Playing with the output

A. Sarti

DIPARTIMENTO DI SCIENZE
DI BASE E APPLICATE
PER L'INGEGNERIA

SAPIENZA
UNIVERSITÀ DI ROMA

Starting point: output of L0

- In the past sessions, we have practised how to process the data and MC information in order to produce the 'basic' building blocks needed for a global event reconstruction.
- DecodeRaw and DecodeMC are the executables that we have learned to used, and that produce an output that can be used to 'combine' vtxs, tracks, TW hits, etc etc into a global track 'fully reconstructed'.
  - Data: ../../bin/DecodeRaw -in data_built.2211.physics_foot.daq.WD.1.dat -out testdata.root -nev 1000 -exp GSI -run 2211
  - MC: ../../bin/DecodeMC -in ~/FOOT/Software/Tutorial_2021/Tutorial/Full/12C_C_200shoe.root -out tetesteringstring.root -exp 12C_200 -run 1

Putting everything together

- Two strategies:
  - 'on the fly'. You can perform global track reconstruction directly from DecodeMC. The executable will a) perform the reconstruction of the building blocks b) run the global tracking.
  - 'in two steps'. You just feed to a specific executable the output of Level0 and it will perform a 'standalone' global tracking reconstruction.
- Prerequisites:
  - You need to have an input file that contains at least the trackers and the TW.
- For now, in the master branch, you can get your hands on fully reconstructed tracks using the TAGactNtuGlbTrack action, that combines VTX, IT, MSD and TW using the TATOE class.

how does it works?

- Global tracking is and action, like the ones you saw up to now:

The input data:
Vtx, IT, Msd, TW

The output: Globaltrack

Then you need the geometry, the magnetic field

For now the tracking implemented is the TOE one, at some point we will provide also the GenFit based one.

```
//! Default constructor.
TAGactNtuGlbTrack::TAGactNtuGlbTrack( const char* name,
                                      TAGdataDsc* p_vtxclus,
                                      TAGdataDsc* p_vtxtrack,
                                      TAGdataDsc* p_vtxvertex,
                                      TAGdataDsc* p_itrclus,
                                      TAGdataDsc* p_msdclus,
                                      TAGdataDsc* p_twpoint,
                                      TAGdataDsc* p_glbtrack,
                                      TAGparaDsc* p_geoG,
                                      TAGparaDsc* p_geodi,
                                      TAGparaDsc* p_geoVtx,
                                      TAGparaDsc* p_geoItr,
                                      TAGparaDsc* p_geoMsd,
                                      TAGparaDsc* p_geoTof,
                                      TADIgeoField* field)
 : TAGaction(name, "TAGactNtuGlbTrack - Global Tracker"),
   fpVtxClus(p_vtxclus),
   fpVtxTrack(p_vtxtrack),
   fpVtxVertex(p_vtxvertex),
   fpItrClus(p_itrclus),
   fpMsdClus(p_msdclus),
   fpTwPoint(p_twpoint),
   fpGlbTrack(p_glbtrack),
   fpGGeoMap(p_geoG),
   fpDiGeoMap(p_geodi),
   fpVtxGeoMap(p_geoVtx),
   fpItrGeoMap(p_geoItr),
   fpMsdGeoMap(p_geoMsd),
   fpTofGeoMap(p_geoTof),
   fField(field),
   fActEvtReader(nullptr),
   fActTOE( SetupAction() )
```

how does it works?

