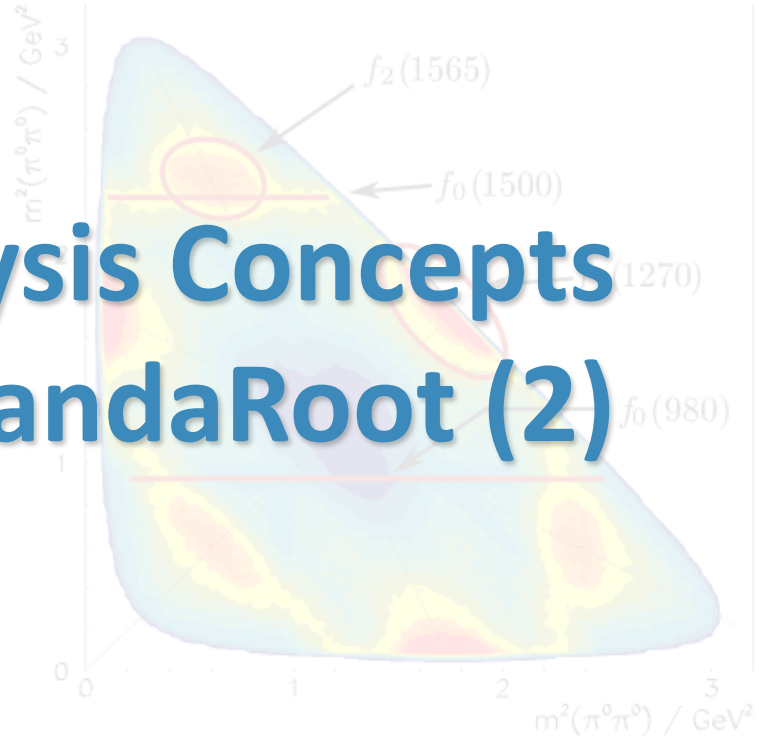
Physics Analysis Concepts with PandaRoot (2)



PANDA Lecture Week 2017

GSI, Dec 11 - 15, 2017

Klaus Götzen
GSI Darmstadt

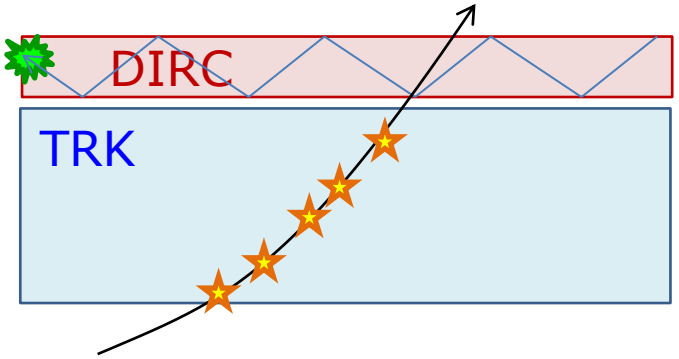
Topics

- Particle Identification
- Particle Selectors *(PndPidCombiner, PndPidSelector)*
- MC Truth Match *(PndAnalysis::McTruthMatch)*
- Fitting Basics
- Fitting Tools *(PndKinVtxFitter, PndKinFitter)*
- Uncertainties

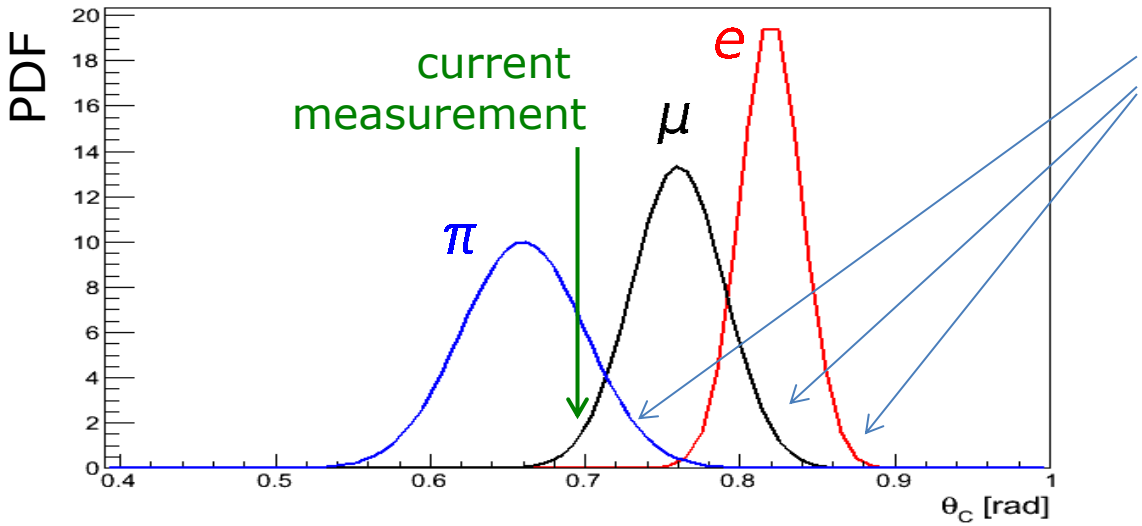
PARTICLE IDENTIFICATION

Particle Identification Basics

How does PID work in principle?



1. Reconstruct track object
2. Match track with PID detector signal
3. Reconstruct PID information
4. Compute likelihood for various particle types
   (e.g. based on Likelihood functions)



expected θ_c distributions for a certain momentum

PID Concept in PandaROOT

- Each PID subdetector delivers individual probability/likelihood information
- PID Combiner, which combines the likelihood values from different detectors

$$\tilde{P}_{tot,i} = \prod_{Det} P_{Det,i} \qquad i \in \{e, \mu, \pi, K, p\}$$

- Normalization via

$$P_{tot,j} = \frac{\tilde{P}_{tot,j}}{\sum_i \tilde{P}_{tot,i}}$$

- PID Selector, which requires certain P_{tot} for positive identification

PID Concept in PandaROOT

PandaROOT objects: **PndAnaPidCombiner, PndAnaPidSelector**

- PndAnaPidCombiner
  - combines *on demand* probabilities from various algorithms by computing product of all P_k *(k=algorithms)*
  - copies resulting probabilities to RhoCandidate/RhoCandList
- PndAnaPidSelector
  - selects particles based on these probabilities
- PndAnalysis::FillList is a short-cut to this funtionallity via

```
pndana.FillList(list, “ElectronLoose“, "PidAlgoEmcBayes;PidAlgoDrc");
```

- Predefined selection with keywords (probability based)

  Electron / Muon / Pion / Kaon / Proton +
  All / VeryLoose / Loose / Tight / VeryTight / Best +
  Plus / Minus *(optional)*
  Simple keywords
  Charged / Plus / Minus / Neutral / All

PID Concept in PandaROOT

- PandaROOT objects:
  PndAnalysis, PndAnaPidCombiner, PndAnaPidSelector

```
PndAnalysis *pndana= new PndAnalysis();
pndana->FillList(eplus,  "ElectronLoosePlus",  "PidAlgoEmcBayes;PidAlgoMvd");
pndana->FillList(eminus, "ElectronLooseMinus", "PidAlgoEmcBayes;PidAlgoMvd");
```

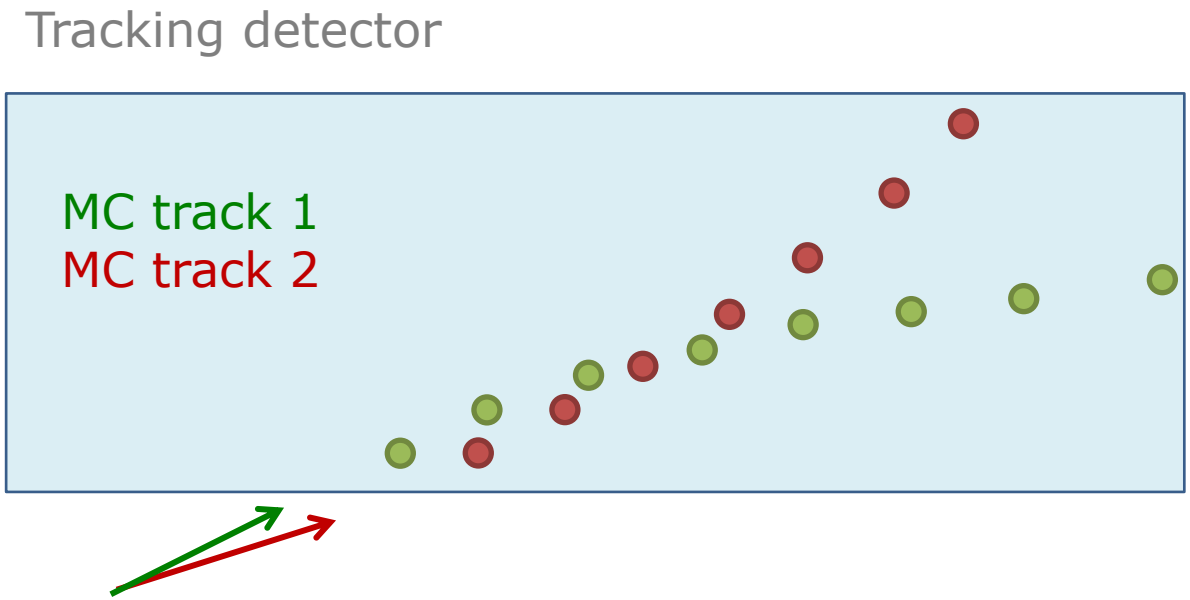
Or 'by hand':

```
RhoCandList charged, kaonLoose;

PndAnaPidSelector kaonSel("KaonSelector");
kaonSel.SetSelection("KaonLoose");            // set selection criterion

PndAnaPidCombiner pidComb("PidCombiner");
pidComb.SetTcaNames("PidAlgoDrc;PidAlgoMvd"); // set algo‘s

while (evr->GetEvent())
{
  pndana->FillList(charged, "Charged");     // start w/ charged candidates

  pidComb.Apply(charged);                   // copy probab. to candidates
  kaonLoose.Select(charged, kaonSel);       // select kaons from charged
}
```

