A Decay Tree Fitter for PANDA

Ralf Kliemt

Helmholtz-Institut Maiz, GSI Darmstadt

Dec. 2015

HIM
Helmholtz-Institut Mainz

GSI

Decay Fits in Rho until now

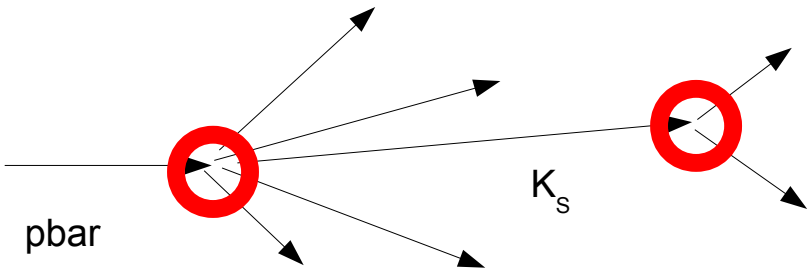
Vertex Fit Corrects final state momenta to one common point along trajectories (PndKinVtxFitter or PndKalmanVtxFitter)

Kinematic Fit Corrects daughter momenta to meet the mass or 4-momentum constraint

Executing fits subsequently and with locking some candidates, a leaf-by-leaf structure is created.

Example

1. Vertex fits for K_S and rest of tracks.
2. Mass constraint fit with vertex fitted K_S daughters
3. Locking K_S daughters
4. 4C fit on rest & K_S



Decay Tree Fitter

Fits the whole decay tree at once. Vertices, known masses, measured tracks & neutrals and beam/target measurement ("4C") are included as constraints. The common approach is the χ^2 fit with Lagrange multipliers.

$\rightarrow$ Very large parameter space and large matrices have to be inverted!

Solution: Kalman Filter approach

- Calculation of χ^2 is linearized
- Each constraint to the fit enters as one separate, scalar term
- Measurements are constraints and are treated similar to, e.g four-momentum conservations
  $\rightarrow$ maximum matrix dimension to be inverted is usually 5 (helices).
- Do not confuse with our track fitting!

Fit Parameters

(3) Primary vertex
(3) Secondary vertices
(3) Final state momenta
(4) Composite's four-momenta

Constraints

(5) Tracks (helix parameters)
(3) Clusters
(4) Initial four-momentum
(4) Internal four-momentum conservations
(1) Mass constraints

Fit Parameters

(3) Primary vertex
(3) Secondary vertices
(3) Final state momenta
(4) Composite's four-momenta

Constraints

(5) Tracks (helix parameters)
(3) Clusters
(4) Initial four-momentum
(4) Internal four-momentum conservations
(1) Mass constraints

Example

$$\psi(2S) \rightarrow J/\psi\ \pi^+\pi^-$$
$$\hookrightarrow \mu^+\mu^-$$

23 Parameters:

12 4 Final State Particles
8 2 Composites
3 Primary vertex

32 Constraints:

20 4 Helices
8 2 P4-Conservations
4 Beam-Target

$\rightarrow$ 9 Degrees of Freedom

Beam: 4 plus Vertex: $2n - 3 = 5$

Existing Decay Tree Fitter

- BaBar & LHCb have a Tree Fitter, written by W.Hulsbergen
- The author provided us the latest stable code.
- Our goal: Implementation into PandaRoot

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Nuclear Instruments and Methods in Physics Research A 552 (2005) 566–575

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

www.elsevier.com/locate/nima

Decay chain fitting with a Kalman filter

Wouter D. Hulsbergen*

University of Maryland, College Park, MD 20742, USA

Received 4 March 2005; received in revised form 21 June 2005; accepted 26 June 2005
Available online 26 July 2005

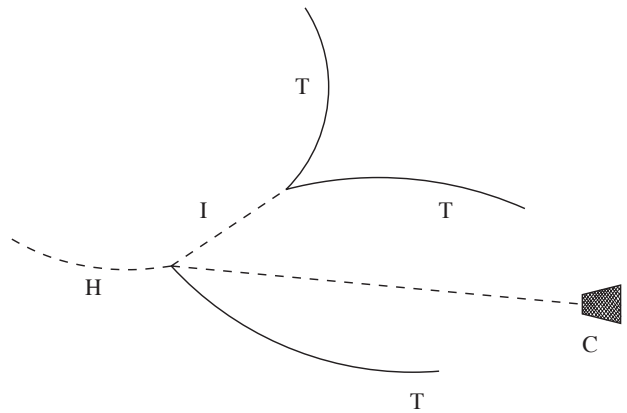


Fig. 1. Schematic picture of a decay tree with three charged particles reconstructed as track segments (T), one photon reconstructed as a calorimeter cluster (C), and two composite particles (I for 'internal' and H for 'head').