- Global tracking is and action, like the ones you saw up to now:

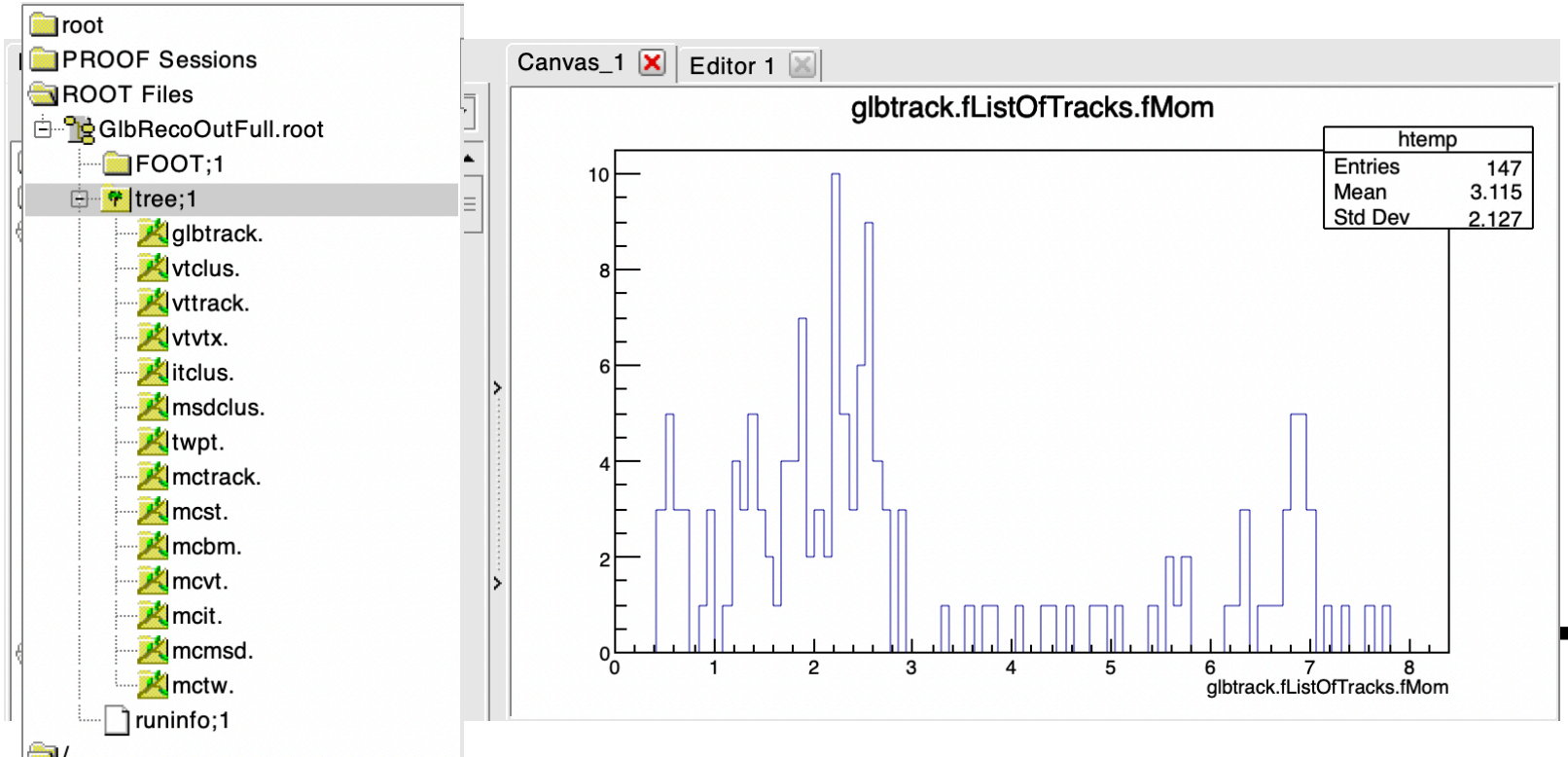
In SetupAction you'll see that the detector information is stored inside a list (that takes the detector input and extract the needed measurements position) and then the TATOEactGlb is initialised.

