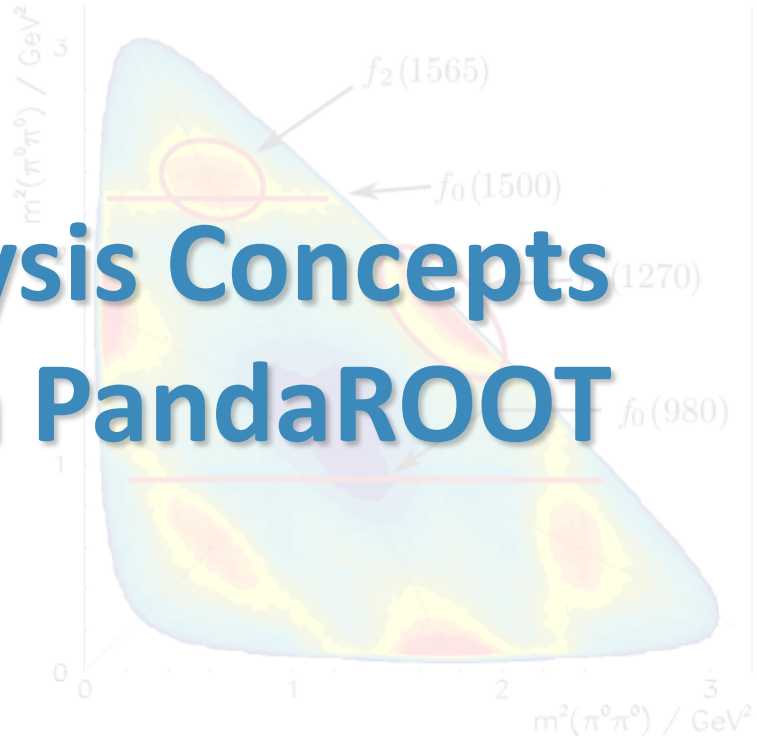
Physics Analysis Concepts with PandaROOT



PANDA Computing Workshop 2012

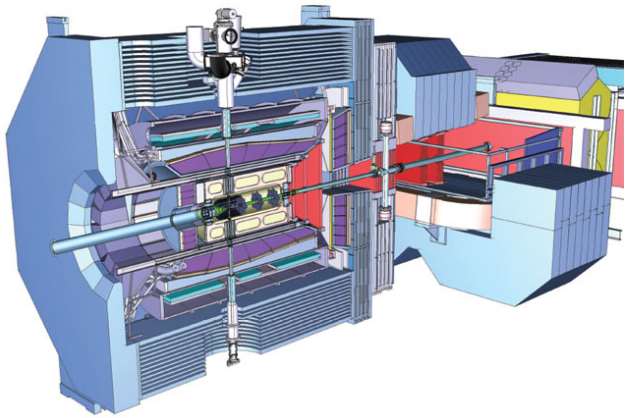
Torino, Italy, July 23-27, 2012

Klaus Götzen
GSI Darmstadt

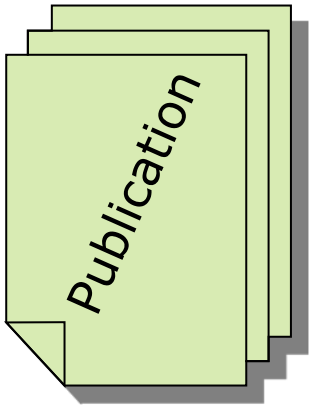
Topics

- Data Levels
- Data Access (*PndAnalysis*)
- Analysis Basics
- Particle Candidates (*TCandidate*, *PndPidCandidate*)
- Combinatorics (*TCandList*)
- Particle Identification
- Particle Selectors (*PndPidCombiner*, *PndPidSelector*)
- MC Truth Match (*PndMcTruthMatch*)
- Fitting Basics
- Fitting Tools (*PndKinVtxFitter*)
- Uncertainties

The Aim



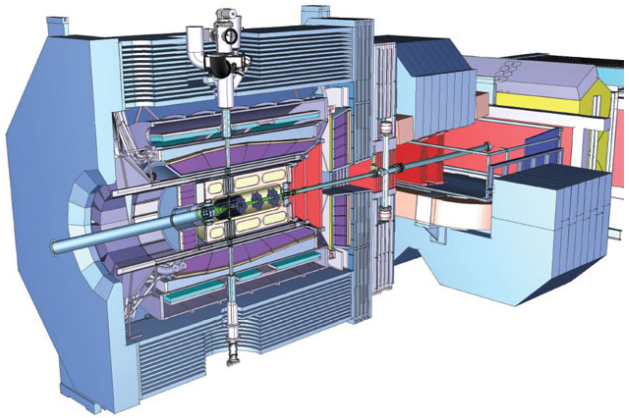
Data Analysis



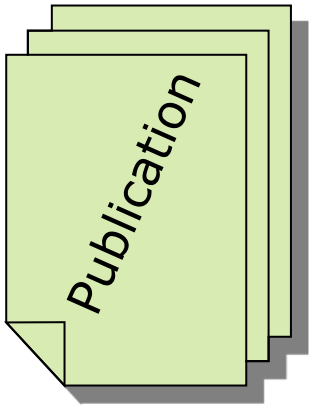
- What do you want to analyse/investigate?
- How can it be measure (what are the observables)?

- Reconstruct the corresponding signal channels!
- Determine the needed quantities!
- Estimate the errors (statistic & systematic)!
- Write paper
- Get invited to Stockholm

The Aim



Data Analysis



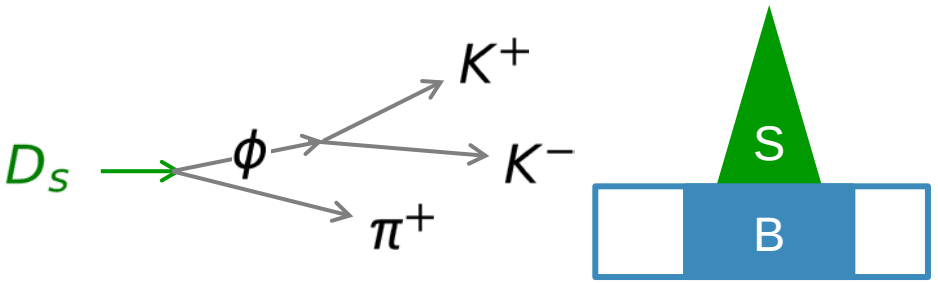
- What do you want to analyse/investigate?
- How can it be measure (what are the observables)?

- Reconstruct the corresponding signal channels!
- Determine the needed quantities!
- Estimate the errors (statistic & systematic)!
- Write paper
- Get invited to Stockholm

Gonna talk about that one!

Example: Count Number of D_s in Data

- Reconstruct signal in data; maximize e.g. significance $\frac{S}{\sqrt{S+B}}$



S = entries in signal peak
B = entries below peak

- Reconstruct signal in dedicated signal Monte Carlo events; number of *MC truth matched* signals give efficiency

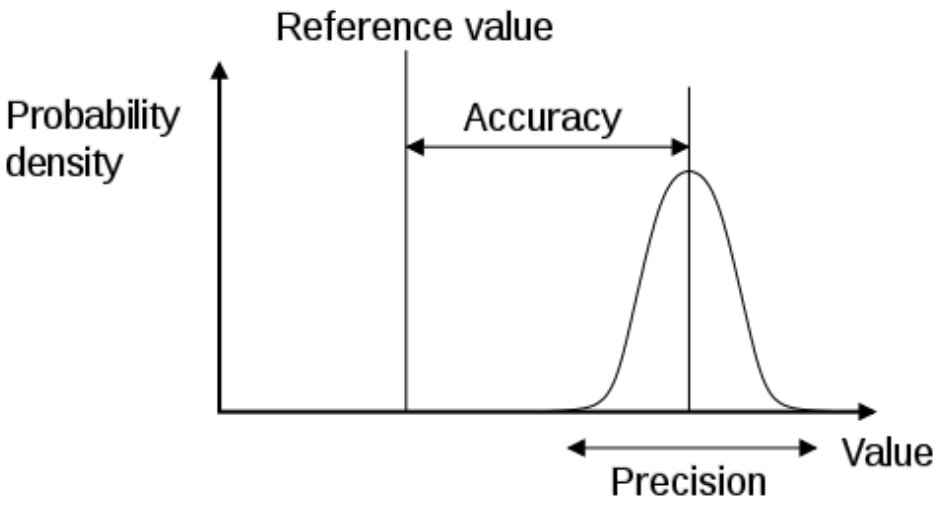
$$\varepsilon = \frac{N_{reco,MC}}{N_{0,MC}}$$

- Number of D_s in data is (taking into account BR's)

$$N_{D_s} = \frac{S}{\varepsilon \cdot \mathcal{BR}(D_s \to \phi\pi^{\pm}) \cdot \mathcal{BR}(\phi \to K^+K^-)}$$

Goal for Experimentalist

- **Theoreticians**: interested in measured **value**
  - $M_{Higgs} \approx$ **125** GeV
- **Experimentalists:** interested in **precision / accuracy (error)**
  - M_{Higgs} = X **±** **2** GeV



High accuracy
(low systematic error)

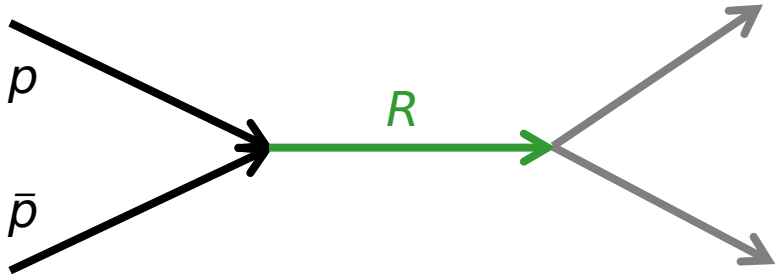
High precision
(low statistic error)

BASICS

Reactions in PANDA

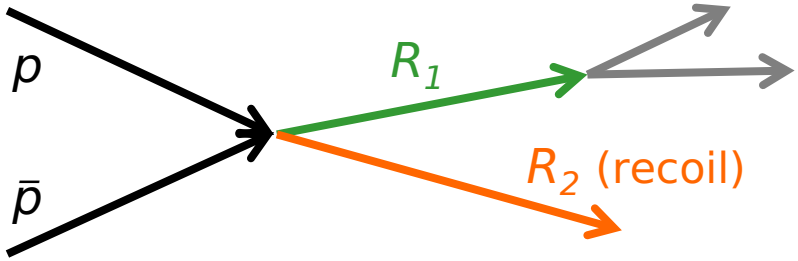
There are two different kinds of principle reactions in PANDA

- **Formation** reactions
  The $\bar{p}p$ system transforms into single resonance state R, which decays afterwards



p̄ beam momentum (→ √s) is fixed by mass of R

- **Production** reactions
  The $\bar{p}p$ system transforms into more than one resonance



p̄ beam momentum can vary due to kinematic recoil R_2

Data Levels

What kind of data (levels) do we have in an experiment?

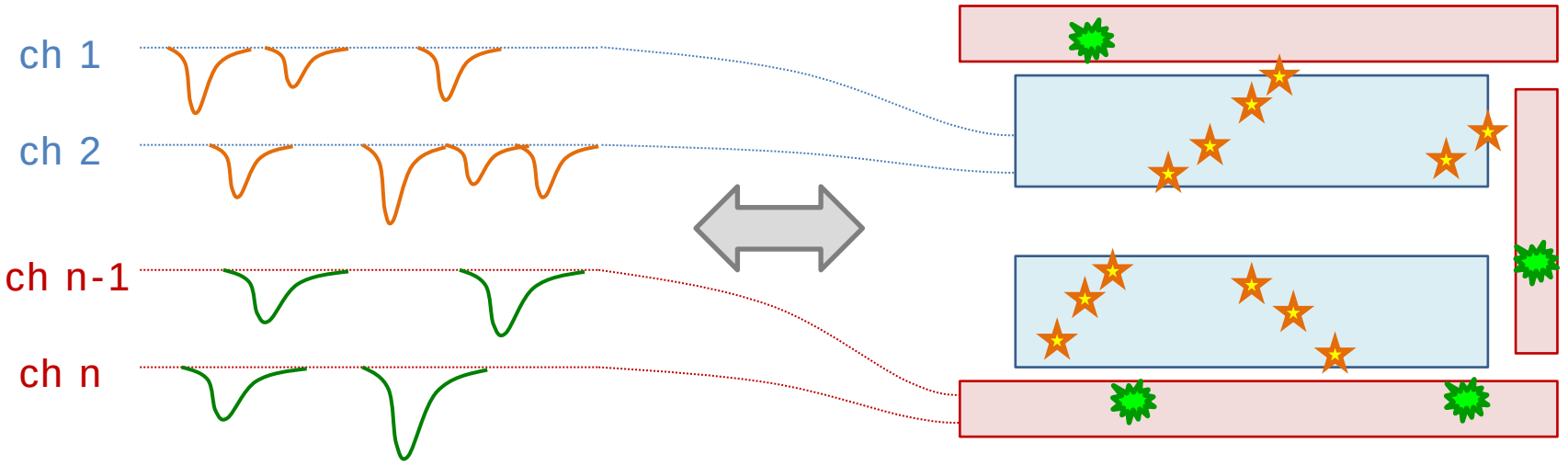
- Raw data
  The stream of digitized hits from the detectors
  Basis for track/neutral cluster/PID reconstruction