MONTE CARLO TRUTH

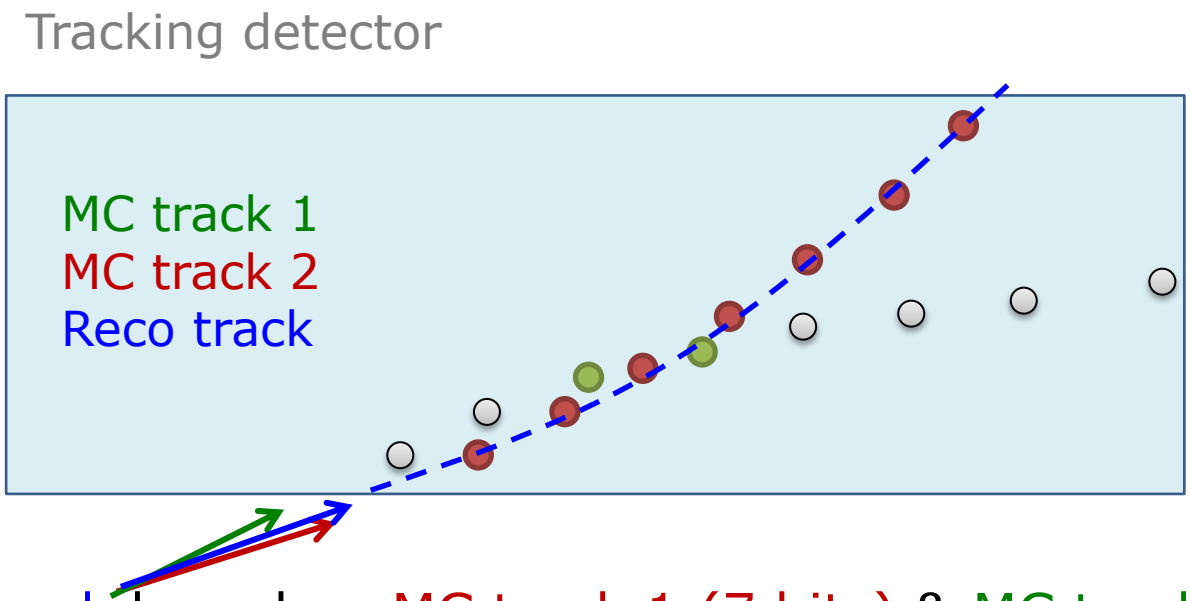
Monte Carlo Truth Match

- In simulated reactions
  → possibility to map reco object to MC object



Monte Carlo Truth Match

- In simulated reactions
  → possibility to map reco object to MC object



- Reco track based on MC track 1 (7 hits) & MC track 2 (2 hits)

  → Monte Carlo Truth(Reco track) = MC track 2 (max hits)

  [realized by FairLink class]

Monte Carlo Truth Access

Use cases:

a) For typical measurement the reco efficiency is needed

b) Sometimes the true signal distribution is of interest

→ Look in MC, whether you combine the correct candidates

- Concerning MC truth, distinguish between

1. access MC truth object (or type) for certain reco object (see last slides)

2. MC truth match for a complete decay tree

MC Truth Access

- For final state particles via RhoCandidate::GetMcTruth

```
ana->FillList( eplus,  "ElectronTightPlus",  "PidAlgoEmcBayes" );
RhoCandidate *truth = eplus[0]->GetMcTruth();
```

- The MC truth particles have the complete genealogy

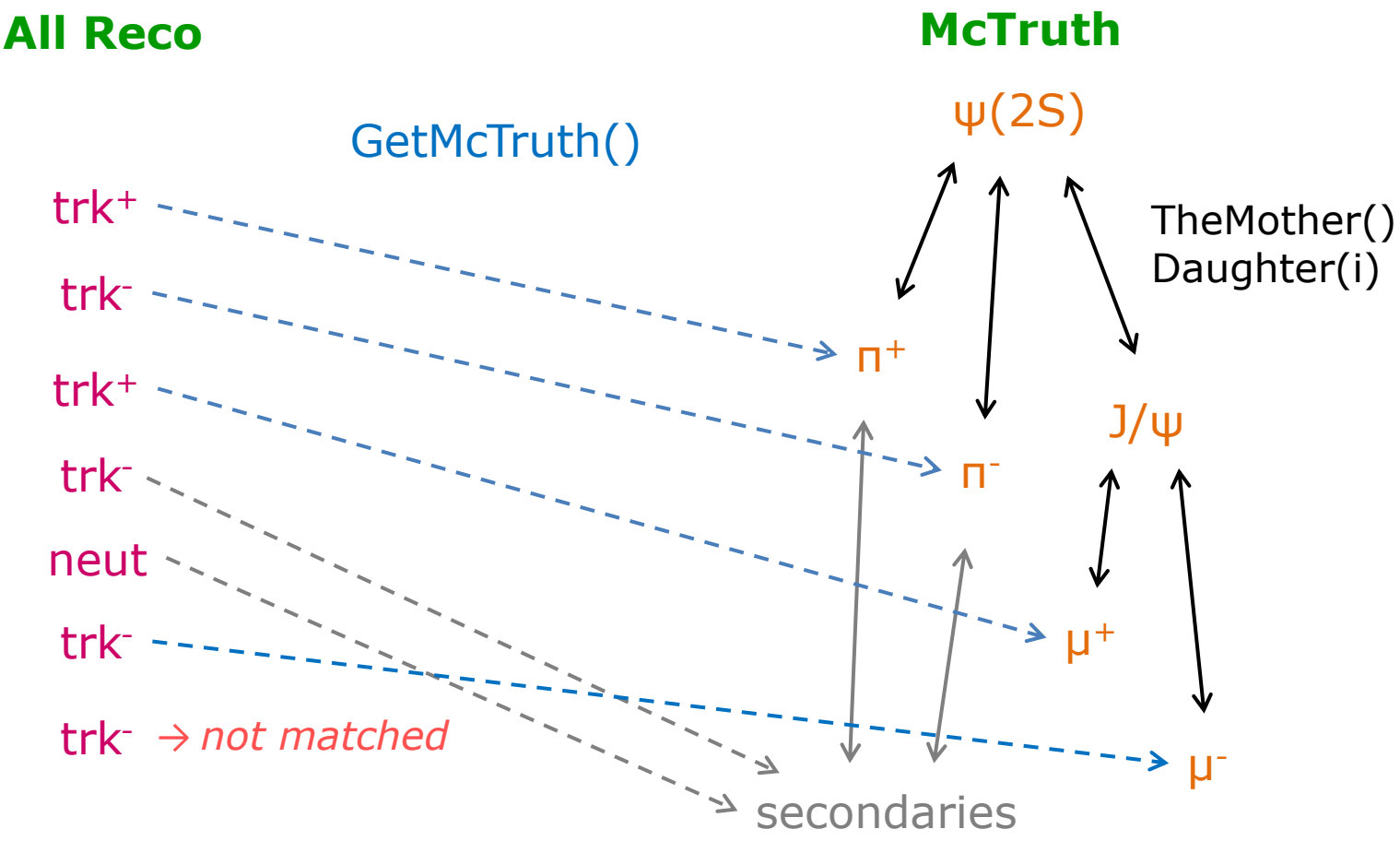
```
if ( truth != 0x0 ) {
   RhoCandidate *mother = truth->TheMother();
   if ( truth->NDaughters() > 1 ) {
      RhoCandidate *firstDaughter  = truth->Daughter(0);
      RhoCandidate *secondDaughter = truth->Daughter(1);
   }
}
```

- Also complete MC truth list available

```
ana->FillList( mct,  "McTruth" );
...
RhoCandidate *firstDaughter = mct[0]->Daughter(0);
```

