Preparing for CLAS12 Data Analysis
HASPECT Working group

Derek Glazier
University of Glasgow

05/09/2017

Experimental Data Analysis

Goal : Extract physics observables from experimental data distributions

Issues: To extract reliable results for theoretical interpretation we must

Account for acceptance

Account for resolutions

Account for backgrounds

...

+ Know how well we have handled these (systematic uncertainties)

Additional Goals

- Do this quickly and reliably
  - So supervisors/theorists do not get bored or annoyed
- Develop well defined procedures and software tools
  - Save time on analysis
  - Save time on analysis reviews
  - Spend time developing more sophisticated analysis rather than re-inventing the wheel
- In line with
  - CLAS Common Tools committee recommendations
  - Analysis Committee of Experts

Software Methodology

- Use ROOT
  - common usage, well maintained
  - Many relevant data handling tools
  - Many relevant statistical tools
  - Supplement with more specific functionality
- Modules for Signal Selection and Observable Extraction
- Separate out event loop action from event handling
  - Run same code on real data, simulated, generated
  - Run same code on ROOT, hipo, LUND … files
- Provide skeleton code builder for users to build on
  - Keep ROOT/C++ flexability, but add structure
- Integrate new tools and techniques for others to use
- Maintained on github with doxygen documentation
- Developed/tested by HASPECT collaborators on CLAS data and CLAS12 simulations

Github and doxygen

HASPECT6 (Peter Pauli, Glasgow)

Hadron spectroscopy data analysis

Main Page | Classes | Files | Search

Class List | Class Index | Class Hierarchy | Class Members

Class Index

R | T

R

RooHS1StepStudy
RooHSAbsEventsPDF
RooHSEventsHistPDF
RooHSEventsPDF
RooHSProtoEventsPDF
RooHSsPlotAndFitStudy
RooHSStudyManager

T

THSBins
THSDataManager
THSDataReader
THSEventWeighter
THSFinalState
THSFinalTemp
THSHipoReader
THSHipoTrigger
THSHisto
THSIsobarPS
THSKinematics
THSLongPS
THSLundReader
THSOutput
THSParticle
THSPartOutput
THSProject
THSProjTemp
THSRADSReader
THSRooFit
THSSkeleton
THSsPlot
THSTxtTreeReader
THSWeights

R | T

Generated on Tue Oct 3 2017 10:33:35 for HASPECT6 by doxygen 1.8.11

https://github.com/dglazier/HASPECT6

dglazier / HASPECT6

Unwatch 1 | Star 0 | Fork 3

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

Hadron spectroscopy data analysis

Edit

Add topics

84 commits | 2 branches | 0 releases | 3 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

dglazier fix THSsPlot CreateSubFit and THSRooFit LoadWorkSpace to delete fWS f… Latest commit 9303cc9 4 days ago

ExtraPackages/rhipo	change rhipo setenv	2 months ago
HaSpect	fix THSsPlot CreateSubFit and THSRooFit LoadWorkSpace to delete fWS f…	4 days ago
Projects	Change Project to FinalState, and copy THSParticle from THSCLAS12Part…	a month ago
RooFitExamples	rm hslogin.C for merging	4 days ago
.gitignore	Add *.root to .gitignore	4 days ago
Doxyfile	Change output path in Doxyfile to local	4 days ago
DoxygenLayout.xml	Add doxygen to HaSpect	4 days ago
README.md	Update README.md	5 months ago
git_guide.txt	Add a how-to for git	4 days ago
gitcommands.txt	Add Hipo2Root to hslogon	5 months ago

README.md

To configure set HSANA variable to HaSpect directory

setenv HSANA /path_to_here/HaSpect

Copy the $HSANA/rootrc file to your top level directory. If you already have a .rootrc file you can just copy the lines from within rootrc

THSFinalState Class Reference

List of all members | Public Member Functions | Protected Mem

Prote

Definition at line 30 of file THSFinalState.h.

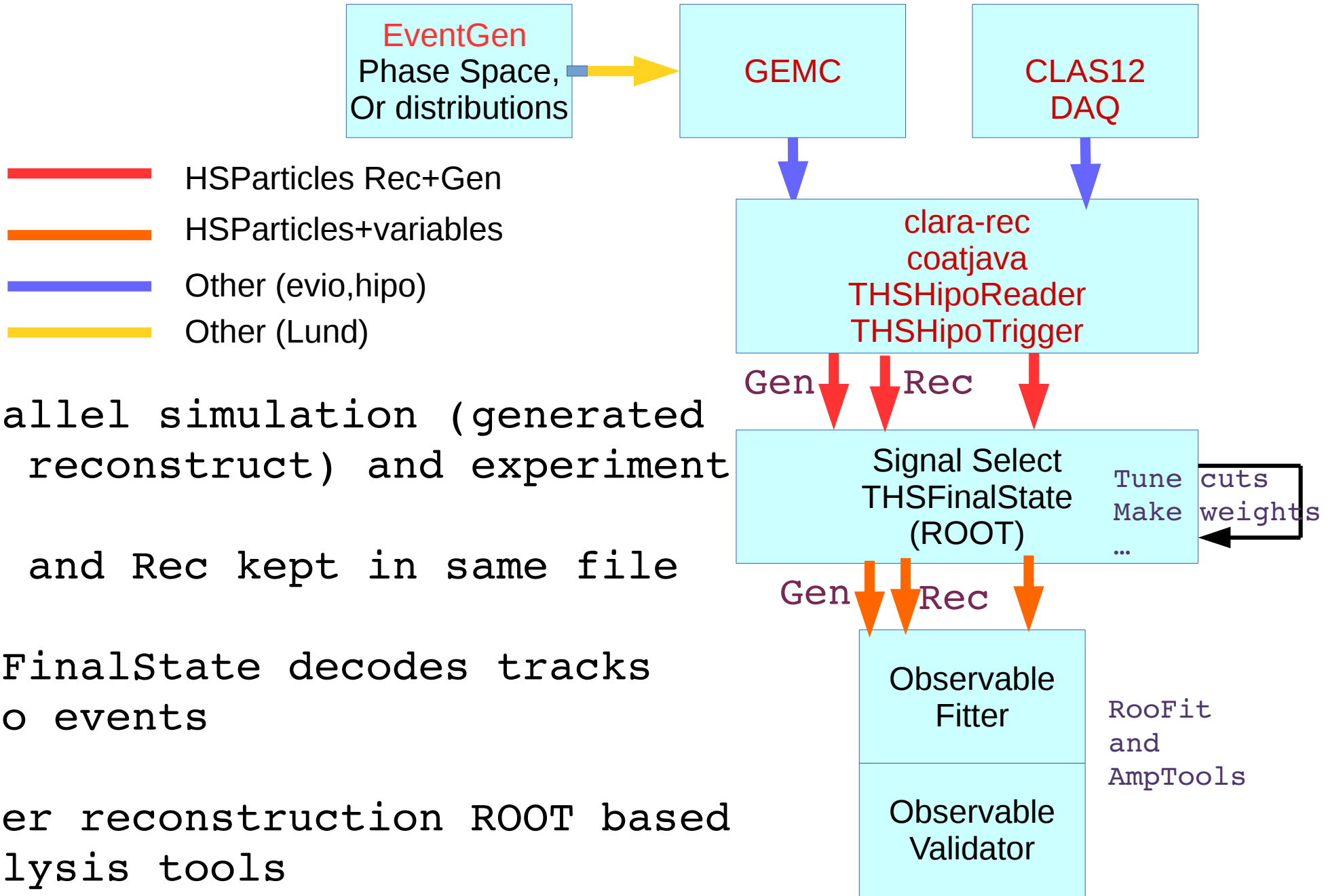
```
#include <THSFinalState.h>
```

Inheritance diagram for THSFinalState:

Public Member Functions

	THSFinalState ()
virtual	~THSFinalState ()
virtual void	GetEvent (Long64_t uid)
virtual Bool_t	WorkOnEvent ()
virtual void	FinaliseEvent ()
virtual void	InitEvent ()
void	SetDetParts (vector< THSParticle * > *dpp)
void	SetGenParts (vector< THSParticle * > *dpp)
Int_t	AddTopology (TString topo)
vector< Int_t >	GetTopology (Int_t i)
Int_t	FindTopology ()
Int_t	FindInclusiveTopology (Int_t incType=-1E9)
void	SetPermutate ()
Bool_t	RotatePartVector (vector< THSParticle * > *vec, Int_t *Nturns)
void	SetGenerated (Bool_t gen=kTRUE)
Bool_t	IsGoodEvent ()
virtual Bool_t	IsPermutating ()
virtual void	ProcessEvent ()
virtual void	MatchWithGen (THSParticle *part)
virtual Bool_t	IsCorrectTruth (THSParticle *part)
virtual void	FinalStateOutTree (TTree *tree)
virtual void	CheckTruth ()

HASPECT Data Analysis



Parallel simulation (generated and reconstruct) and experiment

Gen and Rec kept in same file

THSFinalState decodes tracks into events

After reconstruction ROOT based analysis tools

- HipoInRoot Interface
  - Extract info direct from hipo banks (THSHipoReader)
  - Insert tracks into HSParticle event array (THSHipoReader)
  - Additional trigger information (THSHipoTrigger)
- HSParticle class
  - 4-vector; vertex; path; ToF; time for PID hypothesis; DOCA; truth information; …
- HSFinalState class
  - Decode particle event array into exclusive final state – include all topologies
  - Use COATJAVA PID or just charge/time info
    - Permutate all possible combinations of like tracks, FT electrons,...
  - Same code for simulation and real data
  - Use with ROOT TSelector – PROOF (parallel processing)
- Generate event ntuple for observable extraction

