gluon Plasma

Bose institute, India: 8/11/2021 - 20/11/2021

Recap: Day2

- What we have learned?
- Write a skeleton for analysis task
- Access event
- Access track in a given event

```
 void AliAnalysisTaskMyTask::UserExec(Option_t *)
{
fAOD = dynamic_cast<AliAODEvent*>(InputEvent());
//fESD= dynamic_cast<AliESDEvent*>(InputEvent());
if(!fAOD) return;
//if(!fESD) return;
Int_t iTracks(fAOD->GetNumberOfTracks());
//Int_t iTracks(fESD->GetNumberOfTracks());
for(Int_t i(0); i < iTracks; i++)
 {
AliAODTrack* track = static_cast<AliAODTrack*>(fAOD->GetTrack(i));
//AliESDTrack* track = static_cast<AliESDTrack*>(fESD->GetTrack(i));
fHistPt->Fill(track->Pt());
}
PostData(1, fOutputList);
}
```

Recap: Day3

- Add some event selection cut to select good events

```
 void AliAnalysisTaskMyTask::UserExec(Option_t *)
{
fAOD = dynamic_cast<AliAODEvent*>(InputEvent());
//fESD= dynamic_cast<AliESDEvent*>(InputEvent());
if(!fAOD) return;
//if(!fESD) return;
Bool_t EventAccepted;
        EventAccepted = fEventCuts.AcceptEvent(fAOD);
        if (!EventAccepted)
        {
           PostData(1,fOutputList);
           return;
        }
AliVVertex* vtx = event->GetPrimaryVertex();
Double vtxz=vtx->GetZ();
Int_t iTracks(fAOD->GetNumberOfTracks());
//Int_t iTracks(fESD->GetNumberOfTracks());
for(Int_t i(0); i < iTracks; i++)
 {
AliAODTrack* track = static_cast<AliAODTrack*>(fAOD->GetTrack(i));
//AliESDTrack* track = static_cast<AliESDTrack*>(fESD->GetTrack(i));
fHistPt->Fill(track->Pt());
}
PostData(1, fOutputList);
}
```

Define in header file

```
#include "AliEventCuts.h"

AliEventCuts fEventCuts; //!
```

Today

- Add some event selection cut to select good events

```
 void AliAnalysisTaskMyTask::UserExec(Option_t *)
{
fAOD = dynamic_cast<AliAODEvent*>(InputEvent());
//fESD= dynamic_cast<AliESDEvent*>(InputEvent());
if(!fAOD) return;
//if(!fESD) return;
Bool_t EventAccepted;
        EventAccepted = fEventCuts.AcceptEvent(fAOD);
        if (!EventAccepted)
        {
           PostData(1,fOutputList);
           return;
        }
AliVVertex* vtx = event->GetPrimaryVertex();
Double vtxz=vtx->GetZ();
Int_t iTracks(fAOD->GetNumberOfTracks());
//Int_t iTracks(fESD->GetNumberOfTracks());
for(Int_t i(0); i < iTracks; i++)
 {
AliAODTrack* track = static_cast<AliAODTrack*>(fAOD->GetTrack(i));
//AliESDTrack* track = static_cast<AliESDTrack*>(fESD->GetTrack(i));
Apply Track selection
fHistPt->Fill(track->Pt());
}
PostData(1, fOutputList);
}
```

Define in header file

```
#include "AliEventCuts.h"

AliEventCuts fEventCuts; //!
```

ESD track selection: AliESDtrackCuts

- There are default cut sets available which are a very good starting point for all analyses (however, there is not a default cut set which is perfect for every analysis)

- Implementation:
  AnalysisMyTask.h
  …
  private:
  AliESDtrackCuts *fESDtrackCuts;//!
  ….