Status from last report:

- ✓ Obtain the code & look for showstoppers
- ✓ Matrices & Vectors: CLHEP → ROOT
- • Framework interfaces: Gaudi → FairBase/ROOT
  & LHCB → PandaRoot
- • Candidate Interfaces via Rho (calculations to be transformed)
- × Running Tests & Debugging

Status now:

- ✓ Obtain the code & look for showstoppers
- ✓ Matrices & Vectors: CLHEP → ROOT
- ✓ Framework interfaces: Gaudi → FairBase/ROOT
  & LHCB → PandaRoot
- ✓ Candidate Interfaces via Rho (calculations to be transformed)
- ✓ Move all particle fitters to Rho
- ✓ Intense Debugging & understanding the code
- • Running more Tests & Debugging

→ See our SVN trunk from rev. 28746

$\psi(2S) \to J/\psi\,\pi^+\pi^-$

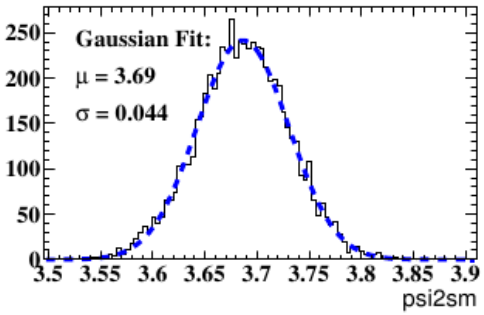
```
theAnalysis->FillList(muplus,  "MuonLoosePlus", pidalg);
theAnalysis->FillList(muminus, "MuonLooseMinus", pidalg);
theAnalysis->FillList(piplus,  "PionLoosePlus", pidalg);
theAnalysis->FillList(piminus, "PionLooseMinus", pidalg);

jpsi.Combine(muplus, muminus);
jpsi.SetType(443);

psi2s.Combine(jpsi, piplus, piminus);
psi2s.SetType(100443);

for (j=0;j<psi2s.GetLength();++j)
{
  // TREE FITTER
  PndDecayTreeFitter treefit(psi2s[j],inie,ipe);
  treefit.SetToleranceZ(0.001); // track propagation precision (cm)
  treefit.setVerbose(0);
  //treefit.setMassConstraint(psi2s[j]->Daughter(0)); //mass from pdt
  //treefit.setMassConstraint(psi2s[j]->Daughter(0)),3.096916);
  treefit.Fit();

  RhoCandidate* psi2sfit=psi2s[j]->GetFit();
```

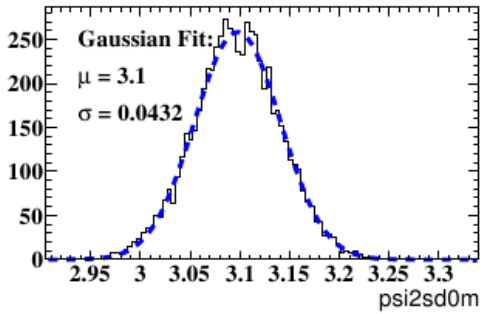
Interaction point

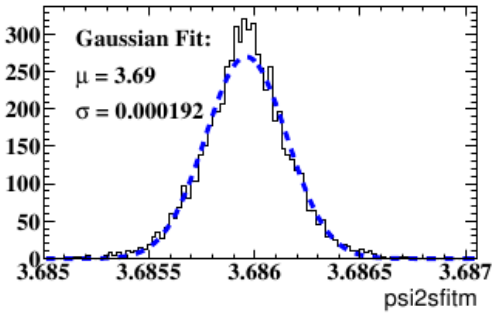
```
// *** the lorentz vector of the initial system
double m0_p  = TDatabasePDG::Instance()->GetParticle("proton")->Mass();
TLorentzVector ini(0, 0, pbarmom, sqrt(m0_p*m0_p + pbarmom*pbarmom) + m0_p);
double beamres = 1e-4*pbarmom;
RhoError inicov(4);
inicov[0][0]=1e-6*1e-6;
inicov[1][1]=1e-6*1e-6;
inicov[2][2]=beamres*beamres;
inicov[3][3]=beamres*beamres/(m0_p*m0_p + pbarmom*pbarmom);
RhoLorentzVectorErr inie(ini,inicov);

// *** beam spot
TVector3 ip(0.,0.,0.);
RhoError ipcov(3);
//ipcov[0][0]=0.005*0.005;
//ipcov[1][1]=0.005*0.005;
//ipcov[2][2]=0.005*0.005;
RhoVector3Err ipe(ip,ipcov);
```