- Tested on simulated CLAS12 and real CLAS data
- Improving user interface, tutorials

- All observables extracted via Extended Maximum Likelihood fits
  - Polarisation Observables, Spin Density Matrix Elements, Angular Moments; Partial Waves;...
  - Simulated events used to correct for acceptance via normalisation integral
  - TOYMC method used to correct for detector distortions
  - Backgrounds accounted for using event weights
    - sWeights, Q-factor, sidebands,…
- Software based on RooFit or IU-AmpTools
  - Adding repository of standard observable fit functions or amplitudes

- Tested on simulated CLAS12 and real CLAS data
- Improving user interface, tutorials

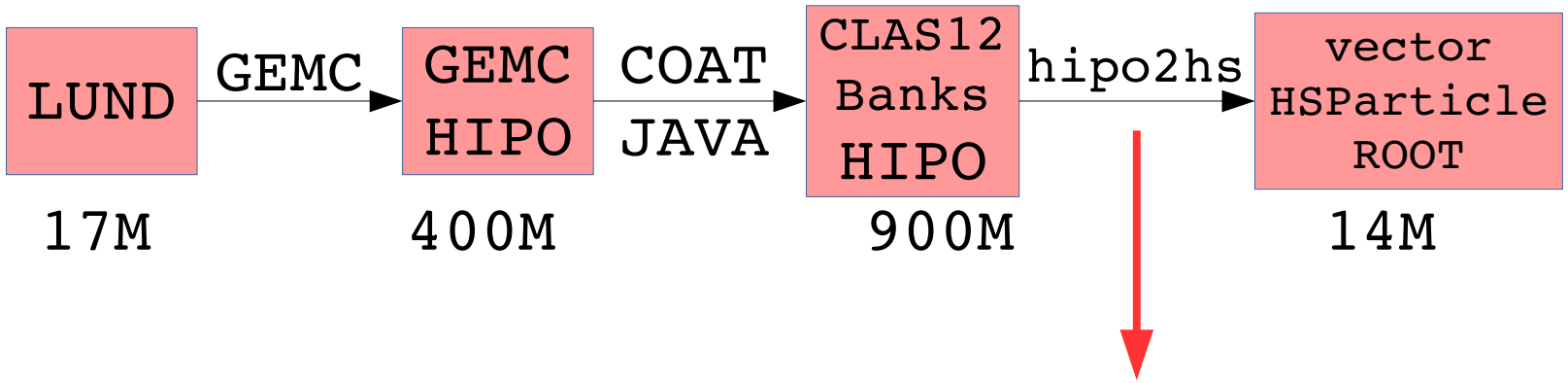
Event Structure : vector<THSParticle>

```
class THSParticle {
 private:
 protected:
  TLorentzVector fP4;     //4-vector
  TVector3 fVertex;       //vertex
  Int_t fPDGCode;         //PDG number
  Double_t fPDGMass;      //PDG mass value
  Double_t fMeasMass;     //Measured Mass
  Double_t fTime;         //TimeOfFlight
  Double_t fPath;         //FlightPath
  Double_t fDoca; //D of Closest approach
  Double_t fEdep;         //Energy deposit
  Int_t fDetector=0;      //detector code

  TLorentzVector fTruthP4; //true P4
  TVector3 fTruthV;          //true vertex
  Int_t fTruthPDG;           //true PDGcode
```

Example usage

```
//sum 2 4-vectors
TLorentzVector pq=p.P4()+q.P4()
//sum 2 HSParticles into a Λ
lambda.Add(p,q,3122);
→Sum 4-vectors, DOCA vertex,
average time, MeasMass=invariant
//Check Mass – Mass(PDG)
p.MassDiff()
//Check ToF-ToF(PDG)
p.DeltaTime()
//Get the generated values
p.TruthP4();...
//Check deviation from true
p.ResTheta(); p.ResPhi();...
```

Convert CLAS12 hipo files




```
hipo2hs –hsin=hipo_dir/ --hsout=root_dir/
```

With Stefan Diehl

* Note 1) additional event information can be added via THSHipoTrigger class
2) For simulated data generated particles are passed on regardless of CLAS12 event
3) Also THipo class which converts to ROOT banks

hipo2hs THSHipoReader Class

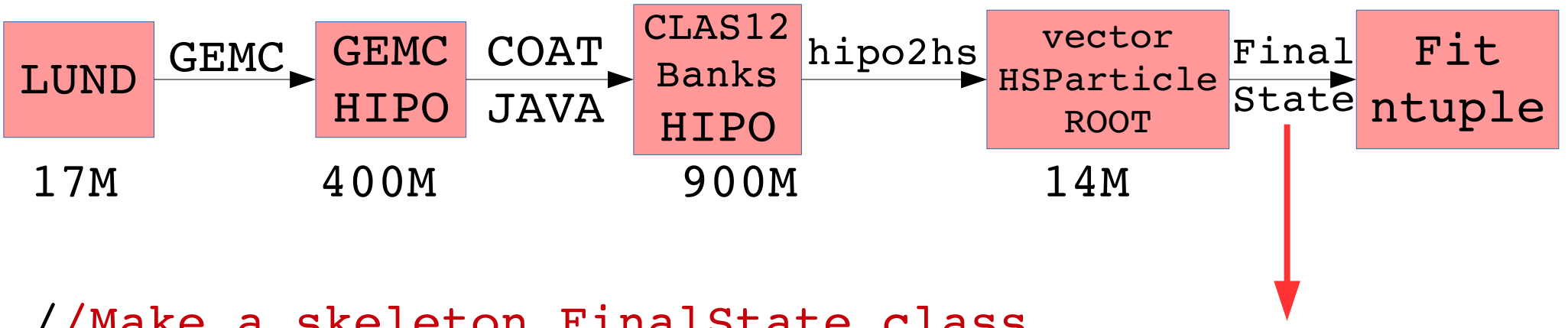
Link class to Hipo banks

```
//Get the EventBuilder banks
fHipo->ConfigBank("REC::Particle");
fHipo->ConfigBank("REC::Scintillator");
fHipo->ConfigBank("REC::ForwardTagger");
fHipo->ConfigBank("REC::Event");

//Get the necessary items from Particle Bank
fPBank=fHipo->GetBank("REC::Particle");
fPx=fPBank->GetItem("px");
fPy=fPBank->GetItem("py");
fPz=fPBank->GetItem("pz");
fVx=fPBank->GetItem("vx");
fVy=fPBank->GetItem("vy");
fVz=fPBank->GetItem("vz");
fPid=fPBank->GetItem("pid");
fBeta=fPBank->GetItem("beta");
fCharge=fPBank->GetItem("charge");

//Get the necessary items from REC::Scintillator Bank
fSBank=fHipo->GetBank("REC::Scintillator");
fSPindex=fSBank->GetItem("pindex");
fSTime=fSBank->GetItem("time");
fSEnergy=fSBank->GetItem("energy");
fSPath=fSBank->GetItem("path");
fSDet=fSBank->GetItem("detector");

//Get the necessary items from REC::FT Bank
fFTBank=fHipo->GetBank("REC::ForwardTagger");
fFTPindex=fFTBank->GetItem("pindex");
fFTTime=fFTBank->GetItem("time");
fFTEnergy=fFTBank->GetItem("energy");
fFTPath=fFTBank->GetItem("path");
fFTDet=fFTBank->GetItem("detector");
```

Event Loop

```
while(fPBank->NextEntry()){//Loop over particles
  THSParticle* particle=fReadParticles->at(ip++);
  fParticles.push_back(particle);
  particle→SetXYZM(fPx→Val(),fPy→Val(),fPz→Val(),0);
  particle->SetPDGcode(fPid->ValI());
  particle->TakePDGMass();
  particle->SetVertex(fVx->Val(),fVy->Val(),fVz->Val());
//Now look for the associated detector info
//match the detector pindex to index of this particle
  while(fSPindex->FindEntry(fPBank->GetEntry())){
//Loop over scintillator
    particle->SetTime(fSTime->Val()-fEvTime->Val());
    particle->SetEdep(fSEnergy->Val());
    particle->SetPath(fSPath->Val()/100);
    particle->SetDetector(fSDet->Val());
   }
   while(fFTPindex->FindEntry(fPBank->GetEntry())){
//Loop over FT
    particle->SetTime(fFTTime->Val()-fEvTime->Val());
    particle->SetEdep(fFTEnergy->Val());
    particle->SetPath(fFTPath->Val()/100);
    particle->SetDetector(fFTDet->Val());
```

Convert HSParticles to fit variables




```
//Make a skeleton FinalState class
THSSkeleton* sk=new THSSkeleton();
//Give your project a name
sk->SetFinalState("KK",kTRUE);//creating the project class with perms

//Set the detected particle combinations you will analyse
//DETECT EVERYTHING,MISSING +VE,MISSING -VE (note diff , and :)
sk->SetFinalTopo("Rootino-:Rootino-:Rootino+:Rootino+,
                  Rootino-:Rootino-:Rootino+,Rootino-:Rootino+:Rootino+");

//Set the actual final state particles of the reaction
//These will just be used as the data member names
sk->SetFinalStateParts("Electron:e-,Proton:proton,Kp:K+,Km:K-");

//Make some code
sk->CreateMyProject();
```