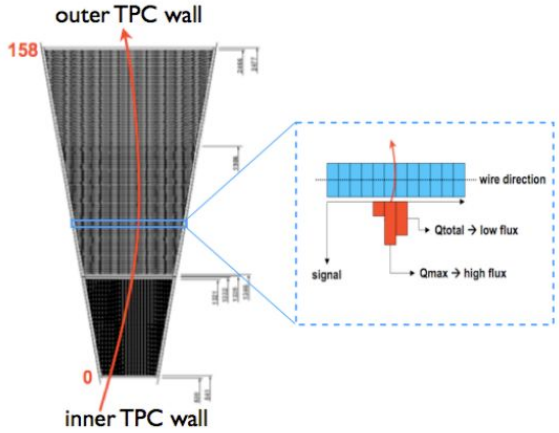
ESD track selection: AliESDtrackCuts

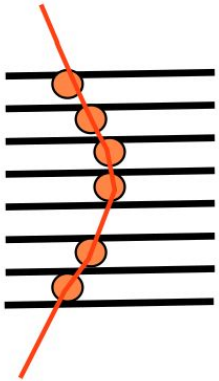
- Implementation:
  AnalysisMyTask.cxx
  UserCreateOutputObjects()
  fESDtrackCuts =
  AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(kTRUE,1);//kTRUE for default and kFALSE for manual change, 0 for cluster cut 1 for crossed row cut
  fESDtrackCuts->SetPtRange(0.35,2.0);
  fESDtrackCuts->SetEtaRange(-0.8,0.8);

  UserExec()
  if(!fESDtrackCuts->AcceptTrack(esdt)) continue;

AliESDtrackCuts:Step-By-Step

```
AliESDtrackCuts* AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(Bool_t selPrimaries, Int_t clusterCut)
{
  /// creates an AliESDtrackCuts object and fills it with standard values for ITS-TPC cuts for pp 2011 data
  /// if clusterCut = 1, the cut on the number of clusters is replaced by
  /// a cut on the number of crossed rows and on the ration crossed
  /// rows/findable clusters

  AliInfoClass("Creating track cuts for ITS+TPC (2011 definition).");

  AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;

  // TPC
  if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
  else if (clusterCut == 1) {
    esdTrackCuts->SetMinNCrossedRowsTPC(70);
    esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
  }
  else {
    AliWarningClass(Form("Wrong value of the clusterCut parameter (%d), using cut on Nclusters",clusterCut));
    esdTrackCuts->SetMinNClustersTPC(50);
  }
  esdTrackCuts->SetMaxChi2PerClusterTPC(4);
  esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
  esdTrackCuts->SetRequireTPCRefit(kTRUE);
  // ITS
  esdTrackCuts->SetRequireITSRefit(kTRUE);
  esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD,
                                         AliESDtrackCuts::kAny);
  if(selPrimaries) {
    // 7*(0.0015+0.0050/pt^1.1)
    esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
    esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
  }
  esdTrackCuts->SetMaxDCAToVertexZ(2);
  esdTrackCuts->SetDCAToVertex2D(kFALSE);
  esdTrackCuts->SetRequireSigmaToVertex(kFALSE);

  esdTrackCuts->SetMaxChi2PerClusterITS(36);

  return esdTrackCuts;
}
```