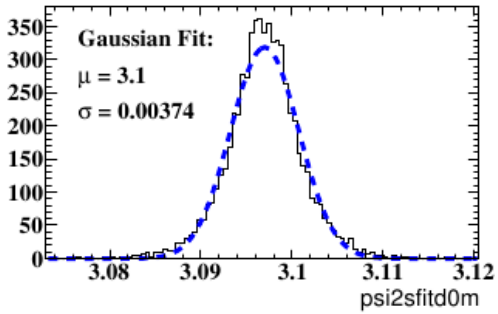
$\psi(2S) \to J/\psi\,\pi^+\pi^-$ (FastSim)

$\psi(2S)$ and J/ψ masses before and after fit



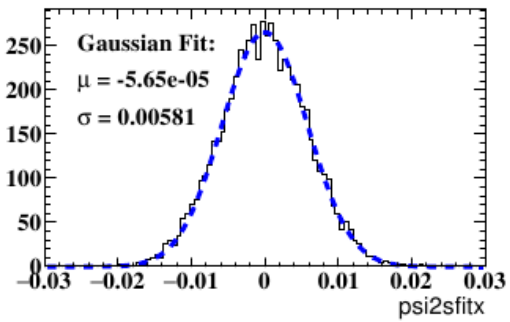


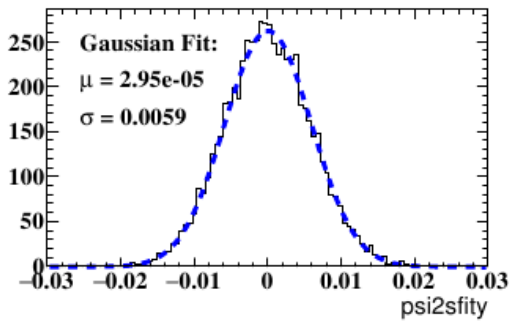


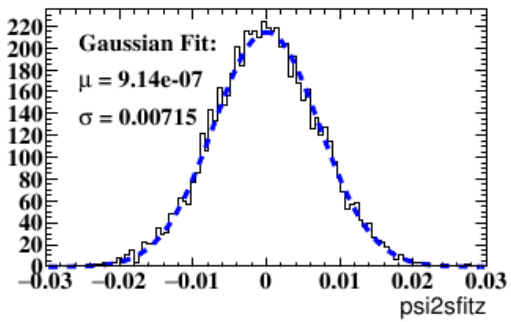


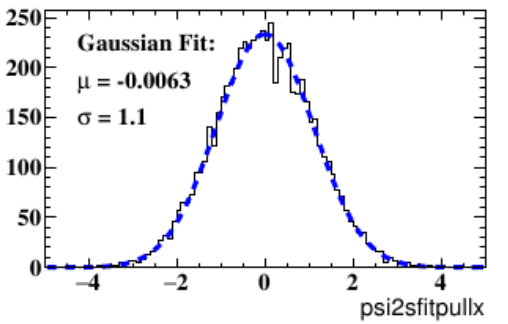
$\psi(2S) \to J/\psi\,\pi^+\pi^-$ (FastSim)

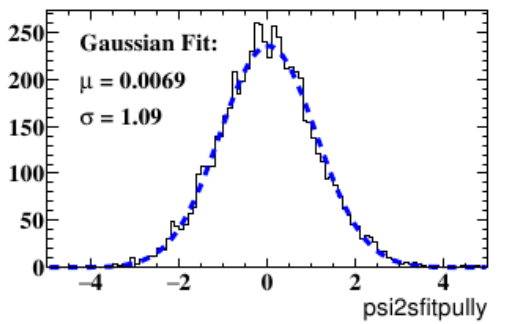
Primary vertex (smeared with $50\ \mu m$) distribution & pulls