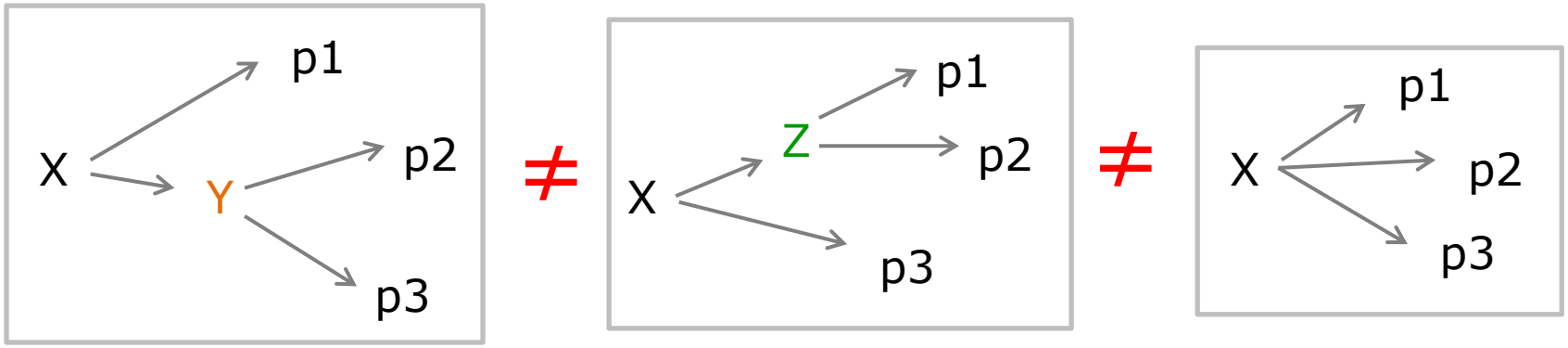
Mc Truth Genealogy

- **Idea**: Each Reco points to an McTruth object, from which the full true tree can be accessed



PndAnalysis: MC Truth Match of Composites

- Physics analysis might require a full mc truth match for composite particles for efficiency calculation, because

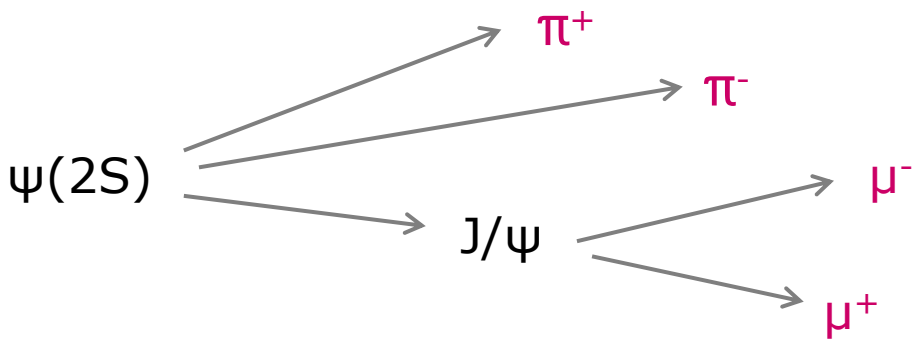


- Functionality in PndAnalysis method

```
PndAnalysis::McTruthMatch( RhoCandidate* );
```

Example: ψ(2S) → J/ψ ($\mu^+\mu^-$) $\pi^+\pi^-$

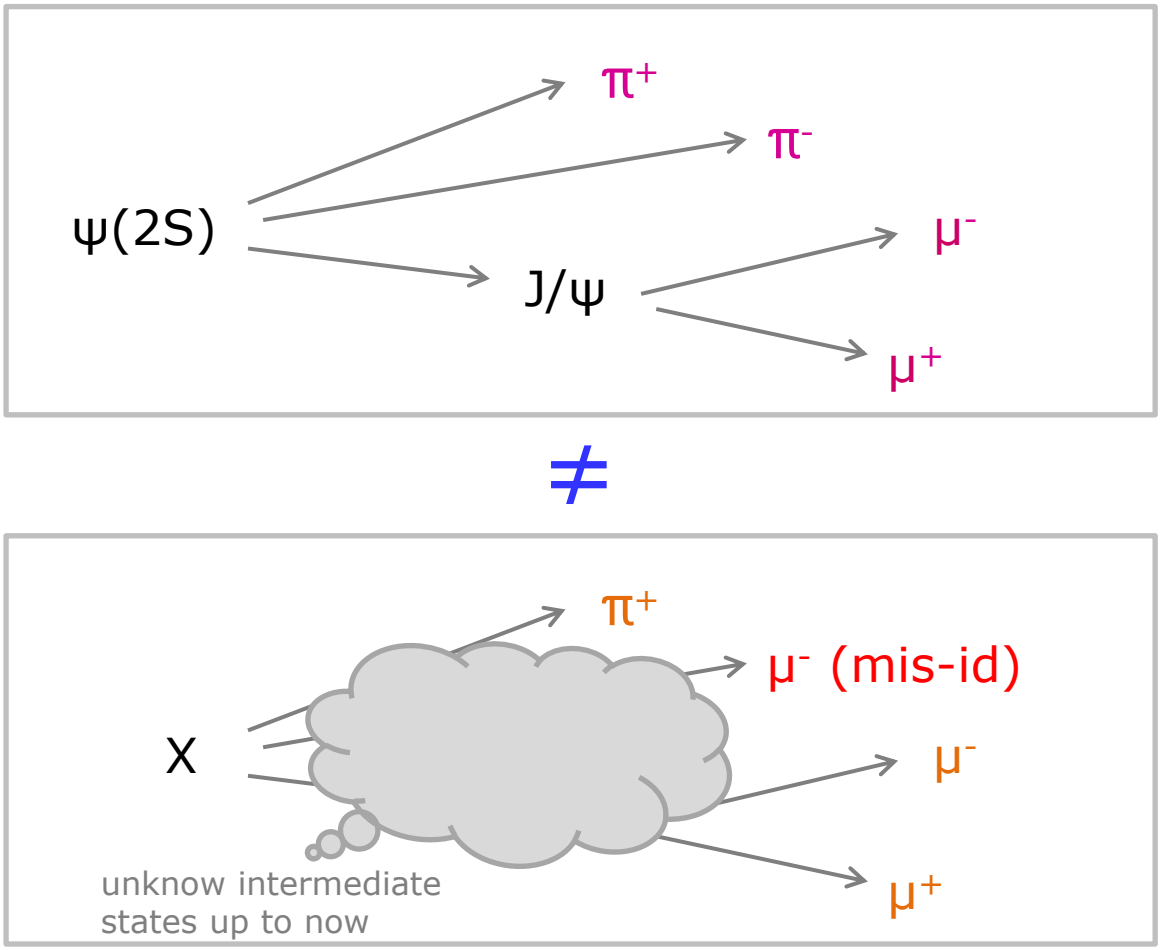
- We want to reconstruct the following decay




```
RhoCandList muplus, muminus, piplus, piminus, jpsi, psi2s;

ana->FillList( muplus,  "MuonLoosePlus",  myPidAlgos );
ana->FillList( muminus, "MuonLooseMinus", myPidAlgos );
ana->FillList( piplus,  "PionLoosePlus",  myPidAlgos );
ana->FillList( piminus, "PionLooseMinus", myPidAlgos );

jpsi.Combine( muplus, muminus );
psi2s.Combine( jpsi, piplus, piminus );
```

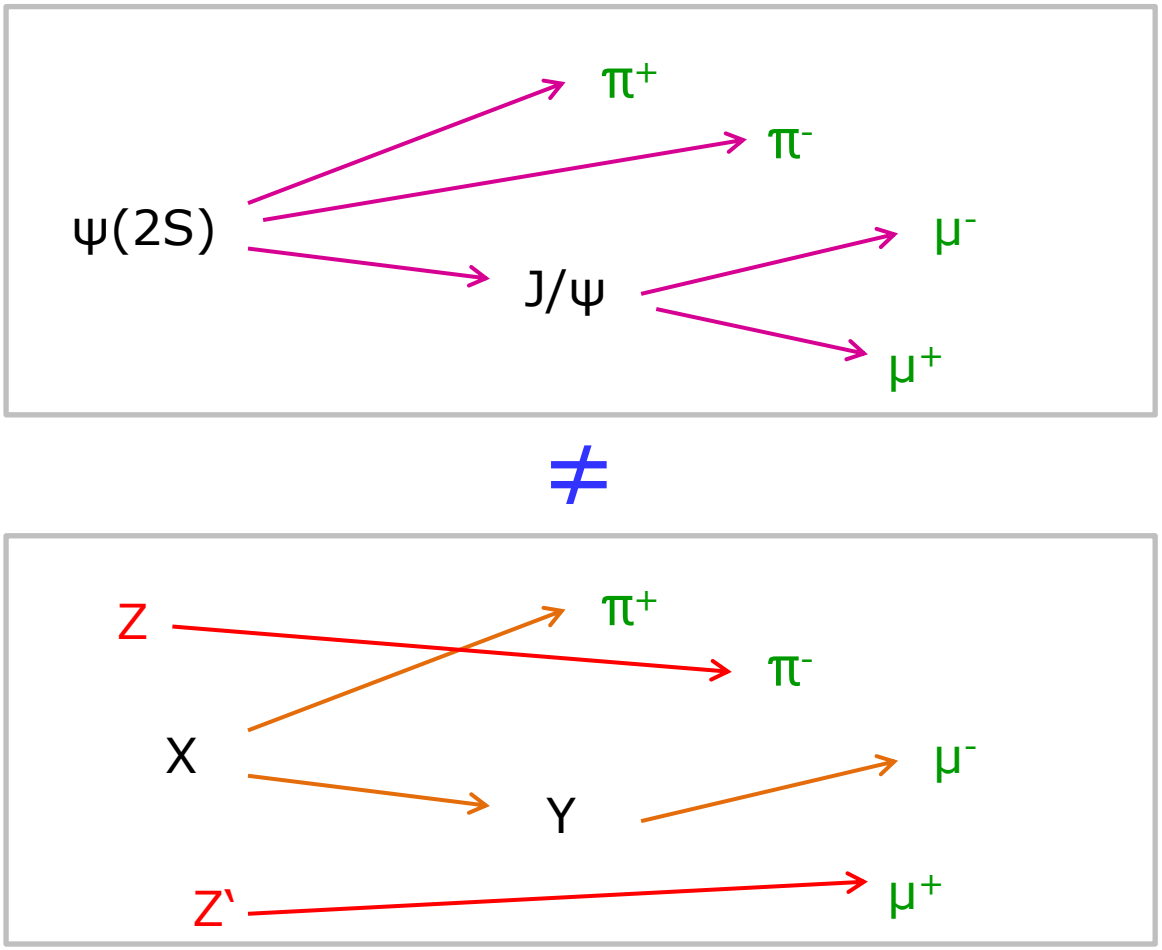
PndMcTruthMatch

- **Checklist** for full tree match:
  1. truth objects of final states have the correct PID types