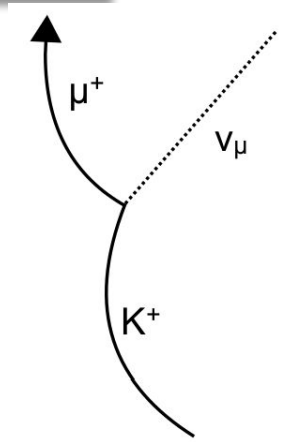




6 clusters / 7 crossed rows

AliESDtrackCuts:Step-By-Step

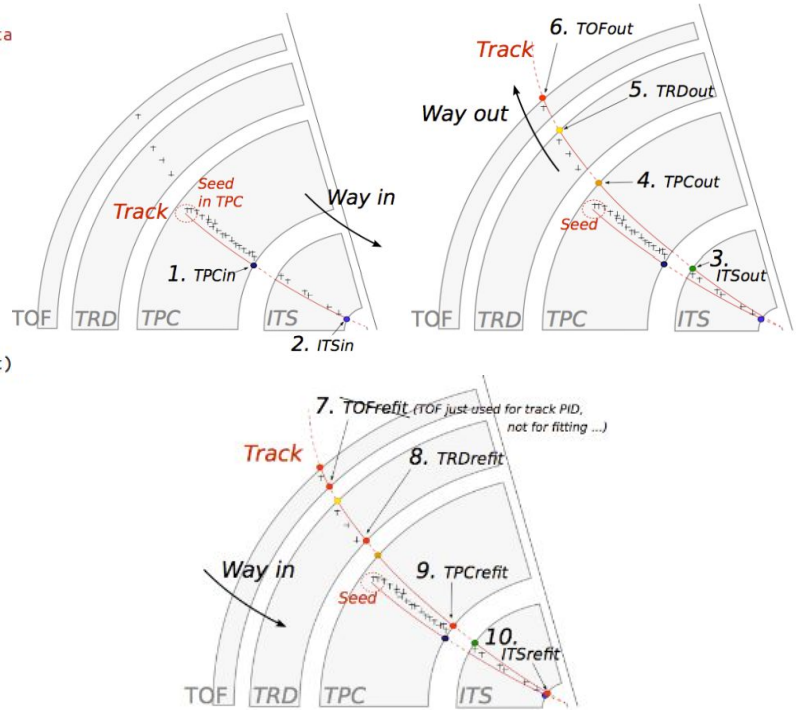
```
AliESDtrackCuts* AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(Bool_t selPrimaries, Int_t clusterCut)
{
  /// creates an AliESDtrackCuts object and fills it with standard values for ITS-TPC cuts for pp 2011 data
  /// if clusterCut = 1, the cut on the number of clusters is replaced by
  /// a cut on the number of crossed rows and on the ration crossed
  /// rows/findable clusters

  AliInfoClass("Creating track cuts for ITS+TPC (2011 definition).");

  AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;

  // TPC
  if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
  else if (clusterCut == 1) {
    esdTrackCuts->SetMinNCrossedRowsTPC(70);
    esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
  }
  else {
    AliWarningClass(Form("Wrong value of the clusterCut parameter (%d), using cut on Nclusters",clusterCut));
    esdTrackCuts->SetMinNClustersTPC(50);
  }
  esdTrackCuts->SetMaxChi2PerClusterTPC(4);
  esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
  esdTrackCuts->SetRequireTPCRefit(kTRUE);
  // ITS
  esdTrackCuts->SetRequireITSRefit(kTRUE);
  esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD,
                                         AliESDtrackCuts::kAny);
  if(selPrimaries) {
    // 7*(0.0015+0.0050/pt^1.1)
    esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
    esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
  }
  esdTrackCuts->SetMaxDCAToVertexZ(2);
  esdTrackCuts->SetDCAToVertex2D(kFALSE);
  esdTrackCuts->SetRequireSigmaToVertex(kFALSE);

  esdTrackCuts->SetMaxChi2PerClusterITS(36);

  return esdTrackCuts;
}
```



- Kink = topological signature of charged particles decaying into 1 charged + 1 neutral particles (e.g. K→μν, π→μν)
- Search for kinks inside the volume of the TPC after the first inward tracking step and assign a kinkstatus for each track
- Kink algorithm depends on angle and E-p conservation
  - Large decay angle kinks = pairs of tracks with same charge that intersect each other in a fiducial volume