THSFinalState : Init Topologies

```
void THSKK::Init_0(){
//define init for detected
Rootino-:Rootino-:Rootino+:Rootino+
//Set detected particles
  SetMinus(&fElectron,0);
  SetPlus(&fProton,0);
  SetPlus(&fKp,1);
  SetMinus(&fKm,1);
//Check for FT electron
  if(fElectron.Detector()!=9)
      {fGoodEvent=kFALSE;return;}
//Check for proton PID time
 if(fProton.DeltaTime()<1)
   {fGoodEvent=kFALSE;return;}
...
 TLorentzVector miss=
      fBeam + fTarget-
      fElectron.P4() - fProton.P4()
      -fKp.P4() - fKm.P4();
  fMissMass=miss.M();
```

And similar for missing Particle topologies

Get +ve -ve particles From the event vector<THSParticles> Or could use specific PID assignment

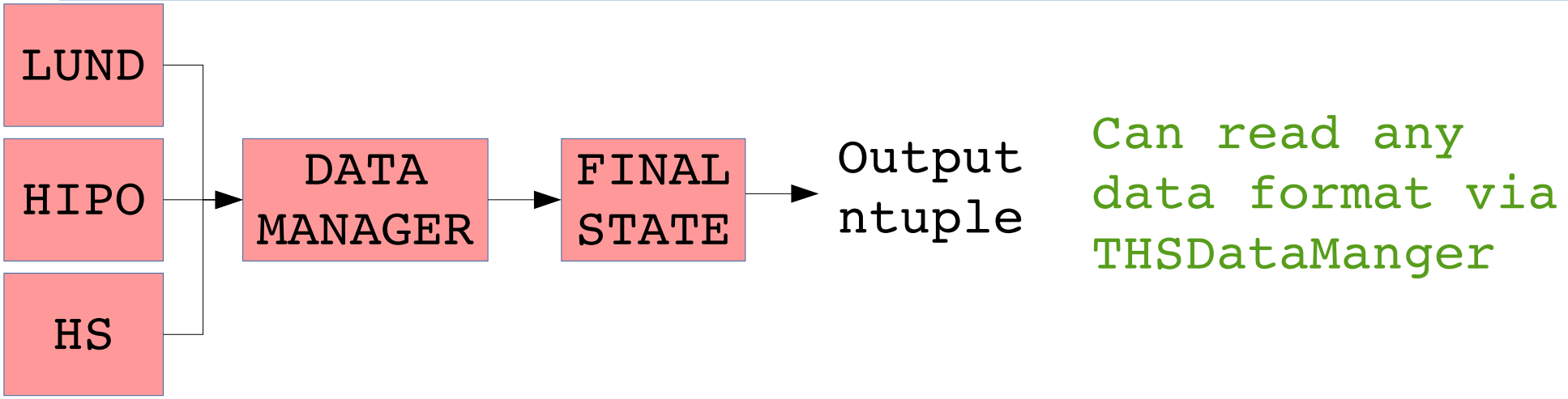
Missing mass is Topology dependent So calculate here

THSFinalState: Kine/Fit variables

```
void THSKK::Kinematics(){
//calculate K+ + K- mesonic state
  TLorentzVector KK=fKp.P4()+fKm.P4();
  fKKMass=KK.M();
//Initialise THSKinematics calculator
  fKine.SetElecsTarget(fBeam,fElectron.P4(),fTarget);
  fKine.SetMesonBaryon(KK,fProton.P4());
  fKine.SetMesonDecay(fKp.P4(),fKm.P4());
//Get Production Kinematics
  f_W = fKine.W();
  f_t = fKine.t(); //t for K+K- meson production
  f_Q2 = fKine.Q2();
  f_Pol = fKine.GammaPol();
//Get Meson+Baryon decay Kinematics
  fKine.ElectroCMDecay();
  fCMCosTh = fKine.CosTheta();
  fCMPhi = fKine.Phi();  //relative to e scattering plane
//Get Meson Decay Kinematics
  fKine.MesonDecayHelicity();
  fHelCosTh = fKine.CosTheta();
  fHelPhi = fKine.Phi();
```

These variables can be written to the Output ntuple

THSFinalState : Further Notes



If asked to permutate it will try all valid particle combinations

If simulated will flag the correct combination *

Can find inclusive events
Can find inclusive events with only 1 particle repeated e.g. FT electrons

Designed to run in HS TSelector framework
-PROOF, histogramming tools, ...

HSFit Interface for fitting data

```
THSPolObsFit* RF=new THSPolObsFit("SF");  //Manager class
RF->LoadVariable("Phi[-180,180]");//azimuthal reaction plane
RF->LoadVariable("CosThx[0,180]");//Lambda decay angle
RF->LoadVariable("CosThy[0,180]");//Lambda decay angle
RF->LoadVariable("CosThz[0,180]");//Lambda decay angle
RF->LoadVariable("Plin[-1,1]");//Linear Polarisation
RF->LoadVariable("PS[-1,1]");//Linear Polarisation State (Para or Perp)
RF->LoadBinVars("Eg",6,1,2.4);//split into 6 Egamma bins
RF->LoadBinVars("Theta",10,0,180);//split into 10 Theta bins

////////////////////////////Make Beam Recoil PDF
RF->Factory("THSPolObsPDF
                    ::Klambda(Phi,CosThx,CosThy,CosThz,Plin,PS,Sigma,Ox,Oz,T,DelLum)");
RF->LoadSpeciesPDF("Klambda");

////////////////////////////Load Data
TChain chain("HSParticles");
chain.Add("data.root");
RF->LoadDataSet(&chain);//import to RooFit

//Weight data for K0Lambda
RF->LoadWeights("out/WeightsTotal.root","HSsWeights");
RF->SetWeightName("Lambda");

////////////////////////////Fit Model to data
RF->FitBins(10);
```

User defined fit fnc

4 variables

5 free parameters

polarisation

From previous sWeight fit

New RooHSEventsPDF class

RooFit PDF classes require normalisation (integral calc.)

- -This can be done by users own method
- -Or RooFit performs is own numerical integration(vegas)

RooHSEventsPDF calculates its own integral by summing over MC events – includes partial integration for plotting

Requires ROOT tree of reconstructed MC phase space events with fit variables (i.e output of THSFinalState)

Additionally can generate events using true values if these are passed through in the tree

- Evaluate function with gen, produce events with rec

Tests if integral is constant (can be ignored for speed)

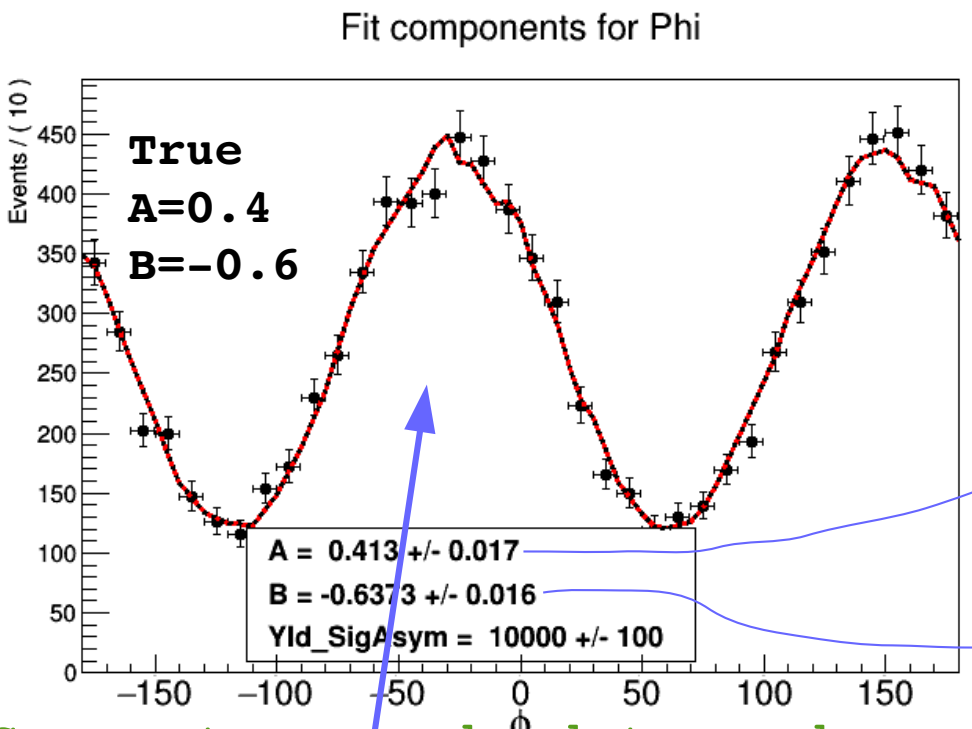
Works any number of fit observables or parameters

