PWG3 analysis (barrel)

https://twiki.cern.ch/twiki/bin/view/ALICE/PWG3Hadron
https://twiki.cern.ch/twiki/bin/view/ALICE/PWG3Electron

Andrea Dainese

(INFN Legnaro)

thanks to R.Bala, C.Lazzeroni, S.Masciocchi, R.Romita, A.Rossi, C.Zampolli

Contents

- Status of heavy-flavour vertexing software: PWG3/vertexingHF
- Main class AliAnalysisVertexinHF: production of candidates
- Production and analysis of candidates on grid (train)
- Starting with CORRFW
- Looking into event mixing

AliAnalysisVertexingHF

- Main class to produce the candidated charm decays: 2-prong, 3-prong, 4-prong, cascades. In one go.
- It is the “core” of AliAnalysisTaskSEVertexingHF
- Output written to AOD event (friend tree in “delta-AOD” file AliAOD.VertexingHF.root)
- Configured by AliAnalysisVertxingHF* ConfigVertexingHF.C
- News since last OW:
  - input is AliVEvent (AOD, ESD, mixed event?...)
  - inclusion of D*→D0π (as a cascade) X.Zhang
  - 4-prong analysis fully functional (was dummy) R.Romita
  - inclusion of like-sign pairs C.Di Giglio
  - AliESDtrackCuts used for single-track selection

AliAnalysisVertexingHF class

single-track cuts using AliESDtrackCuts
[2 sets of cuts: “displaced track”, “soft pion (D*)”, can be extended e.g. electrons]

build (+,-) pairs and like-sign with “displ. tracks” and compute secondary vtx

for $D^0 \rightarrow K\pi$, $J/\psi \rightarrow ee$

create AliAODRecoDecayHF2Prong
apply reco cuts

for $D^* \rightarrow D^0\pi$

special D0 cuts,
loop on “soft pions”
for D* candidates

store D^0

store D^{*+}

store J/ψ

for D^+, D_s^+, Λ_c^+

loop on all tracks (+ & -):
build triplets,
create AliAODRecoDecayHF3Prong,
apply reco cuts (common for the 3 particles?)

store D^+, D_s^+, Λ_c

for $D^0 \rightarrow K\pi\pi\pi$

loop on all tracks (+ & -) …
create AliAODRecoDecayHF
…

store D^0

AliVEvent as input

- Motivation: AOD as well as ESD
  - AODs are smaller in size, will be replicated in more SEs
- Strategy followed:
  - AOD well suited for “kinematics” analysis, not for vertexing
  - ESD well suited for vertexing (AliExternalTrackParam+AliESDVertex)
  - Use “virtual” interfaces to read ESD or AOD with the same code
    - ➢ in case of AOD input, convert to ExternalTrackParam and ESDVertex
  - Do vertexing as from ESD

