Major Update Report on Rho

Ralf Kliemt

PANDA Meeting
June 2013

The Plan (from Jan '13)

- Put data objects from pid to pnddata & rename
- Unclutter Rho classes, i.e. delete obsolete or unnecessary stuff
- Rename all Rho classes to Rho*
- Reorganize Rho to be used more fluently with PandaRoot
  - Remove unnecessary virtual or semivirtual layers
  - Structure fitter interfaces
  - make use of all functionalities/concepts which are not properly used, yet
- Reorganize Pndanalysis & co.

See the trunk from rev. 19627

Setting Up the Analysis

```
FairRunAna* fRun = new FairRunAna();
FairRuntimeDb* rtdb = fRun->GetRuntimeDb();
fRun->SetInputFile(„sim.root“);
fRun->AddFriend(„reco.root“);
fRun->AddFriend(„pid.root“);
fRun->SetOutputFile(„dummyout.root“);
FairParRootFileIo* parIO = new FairParRootFileIo();
parIO->open(„params.root“);
rtdb->setFirstInput(parIO);
rtdb->setOutput(parIO);

fRun->Init();

PndAnalysis *theAnalysis = new
    PndAnalysis("SttMvdGemGenTrack","FtsIdealGenTrack");
```

Event Loop

```
int i=0;
int nevts=1000;
//int nevts = theAnalysis->GetEntries();

//theAnalysis->SetVerbose(true);

while (theAnalysis->GetEvent() && i++<nevts)
{
  …
}
```

Now some things which are different than before...

Particle Candidate Lists

```
RhoCandList pions, kaons, D0s; // almost the same

//Combine PID Algorithm outputs (branch names, separated by semicolon):
//also available: "PidAlgoIdealCharged” (MC Truth based) and "PidAlgoMdtHardCuts"
TString pidalgos("PidAlgoMvd;PidAlgoStt;PidAlgoEmcBayes;PidAlgoDrc;PidAlgoDisc");

//Kaon, Pion, Electron, Muon, Proton
//VeryLoose, Loose, Tight, VeryTight, All, Best, Variable
//   0.0        ,     0.2 ,  0.5  ,     0.9         -     -      (from all.par ascii file)
// (optional) Charged, Plus, Minus
theAnalysis->FillList(pions, "PionLoosePlus",pidalgos);
theAnalysis->FillList(kaons, "KaonLooseMinus",pidalgos);

D0s.Combine(pions,kaons); //as we know it
```

Particle Candidates

```
RhoCandList pions, kaons, D0s;
theAnalysis->FillList(pions, "PionLoosePlus");
theAnalysis->FillList(kaons, "KaonLooseMinus");
D0s.Combine(pions,kaons);

for (int j=0; j < D0s.GetLength(); j++)    {
  //new: Candidates are now always passed as pointers
  RhoCandidate *myD0 = D0s[j];
  RhoCandidate *myPion = myD0->Daughter(0);
  RhoCandidate *theD0 = myPion->TheMother(); //identical to myD0
  //Detailed reconstruction & PID results, if needed:
  PndPidCandidate* recopi = (PndPidCandidate*)myPion->GetRecoCandidate()
}
```

Particle Candidates

```
Double_t        Charge()
TVector3        Pos()
RhoError&       PosCov()
TLorentzVector  P4()
RhoError&       P4Cov()
TVector3        P3()
RhoError&       P3Cov()
Double_t        Px()
Double_t        Py()
Double_t        Pz()
Double_t        M()
Double_t        P()
Double_t        Pt()
Double_t        E()
Double_t        EVar()
TMatrixD&       Cov7()
```

```
void Boost(TVector3& p)

Int_t Ndaughters()
RhoCandidate* Daughter(Int_t n)
RhoCandidate* TheMother()

RhoCandidate* GetFit()
RhoCandidate* GetMcTruth()
FairRecoCandidate* GetRecoCandidate()

Int_t GetTrackNumber()
Int_t Uid()

double GetPidInfo(int hypo)
```