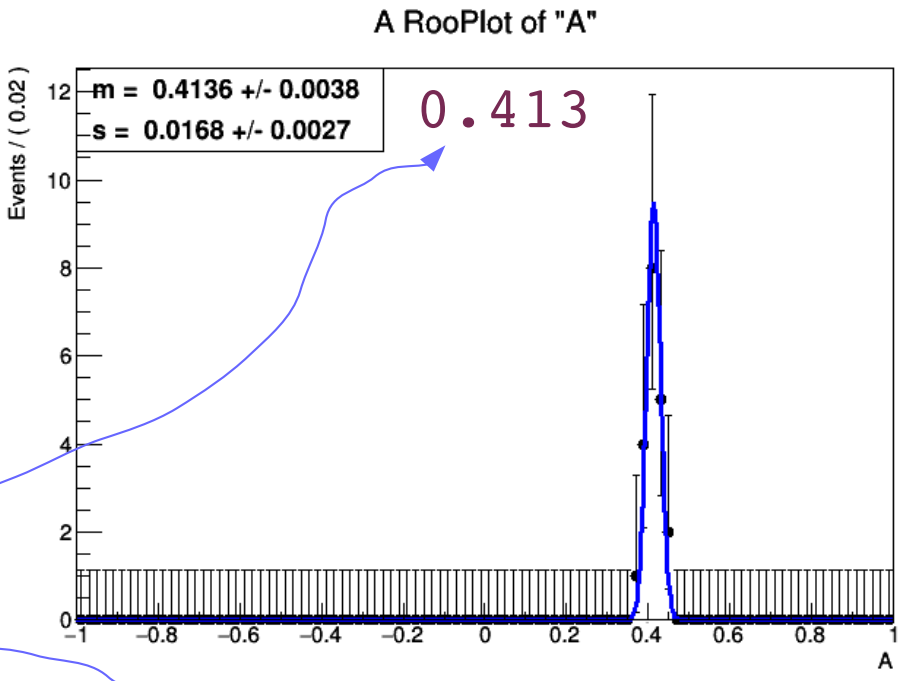
- - i.e. given a Ndim Model it will fit for Mpar parameters accounting for detector acceptance effects

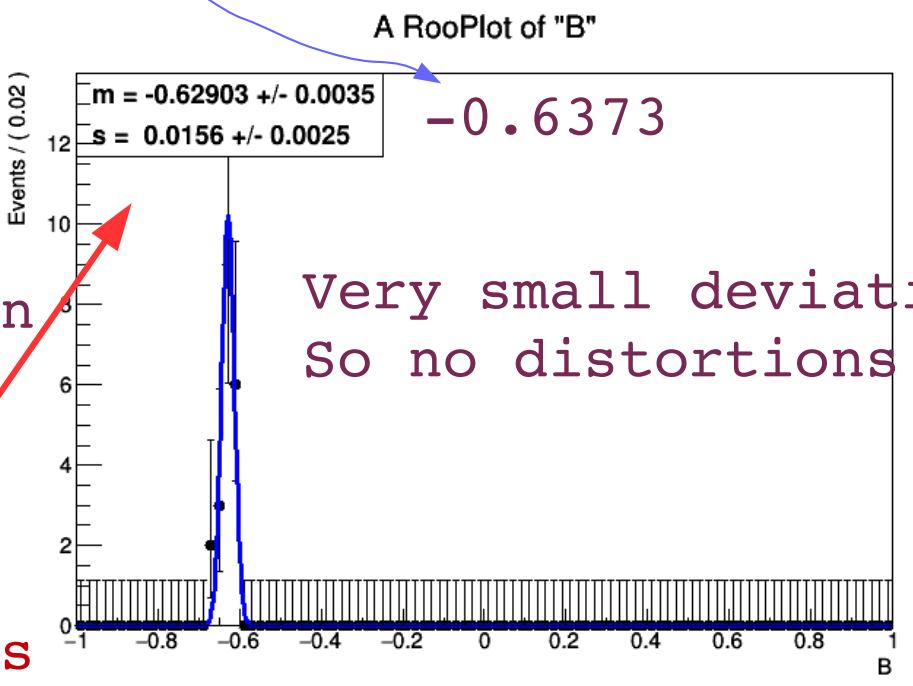
Observable Validation : RooFit MCStudy

- Extracted fit parameters will not necessarily be good estimates of true values
- Systematic errors from acceptance/resolution/bias/…
- Brute force method to estimate these is to perform many similar Toy fits using MC data
- Deviation from true values should correspond to deviations of fits to real data
- This is automated when using RooHSEventsPDF and THSRooFit manager
- Just requires large amount of simulated data!
- Note: Generated(Truth) variables are used to generate, reconstructed variables are used to fit
- Can use either accept/reject or weights for generation
- Can also be used to study planned reaction analysis
  - i.e. for CLAS12

Study Fit on ideal data







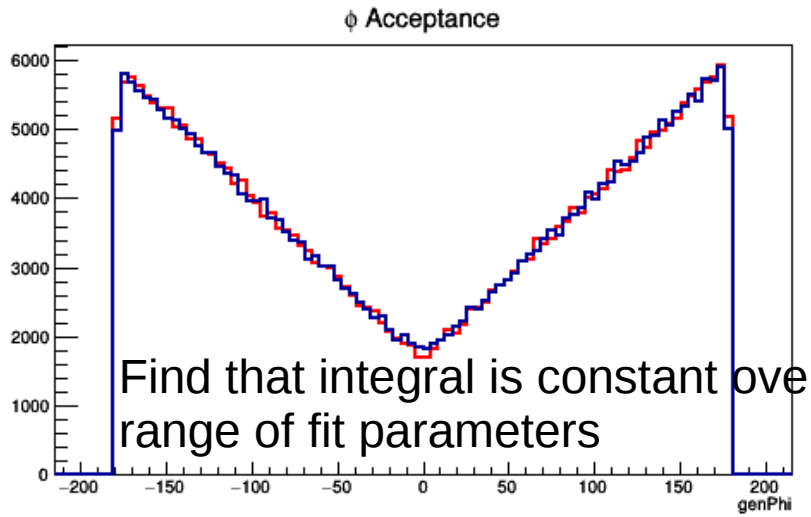
Generate pseudo data and simulation outwith RooFit :

(1+A*P*cos(2Φ)+B*P*sin(2Φ))
P varies from 0.5-1 handled on event level

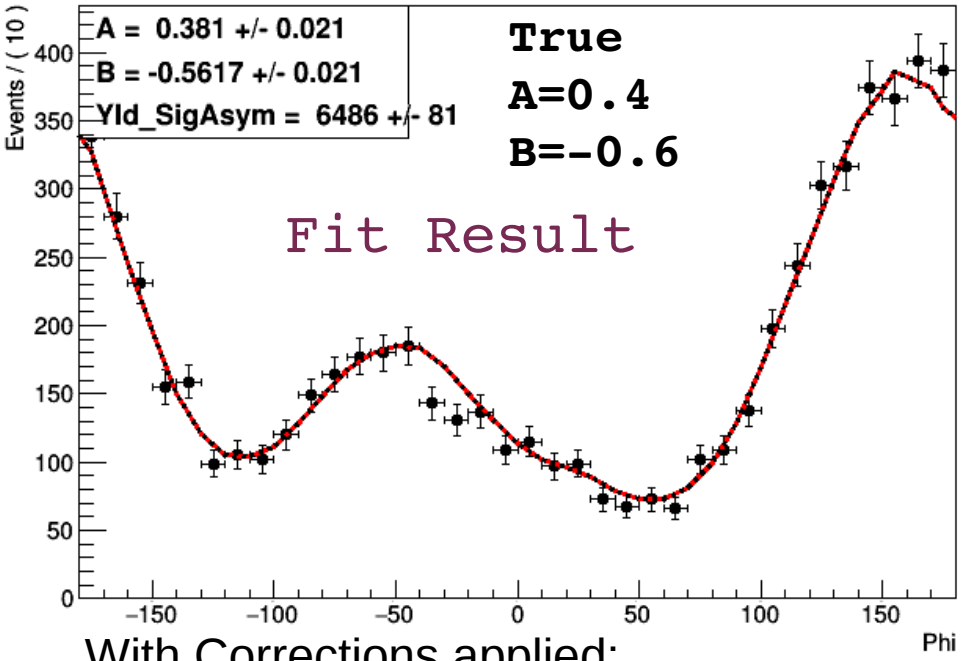
Normal fit to pseudo data
Study 20 fits to toy data based on normal fit parameters