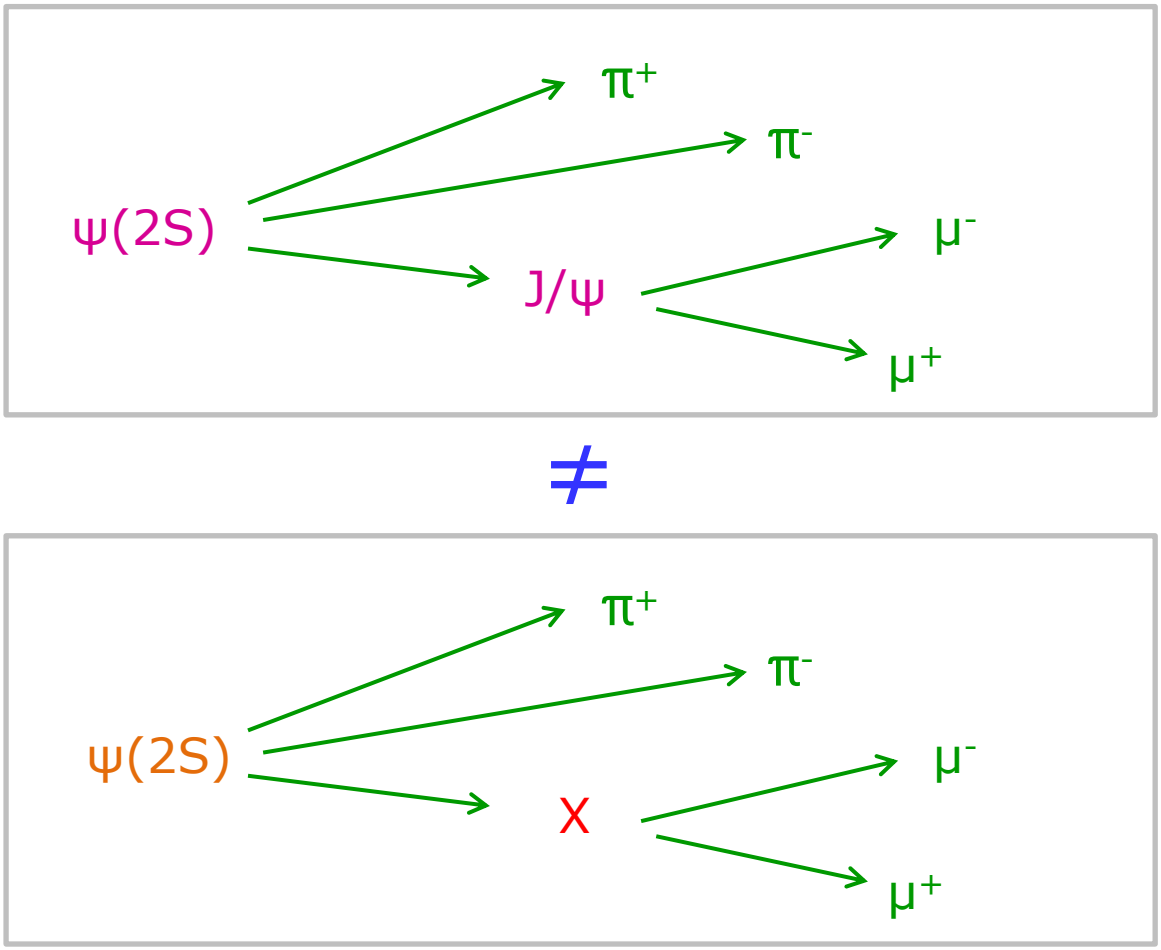
PndMcTruthMatch

- **Checklist** for full tree match:
  2. truth objects of final states have the same mother



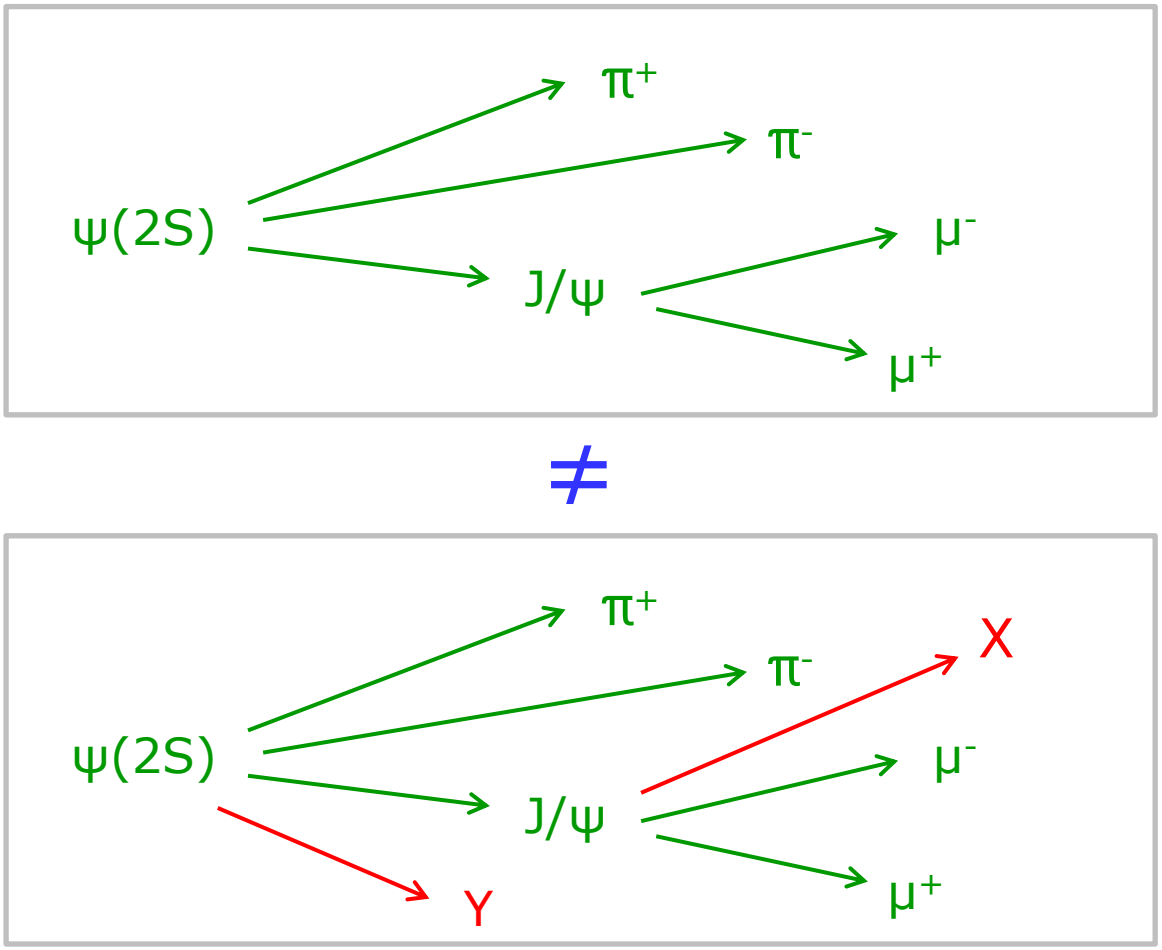
PndMcTruthMatch

- **Checklist** for full tree match:
  3. mother has required type



PndMcTruthMatch

- **Checklist** for full tree match:
  4. mother has correct number of daughters

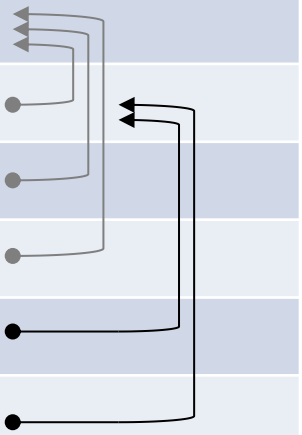


Example: ψ(2S) → J/ψ ($\mu^+\mu^-$) $\pi^+\pi^-$

#	reco
1	trk+
2	trk+
3	trk+
4	trk-
5	trk-
6	trk-
7	trk-
8	neut
9	neut
10	neut
11	neut

GetMcTruth()

#	truth	TheMother()
1	psi(2S)	
2	J/psi	
3	pi+	
4	pi-	
5	mu+	
6	mu-	
7	gamma	
8	gamma	
9	e+	
10	mu-	
11	gamma	
12	gamma	
13	pi-	

