WORKING SESSION: TRACKING

LIA LAVEZZI
INFN PAVIA

PANDA Computing Week Torino, 23 – 27 July 2012

IDEAL TRACK FINDER TASK

- prepare the layout
  - switch to “version5”

```
svn switch
https://subversion.gsi.de/fairroot/pandaroot/development/uhlig/Torino/version5/torinoDetector torinoDetector

svn switch
https://subversion.gsi.de/fairroot/pandaroot/development/uhlig/Torino/version5/macro/torinoDetector macro/torinoDetector
```

  - at this point we have the smeared hits and we want to group them into track candidates

IDEAL TRACK FINDER TASK

PndTorinoDetectorTrackFinderIdeal.h

- It must inherit from FairTask

```
class PndTorinoDetectorTrackFinderIdeal : public FairTask
```

- ...so, you **need** to implement some *virtual* functions

```
virtual void Exec(Option_t* option);
virtual InitStatus Init();
```

- **Class members**

```
TClonesArray *fMCTrackArray
TClonesArray *fPointArray
TClonesArray *fHitArray;
```

INPUT

```
TClonesArray* fTrackCandArray;
TClonesArray* fTrackArray;
```

OUTPUT

IDEAL TRACK FINDER TASK

PndTorinoDetectorTrackFinderIdeal

- Remember you have to add it to the compilation in:
  - CMakeLists.txt:
    - add **this** task

```
reconstruction/PndTorinoDetectorTrackFinderIdeal.cxx
```

    - add the include directories

```
${PANDAROOT_SOURCE_DIR}/torinoDetector/reconstruction
${PANDAROOT_SOURCE_DIR}/pnddata/TrackData
${PANDAROOT_SOURCE_DIR}/trackbase
```

  - PndTorinoDetectorLinkDef.h

```
#pragma link C++ class PndTorinoDetectorTrackFinderIdeal+;
```

IDEAL TRACK FINDER TASK

PndTorinoDetectorTrackFinderIdeal.cxx

- **initialization**

```
InitStatus PndTorinoDetectorTrackFinderIdeal::Init()
```

- you must retrieve all the input TCAs via the FairRootManager

```
// instance of FairRootManager
FairRootManager  *ioman = FairRootManager::Instance();
                                    …
// get MC point array
fPointArray =
 (TClonesArray*) ioman->GetObject("PndTorinoDetectorPoint");
```

FOR EXAMPLE THE MC POINTS

- you must register to ouput the TCAs of PndTrack and PndTrackCand

```
fTrackCandArray = new TClonesArray("PndTrackCand");
ioman->Register("PndTorinoDetectorTrackCand",
      "PndTorinoDetector", fTrackCandArray, kTRUE);
fTrackArray = new TClonesArray("PndTrack");
ioman->Register("PndTorinoDetectorTrack",
      "PndTorinoDetector",fTrackArray, kTRUE);
```

IDEAL TRACK FINDER TASK

PndTorinoDetectorTrackFinderIdeal.cxx

- **execution**

```
void PndTorinoDetectorTrackFinderIdeal::Exec(Option_t* opt)
```

- loop over the hits and get:
  - the associated MC point, via refindex
  - its associated MC track, via trackID

- you will then loop over the list of hits, now associated to the MC track, and add them to the PndTrackCand

```
Int_t size = fTrackCandArray->GetEntriesFast();
trackcand = new ((*fTrackCandArray)[size]) PndTrackCand();

// loop on hits
trackcand->AddHit(detID, hitID, rho)
```

THE ORDERING PARAMETER

```
FairRootManager::Instance()->GetBranchId("PndTorinoDetectorHit")
```

IDEAL TRACK FINDER TASK

PndTorinoDetectorTrackFinderIdeal.cxx

- **execution**

```
void PndTorinoDetectorTrackFinderIdeal::Exec(Option_t* opt)
```

- create the PndTrackCand, taking the values from the MC track

```
// create parameters @ 1st
FairTrackParP first(firstPos, firstMom,
                    firstPosErr, firstMomErr, charge,
                    firstPos, dj, dk);

// create parameters @ last
FairTrackParP last(lastPos, lastMom,
                   lastPosErr, lastMomErr, charge,
                   lastPos, dj, dk);

// fill the track
track = new((*fTrackArray)[size]) PndTrack(first, last, *trackcand);
```

POINTER TO PndTrackCand!