Very small deviations
So no distortions

Acceptance + Resolution 10°



Find that integral is constant over range of fit parameters

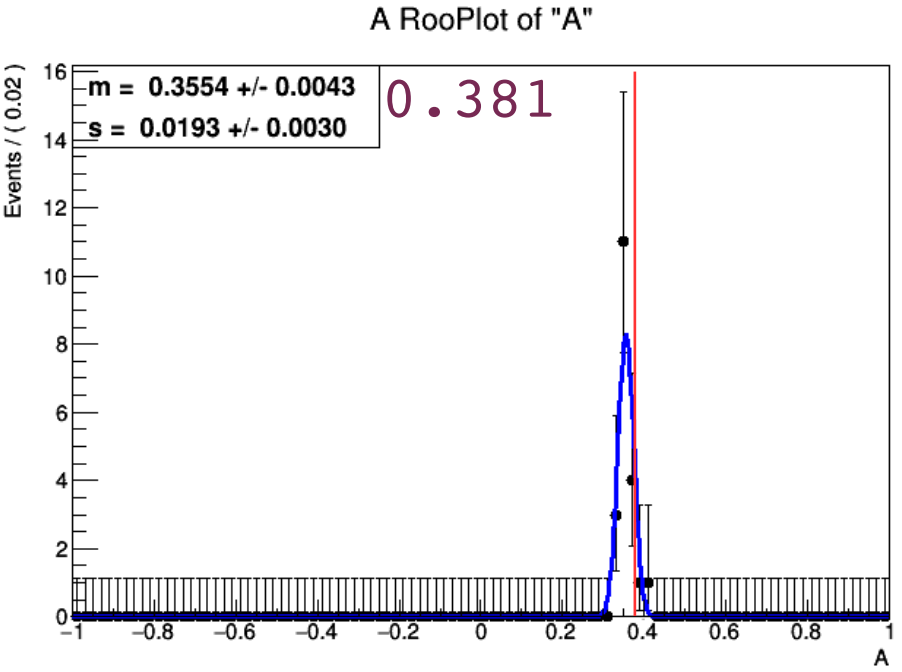


True
A=0.4
B=-0.6

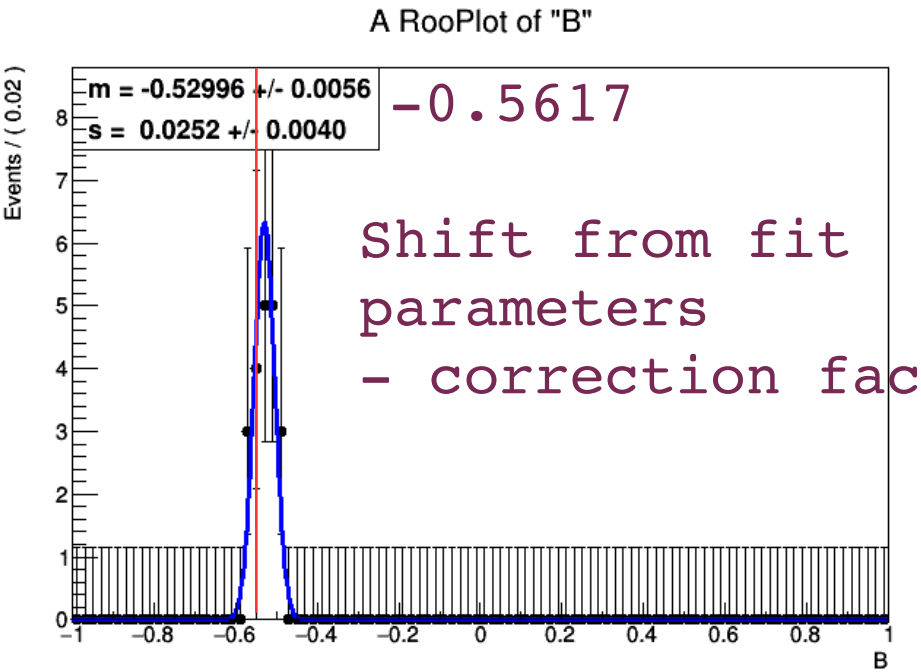
Fit Result

With Corrections applied:

A = 0.405625 +/- 0.02131
B = -0.593406 +/- 0.0207043



0.381



-0.5617

Shift from fit parameters
- correction factor

Status

- A General framework has been developed for “quickly” and “reliably” extracting observables from CLAS12 data
- CLAS12 HIPO files are converted to ROOT then kinematic variables are calculated via THSFinalState analysis class
- These variables are used in Extended Maximum Likelihood fits by an extension to the RooFit package allowing acceptance corrections and background subtration weights
- Results can be validated by integrated ToyMC methods
  - Additional requirement for VeryFastMC !
- Boosted decision trees (TMVA) have been tested for PID at the event level (combining information of all tracks)
- Some details need completed, update to Event Builder banks; THSHipoTrigger; improve THSFinalState; no KinFitting yet (could be done in THSFinalState::Init)

  ...

Extended ML with acceptance

$$L(p) = \prod_i^N \frac{f(\tau_i : p)}{\int f(\tau_i : p)\, d\tau}$$

Standard likelihood, product Prob. function f with obs τ and pars p

$$L_{acc}(p) = \prod_i^N \frac{f(\tau_i : p)\,\eta(\tau_i)}{\int f(\tau_i : p)\,\eta(\tau_i)\, d\tau}$$

With acceptance take product of f and acceptance function

$$A(p) = \int f(\tau_i : p)\,\eta(\tau_i)\, d\tau$$

Probability Normalisation ∫

$$L_{acc}^{ext}(p) = \left[\frac{A(p)^N}{N!} e^{-A(p)}\right] \prod_i^N \frac{f(\tau_i : p)\,\eta(\tau_i)}{A(p)}$$

Extended ML with Poisson statistics

$$-\ln L_{acc}^{ext}(p) \propto -\sum_i^N \ln f(\tau_i : p)\,\eta(\tau_i) + A(p)$$

Simpler to minimise ln ignore terms independent of p

$$= -\sum_i^N \ln f(\tau_i : p) - \sum_i^N \ln \eta(\tau_i) + A(p)$$

Acceptance must be independent of p Make sure this is True!!

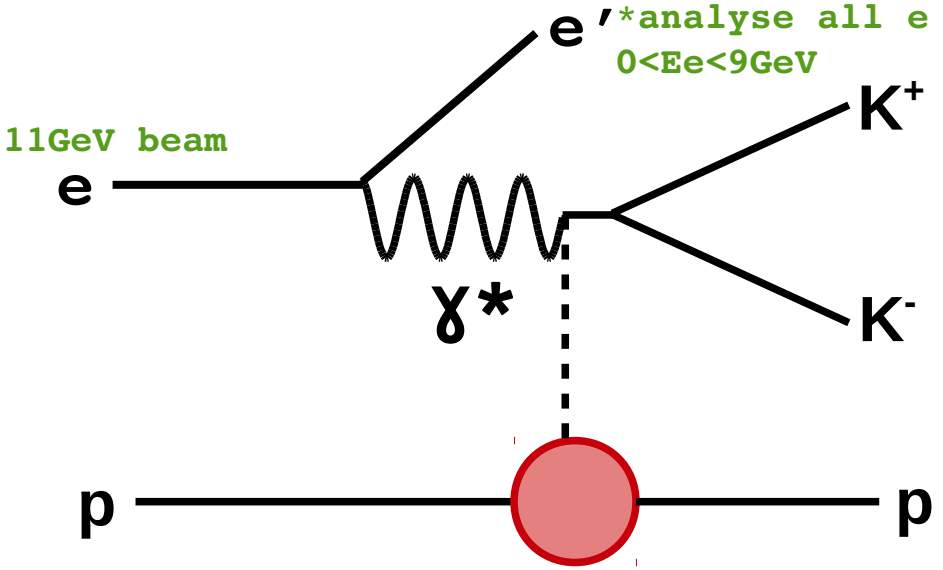
$$\propto -\sum_i^N \ln f(\tau_i : p) + A(p)$$

Need to minimise this

$$A(p) \simeq \sum_j^M f(\tau_j : p)$$

Approximating A as sum of f over M accepted Monte-Carlo events

Toy Example Full Chain Analysis



Simulate 300k phase space events with gemc

Reconstruct with coatjava

Convert to ROOT HSParticle

Calculate Fit variables with THSProject class

Fold in model and fit with hsfit

Production:

t-distribution + photon linear polarisation → $1+\Sigma P_\gamma \cos 2\alpha$

KK Decay parameterised by spherical harmonic moments

In helicity frame :

New HSFIT class

$$I = \sum_L \sum_{M \geqq 0} t_L^M \, Re Y_L^M (\theta, \phi)$$

$$Y_L^M (\theta, \phi) = (-1)^M \sqrt{\frac{(2L+1)}{4\pi} \frac{(L-M)!}{(L+M)!}} P_L^M (\cos\theta) \, e^{iM\phi}$$

Event Algorithm

Filter events based on charges of particles

Exclusive :+ve,+ve,-ve,-ve(FT)

Missing K- :+ve,+ve,-ve(FT)

Missing K+ or p :+ve,-ve,-ve (FT)

Use measured momentum and assign PDG masses for each topology

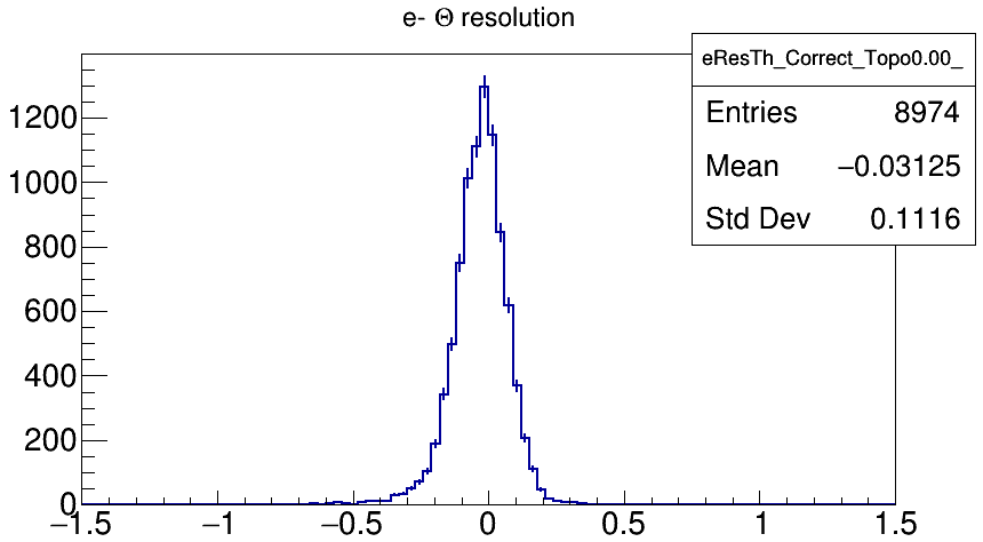
Create new events for each combination of +ve and -ve particles