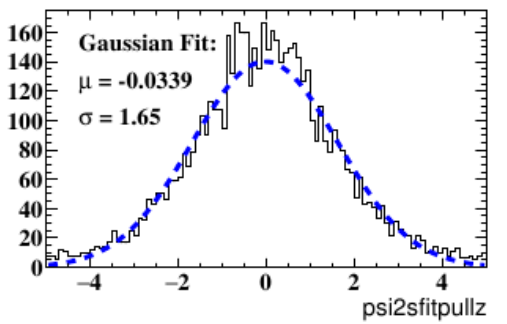






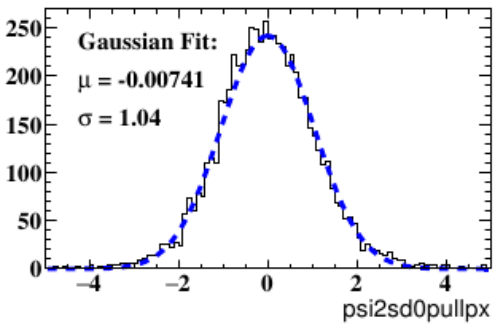


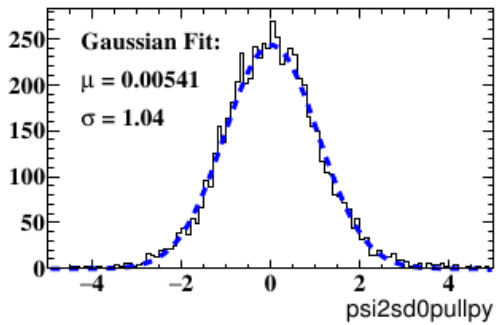


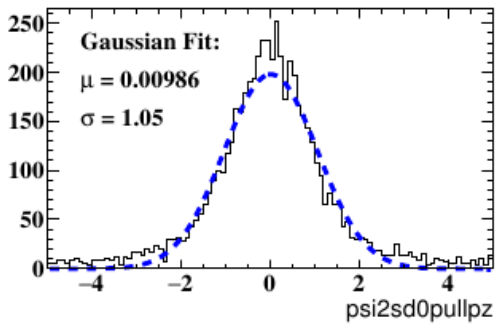


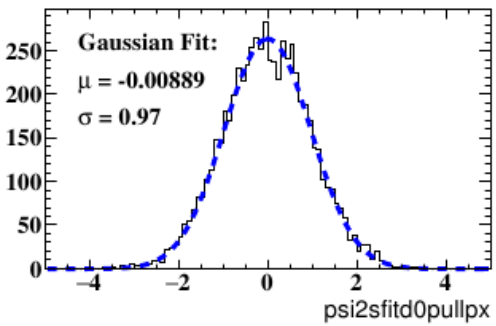
$\psi(2S) \to J/\psi\,\pi^+\pi^-$ (FastSim)

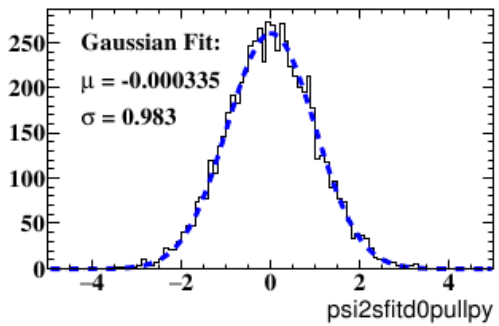
J/ψ momentum pulls before and after fit

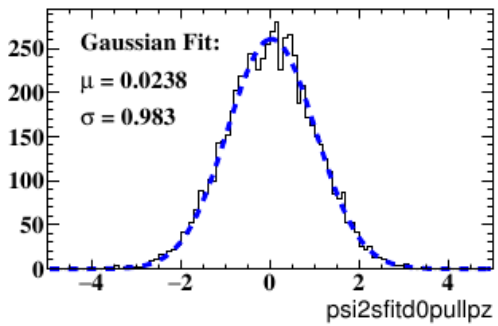






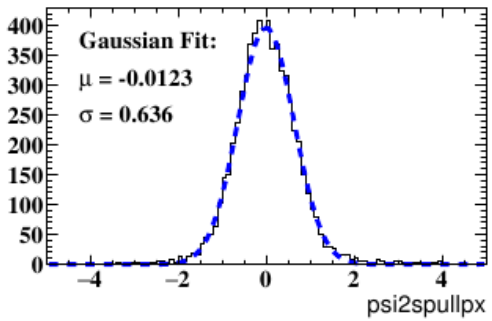


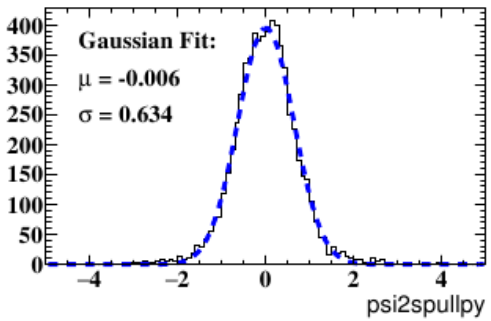


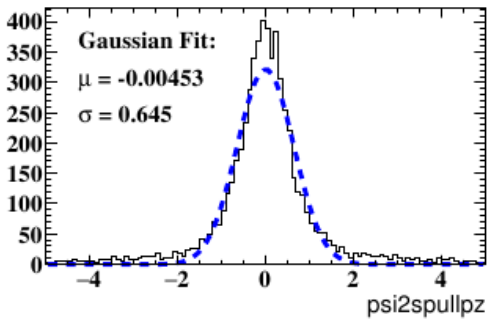


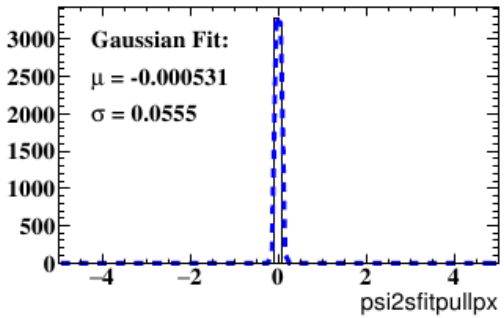
$\psi(2S) \to J/\psi\,\pi^+\pi^-$ (FastSim)