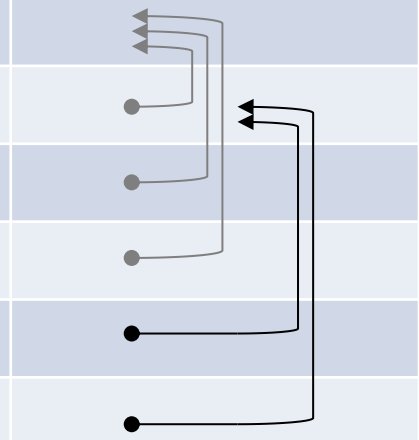


Example: ψ(2S) → J/ψ ($\mu^+\mu^-$) $\pi^+\pi^-$

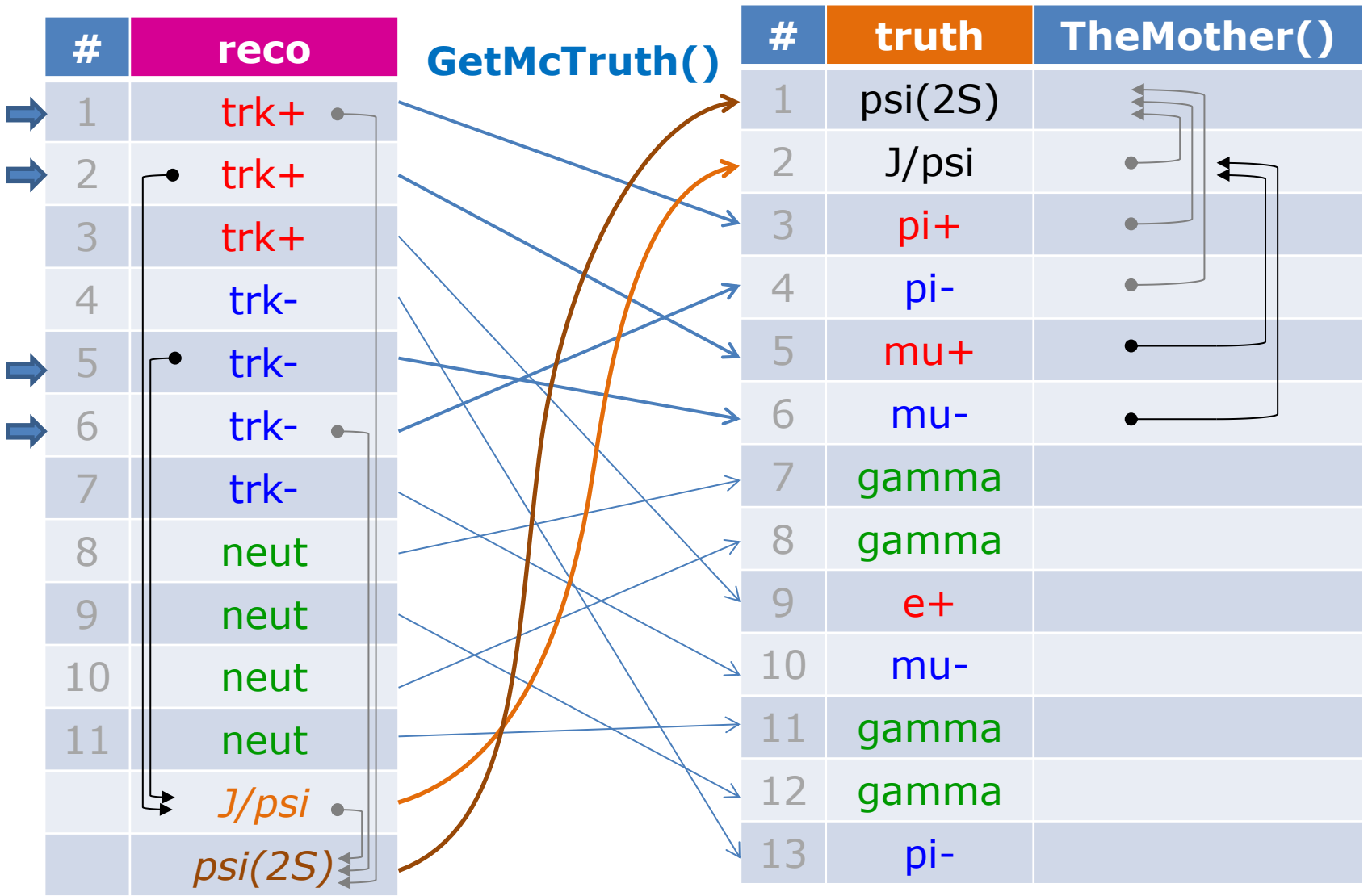
#	reco
1	trk+
2	trk+
3	trk+
4	trk-
5	trk-
6	trk-
7	trk-
8	neut
9	neut
10	neut
11	neut
	J/psi
	psi(2S)

GetMcTruth()

#	truth	TheMother()
1	psi(2S)	
2	J/psi	
3	pi+	
4	pi-	
5	mu+	
6	mu-	
7	gamma	
8	gamma	
9	e+	
10	mu-	
11	gamma	
12	gamma	
13	pi-	



Example: ψ(2S) → J/ψ (μ⁺μ⁻) π⁺π⁻



Usage of PndAnalysis::McTruthMatch

- For matching, composite candidates have to have type set

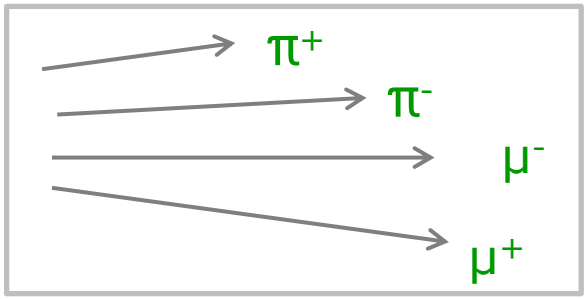
```
PndAnalysis *ana = new PndAnalysis();

while ( ana->GetEvent() )
{
  ana->FillList(muplus, "MuonPlus");
  ...

  jpsi.Combine(muplus, muminus, "J/psi");
  psi2s.Combine(jpsi, piplus, piminus, "pbarpSystem");


  bool match  = ana->McTruthMatch( psi2s[0] );  // match for RhoCandidate
  int nmatch  = ana->McTruthMatch( psi2s );     // match complete RhoCandList


  for (int j=0; j<psi2s.GetLength(); ++j)
  {
    RhoCandidate *truth = psi2s[j].GetMcTruth(); // access truth of composites
    ...
```

although being at psi(2S) energy, particle was generated as pbarpSystem

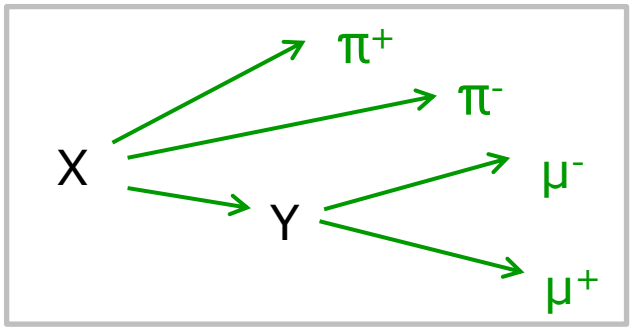
Different levels of matching

- Three different match levels are available via
  `ana->McTruthMatch( psi[0], level );`

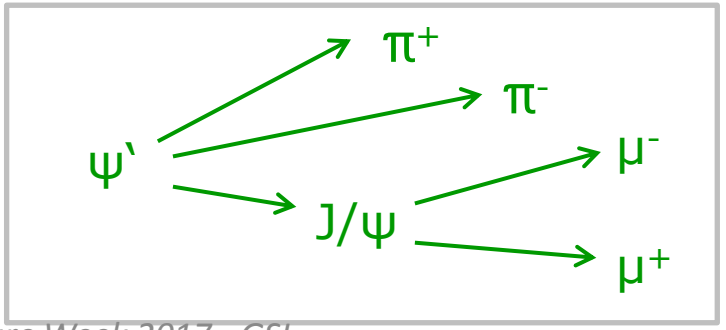
- Level = 0 only looks for correct PID



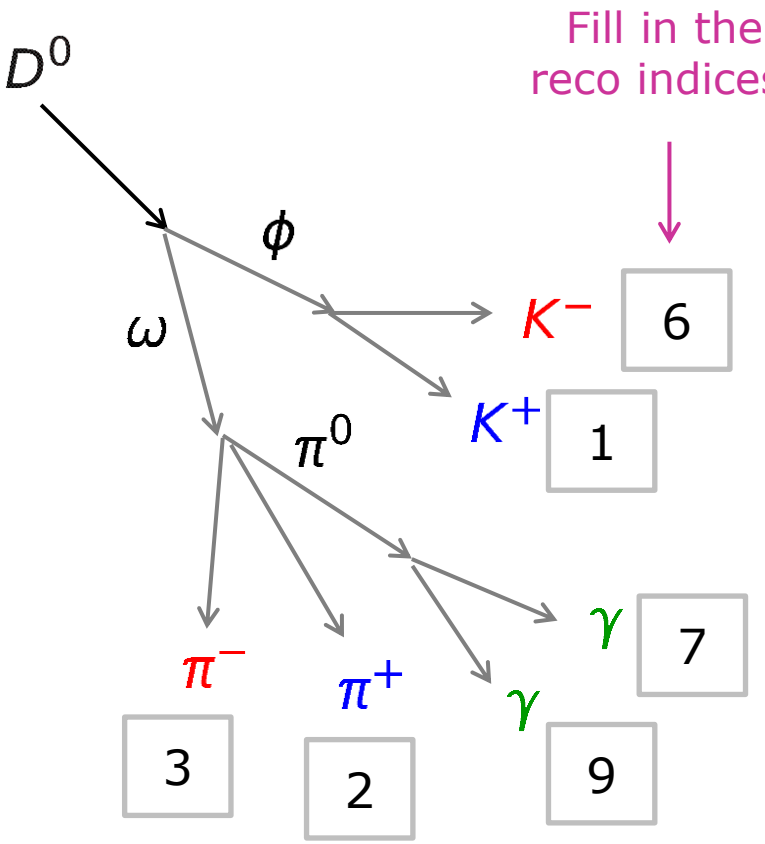
- Level = 1 matches topology in addition