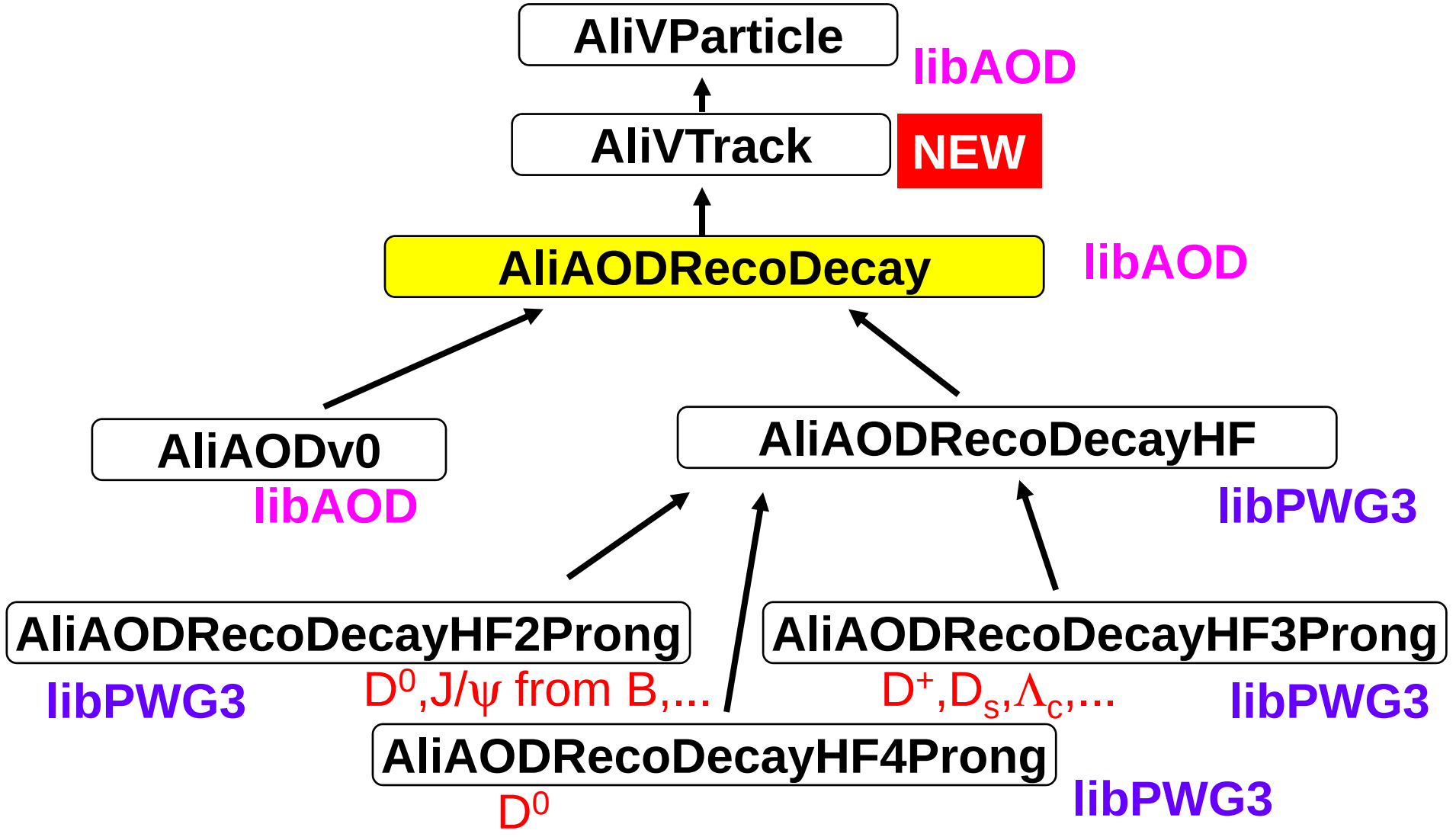
Use virtual interfaces

Virtual interfaces: AliVEvent | AliVParticle | AliVTrack | AliVVertex

ESD world	AOD world
AliESDEvent	AliAODEvent
AliESDVertex	AliAODVertex
AliESDtrack	AliAODTrack

- Possible issues:
  - no common interface for single-track cuts (AliESDtrackCuts used, but a ESD track created from a AOD tracks lacks some info, e.g. nTPC clusters)
  - PID (e.g. electron ID in TRD will need momentum at the TRD)
  - cuts used to create AOD from ESD have to be known!!! stored in alien together with AOD? database?

Storing the candidates in the AOD: AliAODRecoDecay family



AliAODRecoDecay from AliVTrack

- AliAODRecoDecay derived from AliVTrack
  - → can use it for further vertexing
  - use cases:
    - heavy-flavour cascades (e.g. D*→D0π, B→D0*h*, B→D0*e*
    - calculate impact parameters (with error) to primary vertex (feed-down from beauty)
    - use reconstructed V0s (D mesons?) in primary vertex fit
- AliVTrack can be used as input to construct a track
  - AliExternalTrackParam t(vTrack); // charged track
  - AliNeutralTrackParam t(vTrack); // neutral track (e.g. D0, V0)
    - AliNeutralTrackParam: public AliExternalTrackParam with J.Belikov