- Analysis Object Data (AOD) / Data Summary Tape (DST)
  Reconstructed objects like tracks and neutrals with associated PID information, bundled to event based data packages
  Basis for physics analysis

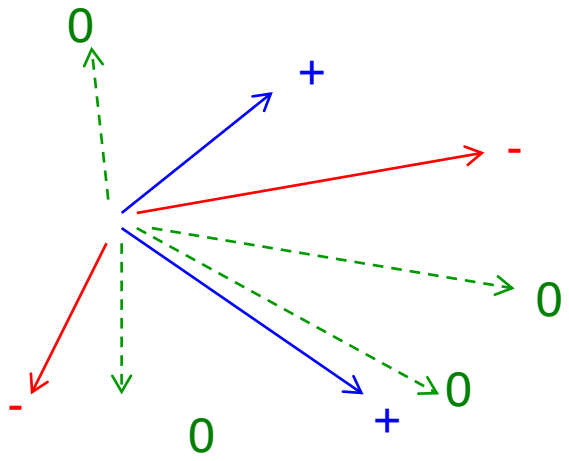
- Tag level data
  Condensed information of events like total energy or momentum, multiplicities (tracks, gammas, kaons, etc.)
  Fast data filtering

Data Levels

RAW data



AOD data



TAG data

2 positive charged (+)
2 negative charged (-)
4 neutral (0)
…

Data Access in PandaROOT

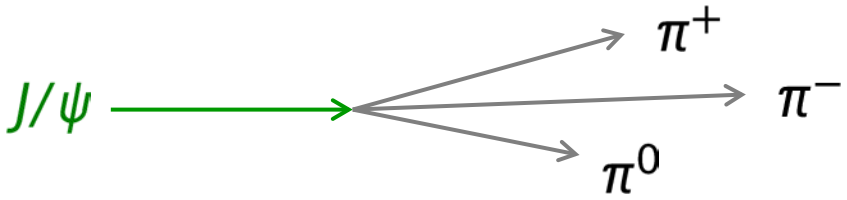
- PandaROOT object: **PndAnalysis**

```
FairRunAna *fRun = new FairRunAna();
fRun->SetInputFile("pid_reco.root");      // output pid/reco stage
fRun->AddFriend("points.root");           // output simulation stage

PndAnalysis *pndana= new PndAnalysis();
...
while (pndana->GetEvent())
{
  pndana->FillList(eplus,  "ElectronLoosePlus");  // access to reco candidates
  pndana->FillList(eminus, "ElectronLooseMinus");
  ...
  pndana->FillList(mcList, "McTruth");    // access to monte carlo truth list

  /* do some analysis things*/

}
```

- Features:
  - Simple access to reco'd candidates and McTruth objects

The basis for Analysis: Invariant masses

- Q: How do we detect a certain resonance decay?



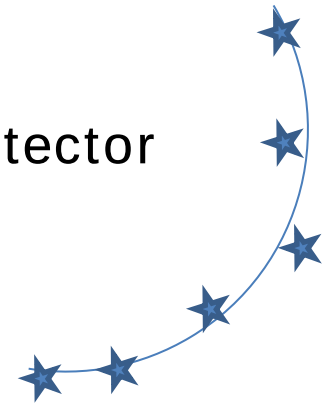
- A: *Estimate* the 4-vectors of the final states (FS), sum up and compute the invariant mass

$$m_{inv} = \sqrt{\left(\sum E\right)^2 - \left(\sum p_x\right)^2 - \left(\sum p_y\right)^2 - \left(\sum p_z\right)^2}$$

It should be ‚close' to the resonance rest mass.

Why did I write estimate?
Usually only momentum (charged) or energy (neutrals) from FS particles is measured; the so-called *mass hypothesis* has to be set during analysis, typically based on PID detector information.

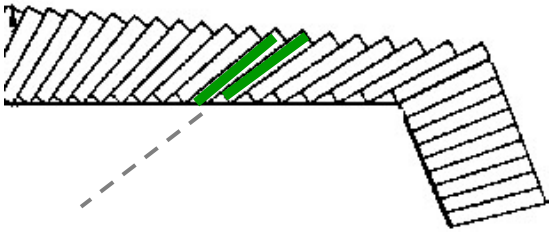
How 4-vectors are reconstructed



Charged particles

- Reconstruction of trajectory with tracking/vertex detector
- Curvature in magnetic field → momentum p
- Assume position on trajectory → 3-vector (p_x, p_y, p_z)
- Assume mass hypothesis m → $E = \sqrt{m^2 + p^2}$

Neutral particles (gammas)

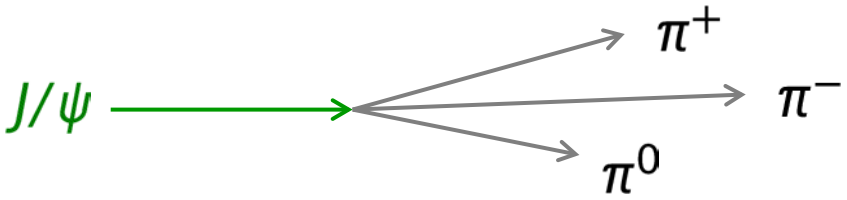


- Reconstruction of cluster in EM-calorimeter
- Cluster energy → particle energy E
- Assume gamma mass $m = 0$ → $p = E$
- Assume gamma comes from IP → 3-vector (p_x, p_y, p_z)

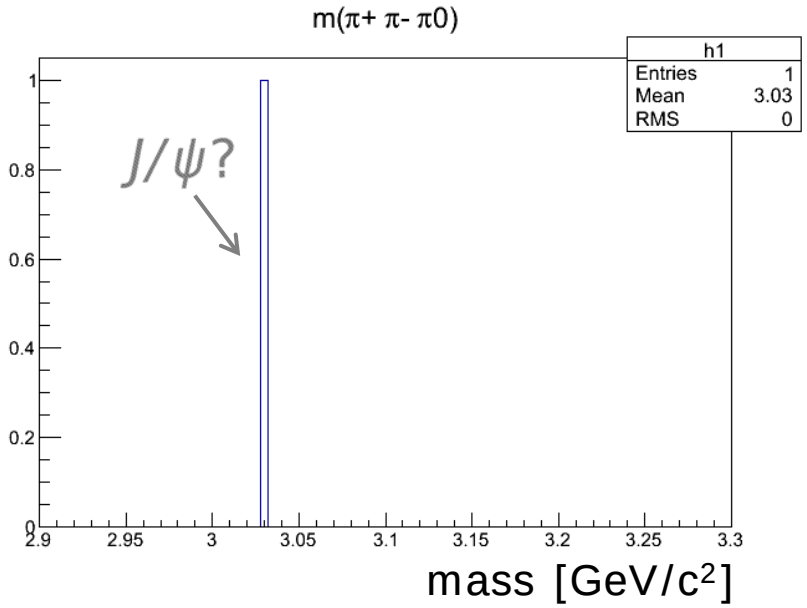
⇒ *Determination of 4-vectors is not an unbiased process!*

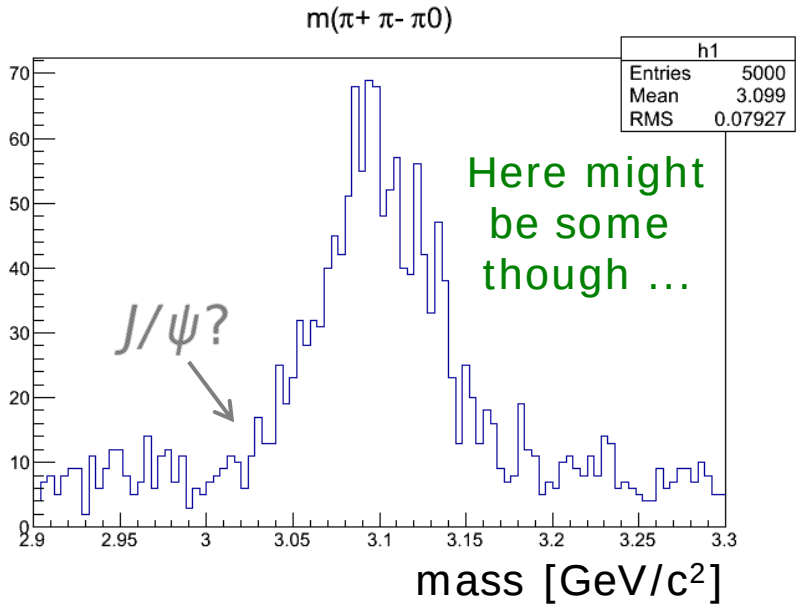
The basis for Analysis: Invariant masses

- Q: How do we detect a certain resonance decay?



- A2: Not at all (in principle) for a single decay!





Particle Candidate in PandaROOT

- PandaROOT object: **TCandidate**
- Some important accessors:

Return type	Method	Information
TLorentzVector	P4()	4-vector
TVector3	P3()	Momentum vector
TVector3	Pos()	Origin
Double_t	Charge()	Charge
Double_t	M(), P(), E()	Mass, Momentum, Energy
Int_t	NDaughters()	Number of daughters
TCandidate*	Daughter(i)	i-th daughter in decay tree
TCandidate*	TheMother()	Mother in decay tree
TCandidate*	Combine(TCandidate &c,...)	Creates composite candidate
Bool_t	Overlaps(TCandidate &c)	Checks for collision in tree
Int_t	GetMcIdx()	Matching index in MC truth list
Int_t	PdgCode()	PDG code (MC truth particle)
VAbsMicroCandidate&	GetMicroCandidate()	Access to reco information

Micro Candidate in PandaROOT

- PandaROOT object: **PndPidCandidate** *(VAbsMicroCandidate)*
- Some accessors:

Return type	Method	Information
Int_t	GetMvdHits()	Number of MVD hits
Float_t	GetMvdDEDX()	dE/dx measurm. from MVD
Int_t	GetSttHits()	Number of STT hits
Float_t	GetSttMeanDEDX()	dE/dx measurm. from STT
Float_t	GetDrcThetaC()	θ_C measurement from DIRC
Int_t	GetDrcNumberOfPhotons()	Number Photons in DIRC
Float_t	GetEmcCalEnergy()	Calibrated EMC cluster energy
Int_t	GetEmcNumberOfCrystals()	Number of crystals in cluster
Int_t	GetMuoNumberOfLayers()	Number of layers hit in MUO det
Float_t	GetMuoProbability()	Probability for being a muon

Work with TCandidate/PndPidMicroCandidate

- Using PndPidCandidate

```
...
PndAnalysis *pndana= new PndAnalysis();

TCandList piplus, piminus, goodpiplus, goodpiminus;

while (pndana->GetEvent())
{
  pndana->FillList(piplus,  „PionAllPlus");  // access to reco candidates
  pndana->FillList(piminus, „PionAllMinus");

  for (int i=0; i<piplus.GetLength(); ++i)
  {
     // get micro candidate and use it for selection
     PndPidCandidate mic = (PndPidCandidate&) piplus[i].GetMicroCandidate();

     if (mic.GetSttHits()>5) goodpiplus.Add(piplus[i]);
  }

 ...
}
```

General handling of TCandidate

- TCandidate = basic analysis object, many instances needed
- **TFactory** does the book-keeping of instances

- Creation of new TCandidates *(when done by hand):*

   TCandidate *c = TFactory::Instance()->NewCandidate();

   TCandidate *c = new TCandidate(); NEVER!!

- Delete TCandidates:

   Not necessary!! *TFactory takes care of it!*

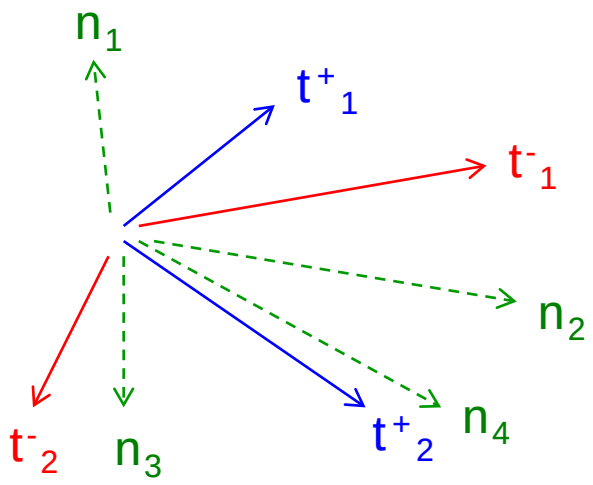
   delete c; NEVER!!

actually, it's not super-bad, but life is easier w/o

COMBINATORICS

Combining Candidates

- Remember the example event shown above



- Question: How do we find out, whether there was a decay

$$J/\psi \rightarrow \pi^+\pi^-\pi^0(\rightarrow \gamma\gamma)$$

Combinatorics

- Answer: Create all combinations of FS particles, which match the decay pattern and create histogram of inv. mass.