- Level = 2 (default) is full match



MC Truth Match Exercise



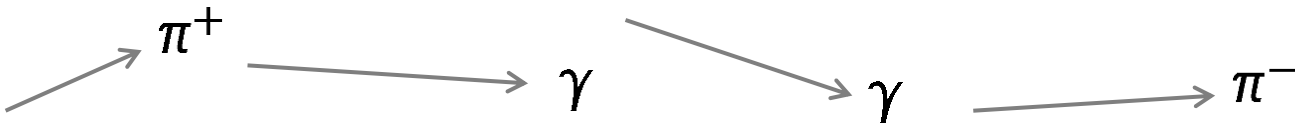
#	reco	mc
1	trk+	7
2	trk+	5
3	trk-	4
4	trk-	11
5	trk-	14
6	trk-	8
7	neut	10
8	neut	12
9	neut	9

#	truth	mum
1	D0	-1
2	omega	1
3	phi	1
4	pi-	2
5	pi+	2
6	pi0	2
7	K+	3
8	K-	3
9	gamma	6
10	gamma	6
11	e-	9
12	gamma	11
13	gamma	11
14	e-	10

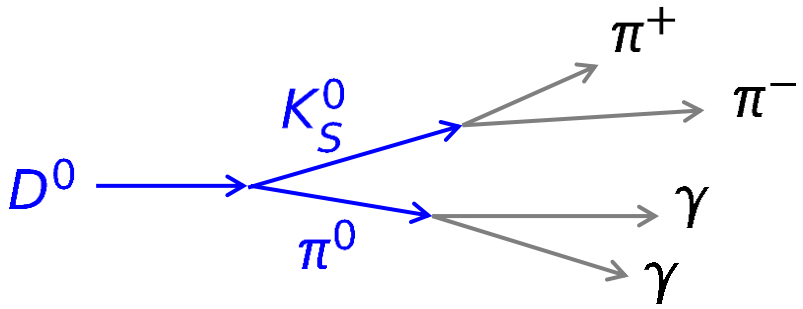
FITTING

Kinematic Fitting

- Let's e.g. consider a set of particle candidates



- You believe what happend is this (hypothesis):

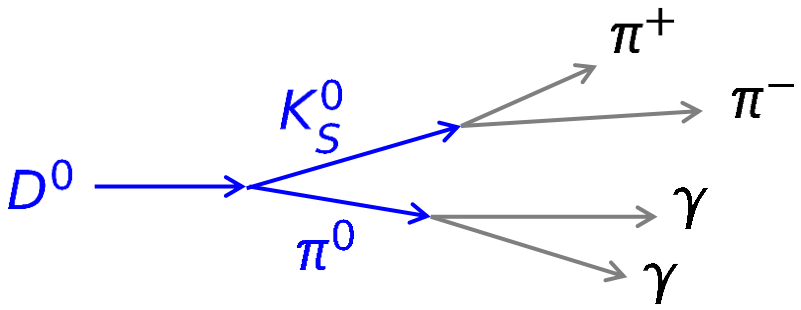


- Kinematic fitting
  - gives a measure for the assumption, that hypothesis is true
  - improves the precision/resolution for kinematic quantities, when hypothesis *actually is* true

More about fitting under **`http://www.phys.ufl.edu/~avery/fitting.html`**

Kinematic Fitting: Constraints

- In order to perform fit, you need to apply so-called constraints
- Constraints correlate kinematics in the tree



- *Which constraints could be applied here?*
  - Vertex constraint K_S: π^+ & π^- from same origin
  - Mass constraint K_S: inv. π^+ π^- mass = K_S rest mass
  - Mass constraint π^0: inv. 2γ mass = π^0 rest mass
  - Mass constraint D^0: inv. K_S π^0 mass = D^0 rest mass

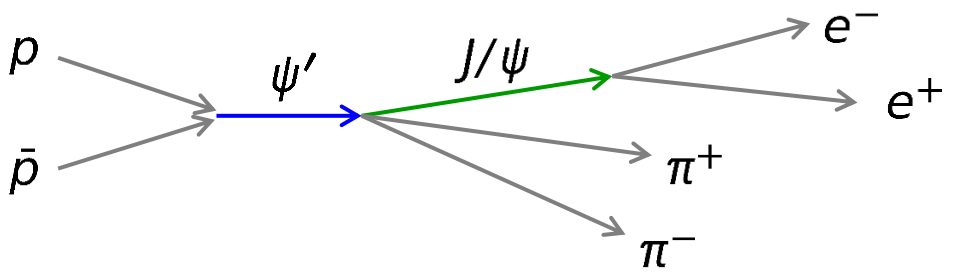
Kinematic Fitting: Constraints

Most common constraints are

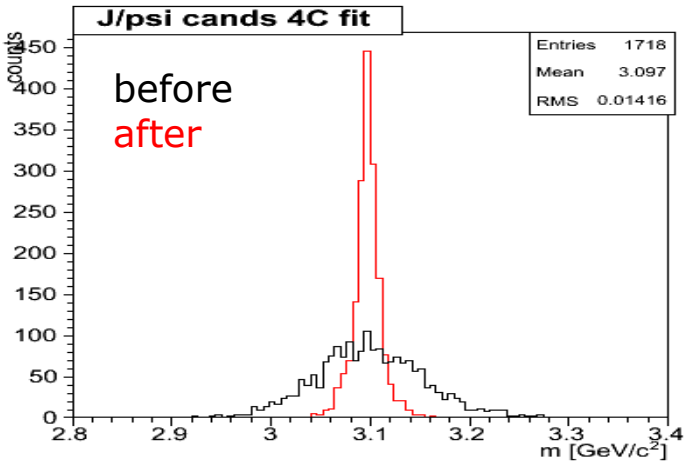
- Mass Constraint
  Composite particle has to have a certain invariant mass (only to be applied to quasi stable particles)
- Vertex Constraint
  Set of 4-vectors originate from common spatial point (which is determined during the fit!). *At least 2 charged tracks needed!*
- 4-vector-Constraint
  4-vector of composite particle has to match a certain 4-vector component-wise (used in exclusive reconstruction)
- Pointing constraint
  3-vector of composite particle is consistent with originating from a certaint point (e.g. the IP)

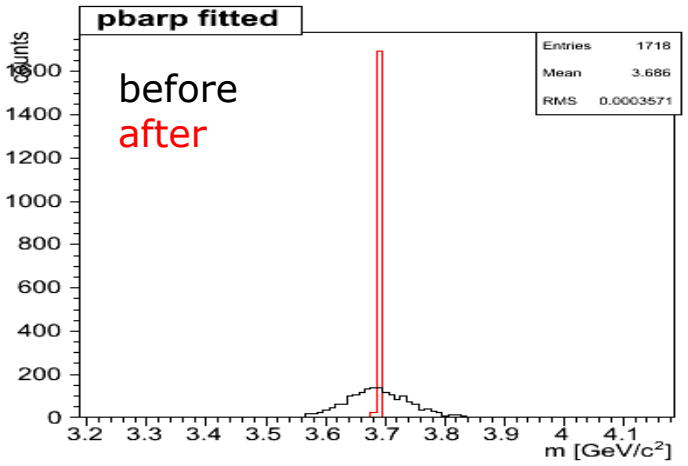
Fitting: 4 Constraint Fit

- Can be applied in *exclusive* analysis, where the initial 4-vector is fully reconstructed



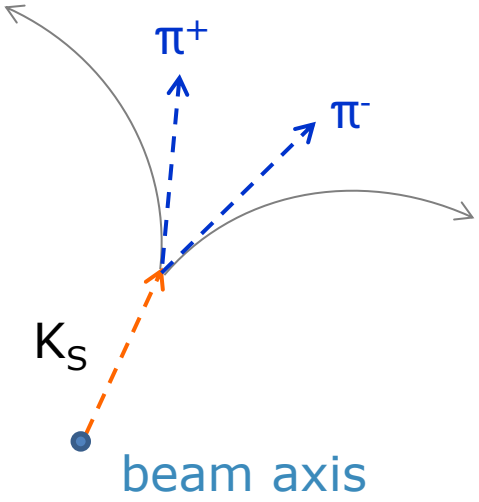
1. Fitted total system is not very meaningful (spiked mass), but intermediate resonances can be improved
2. Can do a cut on the fit probability to reject background