IDEAL TRACK FINDER TASK

- switch to the “version5” in my development directory (the track finder is there)

```
svn switch
https://subversion.gsi.de/fairroot/pandaroot/development/lia
/version5/torinoDetector torinoDetector

svn switch
https://subversion.gsi.de/fairroot/pandaroot/development/lia
/version5/macro/torinoDetector macro/torinoDetector
```

END OF PART 1

RECO HIT

PndTorinoDetectorRecoHit.h

- It must inherit from GFRecoHitIfc<POLICY>

```
class PndTorinoDetectorRecoHit : public GFRecoHitIfc<GFPlanarHitPolicy>
```

WE ARE USING PLANES → THE POLICY IS THE PLANAR ONE

- some functions of GFRecoHitIfc are already implemented by the GFPlanarHitPolicy

```
virtual const GFDetPlane& getDetPlane(GFAbsTrackRep*)
virtual TMatrixT<double> getHitCoord(const GFDetPlane&)
virtual TMatrixT<double> getHitCov(const GFDetPlane&)
virtual const std::string& getPolicyName();
```

- you **need** to implement

```
virtual GFAbsRecoHit* clone();
virtual TMatrixT<double> getHMatrix(const GFAbsTrackRep* state)
```

- and to provide the number of measured variables

```
static const int fNparHitRep = 2;
```

RECO HIT

PndTorinoDetectorRecoHit.cxx

- **ctor**: takes in input the hit

```
PndTorinoDetectorRecoHit::PndTorinoDetectorRecoHit(PndTorinoDetectorH
it *hit) : GFRecoHitIfc<GFPlanarHitPolicy>(fNparHitRep) {

  // coordinates
  fHitCoord[0][0] = hit->GetX();
  fHitCoord[1][0] = hit->GetY();

  // covariance matrix (diagonal)
  fHitCov[0][0] = hit->GetDx() * hit->GetDx();
  fHitCov[1][1] = hit->GetDy() * hit->GetDy();

  // plane
  TVector3 o(0., 0., hit->GetZ());
  TVector3 u(1., 0., 0.);
  TVector3 v(0., 1., 0.);
  fPolicy.setDetPlane(GFDetPlane(o,u,v));
}
```

PndTorinoDetectorRecoHit.cxx

- **H matrix 2 X 5**: transforms from the parameters into the measurement frame

```
TMatrixT<double> PndTorinoDetectorRecoHit::getHMatrix(const GFAbsTrackRep*
stateVector) {
  if (dynamic_cast<const GeaneTrackRep*>(stateVector) != NULL) {
    TMatrixT<double> HMatrix(fNparHitRep,5);
    HMatrix[0][0] = 0.;
    HMatrix[0][1] = 0.;
    HMatrix[0][2] = 0.;
    HMatrix[0][3] = 1.;
    HMatrix[0][4] = 0.;

    HMatrix[1][0] = 0.;
    HMatrix[1][1] = 0.;
    HMatrix[1][2] = 0.;
    HMatrix[1][3] = 0.;
    HMatrix[1][4] = 1.;
    return HMatrix;
  }
  else {throw;}
}
```

CAN HANDLE MORE THAN ONE TRACK REP

KALMAN TASK

PndTorinoDetectorKalmanTask.h

- It must inherit from FairTask

```
class PndTorinoDetectorKalmanTask : public FairTask
```

- ...so, you **need** to implement some *virtual* functions

```
virtual void Exec(Option_t* option);
virtual InitStatus Init();
```

- **Class members**

```
TClonesArray *fHitArray;
TClonesArray *fTrackArray
TClonesArray *fTrackCandArray
```

INPUT

```
TClonesArray* fFitTrackArray;
```

OUTPUT

```
Int_t fPDGcode;
GFRecoHitFactory *fTheRecoHitFactory;
FairGeanePro *fPro;
```