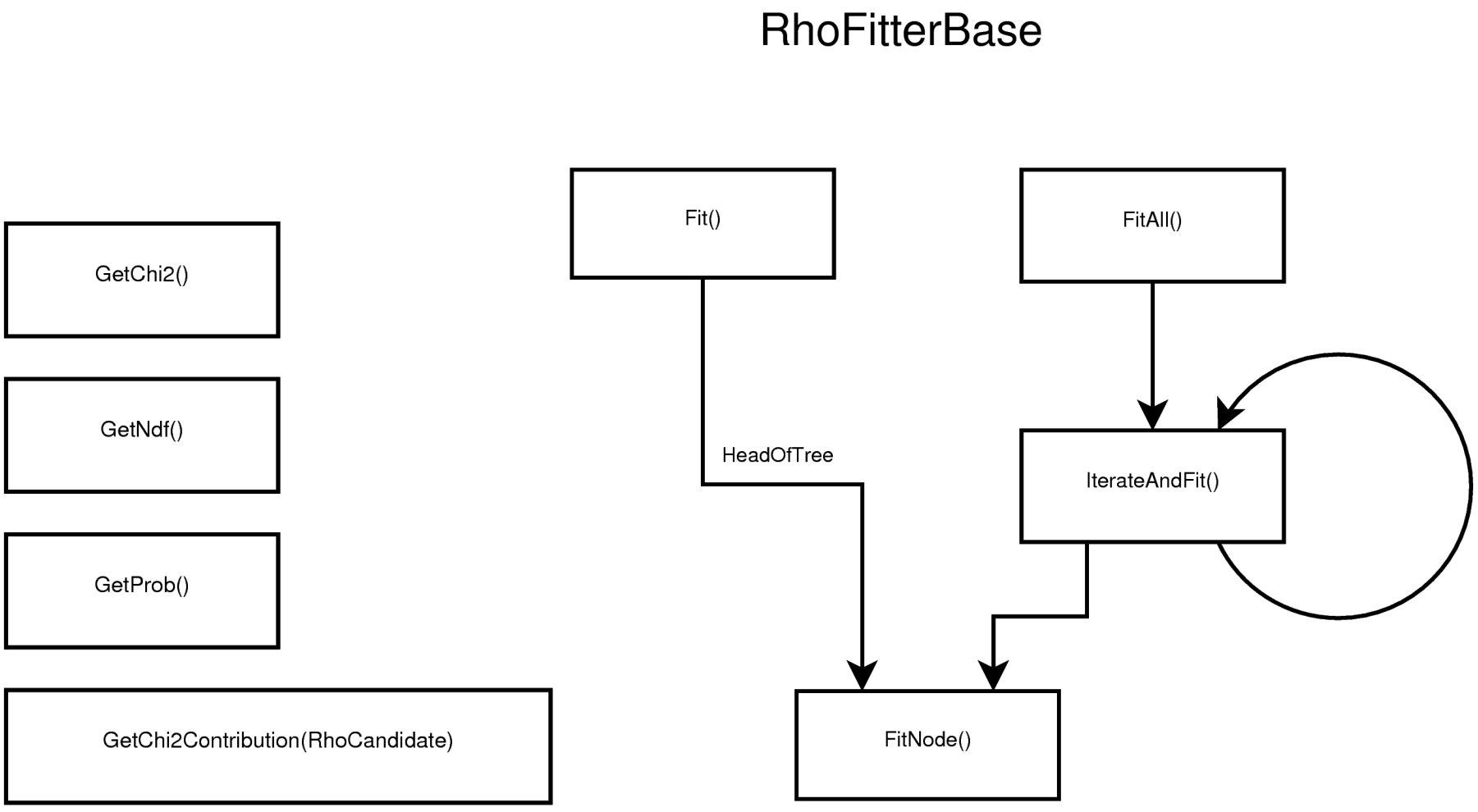
Monte-Carlo Truth

```
PndAnalysis *theAnalysis = new PndAnalysis(...);
PndMcTruthMatch mcmatch;
Int matchlevel = 2; //0: final state pid; 1: tree topology; 2: intermediate particle types
RhoCandList mct;
…
   theAnalysis->FillList(mct,”McTruth”); //get truth as list (if needed)
   for (int j=0; j < D0s.GetLength(); j++) {
     RhoCandidate *myD0 = D0s[j];
     RhoCandidate *myPion = myD0->Daughter(0);
     //new: MC-truth decay tree is always built and set for reconstructed particles
     RhoCandidate *truePion = myPion->GetMcTruth();
     RhoCandidate *truePionMother = truePion->TheMother();
     //now set MC-truth to combined particles in decay tree
     bool check = mcmatch.MctMatch(myD0,mct,matchlevel);
   }
```