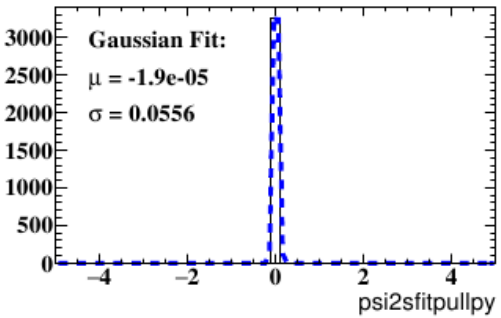
$\psi(2S)$ momentum pulls - before fit not fully understood after fit without meaning (4C)

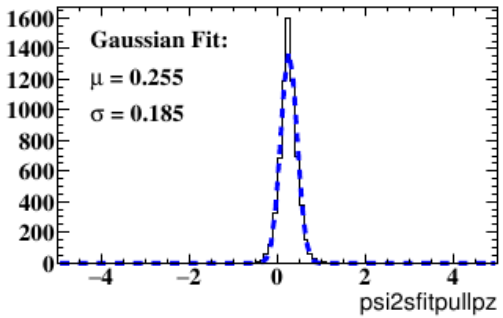






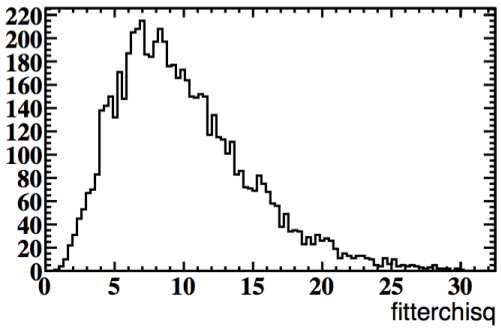


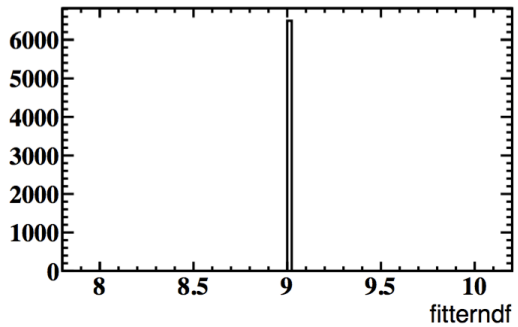


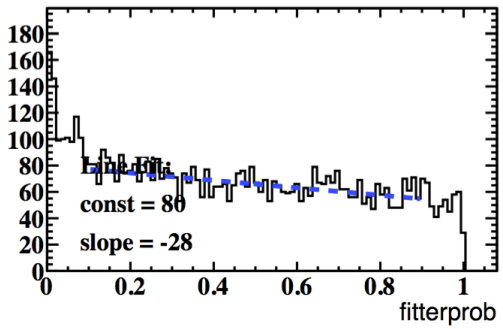


$\psi(2S) \to J/\psi\,\pi^+\pi^-$ (FastSim)

Fit Quality







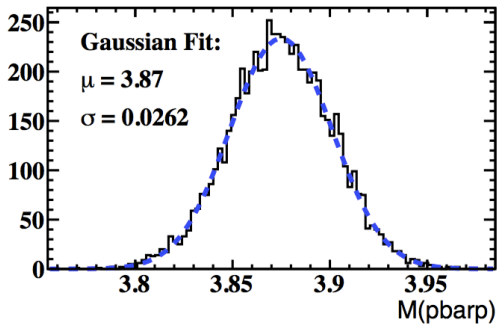
Not perfect, yet.