KALMAN TASK

PndTorinoDetectorKalmanTask.cxx

- **initialization**

```
InitStatus PndTorinoDetectorKalmanTask::Init()
```

- you must retrieve all the input TCAs via the FairRootManager

```
// instance of FairRootManager
FairRootManager  *ioman = FairRootManager::Instance();
                                    …
// get MC point array
fTrackArray =
 (TClonesArray*) ioman->GetObject("PndTorinoDetectorTrack");
```

FOR EXAMPLE THE PR TRACKS

- you must register to ouput the TCAs of PndTrack and PndTrackCand

```
fFitTrackArray = new TClonesArray("PndTrack");
ioman->Register("PndTorinoDetectorKalmanTrack“,
      "PndTorinoDetector", fFitTrackArray, kTRUE);
```

KALMAN TASK

PndTorinoDetectorKalmanTask.cxx

- **initialization**

```
InitStatus PndTorinoDetectorKalmanTask::Init()
```

- you must create the recoHitFactory and add the hit producers

```
fTheRecoHitFactory = new GFRecoHitFactory();

fTheRecoHitFactory
   ->addProducer(detectorID,
   new GFRecoHitProducer<PndTorinoDetectorHit,
       PndTorinoDetectorRecoHit>(fHitArray));
```

- and you will need the GEANE propagator

```
 fPro = new FairGeanePro();
```

KALMAN TASK

PndTorinoDetectorKalmanTask.cxx

- **execution**

```
void PndTorinoDetectorKalmanTask::Exec(Option_t* opt)
```

- loop over the PR found tracks and get the parameters @ 1° plane

```
FairTrackParP par = track->GetParamFirst();
```

- backtrack with GEANE to the poca to (0, 0, 0)

```
fPro->SetPoint(TVector3(0., 0., 0.));
fPro->PropagateToPCA(1, -1);

Int_t ierr = 0;
FairTrackParH *helix = new FairTrackParH(&par, ierr);

FairTrackParH *atvertex = new FairTrackParH();
Bool_t rc =  fPro->Propagate(helix, atvertex, fPDGcode);
```

- "atvertex" will contain the starting position & momentum

KALMAN TASK

PndTorinoDetectorKalmanTask.cxx

GFTrack & co

- to perform the Kalman fit you need GFTrackCand from PndTrackCand

```
PndTrackCand trackCand = track->GetTrackCand();
GFTrackCand *cand = PndTrackCand2GenfitTrackCand(&trackCand);
```

- you need a track representation

```
GeaneTrackRep *grep = new GeaneTrackRep(fPro,
                      startplane, StartMom,
                      StartPosErr, StartMomErr,
                      charge, fPDGcode);
```

- you need a GFTrack

```
GFTrack* trk= new GFTrack(grep);
trk->setCandidate(*candidate);
```

- you need a list of Reco Hits

```
trk->addHitVector(fTheRecoHitFactory->createMany(*candidate))
```

KALMAN TASK

PndTorinoDetectorKalmanTask.cxx

- **propagation**

```
GFKalman genfitter;
genfitter.processTrack(trk);
```

THIS IS THE ACTUAL GENFIT KALMAN

- translate GFTrack to PndTrack

```
fittrack = (PndTrack*)GenfitTrack2PndTrack(trk);
```

PDGCODE & CHARGE

```
switch (abs(pdg)) {
    case 11:
      fPDGcode = charge * (-11); break;
    case 13:
      fPDGcode = charge * (-13); break;
    case 211:
      fPDGcode = charge * (211); break;
    case 321:
      fPDGcode = charge * (321); break;
    case 2212:
      fPDGcode = charge * (2212); break;
    default:
      fPDGcode = charge * (-13); }
```

IDEAL TRACK FINDER TASK

- switch to the “version6” in my development directory (the track finder is there)

```
svn switch
https://subversion.gsi.de/fairroot/pandaroot/development/lia
/version6/torinoDetector torinoDetector

svn switch
https://subversion.gsi.de/fairroot/pandaroot/development/lia
/version6/macro/torinoDetector macro/torinoDetector
```

END OF PART 2