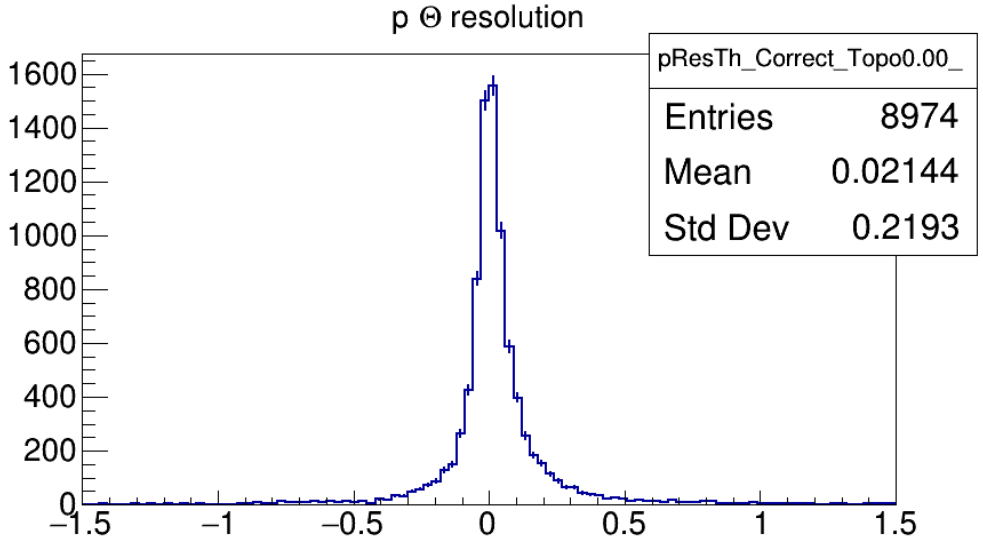
Store true generated values

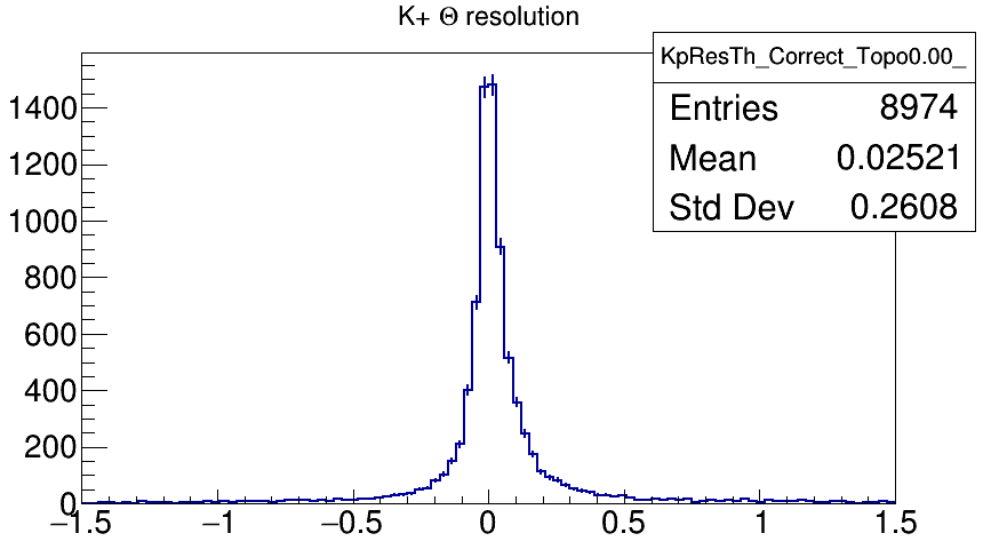
When generated and reconstructed particle identification match ⇒ accept event

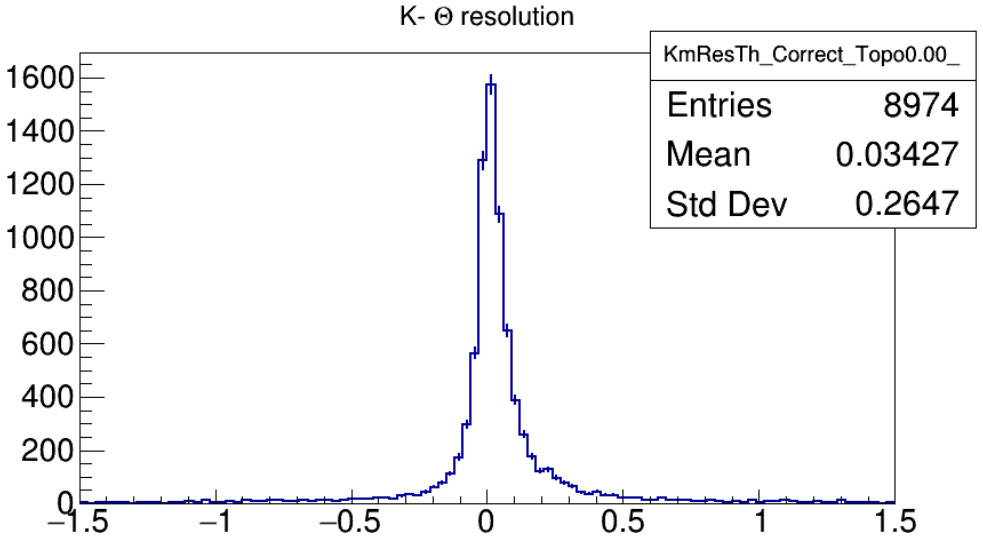
i.e cheating
Otherwise use methods described this morning