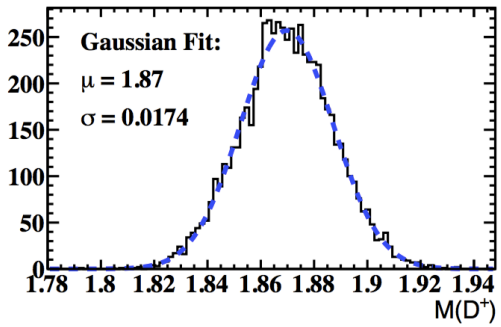
$pbarp \to D^+D^- \to K\pi\pi K\pi\pi$

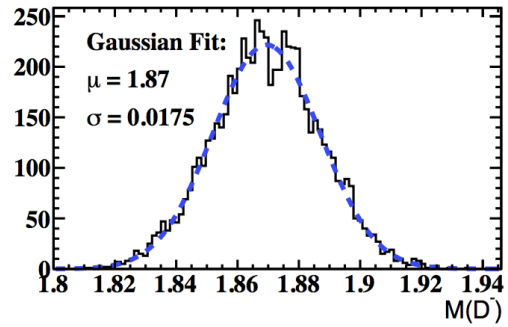
```
Dplus.Combine(Kminus, piplus, piplus);
//DpMassSel.Select(Dplus);
Dplus.SetType(411);
Dminus.Combine(Kplus, piminus, piminus);
//DpMassSel.Select(Dminus);
Dminus.SetType(-411);

pbarp.Combine(Dplus, Dminus);

for (j=0;j<pbarp.GetLength();++j)
{
  cout<<" --- cand "<<j<<" in evt "<<i<<endl;
  // TREE FITTER
  PndDecayTreeFitter treefit(pbarp[j],inie,ipe);
  treefit.SetToleranceZ(0.01); // 0.001cm = 10um, for helix linearization limit
  treefit.setVerbose(0); // full ourtput: (7);
  //pdt constraints  add ",mymass);" for manual mass
  //treefit.setMassConstraint(pbarp[j]->Daughter(0),m0_Dp); //pi0
  //treefit.setMassConstraint(pbarp[j]->Daughter(1),m0_Dp); //pi0
  treefit.Fit();

  RhoCandidate* pbarpfit=pbarp[j]->GetFit();
  ntp->Column("pbarp_fitstat",treefit.status());
  ntp->Column("pbarp_chisq",treefit.GetChi2());
  ntp->Column("pbarp_ndf",treefit.GetNdf());
  ntp->Column("pbarp_prob",treefit.GetProb());
```

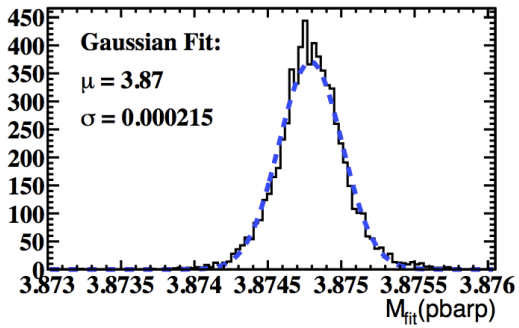
$pbarp \to D^+D^- \to K\pi\pi K\pi\pi$ (FastSim)

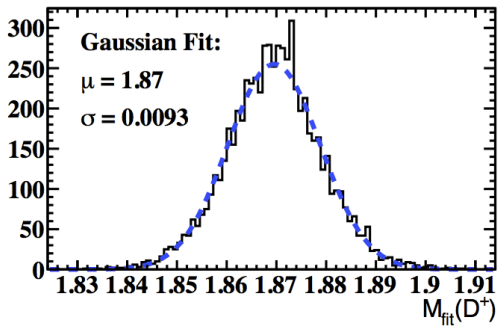
Masses before and after fit

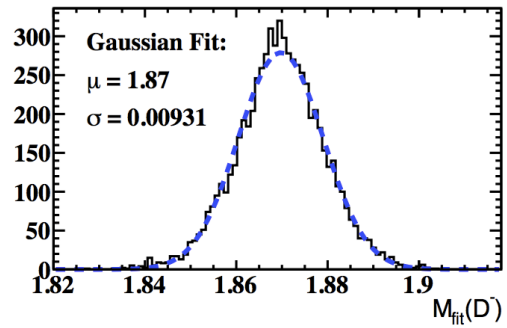






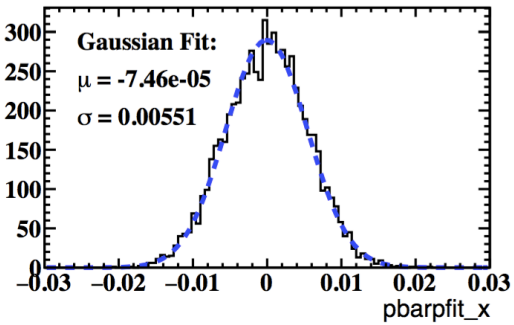


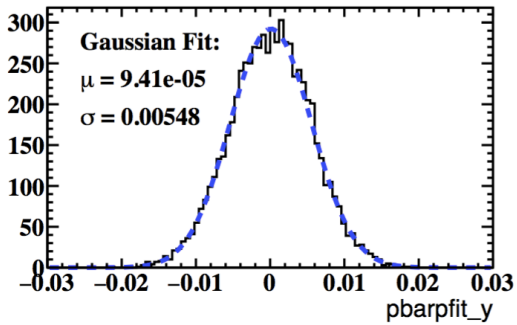


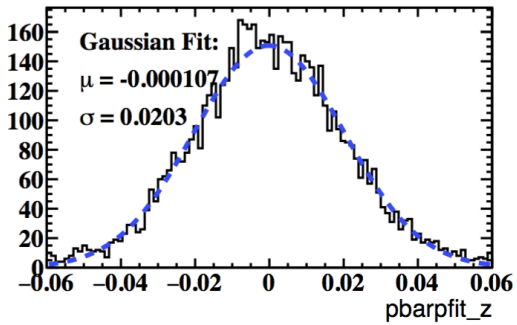


$pbarp \to D^+D^- \to K\pi\pi K\pi\pi$ (FastSim)