Vertex Finding/Fitting

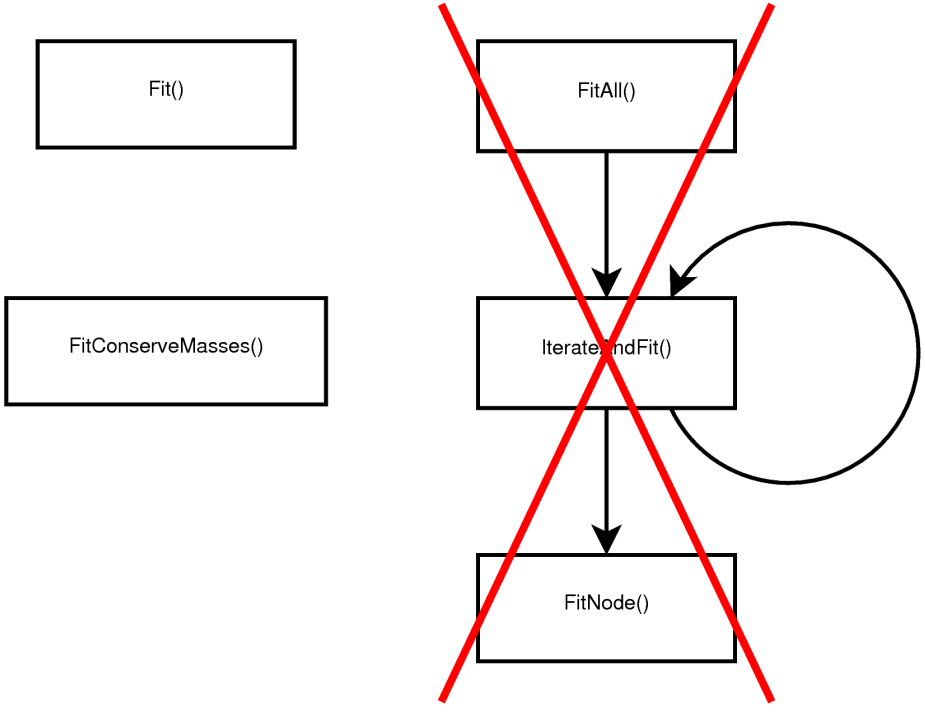
```
RhoCandList pions, kaons, D0s; // almost the same
theAnalysis->FillList(pions, "PionLoosePlus");
theAnalysis->FillList(kaons, "KaonLooseMinus");
D0s.Combine(pions,kaons);

for (int j=0; j < D0s.GetLength(); j++)
{
  PndVtxPRG prgfitter(D0s[j]); //a vertex fitter
  prgfitter.Fit(); //fit the current node (D0 → K- pi+)
  double chi2 = prgfitter.GetChi2();
  RhoCandidate *d0fit = d0[j]->GetFit();
  //new: The fitted candidates live in a separate decay tree
  RhoCandidate *pionfit = d0fit->Daughter(0);
  Tvector3 vtx = pionfit->GetPosition(); //fetch the decay vertex
}
```