- Our example:
  2 positive (t^+_1, t^+_2), 2 negative tracks (t^-_1, t^-_2), 4 neutrals (n_k)

- π^0 candidates → 6 combinations

  $\pi^0_1 = (n_1 + n_2)$, $(n_1 + n_3)$, $(n_1 + n_4)$, $(n_2 + n_3)$, $(n_2 + n_4)$, $\pi^0_6 = (n_3 + n_4)$

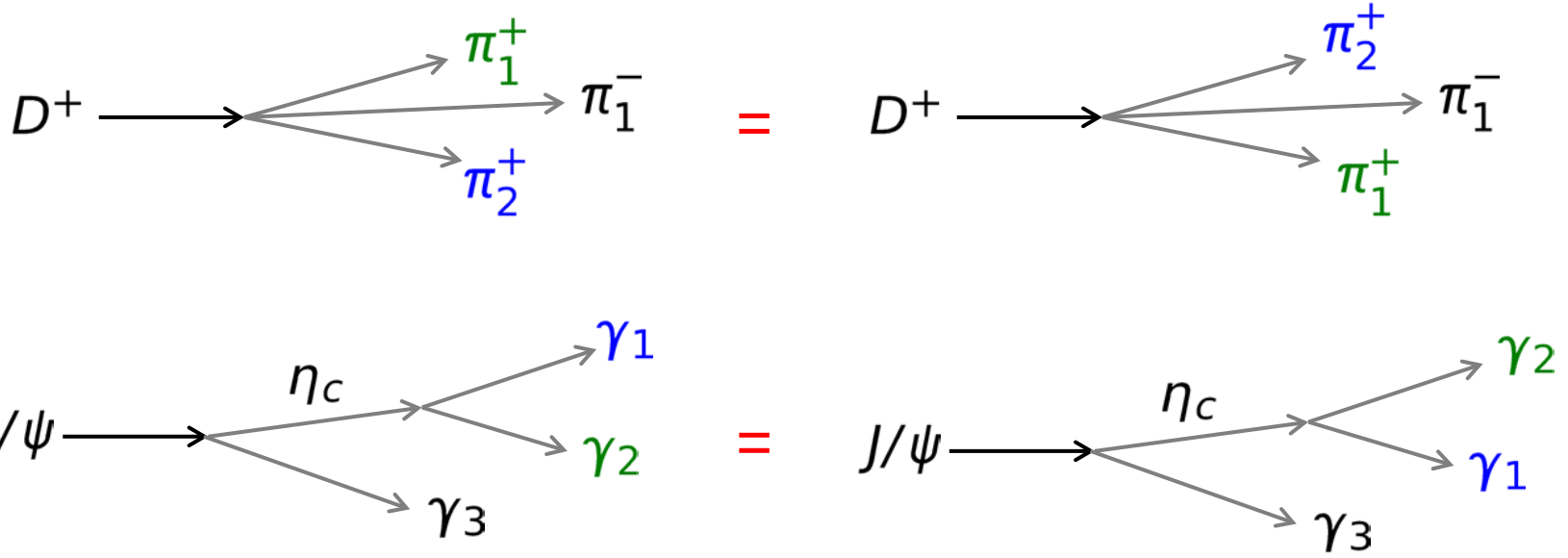
- J/ψ candidates → 24 combinations

  $J/\psi_1 = (t^+_1 + t^-_1 + \pi^0_1)$, $J/\psi_2 = (t^+_1 + t^-_1 + \pi^0_2)$, …
  …
  $J/\psi_{24} = (t^+_2 + t^-_2 + \pi^0_6)$

Caveat: Double counting

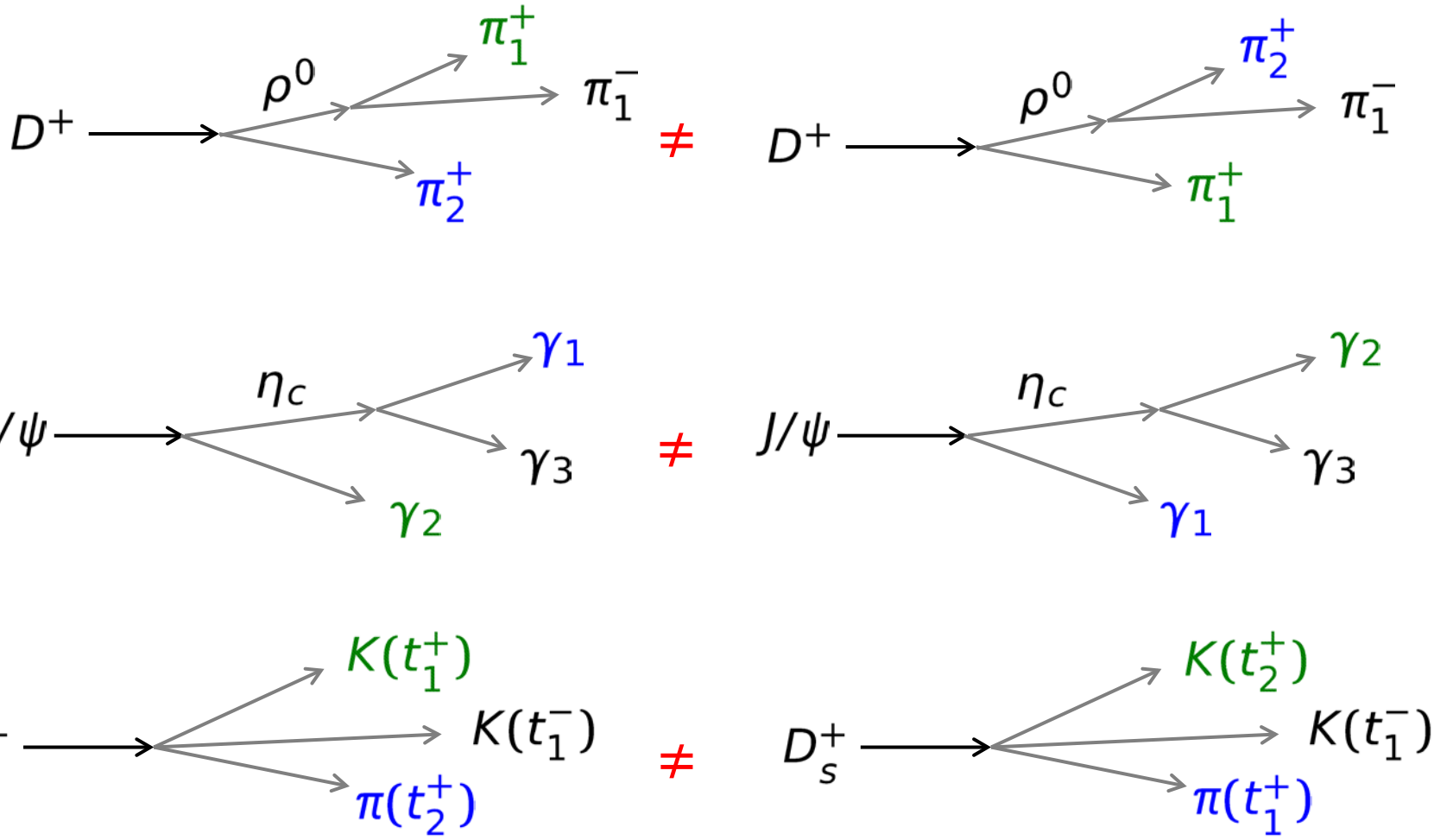
- Double counting = erroneous creation of identical combinations leading to multiple entries in spectra/lists
- Can happen in wrong coded nested loops

- Examples for double counting:



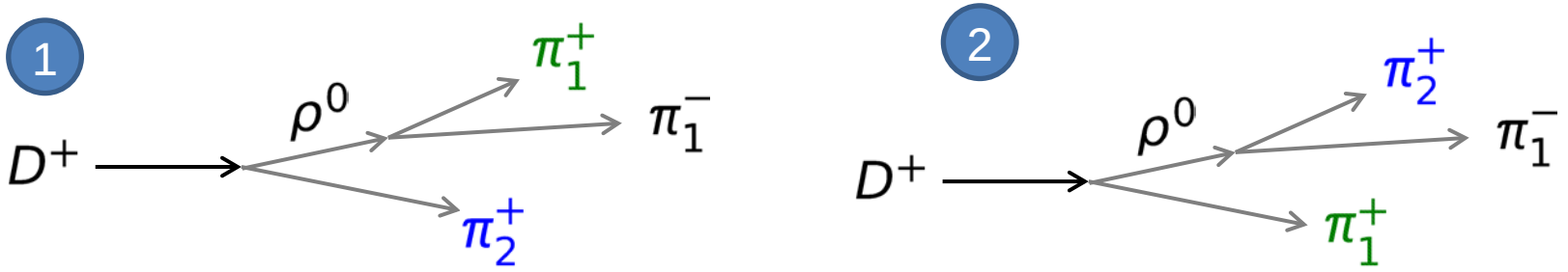
Caveat: Double counting

No double counting here *(decay trees have different topology!)*:



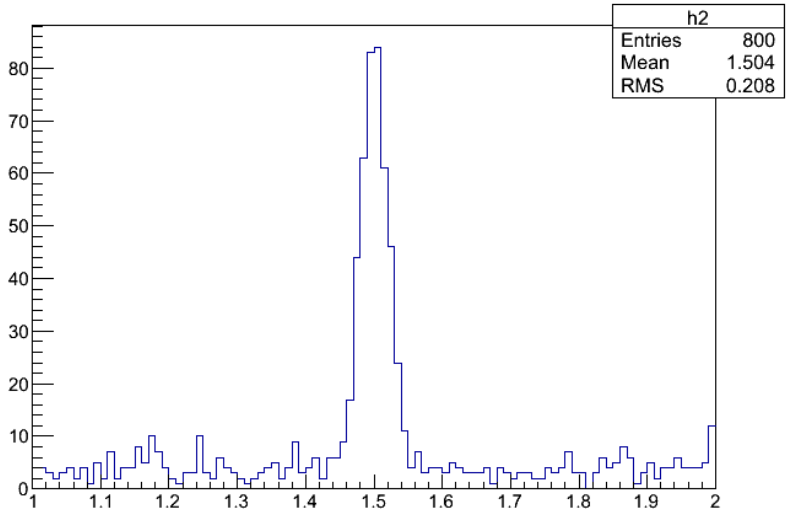
Caveat: Double counting

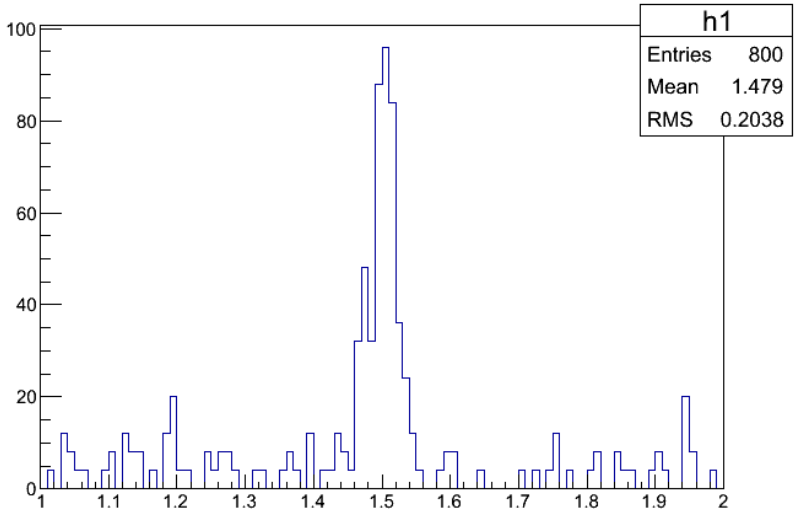
But be aware:

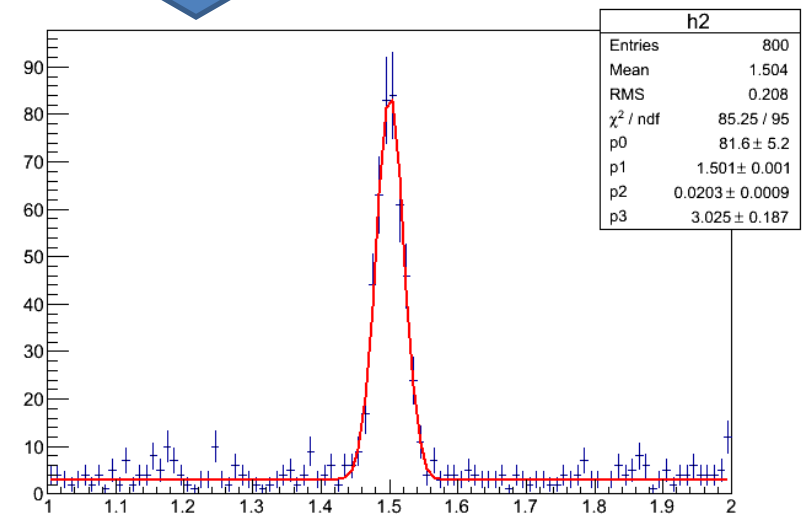


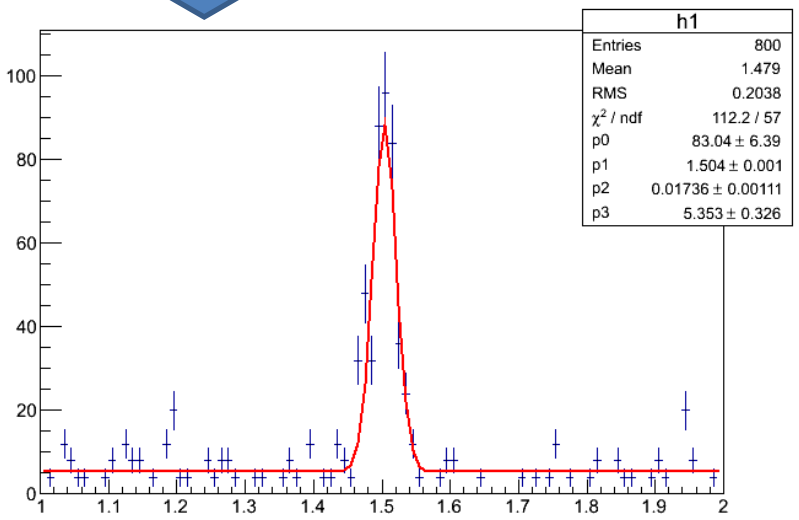
- If e. g. a cut is applied to the ρ^0 mass, only one possibility of (1) and (2) might be selected.
- If both combinations are selected, you have two different decay trees with exactly the same 4-vector of the D^+
- *Is this now double counting?*
- Simple recipe: *Just keep one (the best) solution per event*

Double counting: How it might look like



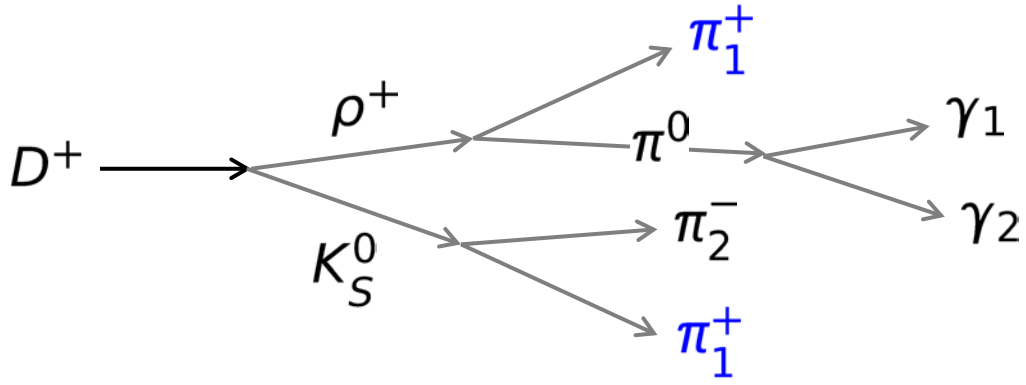


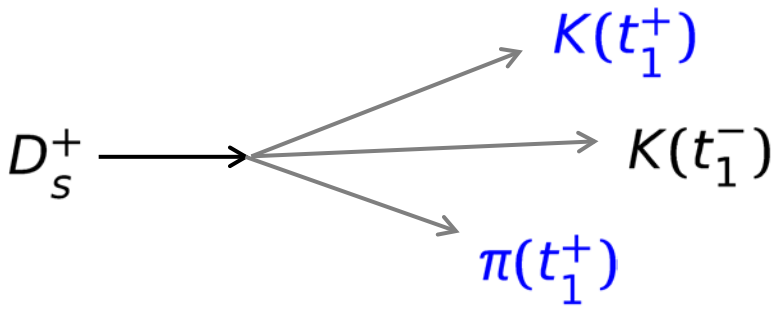




Caveat: Overlaps

- Overlap = erroneous multiple use of the same reco object in a decay tree
- Can happen when working with composite candidates or different mass hypotheses





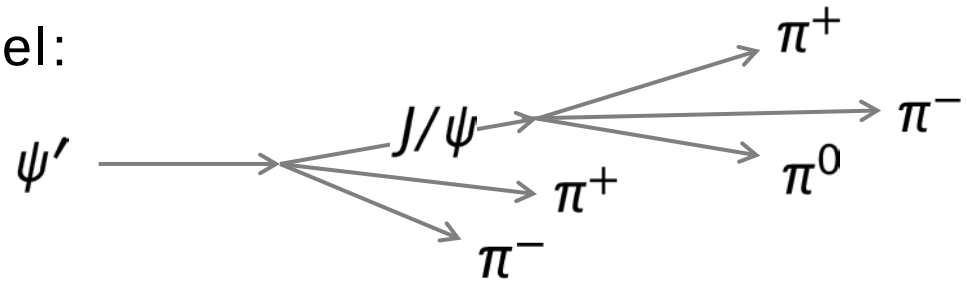
Particle List in PandaROOT

- PandaROOT object: **TCandList**
- Some important accessors:

Method	Information
Int_t GetLength()	number of particles in list
void Add(TCandidate&)	Appends candidate to list
void Remove(TCandidate&)	Removes candidate from list
TCandidate& operator[](Int_t)	Returns a candidate
void Append(TCandList&)	Appends candidates from another list
void Combine(TCandList&,...)	Combinatorics for up to five TCandLists
void Select(VAbsPidSelector*)	Selects (=modifies) list
void Select(TCandList &, VAbsPid...)	Selects from another list
void Cleanup()	Empties list

Combinatorics in PandaROOT

- PandaROOT objects: **TCandList & TCandidate**
- Example channel:




```
TCandList piplus, piminus, gamma, pi0, jpsi, psip;  // define TCandList's

pndana->FillList(piplus,  „PionAllPlus");  // positive chrg trks with pion hypo
pndana->FillList(piminus, „PionAllMinus"); // negative chrg trks with pion hypo

pndana->FillList(gamma, „Neutral“);        // neutral particle candidates

pi0.Combine(gamma, gamma);                 // cares about double counting
jpsi.Combine(piplus, piminus, pi0);
psip.Combine(jpsi, piplus, piminus);       // cares about overlaps

for  (int j=0; j<psip.GetLength(); ++j)    // loop over candidates in TCandList
{
    masshisto.Fill( psip[j].M() );         // and fill e.g. a mass histo
}
```

Combinatorics in PandaROOT (by hand)

- Example channel: $J/\psi \rightarrow \pi^+\pi^-\pi^0(\rightarrow \gamma\gamma)$

```
TCandList piplus, piminus, gamma, pi0, jpsi;              // define TCandList's
...
  pi0.Cleanup(); jpsi.Cleanup();                          // clean lists !!

  for (int i=0; i<gamma.GetLength()-1; i++) {             // nested loops
    for (int j=i+1; j<gamma.GetLength(); j++) {
      if (gamma[i].E()>Emin || gamma[j].E()>Emin) {       // do e.g. some check
         TCandidate *comb = gamma[i].Combine(gamma[j]); // create comb. candidate
         pi0.Add(*comb);                                 // add it to list
      }
    }
  }

  for (int i=0; i<piplus.GetLength(); i++) {              // and the same for
    for (int j=0; j<piminus.GetLength(); j++) {           // the J/psi candidates
      for (int k=0; k<pi0.GetLength(); k++) {
          TCandidate *comb = piplus[i].Combine(piminus[j], pi0[k]);
          jpsi.Add(*comb);
      }
    }
  }
...
```

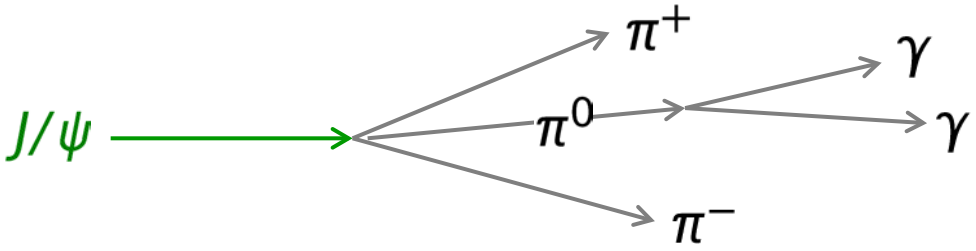
Combinatorics in PandaROOT (by hand)

- Example channel (overlap check): $D_s^{\pm} \to \phi\pi^{\pm} \to K^+K^-\pi^{\pm}$

```
TCandList pipm, kplus, kminus, ds;                    // define TCandList's
...
  ds.Cleanup();                                       // clean list

  for (int i=0; i<kplus.GetLength(); i++)             // nested loops
  {
    for (int j=0; j<kminus.GetLength(); j++)
    {
      TCandidate *phi = kplus[i].Combine(kminus[j]);  // create phi candidate

      for (int k=0; k<pipm.GetLength(); k++)
      {
        if (!pipm[k].Overlaps(*phi))                  // check for overlap
        {
          TCandidate *comb = phi->Combine(pipm[k]);
          ds.Add(*comb);
        }
      }
    }
  }
...
```

Access to Genealogy

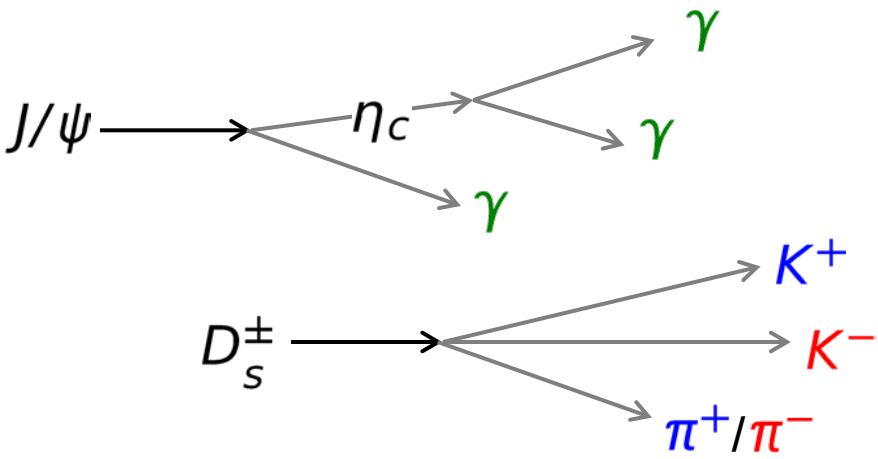
- PandaROOT object: **TCandidate**
- Example channel:

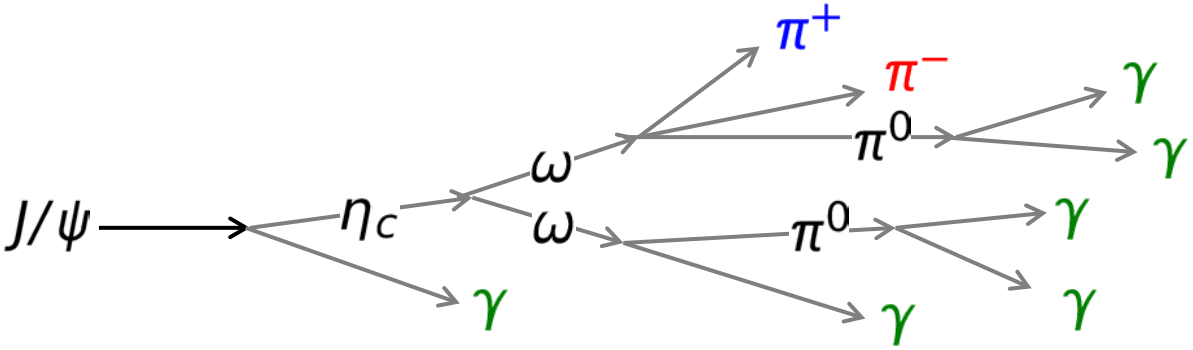



```
pi0->Combine(gamma, gamma);           // create pi0 candidates
jpsi->Combine(piplus, piminus, pi0);  // create J/psi candidates
...
cout << jpsi[i].NDaughters() <<endl;  // prints number of daughters (3, not 4!)

TCandidate *pip = jpsi[i].GetDaughter(0); // the pi+ candidate
TCandidate *pim = jpsi[i].GetDaughter(1); // the pi- candidate
TCandidate *pi0 = jpsi[i].GetDaughter(2); // the pi0 candidate (composite!)

TCandidate *gam1 = pi0->GetDaughter(0);   // the 1st gamma
TCandidate *gam2 = pi0->GetDaughter(1);   // the 2nd gamma
```

Combinatoric Exercises

- Particles in event: $3t^+$, $3t^-$, $6n$; no PID information used
- How many combinations for
  - $J/\psi \to \gamma\eta_c \to \gamma\gamma\gamma$?
  - $D_s^\pm \to K^+ K^- \pi^\pm$?
  - $J/\psi \to \gamma\eta_c \to \gamma\omega\omega \to \gamma\ \pi^+\pi^-\pi^0\ \pi^0\gamma \to \gamma\ \pi^+\pi^-\ \gamma\gamma\ \gamma\gamma\gamma$?