Primary Vertex ($50\ \mu m$ smearing and reconstructed from D's

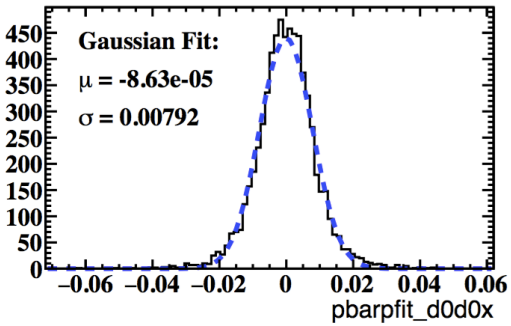


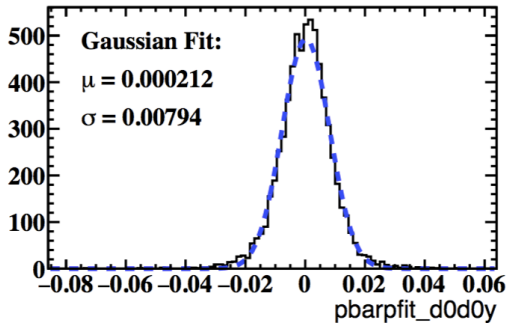


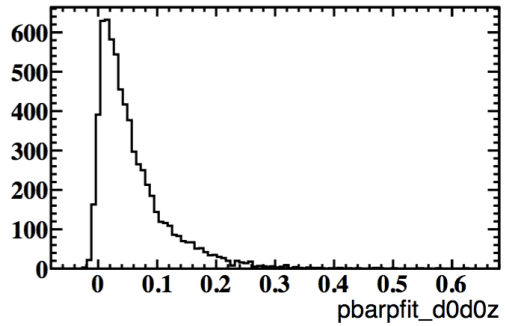


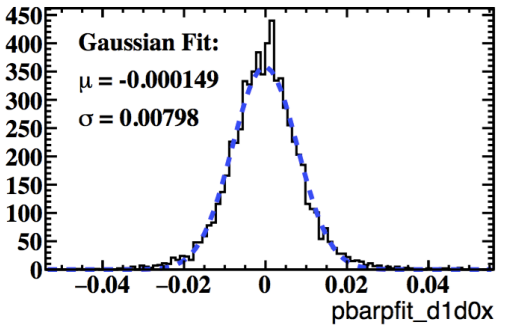
$pbarp \to D^+D^- \to K\pi\pi K\pi\pi$ (FastSim)