Inside TATOE **we have points in space**
We still need to 'preserve' the link btw the glb track and the sub detector items used to build it (that will be our exercise today!)

```
using state_vector =  matrix<4,1> ;
using state_covariance =  matrix<4, 4> ;
using state = state_impl< state_vector, state_covariance  >;

auto * clusterVTX_hc = static_cast<TAVTntuCluster*>( fpVtxClus->Object() );
auto * vertex_hc = static_cast<TAVTntuVertex*>( fpVtxVertex->Object() );
auto * geoVTX_h = static_cast<TAVTparGeo*>( fpVtxGeoMap->Object() );


auto * clusterIT_hc = static_cast<TAITntuCluster*>( fpItrClus->Object() );
auto * geoIT_h = static_cast<TAITparGeo*>( fpItrGeoMap->Object() );

auto * clusterMSD_hc = static_cast<TAMSDntuCluster*>( fpMsdClus->Object() );
auto * geoMSD_h = static_cast<TAMSDparGeo*>( fpMsdGeoMap->Object() );

auto * clusterTW_hc = static_cast<TATWntuPoint*>( fpTwPoint->Object() );
auto * geoTW_h = static_cast<TATWparGeo*>( fpTofGeoMap->Object() );


auto list = start_list( detector_properties<details::vertex_tag>(vertex_hc, clusterVTX_hc,
                                                                 geoVTX_h, 15) )
            .add( detector_properties<details::it_tag>(clusterIT_hc, geoIT_h, {33, 38}) )
            .add( detector_properties<details::msd_tag>(clusterMSD_hc, geoMSD_h, {13, 18, 23}) )
            .add( detector_properties<details::tof_tag>(clusterTW_hc, geoTW_h, 2.2) )
            .finish();


auto ode = make_ode< matrix<2,1>, 2>( model{ GetFootField() } );
auto stepper = make_stepper<data_grkn56>( std::move(ode) );
auto ukf = make_ukf<state>( std::move(stepper) );

return make_new_TATOEactGlb(
                            std::move(ukf),
                            std::move(list),
                            static_cast<TAGntuGlbTrack*>( fpGlbTrack->Object() ),
                            static_cast<TAGparGeo*>( fpGGeoMap->Object() )
                           );
```

The initial step: running glb reco

- To perform a global reconstruction you can: ../../bin/DecodeGlbToe -in ../level0/L0ForGlbReco.root -out GlbRecoOutFull.root -exp 12C_200 -run 1 -nev 200 -**mc**
  - [Grab a cup of coffee… it will take some time]
- Inside fullrec you need to prepare the FootGlobal.par
  - IncludeKalman: n; IncludeTOE: y; EnableLocalReco: y



What you need..

- The output of DecodeMC with the tracking detectors and TW turned on + MC truth info
  - The latest update of Master provides all of this when you run DecodeGlbToe.
- The Macro skeleton from
  - https://drive.google.com/drive/folders/1i6ellUyIqQiU_FKO_UGj31paj0g7pcXt?usp=sharing

Navigating: a tough job!

- To access the info produced by the global tracking exec, you can starting from simulation session macro… same code, different info accessed…
  - Need to add few more data…

```
TAGntuGlbTrack *glbTrack = new TAGntuGlbTrack();
tree->SetBranchAddress(TAGntuGlbTrack::GetBranchName(), &glbTrack);

//MC Data
TAMCntuEve *mcNtuEve; // MC tree
int IncludeMC = 1;
if(IncludeMC>0){
  mcNtuEve = new TAMCntuEve();  // Get MC Tree
  tree->SetBranchAddress(TAMCntuEve::GetBranchName(), &mcNtuEve);
  /*
```

  - Then you can start to do the following: you get the reconstructed tracks, and ask them the measured points (used to build the tracks).
  - Then you loop on the different detectors that contributed and compare the info of the building blocks (e.g. clusters in VTX) to the points in the track and perform a match. Afterwards you can test the MC truth info related to a given track….