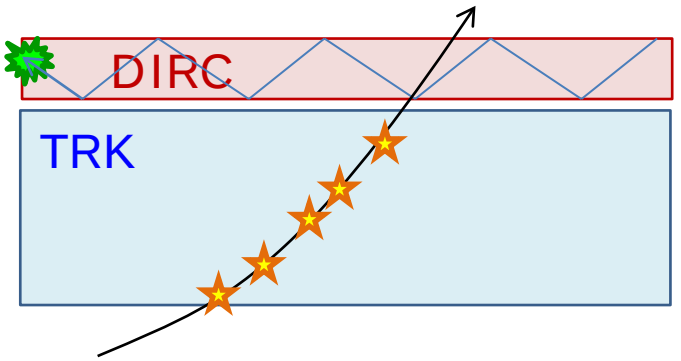




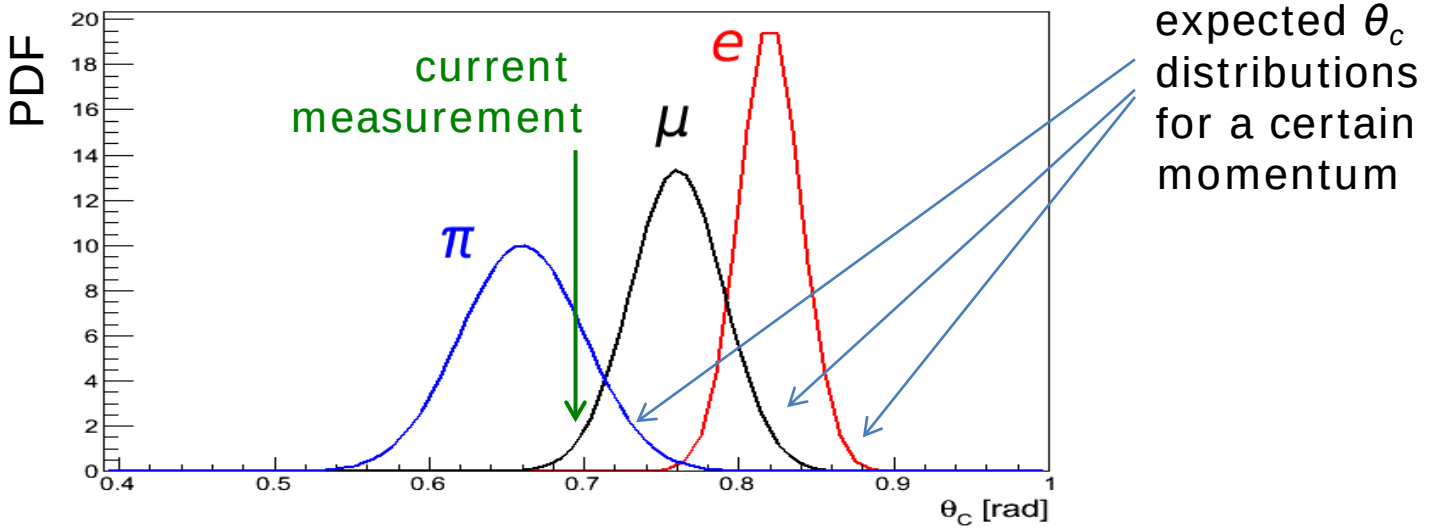
PARTICLE IDENTIFICATION

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 probability for various particle types
   (e.g. based on Likelihood functions)



PID Concept in PandaROOT

- Each PID subdetector delivers individual probability/likelihood information
- PID Combiner, which combines the probability 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 algo‘s by computing product of all P_k *(k=algo‘s)*
  - copies resulting probabilities to TCandidate/TCandList
- PndAnaPidSelector
  - selects particles based on these probabilities
- PndAnalysis::FillList is a short-cut to this funtionallity via

```
evr.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‘:

```
TCandList 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 P to candidates
  kaonSelector.Select(charged, kaonLoose);  // select kaons from charged
}
```

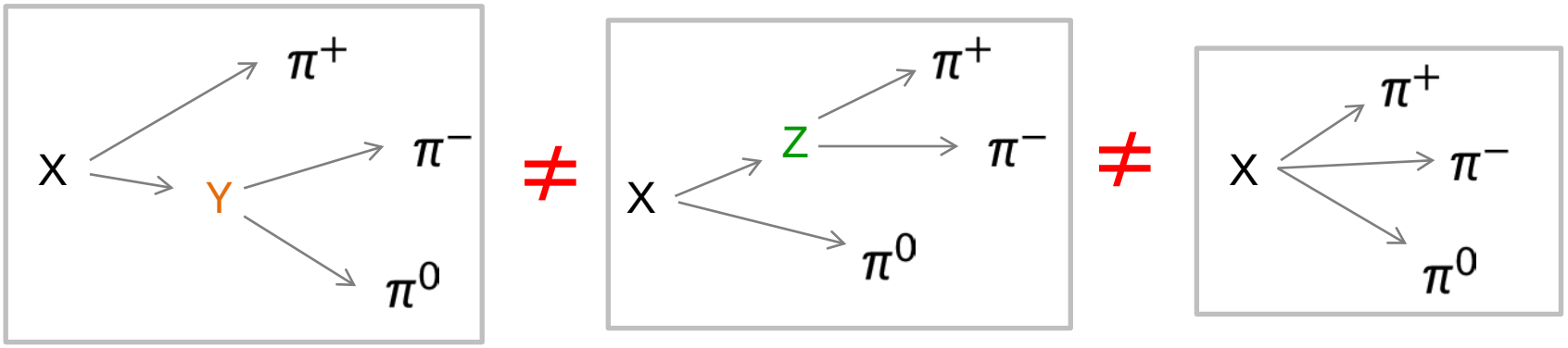
MONTE CARLO TRUTH

Monte Carlo Truth Access

- For typical measurement the reco efficiency is needed
- Look in MC, whether you combine the correct candidates
- Concerning MC truth, distinguish between
  - access MC truth object (or type) for certain reco object

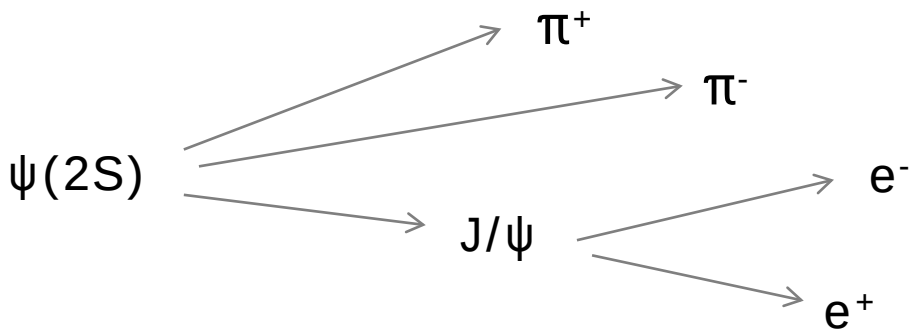
```
evr->FillList(eplus,  "ElectronLoosePlus");
evr->FillList(mcList, "McTruth");
...
TCandidate mc = mcList[eplus[i].GetMcIdx()];
```

  - MC truth match for a complete decay tree, because



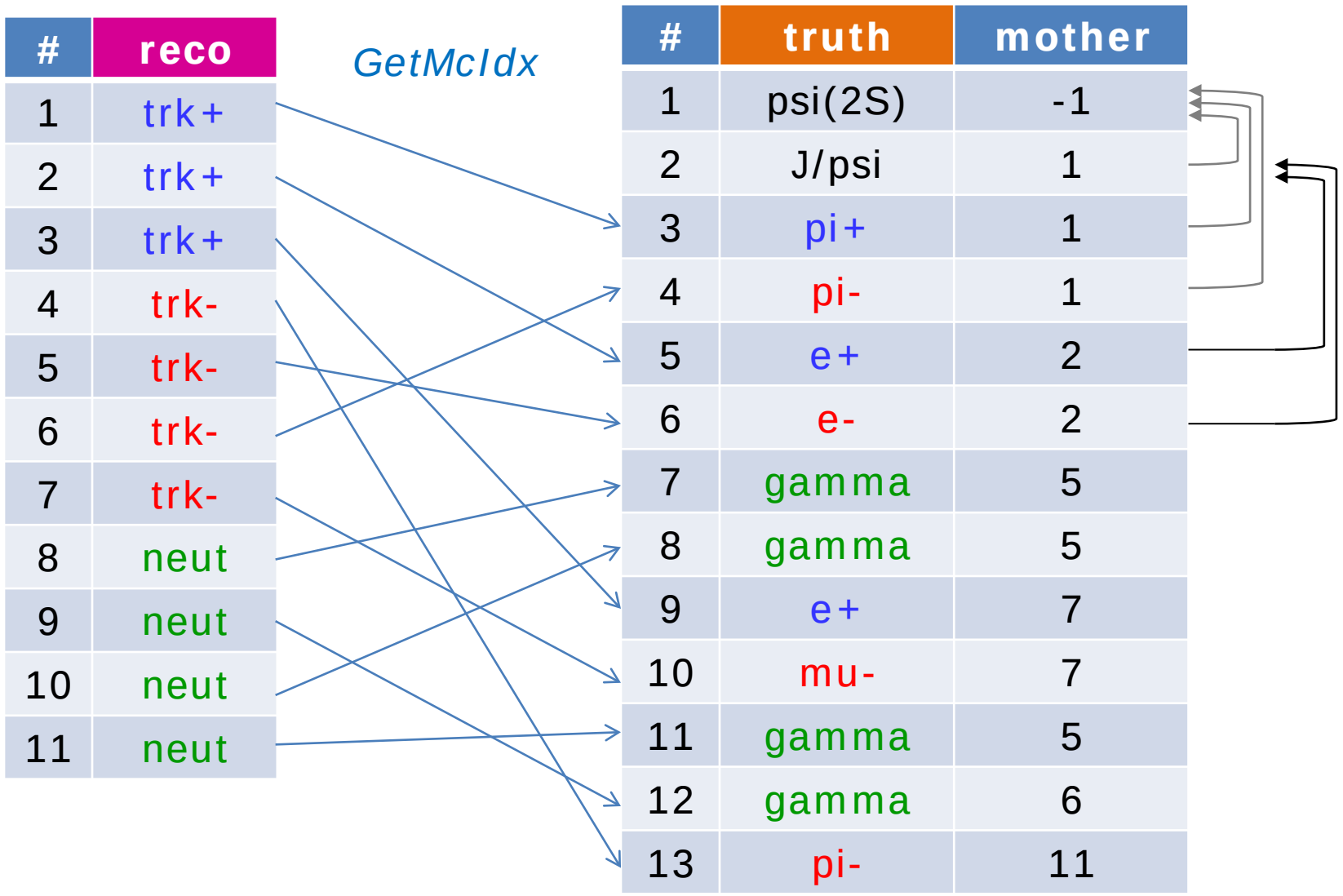
Example: ψ(2S) → J/ψ (e^+ e^-) π^+ π^-

- We want to reconstruct the following decay




```
evr->FillList( eplus,   „ElectronLoosePlus");
evr->FillList( eminus,  „ElectronLooseMinus");
evr->FillList( piplus,  „PionLoosePlus");
evr->FillList( piminus, „PionLooseMinus");

evr->FillList( mcList,  "McTruth");

