Heavy Flavor Reconstruction Tools for ECCE

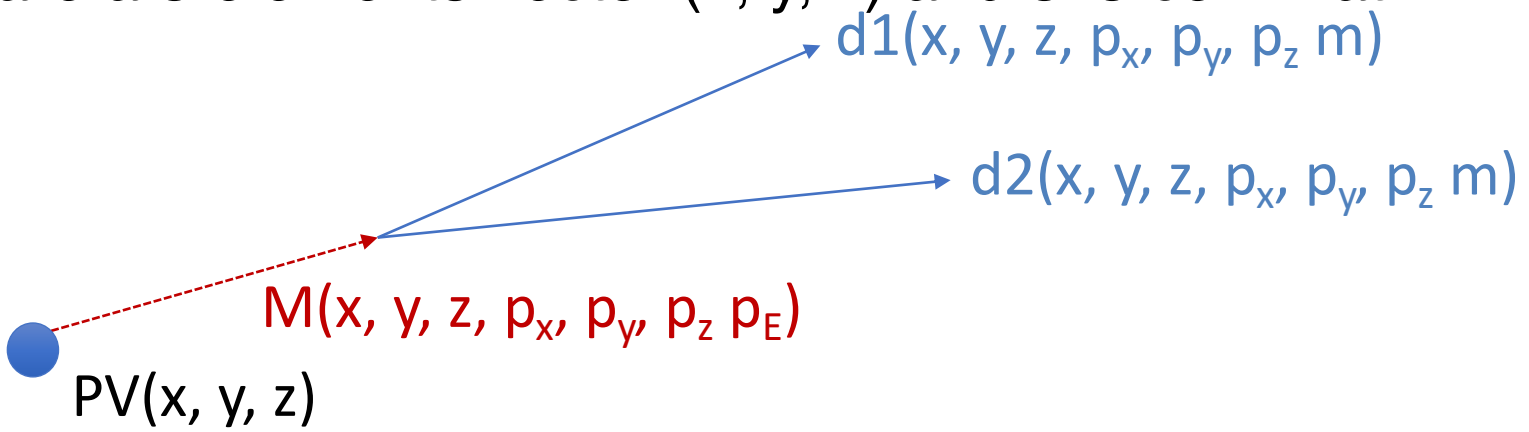
Cameron Dean,
Los Alamos National Lab

05/25/2021

Heavy Flavor and Jets Working Group Meeting

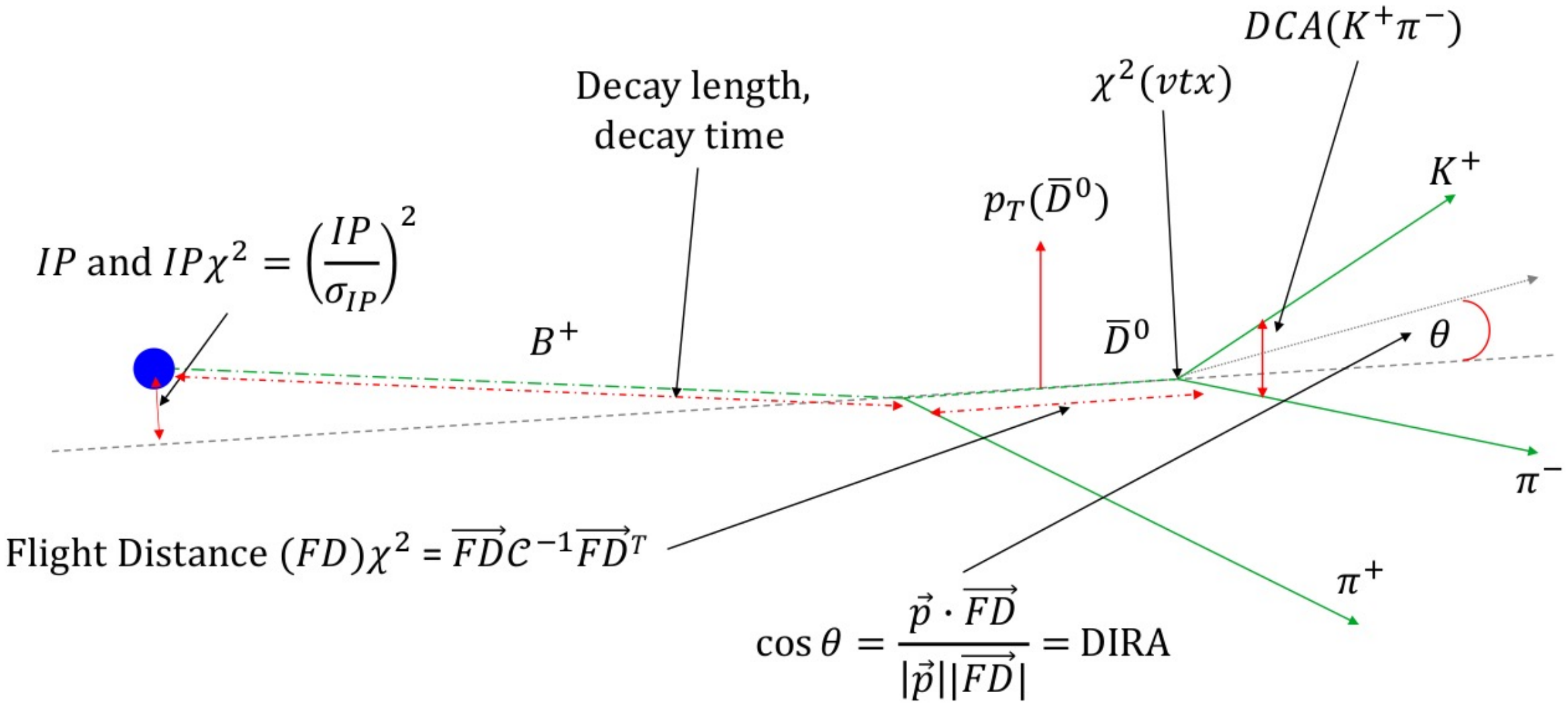
What is KFParticle

- A reconstruction package for heavy flavor decays
- Based around a Kalman Filter (the KF in KFParticle)
- Implemented at ALICE, CBM, STAR and sPHENIX