AliESDtrackCuts:Step-By-Step

```
AliESDtrackCuts* AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(Bool_t selPrimaries, Int_t clusterCut)
{
  /// creates an AliESDtrackCuts object and fills it with standard values for ITS-TPC cuts for pp 2011 data
  /// if clusterCut = 1, the cut on the number of clusters is replaced by
  /// a cut on the number of crossed rows and on the ration crossed
  /// rows/findable clusters

  AliInfoClass("Creating track cuts for ITS+TPC (2011 definition).");

  AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;

  // TPC
  if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
  else if (clusterCut == 1) {
    esdTrackCuts->SetMinNCrossedRowsTPC(70);
    esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
  }
  else {
    AliWarningClass(Form("Wrong value of the clusterCut parameter (%d), using cut on Nclusters",clusterCut));
    esdTrackCuts->SetMinNClustersTPC(50);
  }
  esdTrackCuts->SetMaxChi2PerClusterTPC(4);
  esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
  esdTrackCuts->SetRequireTPCRefit(kTRUE);
  // ITS
  esdTrackCuts->SetRequireITSRefit(kTRUE);
  esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD,
                                         AliESDtrackCuts::kAny);
  if(selPrimaries) {
    // 7*(0.0015+0.0050/pt^1.1)
    esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
    esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
  }
  esdTrackCuts->SetMaxDCAToVertexZ(2);
  esdTrackCuts->SetDCAToVertex2D(kFALSE);
  esdTrackCuts->SetRequireSigmaToVertex(kFALSE);

  esdTrackCuts->SetMaxChi2PerClusterITS(36);

  return esdTrackCuts;
}
```



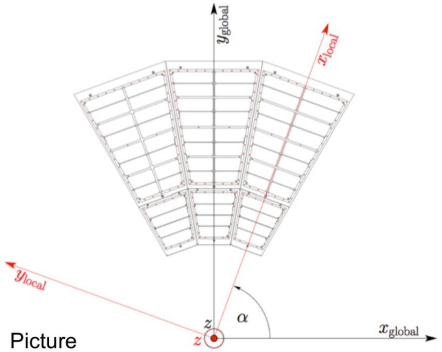
After each reconstruction step detector status is updated
fFlags|=k<DetectorName><step> where step={1:in,2:out,3:refit,}

AliESDtrackCuts:Step-By-Step

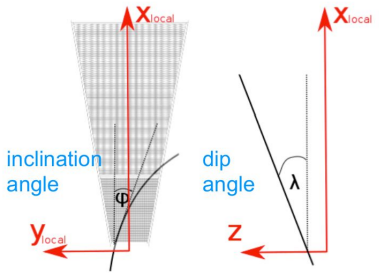
```
AliESDtrackCuts* AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(Bool_t selPrimaries, Int_t clusterCut)
{
  /// creates an AliESDtrackCuts object and fills it with standard values for ITS-TPC cuts for pp 2011 data
  /// if clusterCut = 1, the cut on the number of clusters is replaced by
  /// a cut on the number of crossed rows and on the ration crossed
  /// rows/findable clusters

  AliInfoClass("Creating track cuts for ITS+TPC (2011 definition).");

  AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;

  // TPC
  if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
  else if (clusterCut == 1) {
    esdTrackCuts->SetMinNCrossedRowsTPC(70);
    esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
  }
  else {
    AliWarningClass(Form("Wrong value of the clusterCut parameter (%d), using cut on Nclusters",clusterCut));
    esdTrackCuts->SetMinNClustersTPC(50);
  }
  esdTrackCuts->SetMaxChi2PerClusterTPC(4);
  esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
  esdTrackCuts->SetRequireTPCRefit(kTRUE);
  // ITS
  esdTrackCuts->SetRequireITSRefit(kTRUE);
  esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD,
                                         AliESDtrackCuts::kAny);
  if(selPrimaries) {
    // 7*(0.0015+0.0050/pt^1.1)
    esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
    esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
  }
  esdTrackCuts->SetMaxDCAToVertexZ(2);
  esdTrackCuts->SetDCAToVertex2D(kFALSE);
  esdTrackCuts->SetRequireSigmaToVertex(kFALSE);

  esdTrackCuts->SetMaxChi2PerClusterITS(36);

  return esdTrackCuts;
}
```

χ2 -difference between: TPConly track parameters extrapolated to the primary vertex and global track parameters. Removes fake high-pt tracks due to wrong association of ITS clusters

```
param0:   local Y-coordinate of a track (cm)
param1:   local Z-coordinate of a track (cm)
param2:   local sine of the track momentum azimuthal angle
param3:   tangent of the track momentum dip angle
param4:   1/pt (1/(GeV/c))
```