Basic Fitter Structure

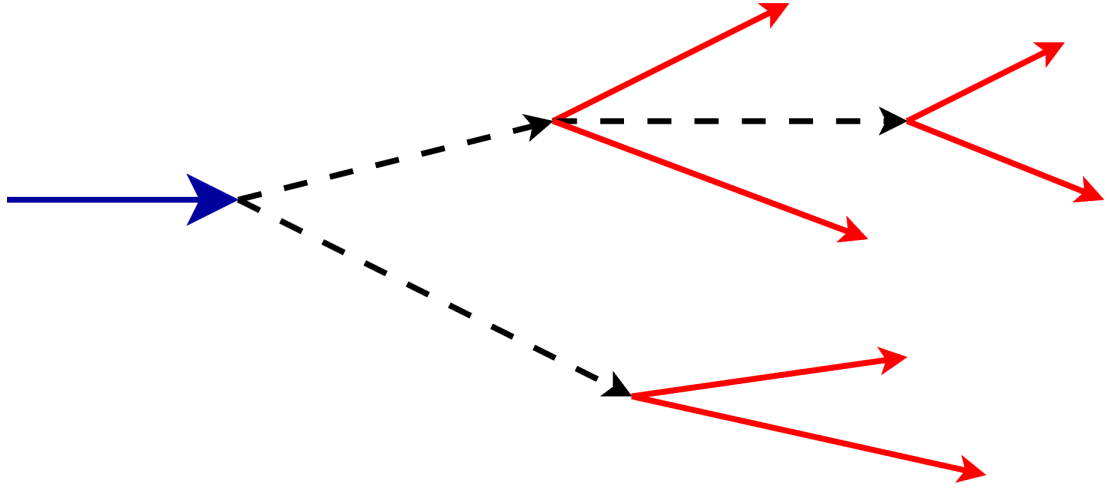


4-Constraint Fitter

Pnd4CFitter



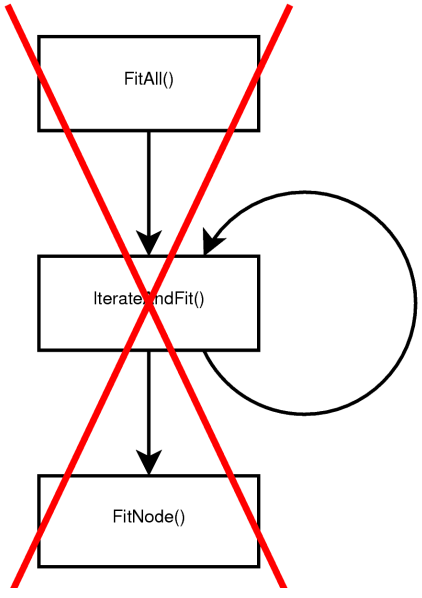
Fitting all final state particles with the initial state.
Intermediate Particles are corrected accordingly.
Final state particle masses may be conserved optionally.



Kinematic Fitter

PndKinFitter

Fit()



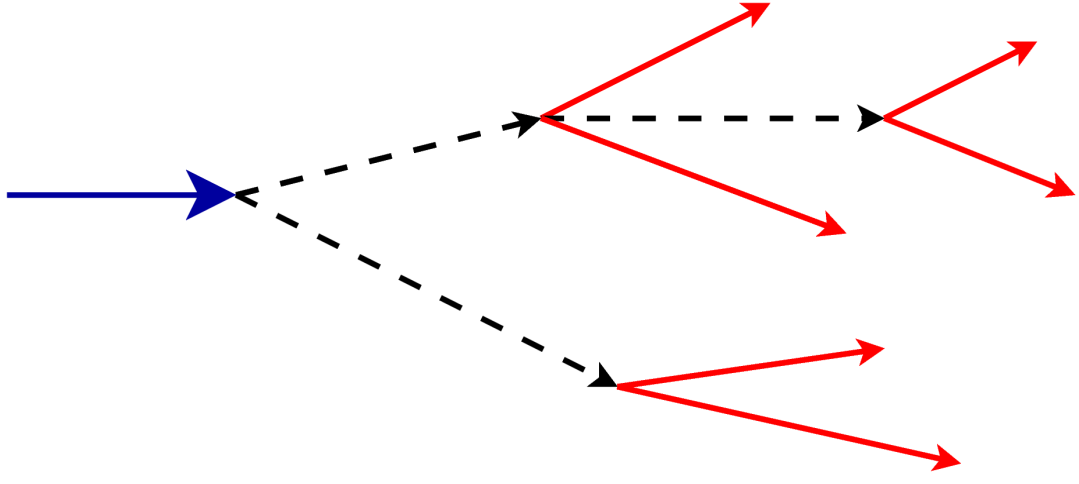
Add4MomConstraint(TLorentzVector)

AddMomConstraint(TVector3)

AddTotEConstraint(double)

AddMassConstraint(double)

AddTotMomConstraint(double)



Fits the final state to the initial state. Only global chisqare available.