Loops!

```
int nTrk = glbTrack->GetTracksN();
cout<<"  "<<nTrk<<endl;
for(int aT=0; aT<nTrk; aT++) {
  TAGtrack* aTr = glbTrack->GetTrack(aT);
  int nMeas = aTr->GetMeasPointsN();
  for(int iMe = 0; iMe<nMeas; iMe++) {
    TAGpoint *aPoi = aTr->GetMeasPoint(iMe);
    TVector3 apos = aPoi->GetPosition();
    //      cout<<"--> "<<iMe<<" "<<apos.X()<<" "<<apos.Y()<<" "<<apos.Z()<<endl;

    //Perform the check against the reconstructed vtx, etc etc
    int nvtx= vtxNtuVertex->GetVertexN();
    for(int iVt = 0; iVt<nvtx; iVt++) {
      TAVTvertex *aVt = vtxNtuVertex->GetVertex(iVt);
      int nVtTr = aVt->GetTracksN();
      for(int ivttr = 0; ivttr <nVtTr; ivttr++) {
        TAVTtrack* avttr = aVt->GetTrack(ivttr);
        int nclus = avttr->GetClustersN();
        for(int iCl = 0; iCl<nclus; iCl++) {
          TAVTbaseCluster *aCl =  avttr->GetCluster(iCl);
          TVector3 clPos = aCl->GetPosition();

          TVector3 glClPos = geoTrafo->FromVTLocalToGlobal(clPos);
          //              cout<<" -----> "<<iCl<<" "<<clPos.X()<<" "<<clPos.Y()<<" "<<clPos.Z()<<" "<<endl;
          //              cout<<" -----> "<<iCl<<" "<<glClPos.X()<<" "<<glClPos.Y()<<" "<<glClPos.Z()<<" "<<endl;
          if(fabs(glClPos.X()-apos.X())<1.e-3 && fabs(glClPos.Y()-apos.Y())<1.e-3) {
            cout<<" found match!! "<<endl;
            int npix = aCl->GetPixelsN();
            for(int ipix = 0; ipix<npix; ipix++) {
              TAVTntuHit* aPix = aCl->GetPixel(ipix);
              int pixI = aPix->GetPixelIndex();
              int idxTrkBlk = aPix->GetMcTrackIdx(pixI);

              //Here we have access to the MCeve block!
              int mcTr = mcNtuEve->GetTracksN();
              if(idxTrkBlk-1<mcTr) {
                TAMCeveTrack* frg = mcNtuEve->GetTrack(idxTrkBlk-1);
                cout<<"Mass, chg:: "<<frg->GetMass()<<" "<<frg->GetCharge()<<endl;
              } else {
```

First of all I get my hands on the global tracks that were reconstructed

Loops!

```
int nTrk = glbTrack->GetTracksN();
cout<<"  "<<nTrk<<endl;
for(int aT=0; aT<nTrk; aT++) {
  TAGtrack* aTr = glbTrack->GetTrack(aT);
  int nMeas = aTr->GetMeasPointsN();
  for(int iMe = 0; iMe<nMeas; iMe++) {
    TAGpoint *aPoi = aTr->GetMeasPoint(iMe);
    TVector3 apos = aPoi->GetPosition();
    //      cout<<"--> "<<iMe<<" "<<apos.X()<<" "<<apos.Y()<<" "<<apos.Z()<<endl;

    //Perform the check against the reconstructed vtx etc etc
    int nvtx= vtxNtuVertex->GetVertexN();
    for(int iVt = 0; iVt<nvtx; iVt++) {
      TAVTvertex *aVt = vtxNtuVertex->GetVertex(iVt);
      int nVtTr = aVt->GetTracksN();
      for(int ivttr = 0; ivttr <nVtTr; ivttr++) {
        TAVTtrack* avttr = aVt->GetTrack(ivttr);
        int nclus = avttr->GetClustersN();
        for(int iCl = 0; iCl<nclus; iCl++) {
          TAVTbaseCluster *aCl =  avttr->GetCluster(iCl);
          TVector3 clPos = aCl->GetPosition();

          TVector3 glClPos = geoTrafo->FromVTLocalToGlobal(clPos);
          //              cout<<" -----> "<<iCl<<" "<<clPos.X()<<" "<<
          //              cout<<" -----> "<<iCl<<" "<<glClPos.X()<<" "
          if(fabs(glClPos.X()-apos.X())<1.e-3 && fabs(glClPos.Y()-apos
            cout<<" found match!! "<<endl;
            int npix = aCl->GetPixelsN();
            for(int ipix = 0; ipix<npix; ipix++) {
              TAVTntuHit* aPix = aCl->GetPixel(ipix);
              int pixI = aPix->GetPixelIndex();
              int idxTrkBlk = aPix->GetMcTrackIdx(pixI);

              //Here we have access to the MCeve block!
              int mcTr = mcNtuEve->GetTracksN();
              if(idxTrkBlk-1<mcTr) {
                TAMCeveTrack* frg = mcNtuEve->GetTrack(idxTrkBlk-1);
                cout<<"Mass, chg:: "<<frg->GetMass()<<" "<<frg->GetCharge()<<endl;
              } else {
```