jpsi.Combine( eplus, eminus );
psi2s.Combine( jpsi, piplus, piminus );
```

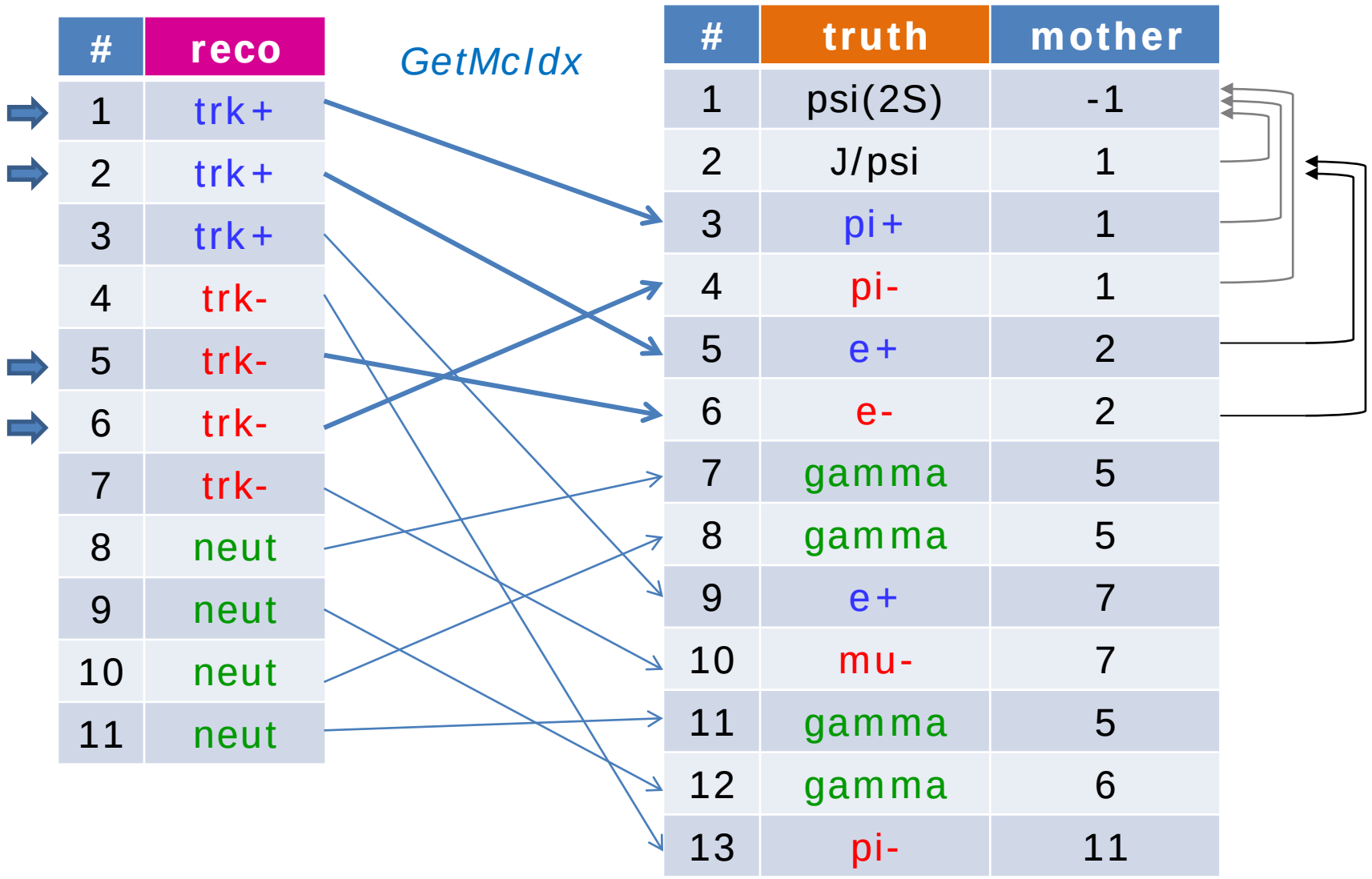
Example: ψ(2S) → J/ψ (e^+ e^-) π^+ π^-



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

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

Example: ψ(2S) → J/ψ (e^+ e^-) π^+ π^-

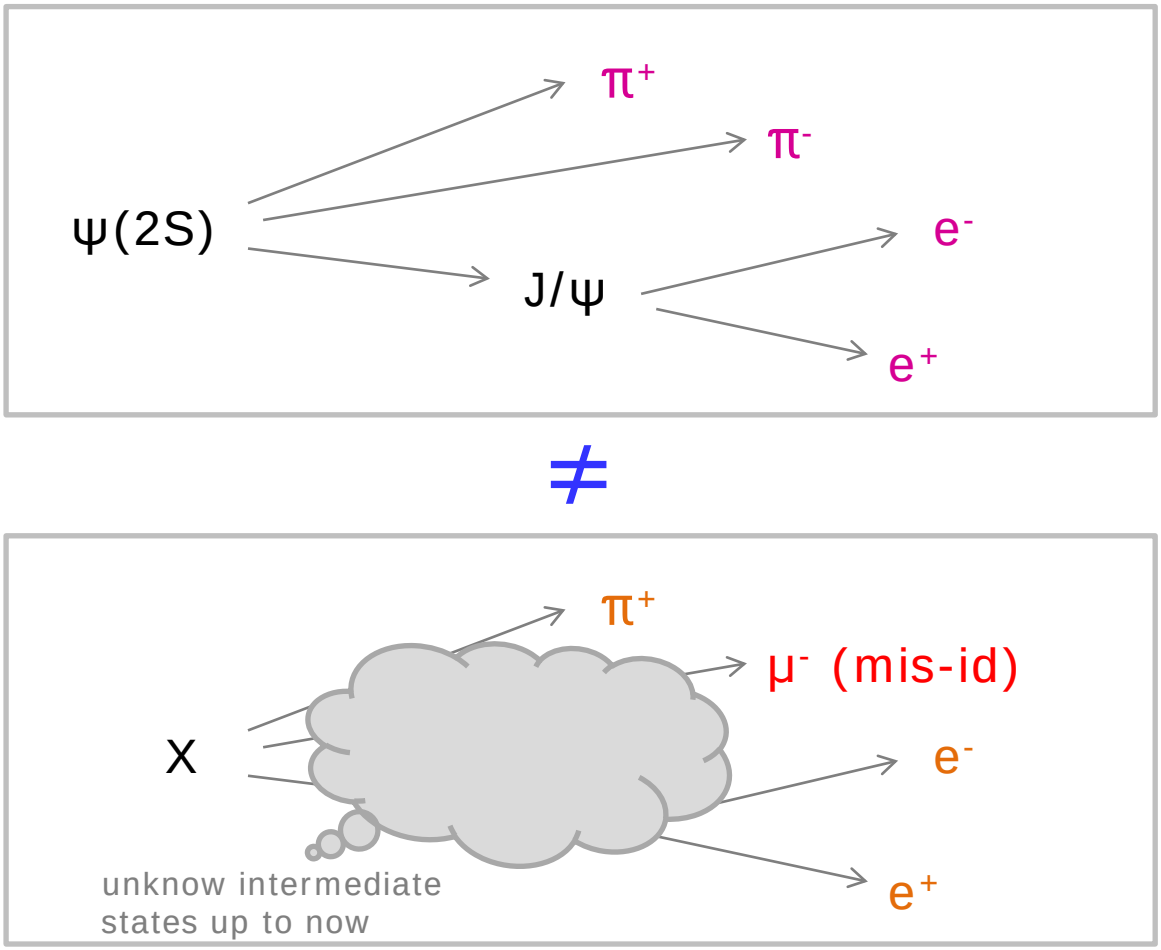


#	reco
1	trk+
2	trk+
3	trk+
4	trk-
5	trk-
6	trk-
7	trk-
8	neut
9	neut
10	neut
11	neut

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

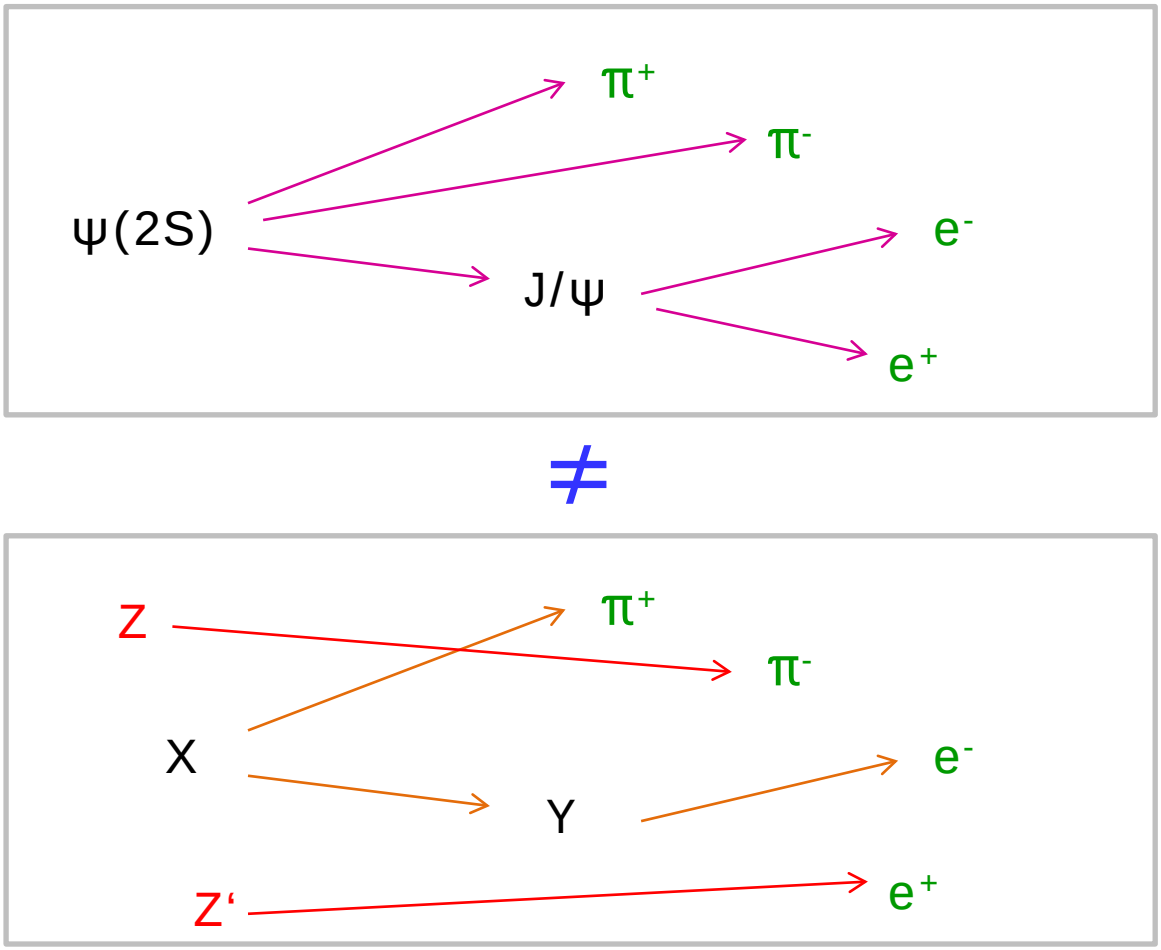
PndMcTruthMatch

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



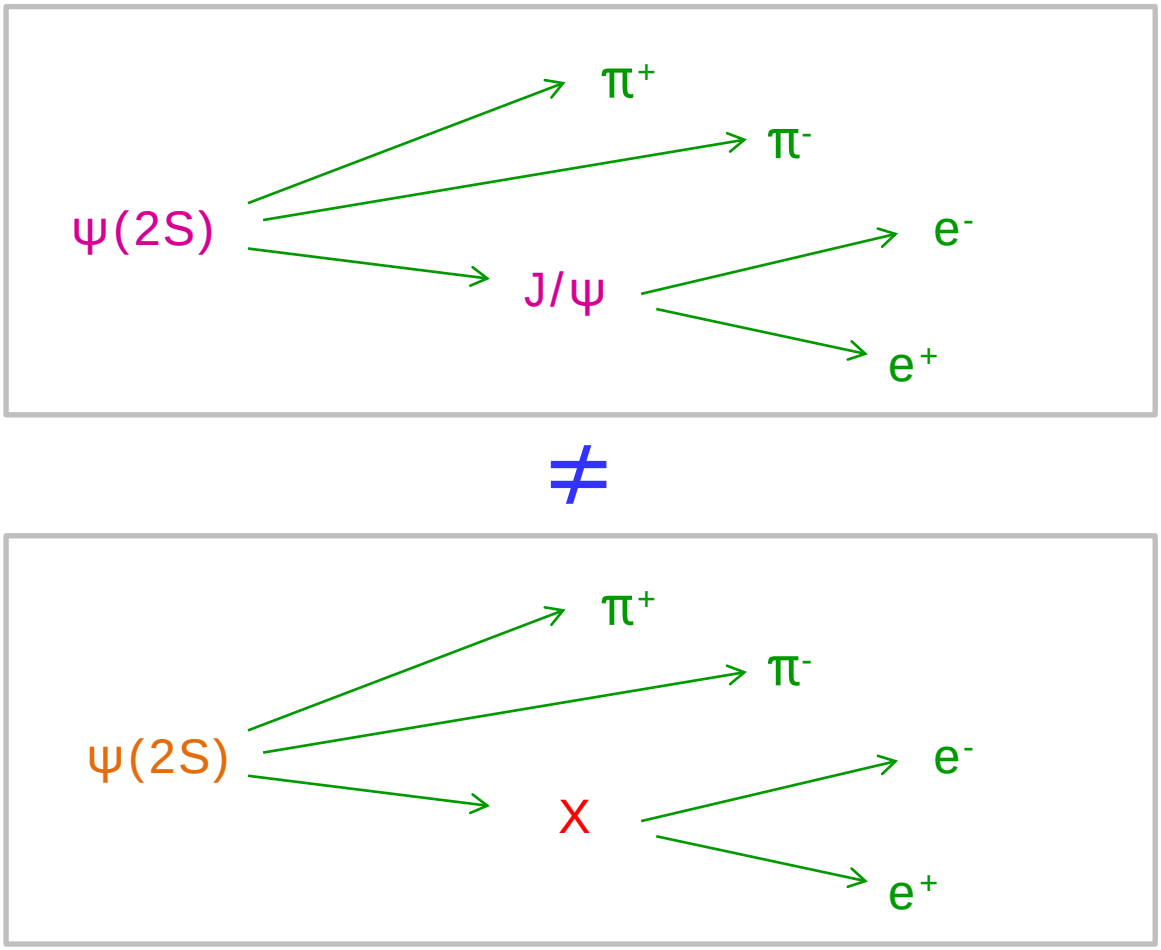
PndMcTruthMatch

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



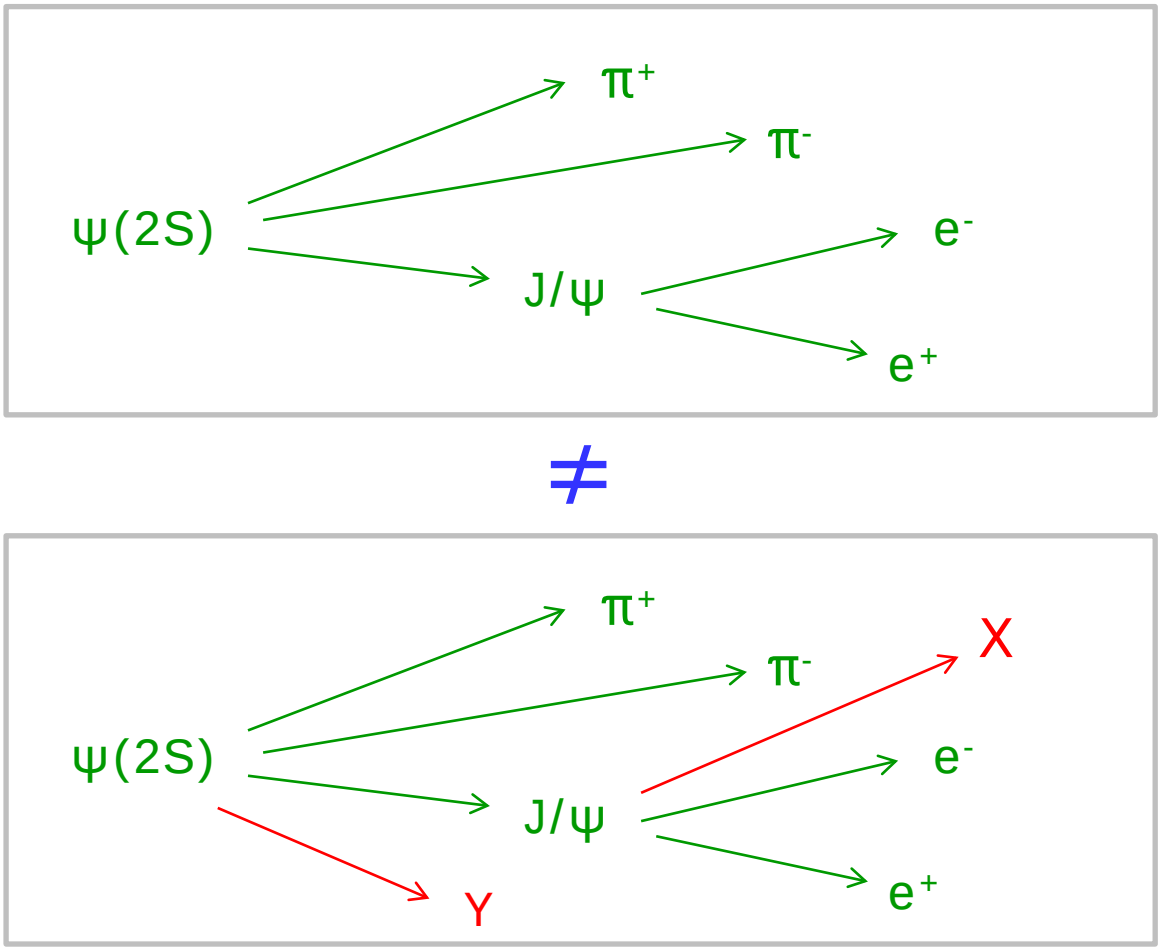
PndMcTruthMatch

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



PndMcTruthMatch

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

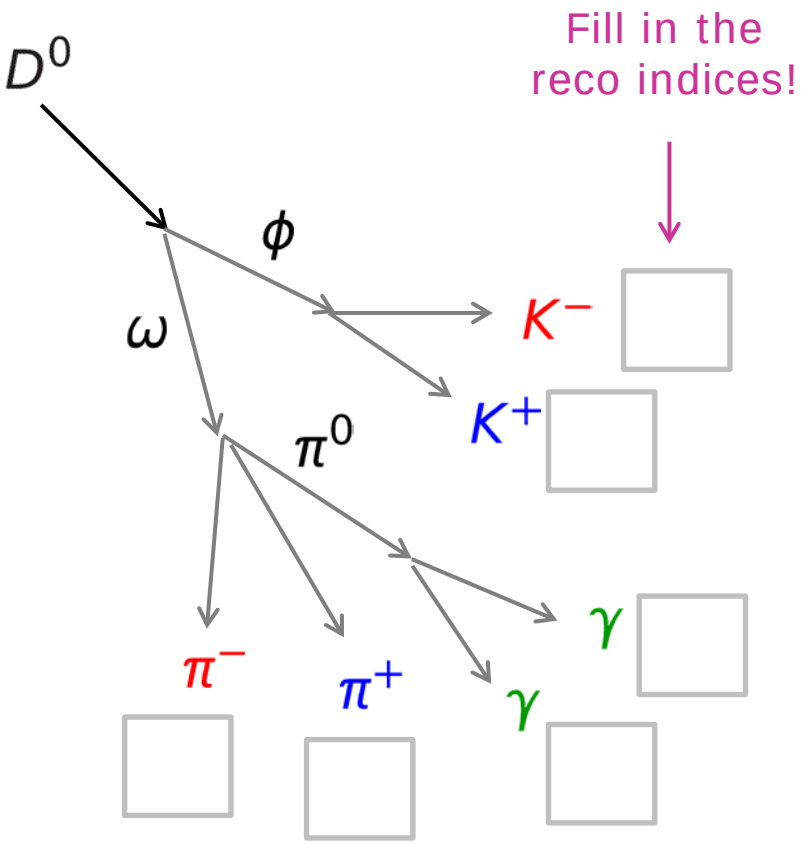


Usage of PndMcTruthMatch

- Only one PndMcTruthMatch object is needed
- For matching, composite candidates have to have type set
- List with Mc truth particles is needed as second parameter