Reconstruction Quality 1 : Angle Resolutions

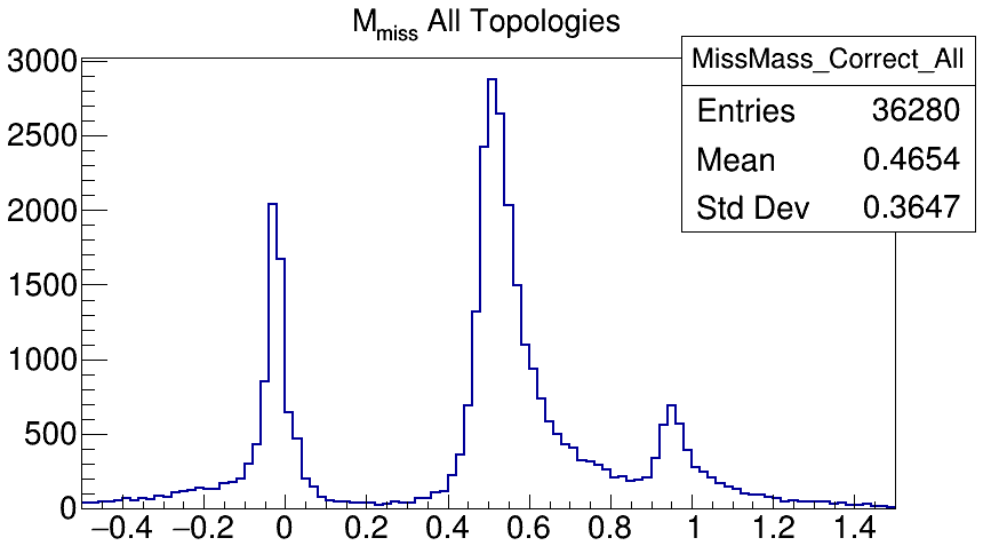


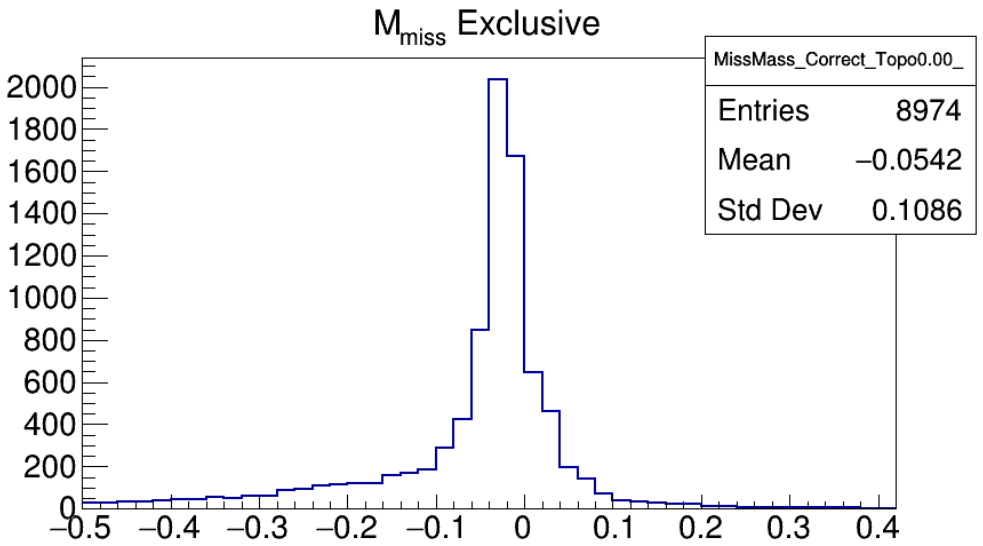


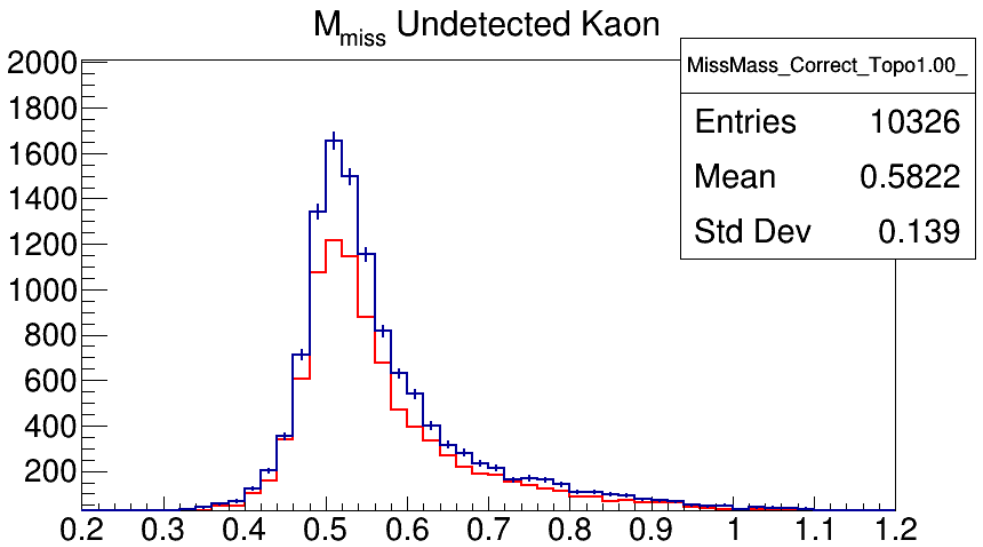


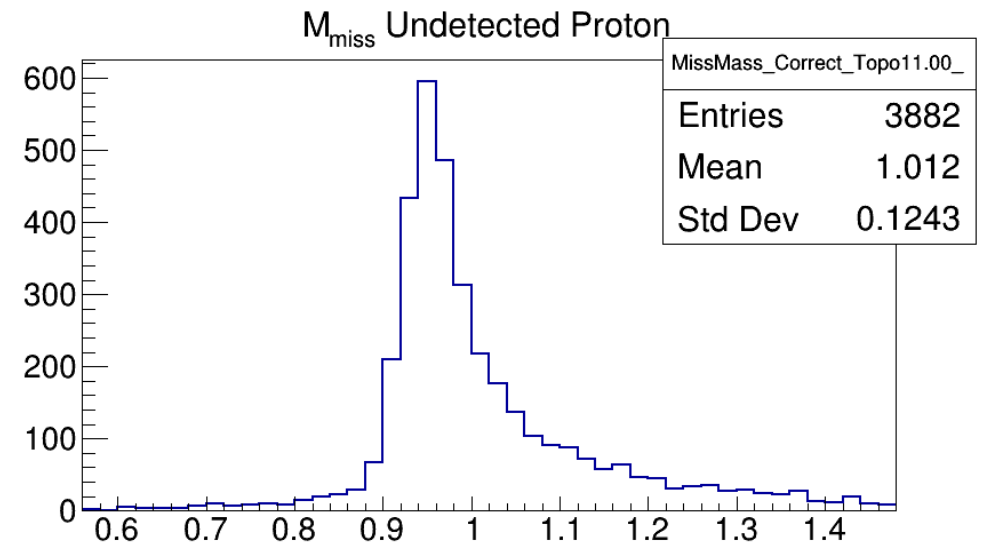


Reconstruction Quality 2 : Missing Mass









Reconstructed Data Fits

Data generated as phase space and passed through gemc + clas12rec

epK+K- Final state reconstructed

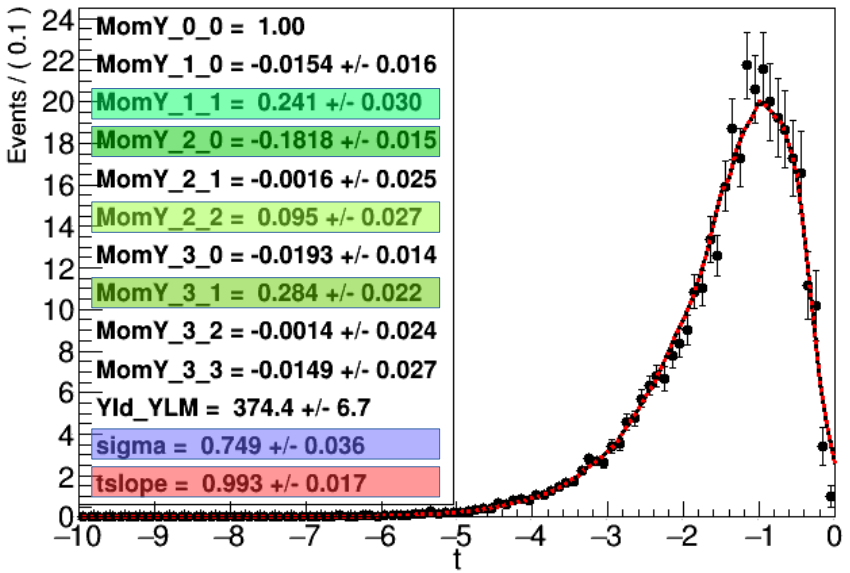
Fold model onto 33k reconstructed events (using true variables)

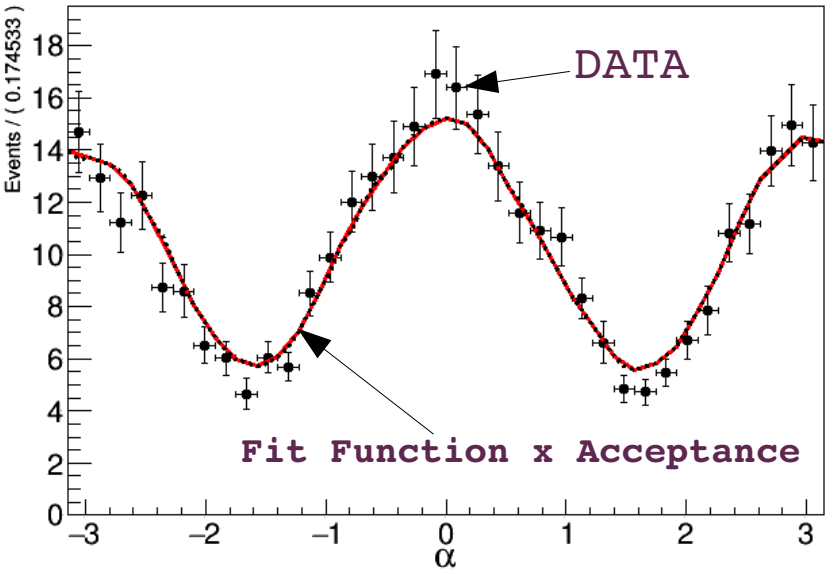
T-distribution **tslope** = 1(GeV/c)$^{-2}$, **Photon Asymmetry Σ** = 0.8
Non-zero moments : **Y11** = 0.3 **Y20** = -0.2 **Y22** = 0.15 **Y31**=0.3

Perform Extended Maximum Likelihood fits with acceptance correction

Production Kinematics

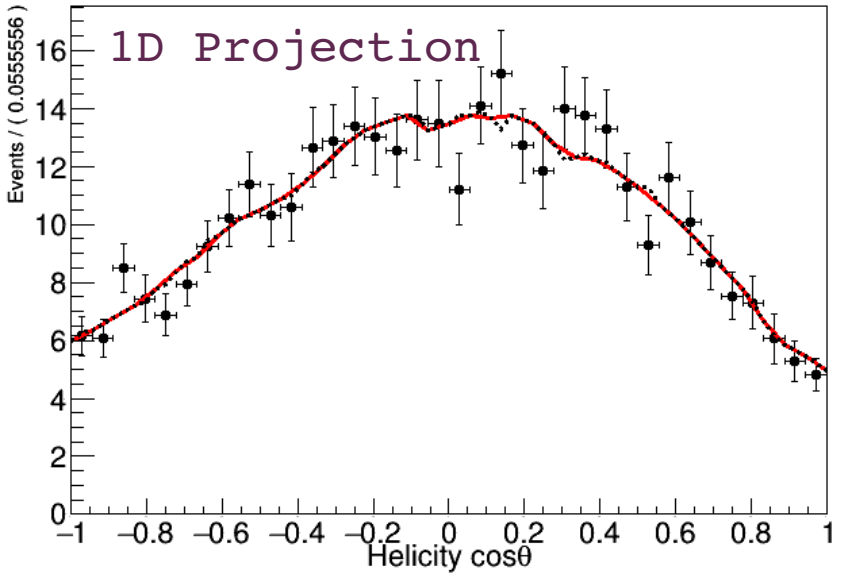
Note phi acceptance ~ flat
May not need correction