GetChi2Contribution(PndCandidate)

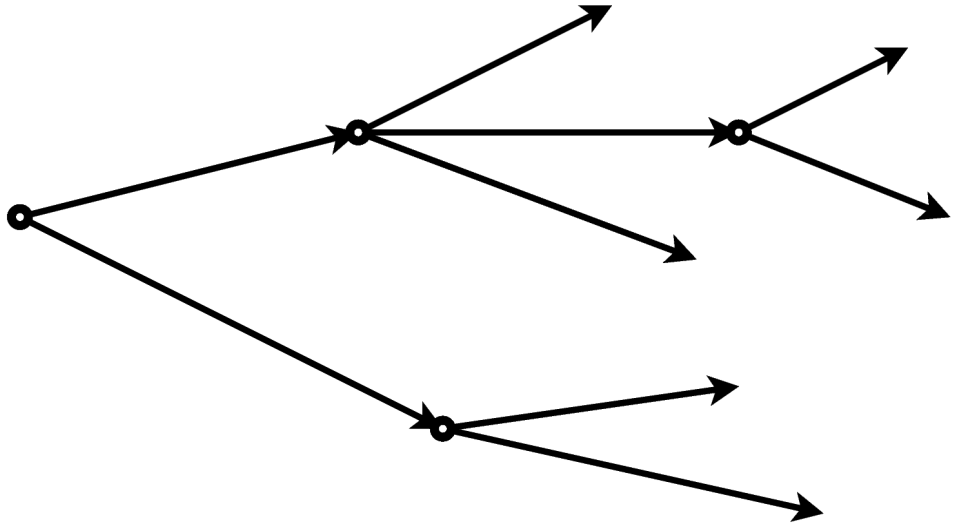
GetPull()

POCA Vertex Finder

PndVtxPoca

GetPocaVtx(TVector3& vertex, RhoCandidate& composite)

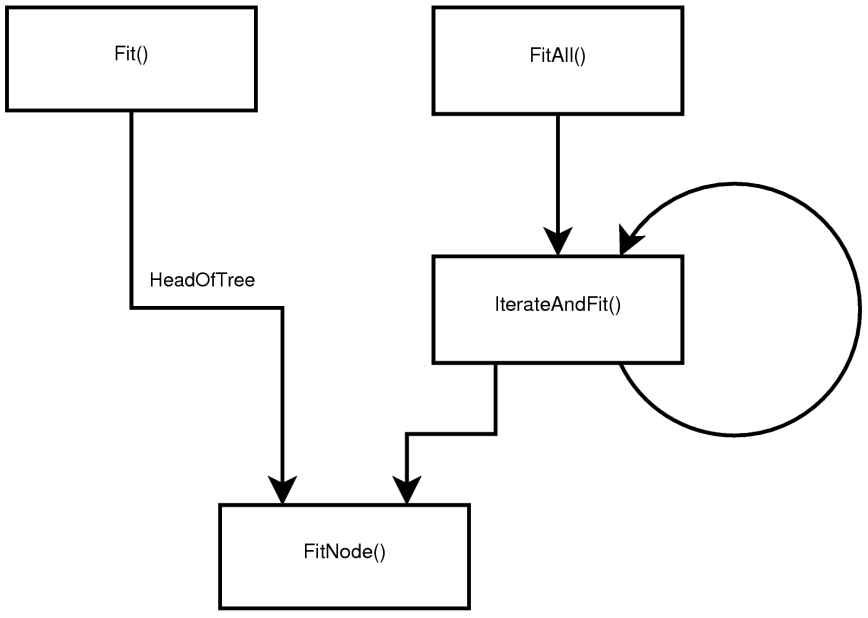
Calculates a vertex position (passed to the TVector3) and returns a quality number. It’s no fit.