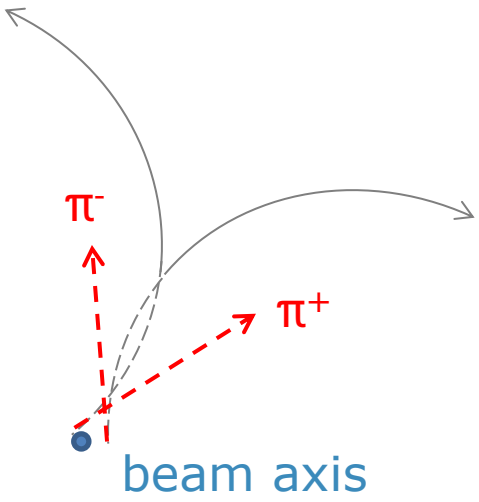




Vertex Fitting for Long-Living Particles

- For long-living particles (e. g. K_S^0, Λ^0), vertex fitting is essential, since charged particles move on helices in magnet field





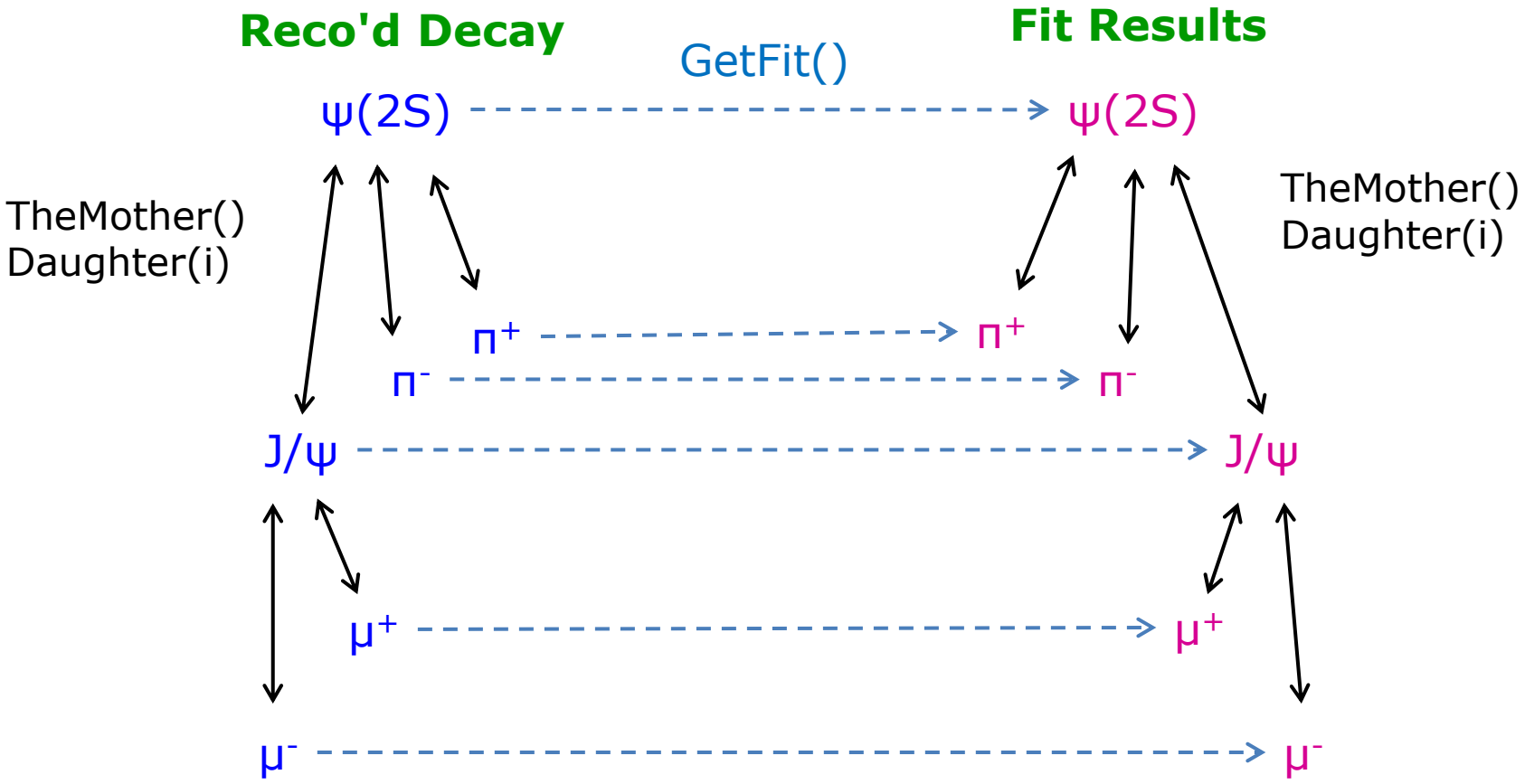
The 4-vector sum of daughters is only consistent with resonance 4-vector when evaluated at true points of origin!

Per default, the track reco computes the 4-vectors at POCA to the IP (or beam axis)
→ resulting 4-vector is bad!

⇒ Good vertex information necessary for proper resonance reco!

Fitting: Access to Results

- After fit → full fitted tree is attached to the reco object



Fitting in PandaROOT

- Fitters basically all have a similar interface
- Fit results are attached to RhoCandidates
- Can be accessed as full tree, allows cascaded fitting!
- E.g. vertex fitting + mass fitting might look like this:

```
RhoCandidate *lambda = pplus->Combine( piminus );
TVector3 IP( 0,0,0 );
  ...
PndKinVtxFitter fitvtx( lambda );                  // setup vertex fitter
fitvtx.Fit();                                      // perform fit

RhoCandidate *lambda_vtx = lambda->GetFit();       // access fit results

PndKinFitter fitmass( lambda_vtx );                // setup mass fitter
fitmass.SetMassConstraint( 1.115 );                // set mass constraint
fitmass.Fit();                                     // perform fit

RhoCandidate *lambda_mass = lambda_vtx->GetFit(); // access cascaded fit results

RhoCandidate *fit_pplus   = lambda_mass->Daughter( 0 );
RhoCandidate *fit_piminus = lambda_mass->Daughter( 1 );
```

Fitting Example: Vertex-Fit

- PandaROOT object: **PndKinVtxFitter**

```
// ... in event loop ...

for (j=0; j<jpsi.GetLength(); ++j)
{
   PndKinVtxFitter vtxfitter(jpsi[j]);       // instantiate vertex fitter
   vtxfitter.Fit();                          // perform fit

   RhoCandidate *jfit = jpsi[j]->GetFit();   // get fitted candidate

   TVector3 jVtx = jfit->Pos();              // and the vertex position
   double chi2_vtx = vtxfitter.GetChi2();    // and the chi^2 of the fit

   if ( chi2_vtx<max_chi )                   // if chi2 is good enough
   {                                         // fill some histos
      hjpsim_vf->Fill( jfit->M() );
      hvpos->Fill( jVtx.X(),jVtx.Y() );
   }
}
```

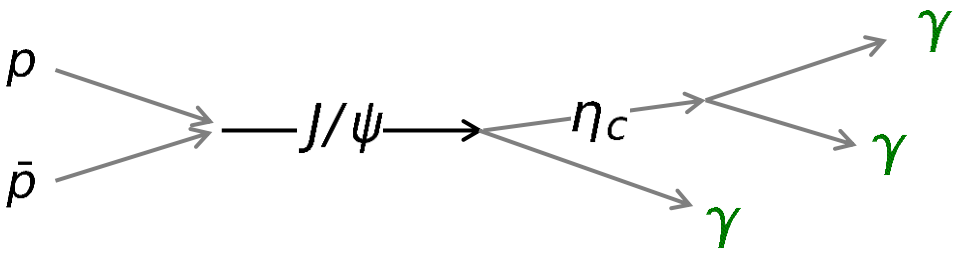
Fitting Example: 4C-Fit

- PandaROOT object: **PndKinFitter**

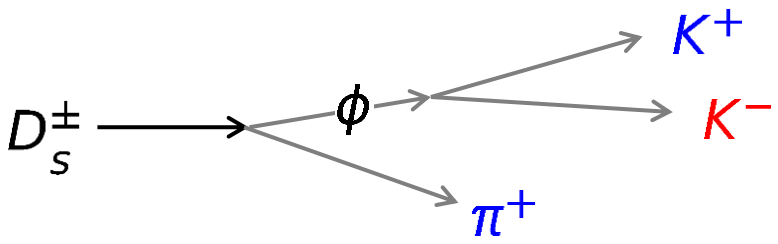
```
// the lorentz vector of the initial system; important for the 4C-fit

TLorentzVector ini(0, 0, 6.232, 7.240);

  // ... in event loop ...
  for (j=0;j<psi2s.GetLength();++j)
  {
    PndKinFitter fitter(psi2s[j]);              // instantiate the kinematic fitter
    fitter.Add4MomConstraint(ini);              // set 4 vector constraint
    fitter.Fit();                               // perform fit

    RhoCandidate *jfit = psi2s[j]->Daughter(0)->GetFit(); // get fitted J/psi

    Double_t chi2 = fitter.GetChi2();           // and the chi^2 of the fit

    if ( 0 != jfit )
    {
        hjpsim_4cf->Fill( jfit->M() );          // fill histogram
    }
}
```

Which constraints do you see?