      GetC(Double_t bz) {return 0.;}

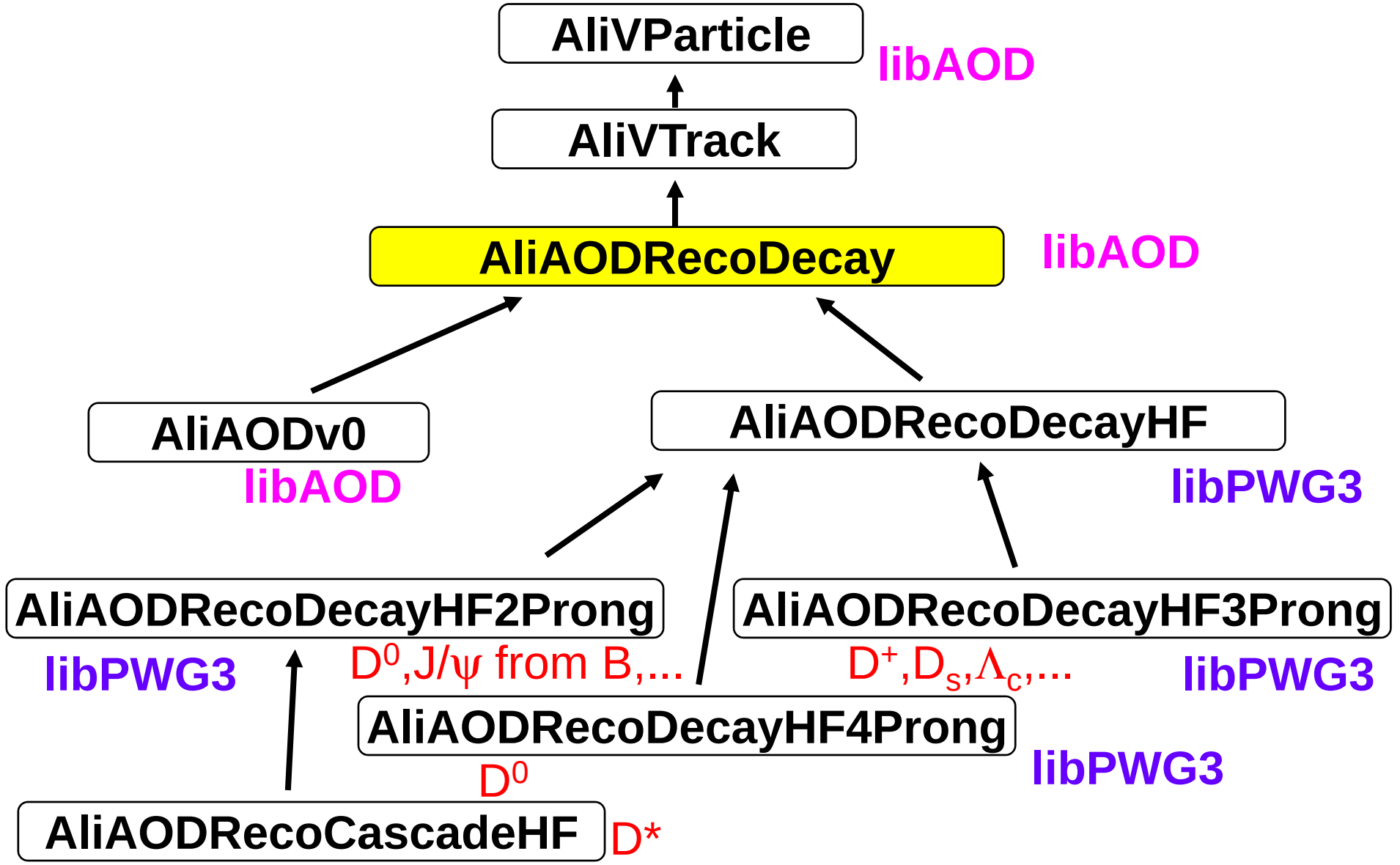
      Charge() {return 0;}
- AliAODRecoDecay::GetCovarianceXYZPxPyPz()
  - xyz from the sec. vertex
  - pxpypz as the sum of the daughter tracks' cov matrices