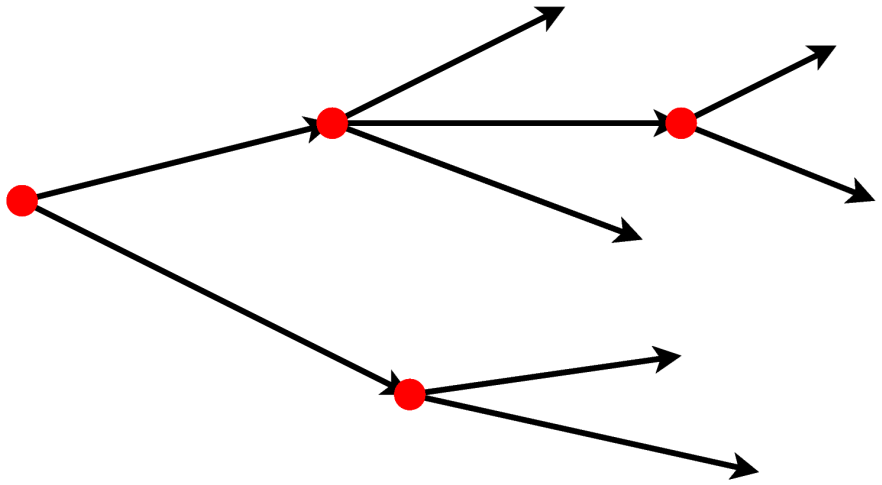


“Kinematic” Vertex Fitter

PndKinVtxFitter

Fitting to vertices at each decay node. Updated Momenta and covariances in each node. Charged particles only.





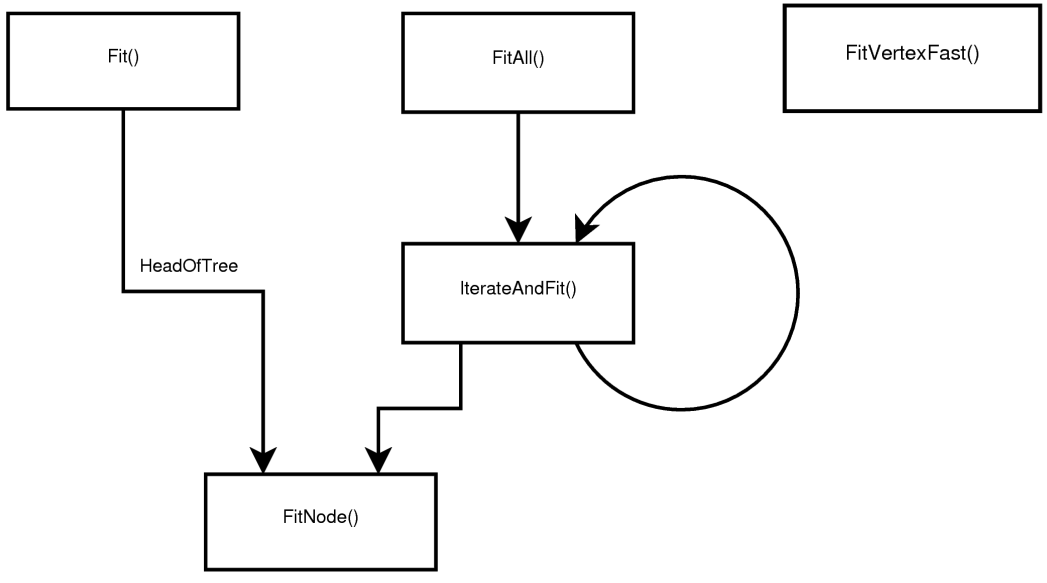
AddPointingConstraint(TVector3)

AddMassConstraint(double)

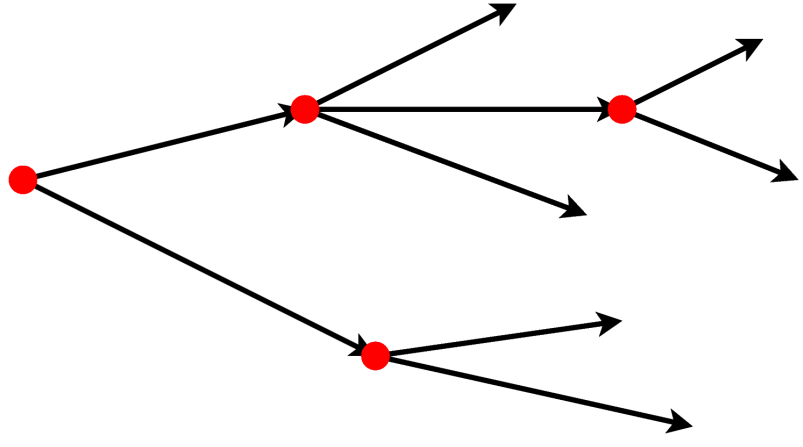
Algorithm as in P. Avery write-ups.