Then I ask all the 'measured points' used to build the track. For now I cannot access the single detector information (this will be available in the future)

Loops!

```
int nTrk = glbTrack->GetTracksN();
cout<<"  "<<nTrk<<endl;
for(int aT=0; aT<nTrk; aT++) {
  TAGtrack* aTr = glbTrack->GetTrack(aT);
  int nMeas = aTr->GetMeasPointsN();
  for(int iMe = 0; iMe<nMeas; iMe++) {
    TAGpoint *aPoi = aTr->GetMeasPoint(iMe);
    TVector3 apos = aPoi->GetPosition();
    //      cout<<"--> "<<iMe<<" "<<apos.X()<<" "<<apos.Y()<<" "<<apos.Z()<<endl;

    //Perform the check against the reconstructed vtx, etc et
    int nvtx= vtxNtuVertex->GetVertexN();
    for(int iVt = 0; iVt<nvtx; iVt++) {
      TAVTvertex *aVt = vtxNtuVertex->GetVertex(iVt);
      int nVtTr = aVt->GetTracksN();
      for(int ivttr = 0; ivttr <nVtTr; ivttr++) {
        TAVTtrack* avttr = aVt->GetTrack(ivttr);
        int nclus = avttr->GetClustersN();
        for(int iCl = 0; iCl<nclus; iCl++) {
          TAVTbaseCluster *aCl =  avttr->GetCluster(iCl);
          TVector3 clPos = aCl->GetPosition();

          TVector3 glClPos = geoTrafo->FromVTLocalToGlobal(clPo
          //              cout<<" -----> "<<iCl<<" "<<clPos.X()
          //              cout<<" -----> "<<iCl<<" "<<glClPos.X
          if(fabs(glClPos.X()-apos.X())<1.e-3 && fabs(glClPos.Y
            cout<<" found match!! "<<endl;
            int npix = aCl->GetPixelsN();
            for(int ipix = 0; ipix<npix; ipix++) {
              TAVTntuHit* aPix = aCl->GetPixel(ipix);
              int pixI = aPix->GetPixelIndex();
              int idxTrkBlk = aPix->GetMcTrackIdx(pixI);

              //Here we have access to the MCeve block!
              int mcTr = mcNtuEve->GetTracksN();
              if(idxTrkBlk-1<mcTr) {
                TAMCeveTrack* frg = mcNtuEve->GetTrack(idxTrkBlk-1);
                cout<<"Mass, chg:: "<<frg->GetMass()<<" "<<frg->GetCharge()<<endl;
              } else {
```

Then I try to understand which clusters from the VTX were used to build the track.. To do so I start from the reconstructed vtxs, take the tracks, take the clusters and get their local positions