Inclusions of $D^{*+} \rightarrow D^0 \pi^+$

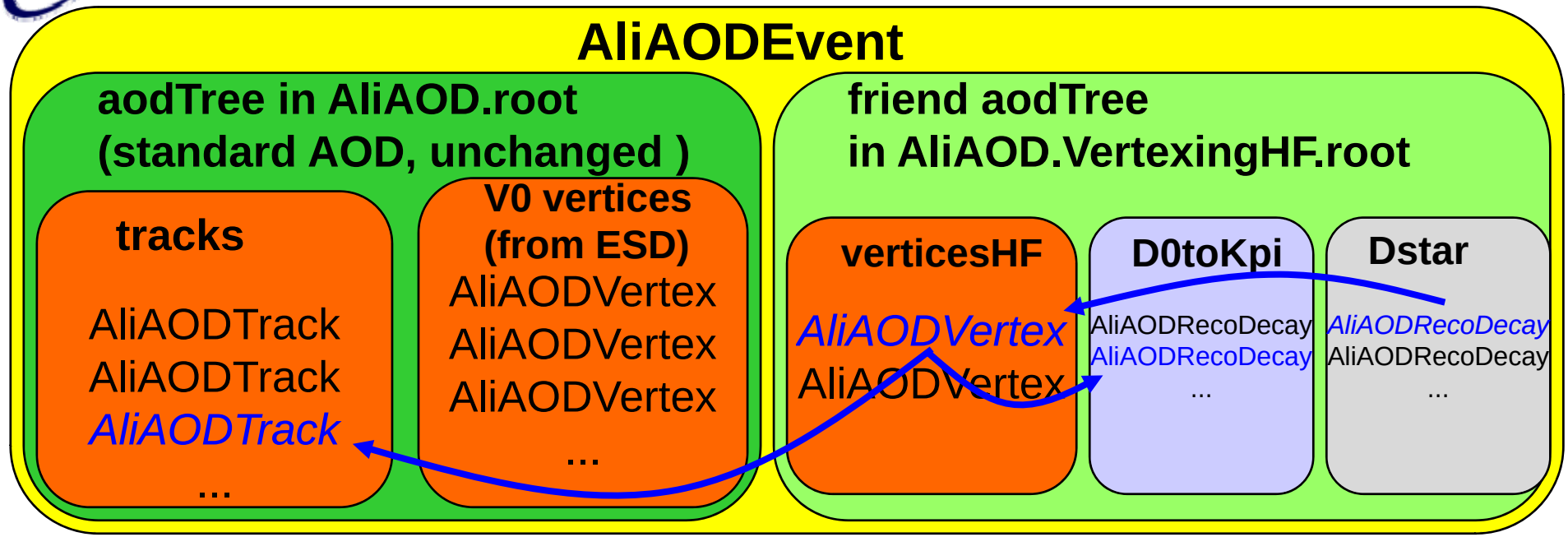
- New class AliAODRecoCascadeHF: public AliAODRecoDecayHF2Prong
  - no members, only methods
  - use the TRef of AliAODVertex to point to the 2Prong (D0) and to the bachelor (π_s)
    - use TRef * instead of TRefArray because the latter requires all TRefs to have the ProcessID (thanks to Andreas)

```
// 2prong (D0 for Dstar)
AliAODRecoDecayHF2Prong* Get2Prong() const {return (AliAODRecoDecayHF2Prong*)GetDaughter(1);}

// Bachelor (soft pion for Dstar)
AliAODTrack* GetBachelor() const {return (AliAODTrack*)GetDaughter(0);}

// D*->D0pi, D0->Kpi
Double_t EDstar() const {return E(413);}
Double_t YDstar() const {return Y(413);}
Bool_t   SelectDstar(const Double_t *cutsDstar,const Double_t *cutsD0,Bool_t testD0=kTRUE) const;
Double_t InvMassD0() const {return (Charge()>0 ? Get2Prong()->InvMassD0() : Get2Prong()->InvMassD0bar());}
Double_t InvMassDstarKpipi() const;
Double_t DeltaInvMass() const {return (InvMassDstarKpipi()-InvMassD0());}
```

New class AliAODRecoCascadeHF



HF candidates in AliAODEvent



- Access to daghters:

```
AliAODRecoCascadeHF *c = arrDstar->At(i);
AliAODTrack *pion = c->GetBachelor();
AliAODRecoDecayHF2Prong *dzero = c->Get2Prong()
```

Production of candidates on Grid

- Optimal running mode: from AOD (rather than ESD)
  - full TRef functionality
  - AODs have smaller granularity and more replicas
- Official Train runs on ESDs and produces AODs (Mihaela)
- VertexingHF in Train (adapted from Mihaela's example): two macros execute in the same jdl:
  - root.exe –q AnalysisTrain.C // official train (ESD→AOD) using xml
  - root.exe –q AliAnalysisTaskSEVertexingHFTest.C // runs in "local" mode (no xml) using as input the file ./AliAODs.root produced by the prev macro
- Tested on grid by R.Bala on productions LHC08x, LHC08w, LHC08s, using ROOT v5-23-02+par-files from trunk → OK
  - LHC08x (7M evts with charm) re-generated by Latchezar (thanks!)
- Official Train to be updated
- Do we go this way also for the MB pp production?
- Or is it better to try to have it in AnalysisTrain.C?

Analysis of candidates

- Input: AliAODs.root + AliAOD.VertexingHF.root
- Analysis tasks derived from AliAnalysisTaskSE
- Analysis examples:
  - CORRFW tasks (see later)
  - AliAnalysisTaskSECompareHF: association with MC in the AOD (AliAODMCParticle) and resolutions
    - helper method AliAODRecoDecay::MatchToMC()
  - AliAnalysisTaskSELikeSignAnalysis: compare unlike-sign and like-sign background for 2-prong (D0 and J/psi)
  - AliAnalysisTaskSEBtoJpsiFit: simulatenous fit of J/psi mass and pseudo-proper decay time (a la CDF)
  - In preparation:
    - general invariant mass fitter
    - analysis for B→D feed-down