“Fast” Vertex Fitter

PndVtxPRG



Vertexing region (expansion point/seed) adjustable.
Optional fast fit (vertex position output only).



SetExpansionPoint(TVector3)

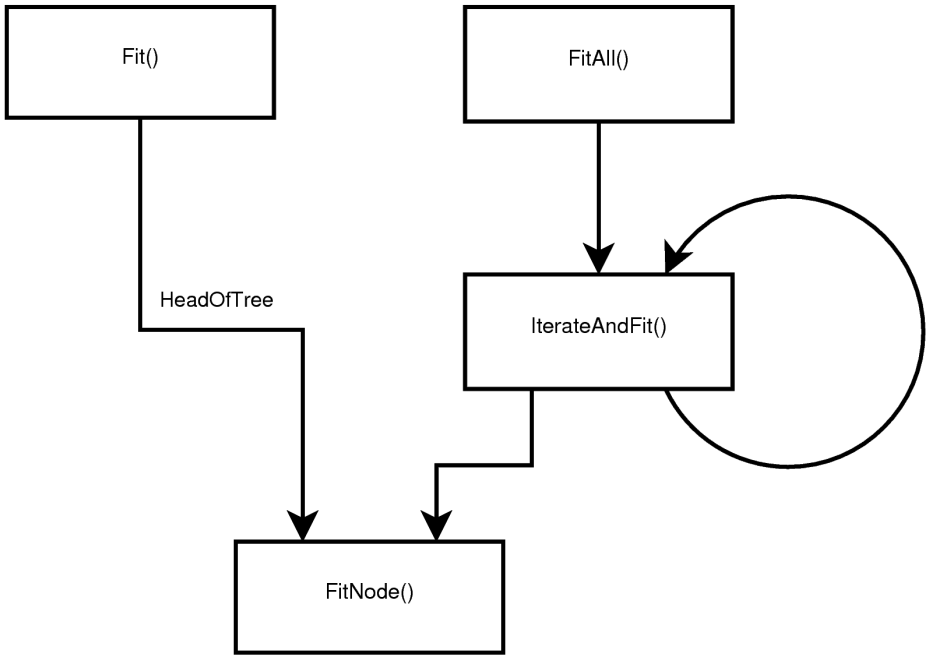
SetNIterations()

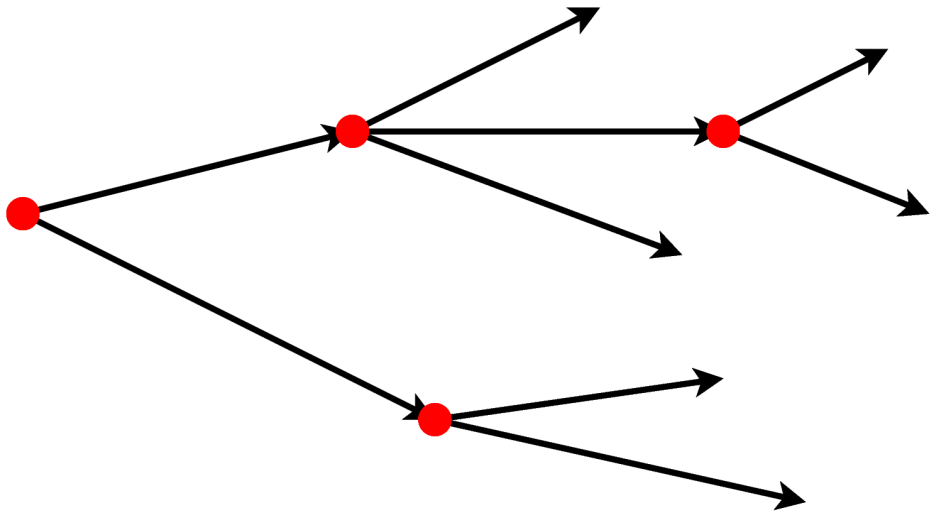
Algorithm from Billoir/Qian.