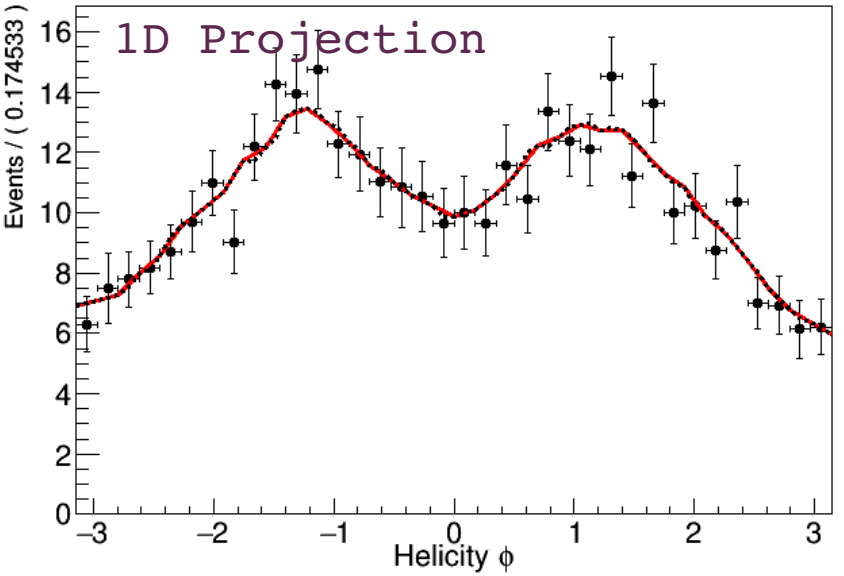


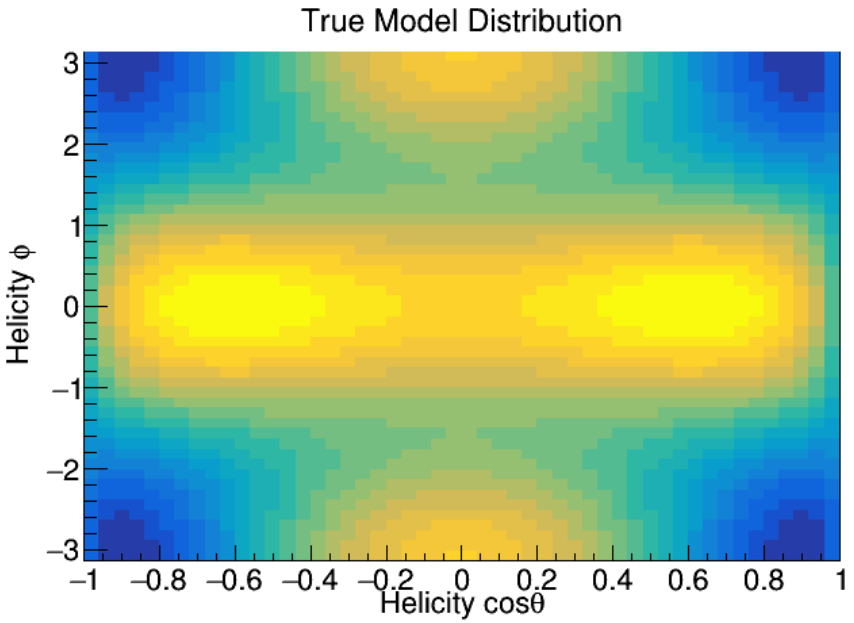


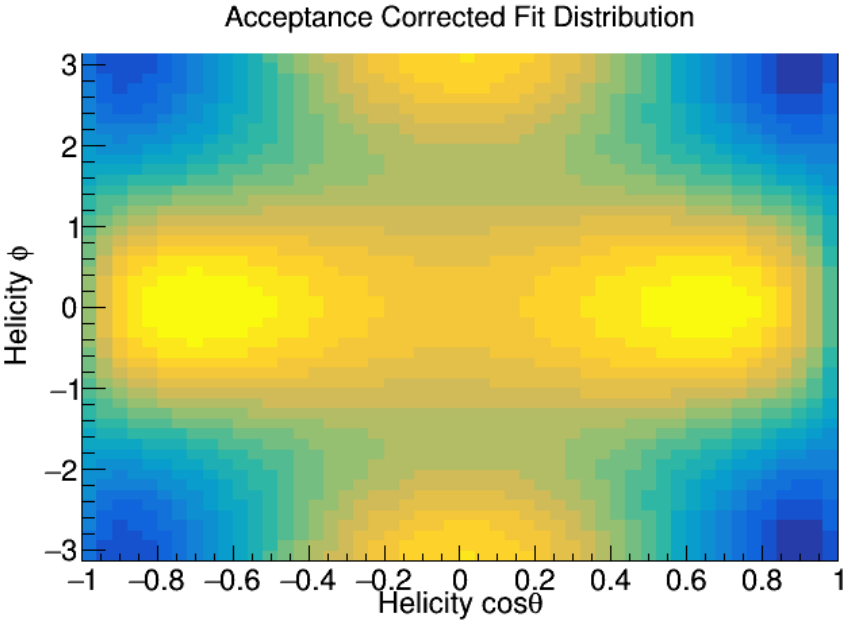
Reconstructed Data Fits

Decay Kinematics









Running hsroot

- hslogon.C
  - can be used to define extra useful functions to be run in ROOT session
  - Interpret command line arguments
- hsroot Command line args

  --hssel load HSSelector classes

  --hsfit load HSRooFit classes

  --proof=N initialse proof for N workers

  --THSMyClass.C compile and load new class to be used in this session

  --hsin set input file directory

  optional

  --hsout set output destination

  Can be file or dir

  --macrodir=extradir add additional path to your class

  --makeall compile all classes

Commands

```
root --hssel
hsroot[0] ...do stuff with hssel classes

root --hsfit
hsfit[0] ...do stuff with hsfit classes

root --hssel --proof=3 MyMacro.C

...run Control.C with proof and hssel classes

root --hssel MyMacro.C --hsout=test.root
*
```

* Can use function hsout() in Control.C to set ouput file to test.root (similarily for hsin)

Can also set in shell via env variables e.g. setenv HSOUT /destination/

Or in MyMacro.C :Hout("/destination/");

Signal Selection with THSProject

Users derive their own ROOT C++ class

- Handles EVENTS
- Take a detected state of HSParticles
- Produce quantities for
  - Fitting: final 4-vectors, variables (angles,masses), discriminators (signal/back)
  - Diagnostics: compare simulated/real,...
- Define behaviour for all topologies
- Permutate all combinations
  - Call WorkOnEvent for each
  - Also works when PID not known, just charge
- Same code for simulated and real experiment

```
WorkOnEvent(){
fGoodEvent=kTRUE;
//Find the detected particles for event
FindTopology();
//Do they have a defined topology?
if(fCurrTopo==fTIDprot_pip_pim_pi0_omega)
Init_prot_pip_pim_pi0_omega();
  else
if(fCurrTopo==fTIDprot_pip_pim_pi0)
Init_prot_pip_pim_pi0();
  else if(fCurrTopo==fTIDprot_pip_pim)
Init_prot_pip_pim();
  else fGoodEvent=kFALSE;
//From here on same for each topology
//Calc kinematics
ProductionKinematics();
}
```

Define topology

- How different final state particles are reconstructed will differ for different detected final states
- Handle via topology Init functions
- THSProject determines topology for event
- User codes behaviour

```
void THSPhotoOmega::Init_prot_pip_pim(){
  //Fill our data member particles
  //User is responsible for indicing right
  fProton=(fVecProtons.at(0));
  fPip=(fVecPiPs.at(0));
  fPim=(fVecPiMs.at(0));
  //make pi0 missing particle
  fPi0.SetP4(fBeam+fTarget-fProton.P4()-
                  fPip.P4()-fPim.P4());
  fPi0.SetMeasMass(fPi0.P4().M());
  fPi0.TakePDGMass();
//Reconstruct Omega
  fOmega.SetP4(fPip.P4()+fPim.P4()
+fPi0.P4());
  fOmega.SetMeasMass(fOmega.P4().M());
  fOmega.TakePDGMass();
  //Reconstruct discriminators for this
topology
  fOmegaMass=fOmega.MassDiff();
  fMissMass=fPi0.MassDiff();
}
```

THSProject output tree

- User defines information to be saved in tree for fitting
- User takes care of tree filling/saving outwith project
- Pass project the tree before start loop

```
THSPhotoOmega::ProjectOutTree
                    (TTree* tree){
//fFinal=Final State HSParticles
tree->Branch("Final",&Final);
tree→Branch
("MissM",&fMissM,"MissM/D");
tree→Branch
("OmMass",&fOmMass,"OmMass/D");
tree→
Branch("Topo",&fCurrTopo,"Topo/I");
tree->Branch("t",&f_t,"t/D");
tree->Branch("W",&f_W,"W/D");
}
```

Simulated Events with THSProject

- Access true values from fGenParts
- Directly set recon particle truth
- THSProject attempts to determine if correct tracks reconstructed
- Sets “Correct” flag
- Can then determine resolution, combitorial background,...

```
//When only analysing generated
Init_Generated(){
fElectron=frGenParts->At(0);
//When analysing recon
fElectron.SetTruth(frGenParts→At(0))

WorkOnEvent(){
//Check if recon match gen
CheckTruth();

//Check difference in recon angle
fElectron.ResTheta();
...
```