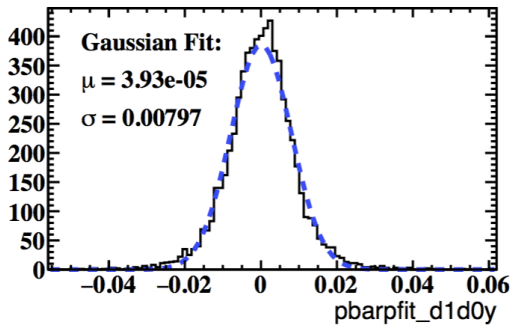
Secondary Vertices D^+ and D^-

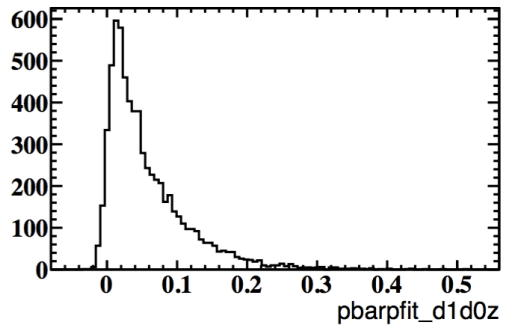






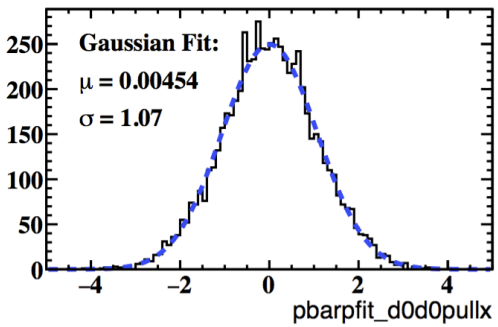


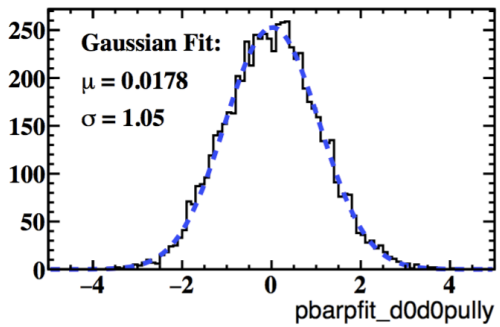


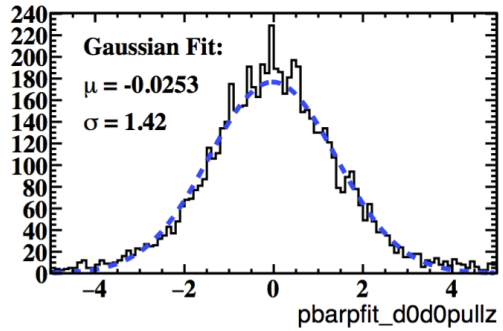


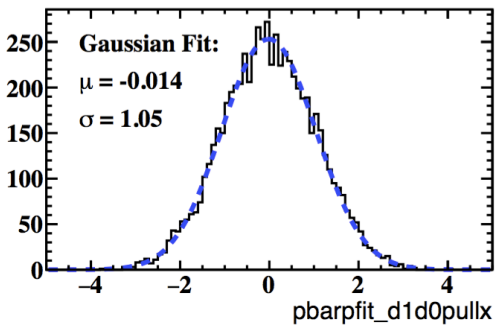
$pbarp \to D^+D^- \to K\pi\pi K\pi\pi$ (FastSim)

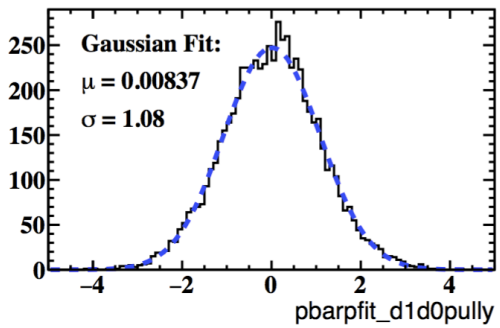
Secondary Vertex Pulls D^+ and D^-

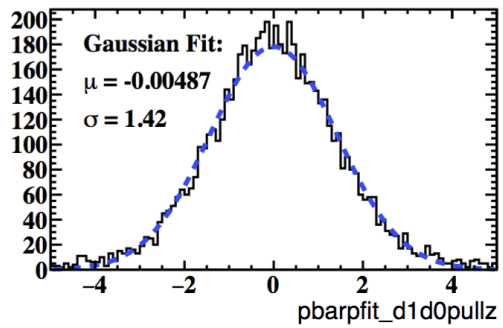






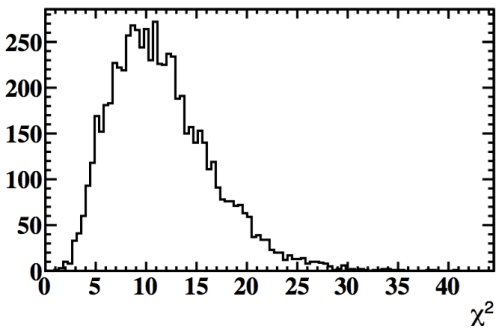


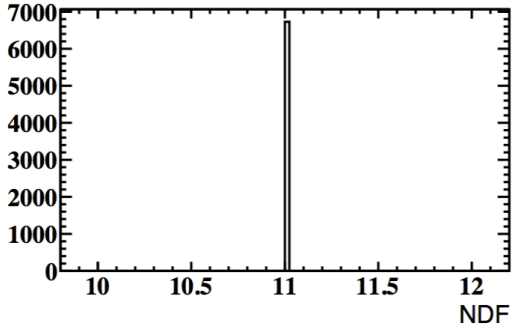


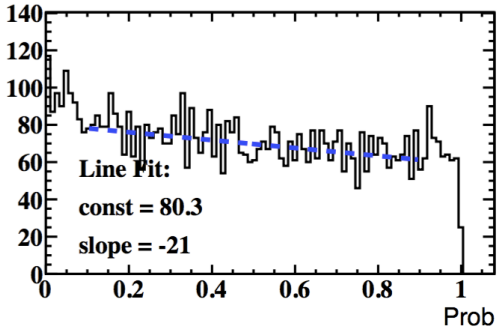


$pbarp \to D^+D^- \to K\pi\pi K\pi\pi$ (FastSim)

Fit Quality







Outlook

- Covariance issue to be investigated.
- Neutrals need possibly fixing (may be an fsim thing)
- Treat "stabe" particles (Kshort) properly
- Test delayed Vertices (D's, Lambdas)
- Compare performance with available fitters

Outlook

- Covariance issue to be investigated.
- Neutrals need possibly fixing (may be an fsim thing)
- Treat "stabe" particles (Kshort) properly
- Test delayed Vertices (D's, Lambdas)
- Compare performance with available fitters

Thanks for your attention.