Loops!

```
int nTrk = glbTrack->GetTracksN();
cout<<"  "<<nTrk<<endl;
for(int aT=0; aT<nTrk; aT++) {
  TAGtrack* aTr = glbTrack->GetTrack(aT);
  int nMeas = aTr->GetMeasPointsN();
  for(int iMe = 0; iMe<nMeas; iMe++) {
    TAGpoint *aPoi = aTr->GetMeasPoint(iMe);
    TVector3 apos = aPoi->GetPosition();
    //      cout<<"--> "<<iMe<<" "<<apos.X()<<"
    //Perform the check against the reconstructed vt
    int nvtx= vtxNtuVertex->GetVertexN();
    for(int iVt = 0; iVt<nvtx; iVt++) {
      TAVTvertex *aVt = vtxNtuVertex->GetVertex(iVt);
      int nVtTr = aVt->GetTracksN();
      for(int ivttr = 0; ivttr <nVttr; ivttr++) {
        TAVTtrack* avttr = aVt->GetTrack(ivttr);
        int nclus = avttr->GetClustersN();
        for(int iCl = 0; iCl<nclus; iCl++) {
          TAVTbaseCluster *aCl =  avttr->GetCluster(iCl);
          TVector3 clPos = aCl->GetPosition();
          TVector3 glClPos = geoTrafo->FromVTLocalToGlobal(clPos);
          //              cout<<" -----> "<<iCl<<" "<<clPos.X()<<" "<<clPos.Y()<<" "<<clPos.Z()<<" "<<endl;
          //              cout<<" -----> "<<iCl<<" "<<glClPos.X()<<" "<<glClPos.Y()<<" "<<glClPos.Z()<<" "<<endl;
          if(fabs(glClPos.X()-apos.X())<1.e-3 && fabs(glClPos.Y()-apos.Y())<1.e-3) {
            cout<<" found match!! "<<endl;
          int npix = aCl->GetPixelsN();
          for(int ipix = 0; ipix<npix; ipix++) {
            TAVTntuHit* aPix = aCl->GetPixel(ipix);
            int pixI = aPix->GetPixelIndex();
            int idxTrkBlk = aPix->GetMcTrackIdx(pixI);

            //Here we have access to the MCeve block!
            int mcTr = mcNtuEve->GetTracksN();
            if(idxTrkBlk-1<mcTr) {
              TAMCeveTrack* frg = mcNtuEve->GetTrack(idxTrkBlk-1);
              cout<<"Mass, chg:: "<<frg->GetMass()<<" "<<frg->GetCharge()<<endl;
            } else {
```

To match the cluster position with the one from the track one needs to convert the detector local reference frame in the global one.. Then one can match the measured points!

Loops!

```
int nTrk = glbTrack->GetTracksN();
cout<<"  "<<nTrk<<endl;
for(int aT=0; aT<nTrk; aT++) {
  TAGtrack* aTr = glbTrack->GetTrack(aT);
  int nMeas = aTr->GetMeasPointsN();
  for(int iMe = 0; iMe<nMeas; iMe++) {
    TAGpoint *aPoi = aTr->GetMeasPoint(iMe);
    TVector3 apos = aPoi->GetPosition();
    //      cout<<"--> "<<iMe<<" "<<apos.X()<<" "<<ap

    //Perform the check against the reconstructed vtx
    int nvtx= vtxNtuVertex->GetVertexN();
    for(int iVt = 0; iVt<nvtx; iVt++) {
      TAVTvertex *aVt = vtxNtuVertex->GetVertex(
      int nVtTr = aVt->GetTracksN();
      for(int ivttr = 0; ivttr <nVtTr; ivttr
        TAVTtrack* avttr = aVt->GetTrack(iv
        int nclus = avttr->GetClustersN()
        for(int iCl = 0; iCl<nclus; iCl
          TAVTbaseCluster *aCl =  avtt
          TVector3 clPos = aCl->GetP

          TVector3 glClPos = geoT
          //              cout<
          //              cou
          if(fabs(glClPos.X
            cout<<" found
            int npix = aCl->GetPixelsN();
            for(int ipix = 0; ipix<npix; ipix++) {
              TAVTntuHit* aPix = aCl->GetPixel(ipix);
              int pixI = aPix->GetPixelIndex();
              int idxTrkBlk = aPix->GetMcTrackIdx(pixI);

            //Here we have access to the MCeve block!
            int mcTr = mcNtuEve->GetTracksN();
            if(idxTrkBlk-1<mcTr) {
              TAMCeveTrack* frg = mcNtuEve->GetTrack(idxTrkBlk-1);
              cout<<"Mass, chg:: "<<frg->GetMass()<<" "<<frg->GetCharge()<<endl;
            } else {
```