```
PndMcTruthMatch mcm;

while (evr->GetEvent())
{
  evr->FillList(eplus, "ElectronPlus");
  ...
  evr->FillList(mct, "McTruth");     // mc truth list needed for match

  jpsi.Combine(eplus, eminus);
  mcm.SetType(jpsi, "J/psi");       // set type for J/psi (names like EvtGen)

  psi2s.Combine(jpsi, piplus, piminus);
  mcm.SetType(psi2s, "psi(2S)");             // set type for psi(2S)

  bool match = mcm.MctMatch(psi[0], mct);    // perform the match

  TCandidate mcPsi;
  if (match) mcPsi = mct[psi[0].GetMcIdx()]; // access truth of composites!
```

MC Truth Match Exercise



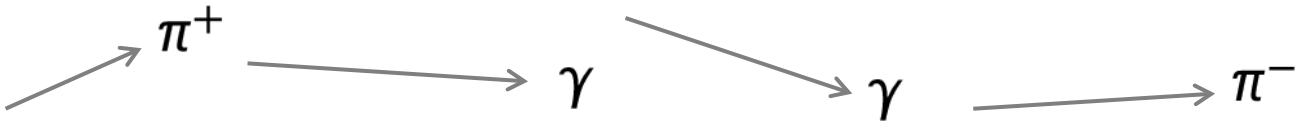
#	reco	mc
1	trk+	7
2	trk+	5
3	trk-	4
4	trk-	11
5	trk-	14
6	trk-	8
7	neut	10
8	neut	12
9	neut	9

#	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

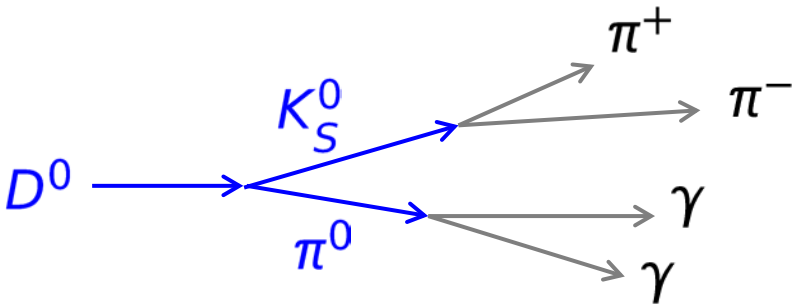
FITTING

Kinematic Fitting

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



- You believe what happend is this (hypothesis):

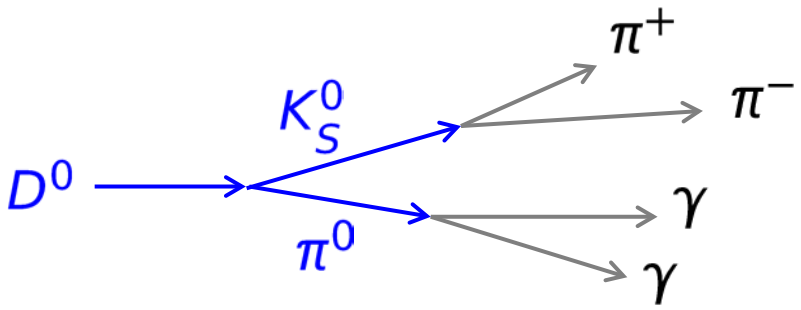


- Kinematic fitting
  - gives a measure for the assumption, that hypothesis is true
  - improves the precision/resolution for kinematic quantities, when hypothesis *actually is* true

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

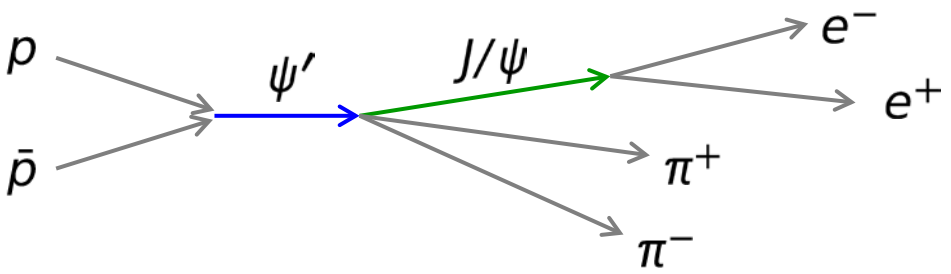
Kinematic Fitting: Constraints

Most common constraints are

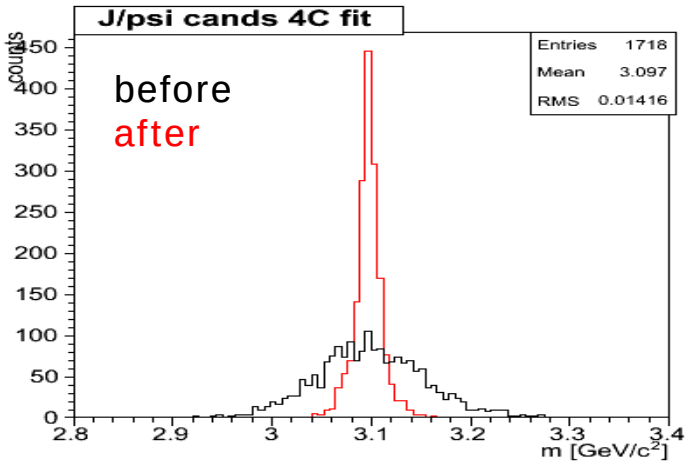
- Mass Constraint
  Composite particle has to have a certain invariant mass (only to be applied to quasi stable particles)
- Vertex Constraint
  Set of 4-vectors originate from common spatial point (which is determined during the fit!). *At least 2 charged tracks needed!*
- 4-vector-Constraint
  4-vector 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)

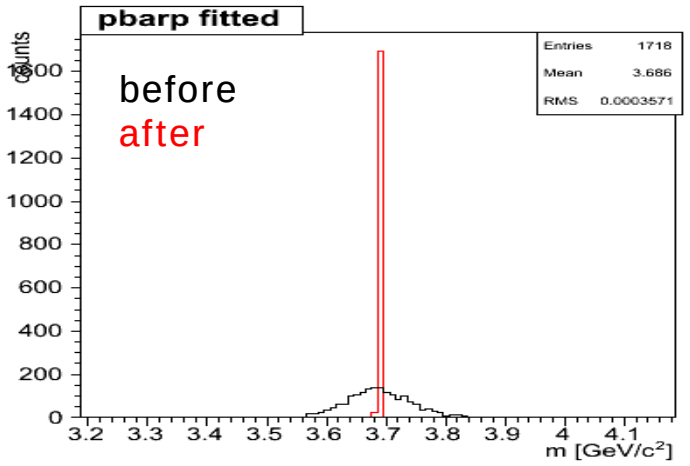
Fitting: 4 Constraint Fit

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



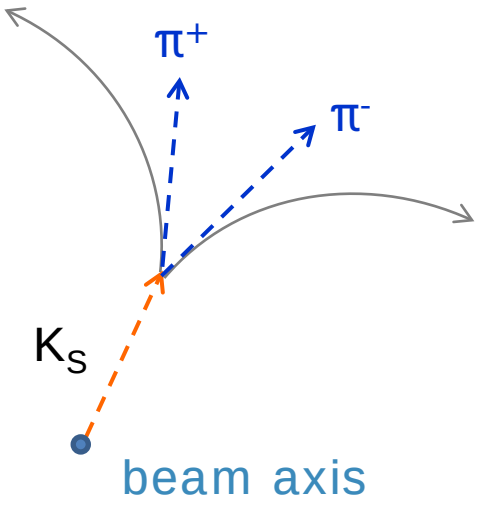
1. Fitted total system is not very meaningful (spiked mass), but intermediate resonances can be improved
2. Can do a cut on the fit probability to reject background

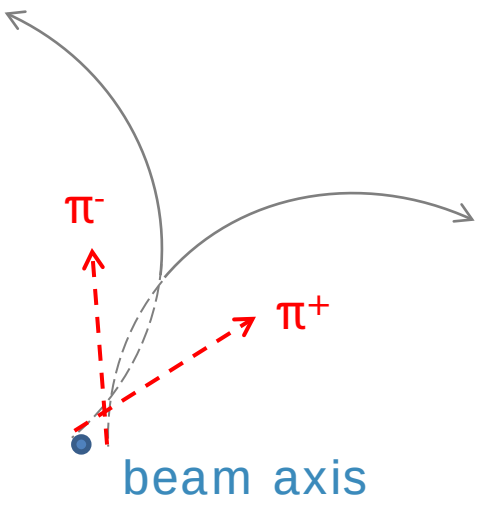




Vertex Fitting for Long-Living Particles

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





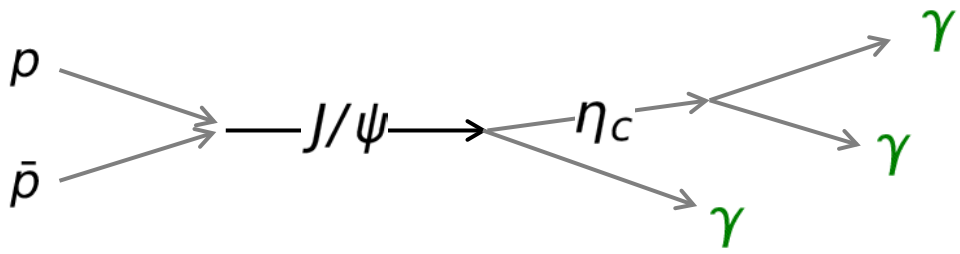
The 4-vector sum of daughters is only consistent with resonance 4-vector when evaluated at true points of origin!

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

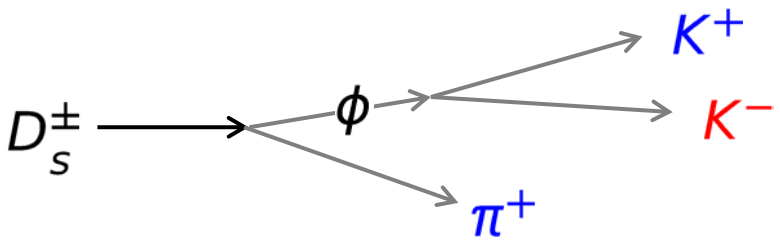
⇒ Good vertex information necessary for proper resonance reco!

Which constraints do you see?

1

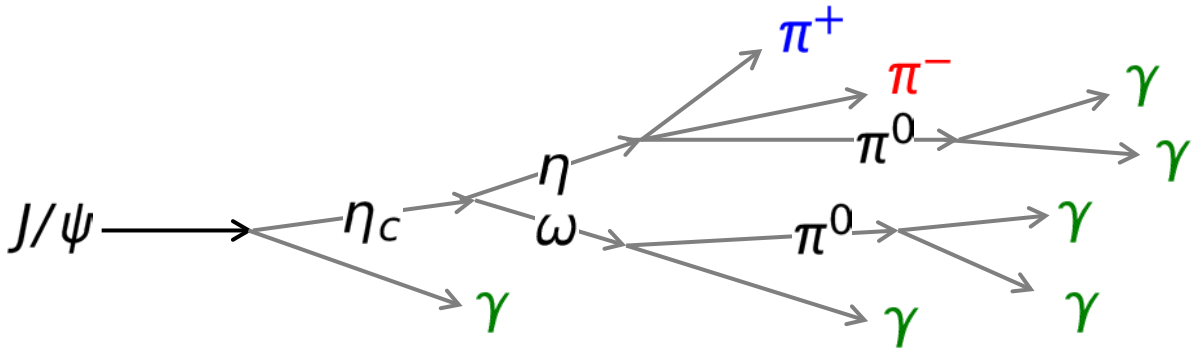


2



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

3



Fitting in PandaROOT: Vertex

- PandaROOT object: **PndKinVtxFitter**

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

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

   TCandidate *jfit
            = vtxfitter.FittedCand(jpsi[j]); // get fitted candidate

   TVector3 jVtx = jfit->Pos();              // and the vertex position
   double chi2_vtx = vtxfitter.GlobalChi2(); // 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 in PandaROOT: 4C

- PandaROOT object: **Pnd4CFitter**