“Chi” Vertex Fitter

PndChiVtxFitter

Vertexices found as position of daughters.
No updated covariances.





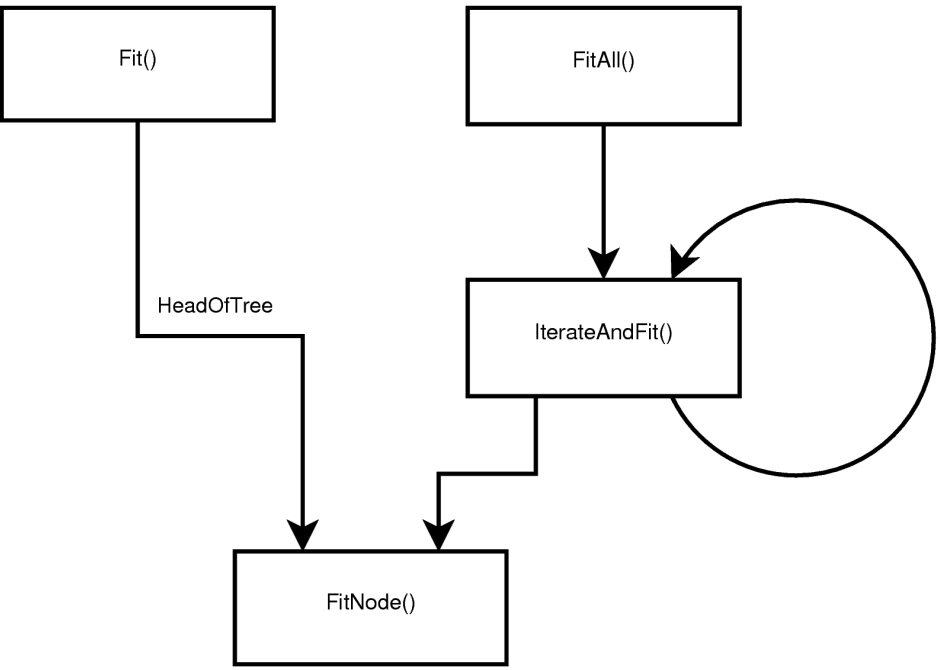
SetIterations()

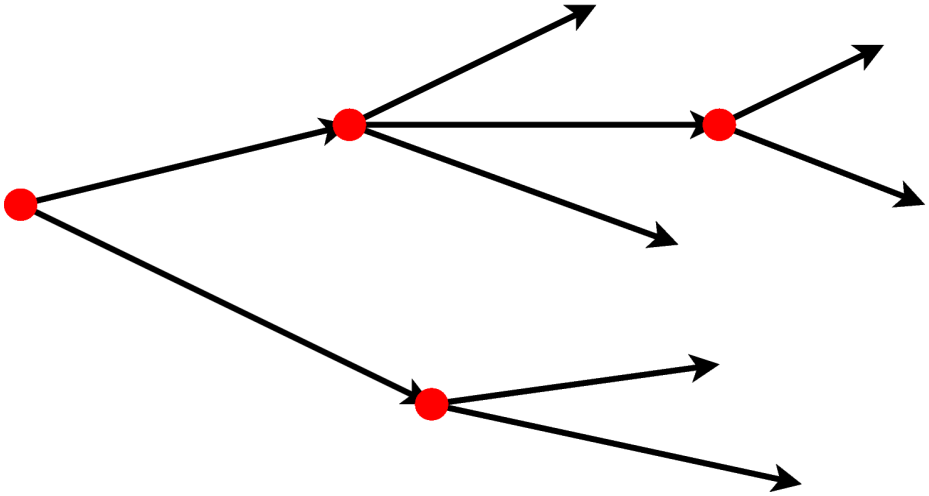
Algorithm as in P. Avery write-ups.

Vertex Fitter

PndVtxFitter

Only global chisquare, and only vertex position covariance.





SetBeamConstraint(int)

Algorithm as in P. Avery write-ups?

Conclusion & Known Issues

- Rho is cleaned from much “waste”
- Fitter interface is redesigned
- We now know what Rho does

Known Issues

- No neutral particles in Vertex Fitters allowed (ideas present)
- Fitters give small chisqares (under investigation)