In order to associate the vex track to the MC truth one has to go down to the pixel level. I will now get the pixels from a cluster, and ask to the pixel from which track were generated (see talk from G. Battistoni)

Loops!

```
int nTrk = glbTrack->GetTracksN();
cout<<"  "<<nTrk<<endl;
for(int aT=0; aT<nTrk; aT++) {
  TAGtrack* aTr = glbTrack->GetTrack(aT);
  int nMeas = aTr->GetMeasPointsN();
  for(int iMe = 0; iMe<nMeas; iMe++) {
    TAGpoint *aPoi = aTr->GetMeasPoint(iMe);
    TVector3 apos = aPoi->GetPosition();
    //      cout<<"--> "<<iMe<<" "<<apos.X()<<" "<<apos.Y()<<" "<<apo

    //Perform the check against the reconstructed vtx, etc etc
    int nvtx= vtxNtuVertex->GetVertexN();
    for(int iVt = 0; iVt<nvtx; iVt++) {
      TAVTvertex *aVt = vtxNtuVertex->GetVertex(iVt);
      int nVtTr = aVt->GetTracksN();
      for(int ivttr = 0; ivttr <nVtTr; ivttr++) {
        TAVTtrack* avttr = aVt->GetTrack(ivttr);
        int nclus = avttr->GetClustersN();
        for(int iCl = 0; iCl<nclus; iCl++) {
          TAVTbaseCluster *aCl =  avttr->GetCluster(iCl);
          TVector3 clPos = aCl->GetPosition();

          TVector3 glClPos = geoTrafo->FromVTLocalToGlobal(clPos);
          //              cout<<" -----> "<<iCl<<" "<<clPos.X()<<" "<<clPos.Y()<<" "<<clPos.Z()<<" "<<endl;
          //              cout<<" -----> "<<iCl<<" "<<glClPos.X()<<" "<<glClPos.Y()<<" "<<glClPos.Z()<<" "<<endl;
          if(fabs(glClPos.X()-apos.X())<1.e-3 && fabs(glClPos.Y()-apos.Y())<1.e-3) {
            cout<<" found match!! "<<endl;
            int npix = aCl->GetPixelsN();
            for(int ipix = 0; ipix<npix; ipix++) {
              TAVTntuHit* aPix = aCl->GetPixel(ipix);
              int pixI = aPix->GetPixelIndex();
              int idxTrkBlk = aPix->GetMcTrackIdx(pixI);

              //Here we have access to the MCeve block!
              int mcTr = mcNtuEve->GetTracksN();
              if(idxTrkBlk-1<mcTr) {
                TAMCeveTrack* frg = mcNtuEve->GetTrack(idxTrkBlk-1);
                cout<<"Mass, chg:: "<<frg->GetMass()<<" "<<frg->GetCharge()<<endl;
              } else {
```

Once I know the track idx in the track block, I'm done. I can finally know which particle generated the vtx track that I have used to build the global track!

Exercise:

- From this point onward you can do two things:
  - a) check if the TW info matches the VTX info (e.g. if the candidate is performed with the same fragment or if you have a combinatorial event)
  - b) check if the tracking is efficient: loop on ALL the tracks that are produced in the FOOT acceptance and check if they are reconstructed or not.