```
// the lorentz vector of the initial psi(2S); important for the 4C-fit

TLorentzVector ini(0, 0, 6.231552, 7.240065);

  // ... in event loop ...
  for (j=0;j<psi2s.GetLength();++j)
  {
    Pnd4CFitter fitter(psi2s[j], ini);                    // instantiate the vertex fitter
    fitter.FitConserveMasses();                           // perform fit

    TCandidate *jfit = fitter.FittedCand(*(psi2s[j].Daughter(0))); // get fitted J/psi candidate

    TCandidate *epfit
      = fitter.FittedCand(*(psi2s[j].Daughter(0)->Daughter(0)));   // get the fitted positron

    TCandidate *emfit
      = fitter.FittedCand(*(psi2s[j].Daughter(0)->Daughter(1)));   // get the fitted electron

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

    TLorentzVector tlvepf = epfit->P4();
    TLorentzVector tlvemf = emfit->P4();
    Double_t j_fitmass = (tlvepf+tlvemf).M();          // compute the fitted J/psi mass
                                                       // (by adding the e+ and e- 4-vectors)
    hjpsim_4cf->Fill( (tlvepf+tlvemf).M() );           // fill histogram
}
```

UNCERTAINTIES

Uncertainties: Figure of merit

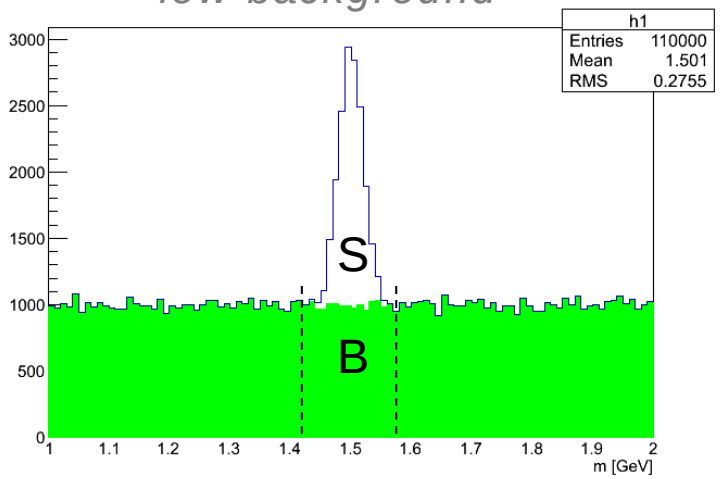
- Figure of Merit for analysis: Minimization of the relative error!
  (not, as you might think, the measured value itself...)

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

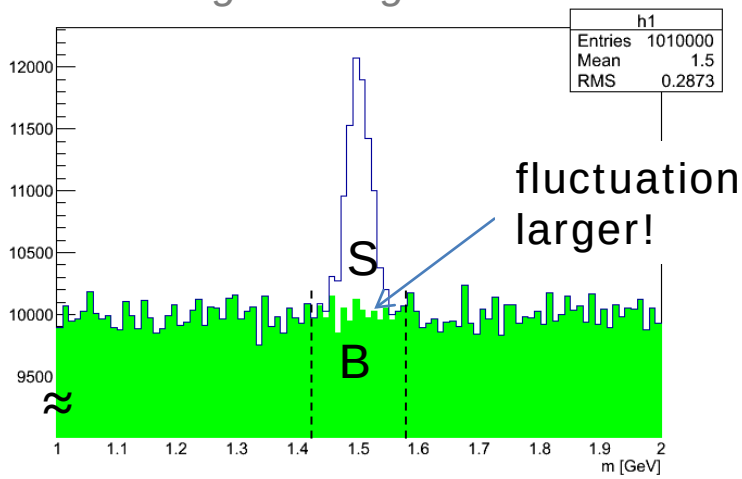
- Explanation: Statistical fluctuation of S is $\sqrt{S+B}$ *(see below)*

→ 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)*

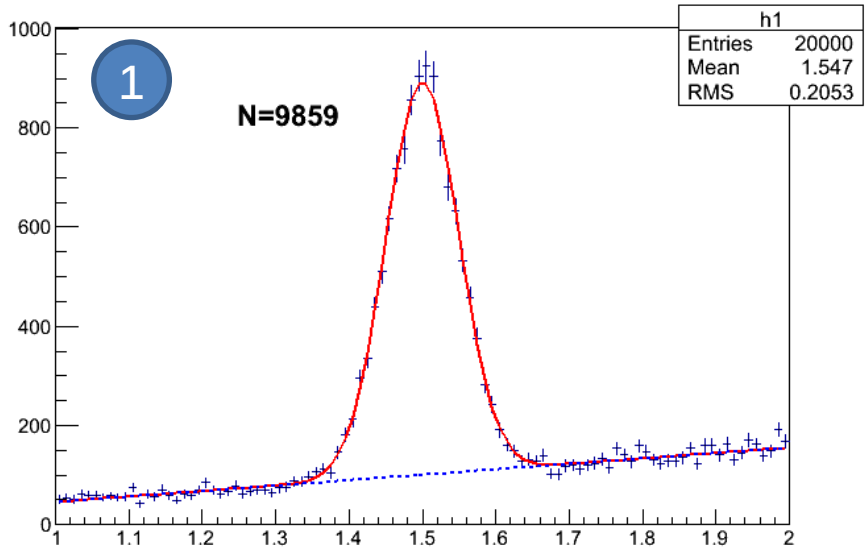
Systematics Example: Fit model uncertainty

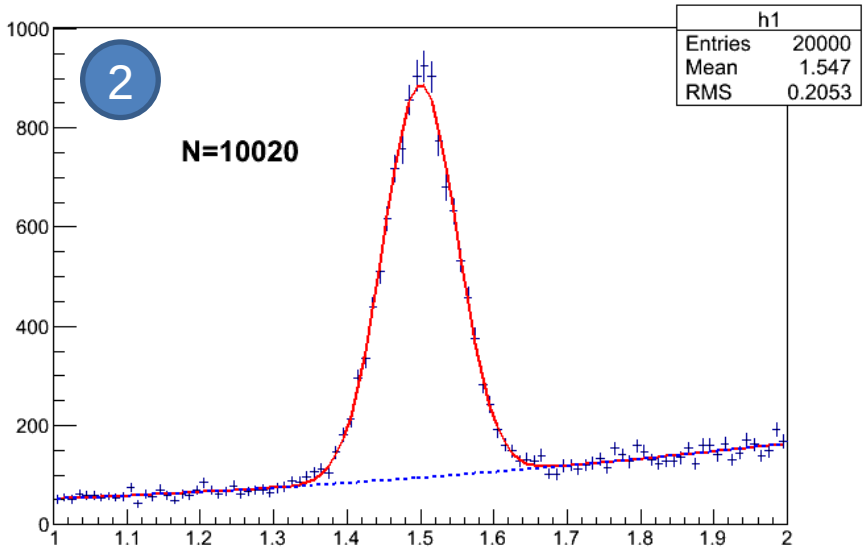
Task: Determine number of signals ontop of background

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

- Estimate for systematic error (fit model) could be

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





And now ... *try it yourself!*

- Take a look at the Wiki Tutorial page
  http://panda-wiki.gsi.de/cgi-bin/view/Computing/PandaRootTutFeb2012

- Try to perform simulation and analysis for one of the channels
  (Reconstruct signal, determine efficiency & resolution)

1. $\bar{p}p \to J/\psi \to \eta_c \gamma \to K_S^0(\to \pi^+\pi^-)\, K^\pm \pi^\mp \gamma$
2. $\bar{p}p \to \Lambda^0 \bar{\Lambda}^0 \to p\pi^-\, \bar{p}\pi^+$ ☆
3. $\bar{p}p \to D_s^\pm D_s^\mp \to \phi(\to K^+K^-)\pi^\pm X$ ☆ *(inclusive)*
4. $\bar{p}p \to \omega\omega \to \pi^+\pi^-\pi^0(\to \gamma\gamma)\, \pi^+\pi^-\pi^0(\to \gamma\gamma)$ ☆

☆ *Production modes, require choice of a $\bar{p}$ momentum*