Analysis of candidates on Grid

- Input: AliAODs.root + AliAOD.VertexingHF.root
  - The two files have to be read in parallel
  - Each pair should be on the same SE
- Possible issues for analysis jobs:
  - how to split the job?
  - one of the two files is not present for a given pair
- A working solution prepared and tested (A.Rossi)
  - create two xml: one with AOD, the other with AOD.VertexingHF
  - use the AOD xml as the InputDataCollection
  - bring the full AOD.VertexingHF xml to the WN as InputFile
  - run a helper macro that
    - creates the input chain with AOD from the xml resulting after the splitting (InputDataList)
    - finds corresponding entries in the AODVertexingHF xml (assuming the two files of the same pair have the same path in the catalogue!)
    - rejects lonely files (no friend)

Example JDL

A.Rossi

```
InputDataCollection="LF:/alice/cern.ch/user/a/arossi/files/TEST/AODcoll_Global.xml, nodownload";
InputDataList="AODcoll.xml";
InputDataListFormat="xml-single";

InputFile= "LF:/alice/cern.ch/user/a/arossi/files/TEST/VertexingColl.xml"};
Split="se";
```

Carry the XML of the friends as an input file on the working dir

Compare the entries of (= create the indexes for) the AODcoll.xml resulting from the splitting of the global AODcollection with the entries of the friend collection VertexingColl.xml

In this way

- never lost the right correspondence between a AOD.root and its Friend
- all the cases in which both files are present are considered
- can split by "se"

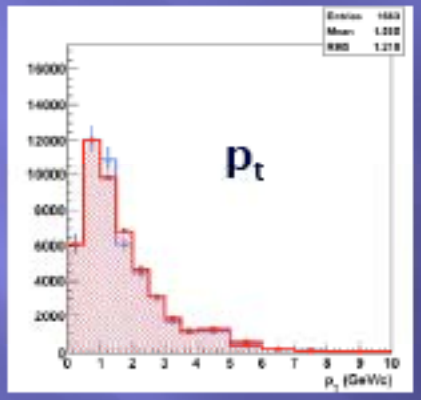
Corrections, with CORRFW

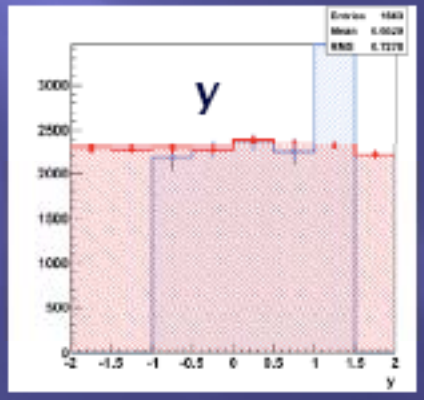
C.Zampolli

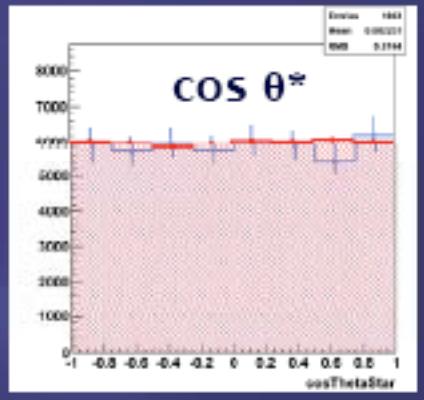
- Important missing part in the analysis chain: Acceptance and efficiency (reco & selection) corrections
- Started to use CORRFW
- Main HF specific requirement: HF vertices are not in ESD but in AOD
- R.Vernet implemented possibility to use CF from AOD reading the MC info in AOD (AliAODMCParticle) →OK
  - open issue: working with AOD we have no TrackRefs → acceptance step based on kine cuts on MC particles
- Already implemented 3 examples classes for D0→Kp
  - simple grid in pt vs y
  - grid with multi vars: pt, y, cosθ*, ct, pt prongs
  - multi-step procedure (in progress): acceptance, rec, rec+single track cuts, rec+D0 cuts

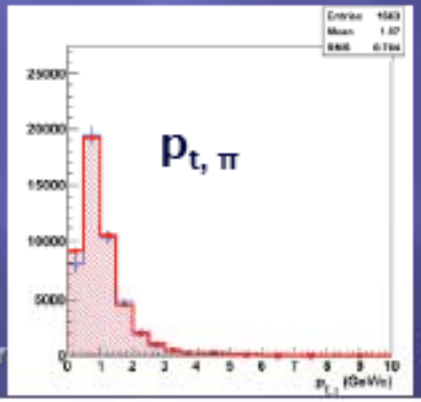
Correcting a Different Sample

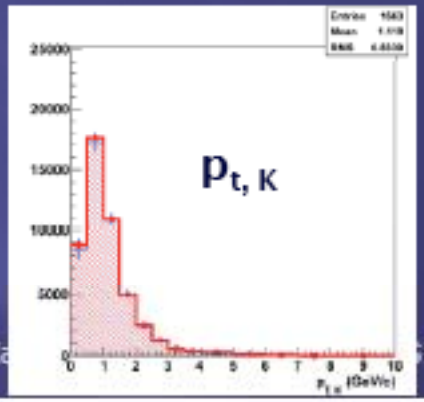
- Complete agreement between corrected data and MC data when corrected data are obtained using efficiencies calculated from the same sample
- Using the correction maps obtained from a sample to correct a different one (using REC info), things change… (NB: independent errors)