- Benefits from information on uncertainties (covariance matrices)
- Particles are a 7 element vector (x, y, z, p_x, p_y, p_z p_E) and 7x7 cov. Matrix
- p_E can be left unknown and then calculated by conservation of 4-mom.
- Vertices are a 3 elements vector (x, y, z) and 3x3 cov. matrix



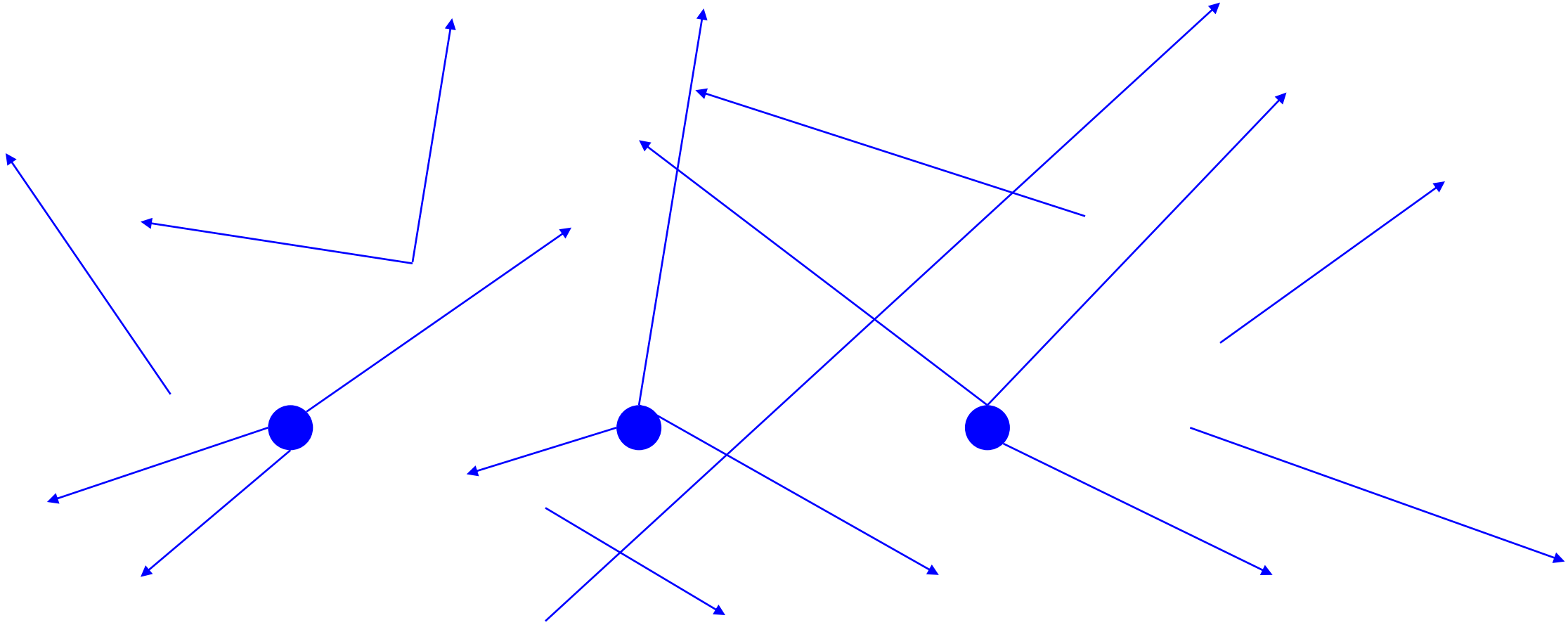
[CBM-SOFT-note-2006-001]
[CBM-SOFT-note-2006-002]
[CBM-SOFT-note-2007-003]
[GSI Talk. Nov 25th, 2008]