$(y, z, \sin(\varphi), \tan(\lambda), 1/p_t)$



Picture from J. Wiechula

AliESDtrackCuts:Step-By-Step

```
AliESDtrackCuts* AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(Bool_t selPrimaries, Int_t clusterCut)
{
  /// creates an AliESDtrackCuts object and fills it with standard values for ITS-TPC cuts for pp 2011 data
  /// if clusterCut = 1, the cut on the number of clusters is replaced by
  /// a cut on the number of crossed rows and on the ration crossed
  /// rows/findable clusters

  AliInfoClass("Creating track cuts for ITS+TPC (2011 definition).");

  AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;

  // TPC
  if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
  else if (clusterCut == 1) {
    esdTrackCuts->SetMinNCrossedRowsTPC(70);
    esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
  }
  else {
    AliWarningClass(Form("Wrong value of the clusterCut parameter (%d), using cut on Nclusters",clusterCut));
    esdTrackCuts->SetMinNClustersTPC(50);
  }
  esdTrackCuts->SetMaxChi2PerClusterTPC(4);
  esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
  esdTrackCuts->SetRequireTPCRefit(kTRUE);
  // ITS
  esdTrackCuts->SetRequireITSRefit(kTRUE);
  esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD,
                                         AliESDtrackCuts::kAny);
  if(selPrimaries) {
    // 7*(0.0015+0.0050/pt^1.1)
    esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
    esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
  }
  esdTrackCuts->SetMaxDCAToVertexZ(2);
  esdTrackCuts->SetDCAToVertex2D(kFALSE);
  esdTrackCuts->SetRequireSigmaToVertex(kFALSE);

  esdTrackCuts->SetMaxChi2PerClusterITS(36);

  return esdTrackCuts;
}
```

Cut at ≈ 7σ of impact parameter resolution.

AOD trackselection: filter bits

- AOD tracks have a filter-bit mask
- Store information about whether the track satisfies standard sets of quality criteria
- Each bit corresponds to a given set of cuts, e.g.:

```
enum AODTrkFilterBits_t {
  kTrkTPCOnly            = BIT(0), // Standard TPC only tracks
  kTrkITSsa              = BIT(1), // ITS standalone
  kTrkITSConstrained     = BIT(2), // Pixel OR necessary for the electrons
  kTrkElectronsPID       = BIT(3),    // PID for the electrons
  kTrkGlobalNoDCA        = BIT(4), // standard cuts with very loose DCA
  kTrkGlobal             = BIT(5),  // standard cuts with tight DCA cut
  kTrkGlobalSDD          = BIT(6), // standard cuts with tight DCA but with requiring the first SDD
  kTrkTPCOnlyConstrained = BIT(7) // TPC only tracks: TPConly information constrained to SPD vertex
};
```

https://twiki.cern.ch/twiki/bin/view/ALICE/AODsets

- The usage of a filter bit is recommended but might not be enough for your analysis! You should be aware what it corresponds to!

AOD trackselection: filter bits

Bit	Cuts	Methods
Bit 0 (001)	Standard cuts on primary tracks	GetStandardTPCOnlyTrackCuts() (*)
Bit 1 (002)	ITS stand-alone tracks(ESD Track Cuts)	SetRequireITSStandAlone(kTRUE)
Bit 2 (004)	Pixel OR (necessary for the electrons) AND Standard track cuts (SetFilterMask(1) of AliESDtrackCuts)	SetClusterRequirementITS(AliESDtrackCuts::kSPD, AliESDtrackCuts::kAny)
Bit 3 (008)	PID for the electrons AND Pixel Cuts (SetFilterMask(4) of AliESDpidCuts)	SetTPCnSigmaCut(AliPID::kElectron, 3.5)
Bit 4 (016)	Standard Cuts with very loose DCA	GetStandardITSTPCTrackCuts2011(kFALSE) SetMaxDCAToVertexXY(2.4) SetMaxDCAToVertexZ(3.2) SetDCaToVertex2D(kTRUE)
Bit 5 (032)	Standard Cuts with tight DCA cut	GetStandardITSTPCTrackCuts2011()
Bit 6 (064)	Standard Cuts with tight DCA but with requiring the first SDD cluster instead of an SPD cluster tracks selected by this cut are exclusive to those selected by the previous cut	GetStandardITSTPCTrackCuts2011() SetClusterRequirementITS(AliESDtrackCuts::kSPD,AliESDtrackCuts::kNone) SetClusterRequirementITS(AliESDtrackCuts::kSDD,AliESDtrackCuts::kFirst)
Bit 7 (128)	TPC only tracks,constrained to SPD vertex in the filter	GetStandardTPCOnlyTrackCuts esdfilter->SetTPCOnlyFilterMask(128)
Bit 8 (256)	Extra cuts for Hybrids: - first the global tracks we want to take	AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(kFALSE) SetMaxDCAToVertexXY(2.4) SetMaxDCAToVertexZ(3.2) SetDCAToVertex2D(kTRUE) SetMaxChi2TPCConstrainedGlobal(36) SetMaxFractionSharedTPCClusters(0.4) esdfilter->SetHybridFilterMaskGlobalConstrainedGlobal((1<<8)); // these normal global tracks will be marked as hybrid