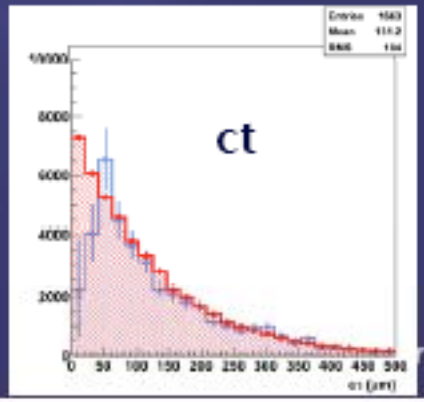










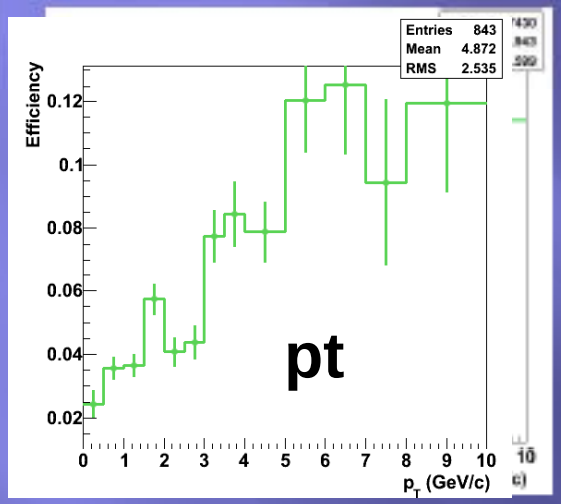


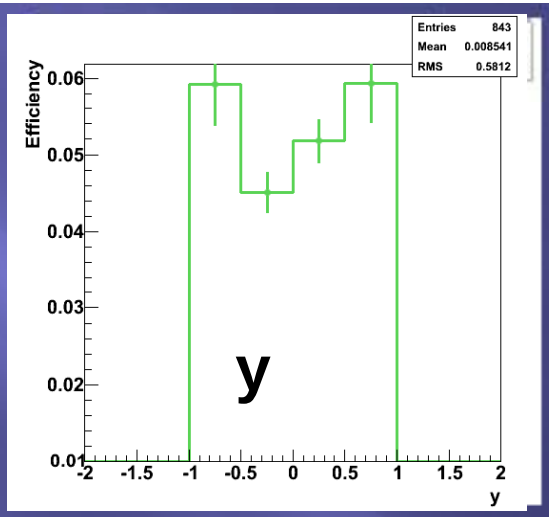
MC data

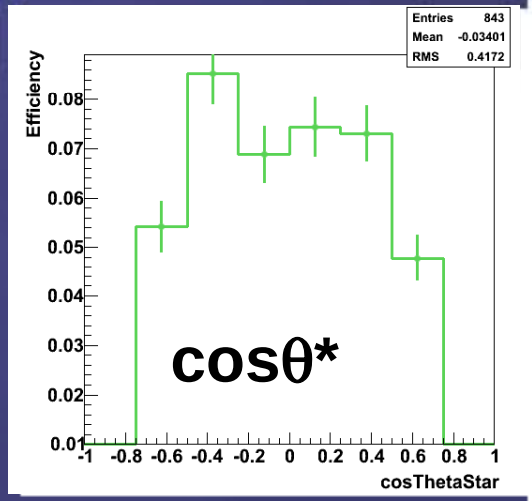
Reconstruct data, corrected with 1d eff

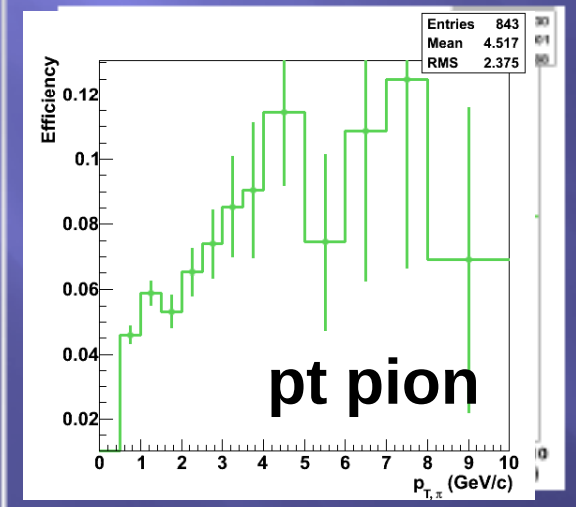
A Closer Look at the Efficiency Maps

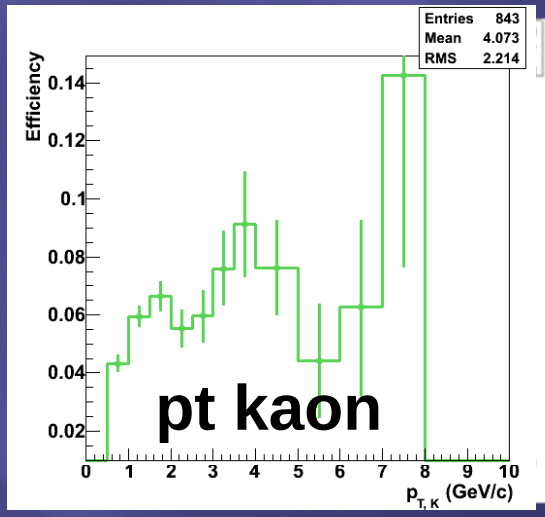
D0 selected (acc & PPR cuts) / generated (acceptance cuts)