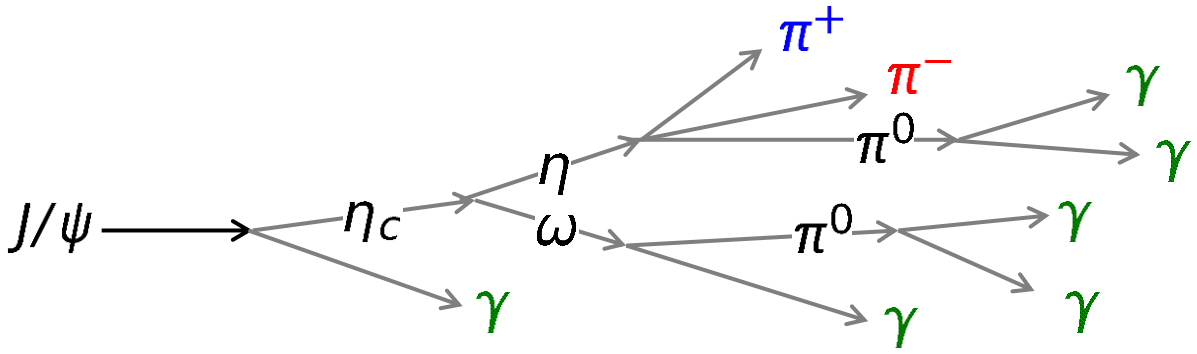
1



2



3



Particle	Γ [MeV]
J/ψ	0.09
η_c	28.6
D_s	0
ϕ	4.26
ω	8.49
η	0.001
π^0	0

UNCERTAINTIES

Uncertainties: Figure of merit

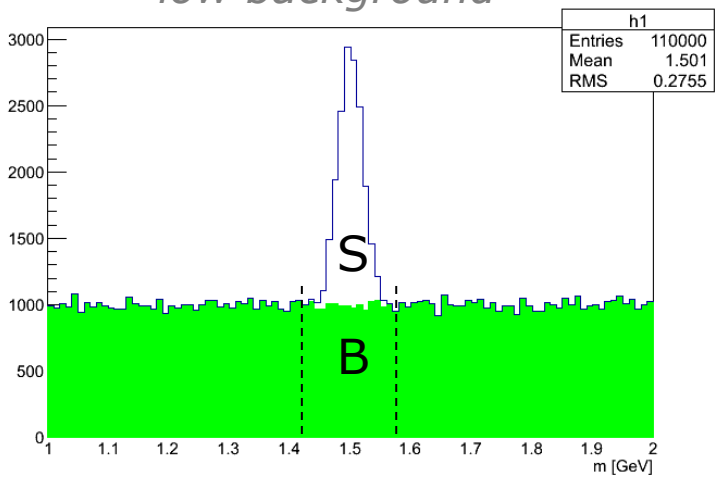
- Figure of Merit for analysis: Minimization of the relative error!
  (not, as you might think, the measured value itself...)

- In cut & count analyses, usually maximize significance $\frac{S}{\sqrt{S+B}}$

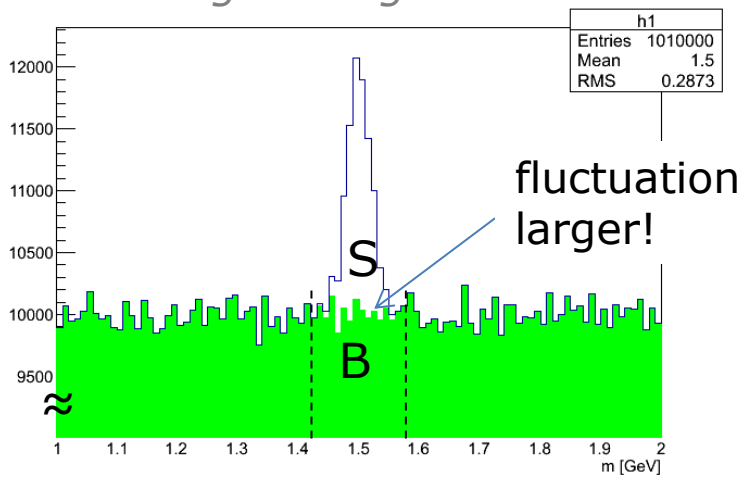
- Explanation: Statistical fluctuation of S is $\sqrt{S+B}$ *(see below)*

→ relative uncertainty = $\frac{\sqrt{S+B}}{S}$ = 1/significance!

low background



high background



Statistic and Systematic Uncertainties

The most common kinds are

- Statistic uncertainties
  - Rule of thumb: Errors, which get smaller when taking more data are statistic errors. *(sometimes also sys. errors get smaller with more statistics)*

- Systematic uncertainties
  - Usually do not decrease with more data (when systematics dominate, you can stop measuring longer)
  - *Examples*
    - Track reconstruction efficiency *(BABAR: 1-2%/track)*
    - PID selection efficiency *(BABAR: 1-2%/particle)*
    - Choice of fit model *(e.g. 2nd vs. 3rd order bgk-polynomial)*

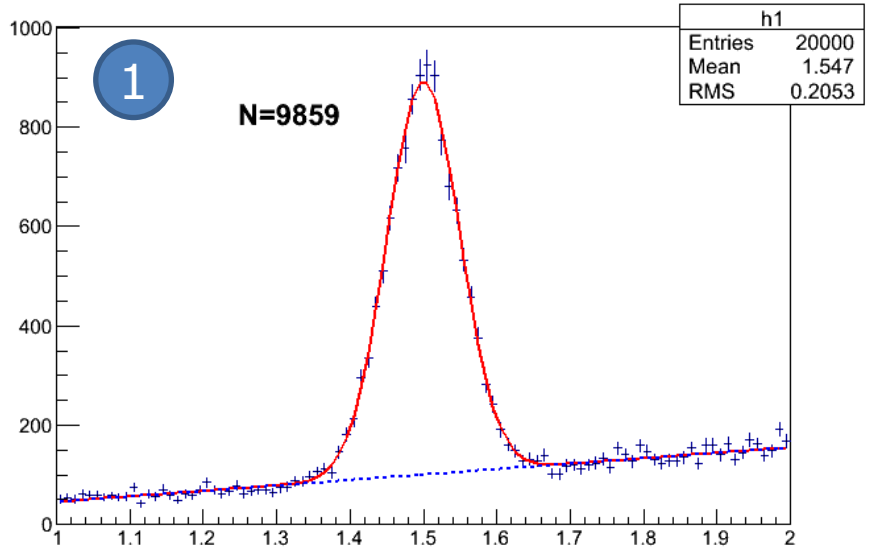
Systematics Example: Fit model uncertainty

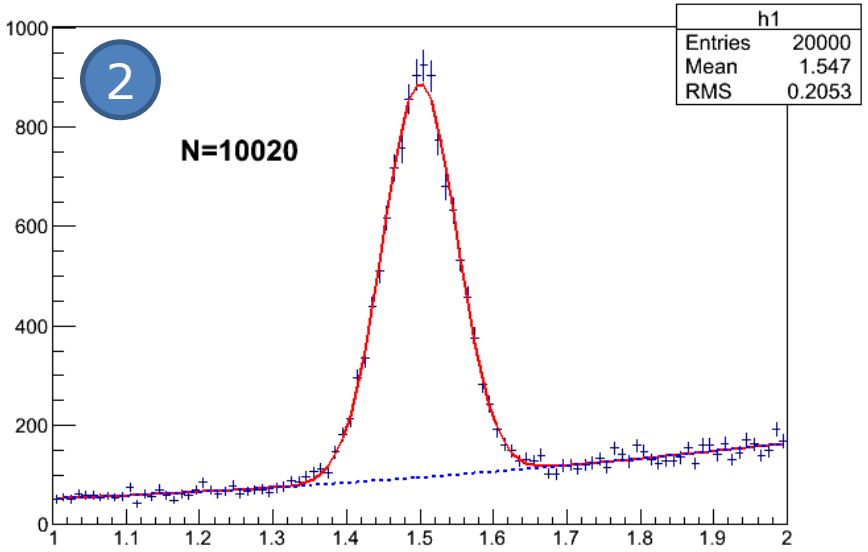
Task: Determine number of signals ontop of background

- *Bgk-Model 1:* Linear; integral $N_1 = 9859$
- *Bgk-Model 2:* Parabolic; integral $N_2 = 10020$

- Estimate for systematic error (fit model) could be

$$\Delta N_{sys,fit} = \frac{|N_1 - N_2|}{(N_1 + N_2)/2} = 1.6\%$$





EXERCISES

Exercises Suggestions

- Rho Tutorial website:
  http://panda-wiki.gsi.de/cgi-bin/view/Computing/PandaRootRhoTutorial
- Take a look to `tutorials/thailand2017/README`
  `Exercises: #2 - #6`

1. Apply different PID selections and algorithms and compare

2. Apply MC truth match for your reconstructed channels

3. Try different fitters/constraints and compare resolution

Exercises Preparation

Preparation/hints in tutorials/thailand2017/README

- **FIRST setup environment: source ~/workshop/build/PandaRoot_trunk.../config.sh**

- To have some data for tutorial macros, do one of the following

```
a) . ./tut_runall.sh 1000   # sim/reco 1000 events pbarp -> J/psi pi+ pi-
b) cp data/signal_p*root .  # preproduced default data
```

- Macros named 'tut_ana...C' are stubs and should be completed by you.

- At places marked with ' #### EXERCISE: ...' some code needs to be added;

```
Run macros (default or different input data) with
> root -l tut_ana...C                 # signal_pid.root, signal_par.root
> root -l 'tut_ana...C(0,"mydata")'   # mydata_pid.root, mydata_par.root
```

- If getting stuck, sample solutions are in the subfolder 'solution'

```
Run solution macros directly (default or different input data) with
> root -l solution/tut_ana...C
> root -l 'solution/tut_ana...C(0,"mydata")'
```