Bit 9 (512)	Than the complementary tracks which will be stored as global
	constraint, complement is done in the ESDFilter task
		SetClusterRequirementITS(AliESDtrackCuts::kSPD,AliESDtrackCuts::kOff)
		SetRequireITSRefit(kTRUE)
		esdfilter->SetGlobalConstrainedFilterMask(1<<9); // these tracks are written out as global constrained tracks
		esdfilter->SetWriteHybridGlobalConstrainedOnly(kTRUE); // write only the complement

Bit 10(1024)	Standard Cuts with tight DCA cut, using cluster cut instead of crossed rows	GetStandardITSTPCTrackCuts2011(kTRUE,0) (**)

https://twiki.cern.ch/twiki/bin/view/ALICE/PWGPPAODTrackCuts

AOD trackselection: filter bits

Track selection can be done using the filter bits using these methods of AliAODTrack

```
for(Int_t iTrack=0;iTrack<nTracks;iTrack++){
 AliAODTrack *aodTrack = aod->GetTrack(iTrack);
 if(!aodTrack) continue;
 // filter bit 128 denotes TPC-only tracks, use only them
 if(!aodTrack->TestFilterBit(128)) continue;
```

```
Bool_t IsAcceptedTrack(const AliAODTrack *aodTrack) {
  if (!aodTrack) return kFALSE;
  if(!aodTrack->TestFilterMask(BIT(5))) return kFALSE; // standard TPCITS with tight DCA
```

```
if(!(aodtrack->TestFilterBit(AliAODTrack::kTrkGlobalNoDCA) ||
     aodtrack->TestFilterBit(AliAODTrack::kTrkITSsa))) return kFALSE;
```

Extracting detector information: ITS

- Number of ITS clusters
  - It is the number of ITS points that are included in the Kalman filter fit of the track
  - Retrieved with: AliESDtrack::GetITSNcls() and AliAODTrack::GetITSNcls()
  - NOTE: tracks with kITSrefit have >=2 ITS clusters
- Hit in ITS layer
  - Method to test if a track has a point on a given layer:
    AliESDtrack::HasPointOnITSLayer(Int_t i) and AliAODTrack::HasPointOnITSLayer(Int_t i)
    - NOTE: index i from 0 (inner SPD) to 5 (outer SSD)
- chi-square of the track fit in ITS
  - Can be retrieved via AliESDtrack::GetITSchi2() and AliAODTrack::GetITSchi2()