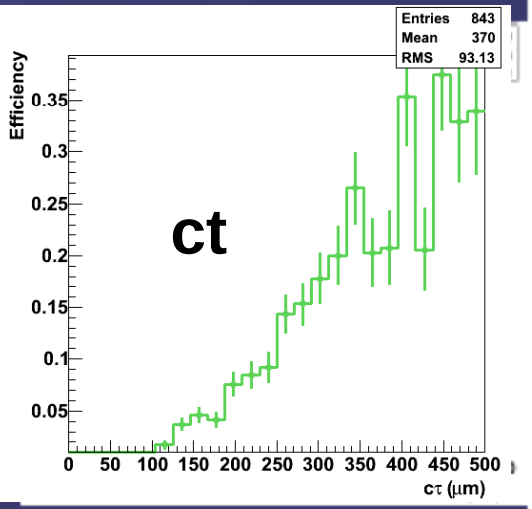












February 2009 C. Zampolli – D2H Meeting – PWG3 Corr Fram, 7

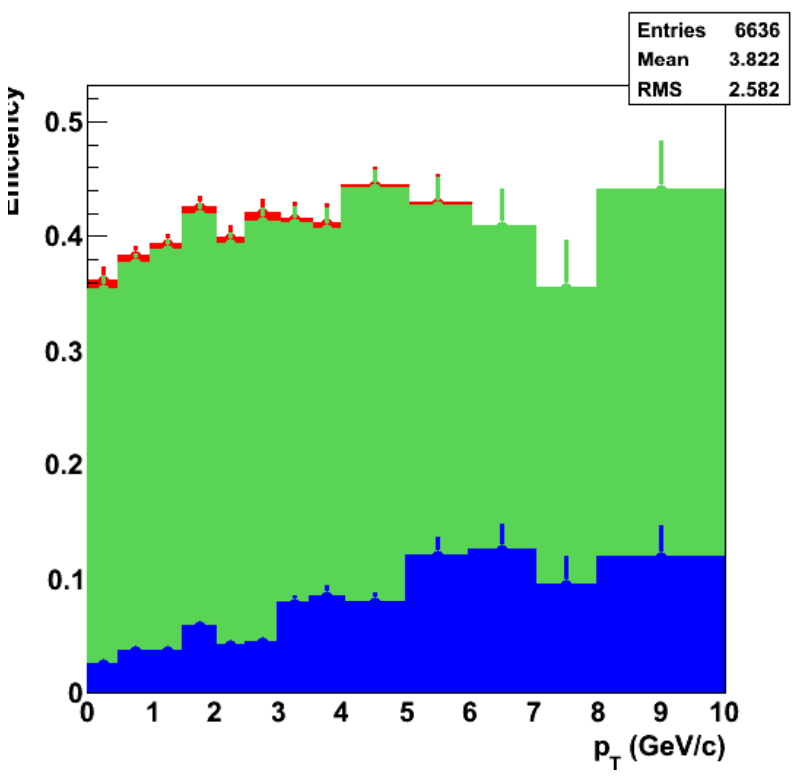
A Closer Look at the Efficiency Maps

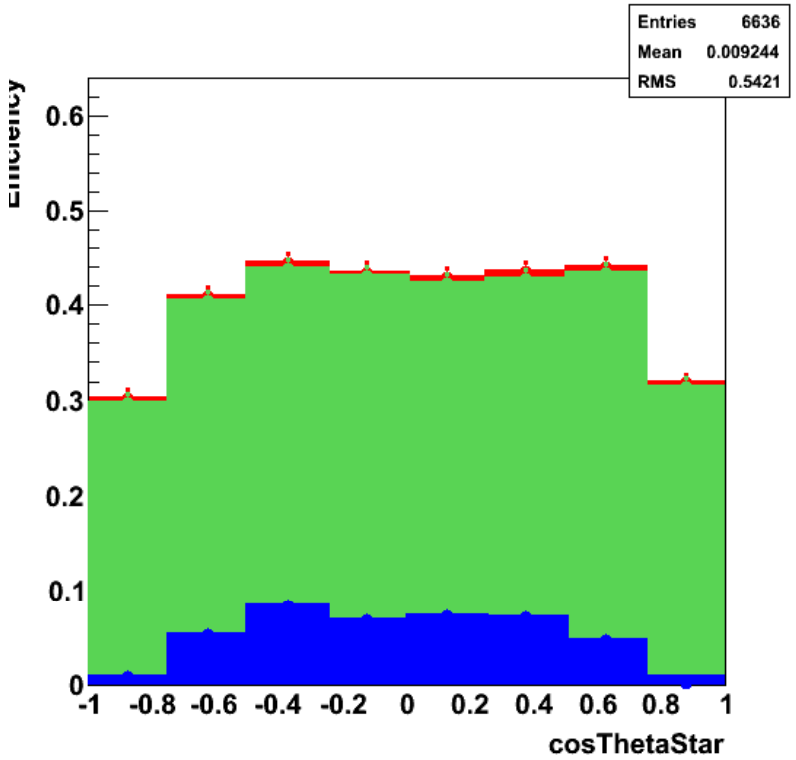
Step by step

Rec(Acc cuts)/ Gen(Acc cuts)

Rec(Acc+trk cuts)/ Gen(Acc cuts)

Sele(Acc+trk+PPR cuts)/ Gen(Acc cuts)





Looking into mixing

R.Romita

- Goal: possibility to use mixed events to study background
- Caveats:
  - should not be crucial in pp (if S/B not too small)
  - may not “work” (residual correlations due to event shape or vertexing)
- Want to use AliAnalysisTaskME
- Requests: additional event selection
  - multiplicity (OK)
  - vertex position in z
  - event shape/jettiness (?, to be studied first)
- Try to minimize changes in the analysis code
  - can we use a AliVEvent as mixed-event input
  - Andreas proposed and implemented a AliMixedEvent : AliVEvent
    - looks promising, to be tried soon
- Change AliAnalysisTaskSE to AliAnalysisTaskME

Can we do mixing with the ESD?

- Some people/groups in PWG3 request possibility to mix ESDs using AliAnalysisTaskME

Advantages

1. At the very beginning of the data taking, we will use mainly ESDs (for example TRD PID will need ESD (momentum at the TRD)) →the event mixing will be feasible from “day one” (pp events)
2. Can be used for a **quick debugging** of the mixing framework:

→ most of the analysis start from ESDs and don’t use the event mixing framework: if the framework is not used, it cannot be debugged...

Drawbacks

1. Memory consumption: maybe too much for Pb-Pb events, but acceptable for pp events...

AOB

- AliEVE:
  - in progress: start from V0 example → AliEveHF, AliEveHFList, AliEveHFListEditor (generalisation to N prongs) D. Caffarri
- User feedback:
  - “unpractical to use par files”, mainly compilation issues, often trivial errors...
  - on the other hand, this is necessary because our code evolved a lot since the last branching (v4-16-Release)
  - need min bias production!
- Production requests:
  - LHC08x was re-generated in 3 days with original AliRoot (v4-13-Rev-03)
  - Pending requests for J/psi production: needed by many people, but have to wait for AliRoot...

82		PWG3	p-p	14 TeV	50,000	0	04 Feb 2009	28 Feb 2009
81		PWG3	p-p	14 TeV	100,000	0	04 Feb 2009	28 Feb 2009
79		PWG3	p-p	14 TeV	300,000	0	04 Feb 2009	28 Feb 2009

To do

- Checks of reco cuts and AOD size increase
- Port recent development to v4-16-Release
- Now that we are starting to have large set of AODs, start organizing AOD analysis exercises (end-user analysis)
  - few analysis tasks in a common train
    - D0toKpi selection
    - Dplus selection
    - Significance maximisation
    - Correction maps
    - ...
  - read common AOD set
  - will take time because most analyses are not advanced enough...
- Some guidelines from offline, on how to organise this, would be useful