Extracting detector information: TPC

- Number of TPC clusters
  - It is the number of clusters that is included in the Kalman filter fit of the track
  - Retrieved with: AliESDtrack::GetTPCNcls() and AliAODTrack::GetTPCNcls()
- Number of crossed rows
  - It is the number of clusters that were considered for the track fitting.
  - Retrieved with: AliESDtrack::GetTPCCrossedRows() and AliAODTrack::GetTPCCrossedRows()
- Number of findable clusters
  - It is the number of TCP pad rows crossed by the track, computed taking into account dead zones
  - Retrieved with: AliESDtrack::GetTPCNclsF() and AliAODTrack::GetTPCNclsF()
  - Ratio of number of crossed rows / number of findable clusters used in track selection
- chi-square of the track momentum fit in TPC
  - ESD tracks: retrieved with AliESDtrack::GetTPCchi2
  - AOD tracks: the chi2 in TPC is not stored as a data member of AliAODTrack, which instead has as data member fChi2perNDF
    - Can be retrieved via: AliAODTrack::Chi2perNDF

ndof = 2ncl - 5

Impact parameter, momentum and charge

```
Double_t p[3]={-999.0,-999.0,-999.0};
Int_t trackcharge      = -999;
Float_t dcaxy = -999.0;
Float_t dcaz  = -999.0;


////////Track information/////////////////
      track->GetImpactParameters(dcaxy,dcaz);
      track->PxPyPz(p);
      trackcharge=track->Charge();t;
```

PID information: TPC

runplugin.C

```
// === PID RESPONSE=================================================
        gROOT->LoadMacro("$(ALICE_ROOT)/ANALYSIS/macros/AddTaskPIDResponse.C");
        if(bMCphyssel)AddTaskPIDResponse(bMCphyssel,kTRUE,kTRUE);
        else AddTaskPIDResponse(bMCphyssel);
```

AnalysisMyTask.h

```
private:
….
AliPIDResponse    *fPIDResponse;//!
……
```

AnalysisMyTask.cxx

```
#include "AliPIDResponse.h
```

Initialize in two constructor

PID information: TPC

AnalysisMyTask.cxx

UserCreateOutputObjects()

```
//---------------------------------------------
 // Particle Identification Setup
 //---------------------------------------------
AliAnalysisManager *man=AliAnalysisManager::GetAnalysisManager();
AliInputEventHandler* inputHandler = (AliInputEventHandler*) (man->GetInputEventHandler());
fPIDResponse = inputHandler->GetPIDResponse();
```

UserExec()

```
nSigma_proton_TPC = fPIDResponse->NumberOfSigmasTPC(esdt, AliPID::kProton);
nSigma_deuteron_TPC = fPIDResponse->NumberOfSigmasTPC(esdt, AliPID::kDeuteron);

TPCdedx=fPIDResponse->GetTPCsignal()
```

PID information: TOF

AnalysisMyTask.cxx

UserExec()

```
 AliVTrack   *track = (AliVTrack*)lESDevent->GetTrack(itr);
 AliESDtrack *esdt  = dynamic_cast<AliESDtrack*>(track);
///condition for TOF matching track/////////
Double_t l = esdt->GetIntegratedLength();
if(l<350) return kFALSE;
EDetPidStatus status = fPIDResponse->CheckPIDStatus(AliPIDResponse::kTOF,vtrack);
```

Before to use TOF you then need to check the returned status is equal to `AliPIDResponse::kDetPidOk`

Or

```
Double_t l = esdt->GetIntegratedLength();
if(l<350) return kFALSE;
if (!(vtrack->GetStatus() & AliESDtrack::kTOFout)) return kFALSE;
if (!(vtrack->GetStatus() & AliESDtrack::kTIME  )) return kFALSE;

//////////////////////////
nSigma_proton_TOF = fPIDResponse->NumberOfSigmasTOF(esdt, AliPID::kProton);
nSigma_deuteron_TOF = fPIDResponse->NumberOfSigmasTOF(esdt, AliPID::kDeuteron);
```

PID information: TOF mass information

```
AliESDtrack *esdtrack  = dynamic_cast<AliESDtrack*>(vtrack);
 if(!esdtrack) return -1;
 const Double_t c = 2.99792457999999984e-02;
 Double_t p = esdtrack->GetTPCmomentum();
 Double_t l = esdtrack->GetIntegratedLength();
 Double_t trackT0 = fPIDResponse->GetTOFResponse().GetStartTime(p);
 Double_t timeTOF = esdtrack->GetTOFsignal()- trackT0;
 Double_t mass_square=(p*p)*(TMath::Power(c*timeTOF/l,2.0)-1);
```