What is KFParticle



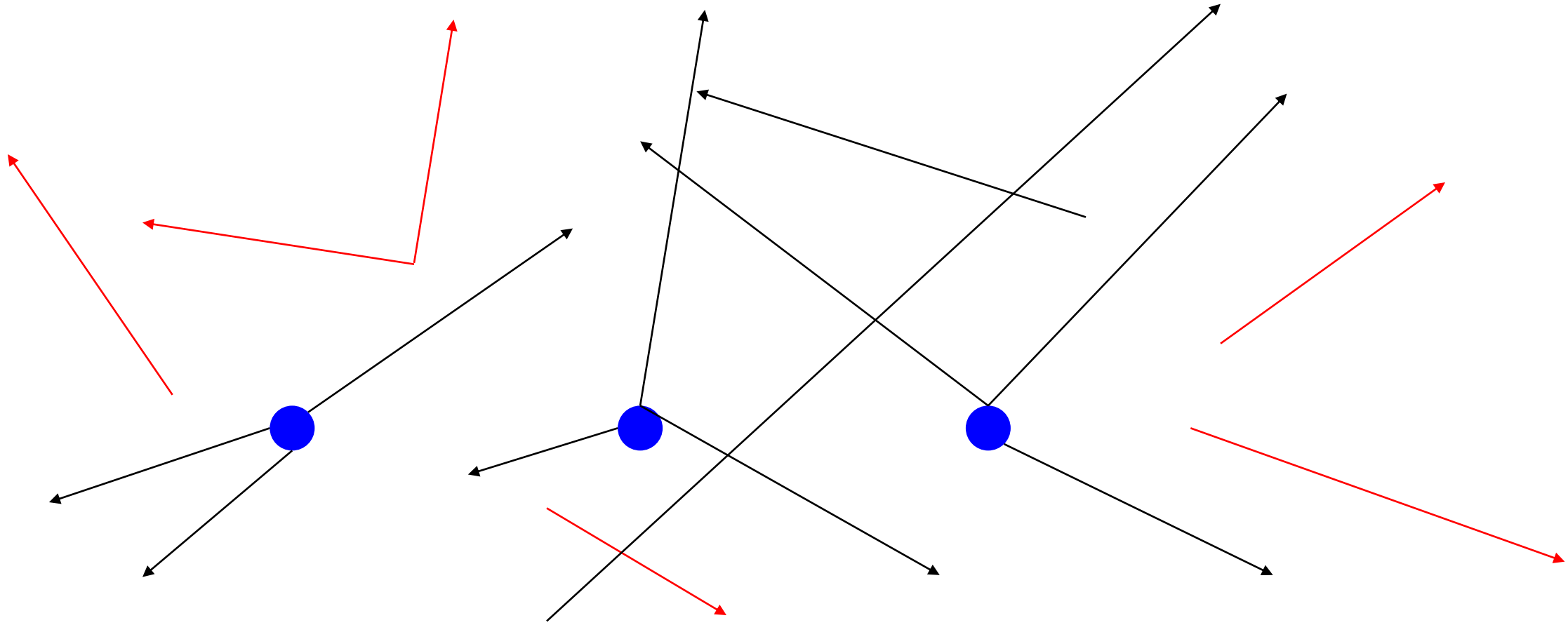
Step 1

- Unpack vertices and tracks



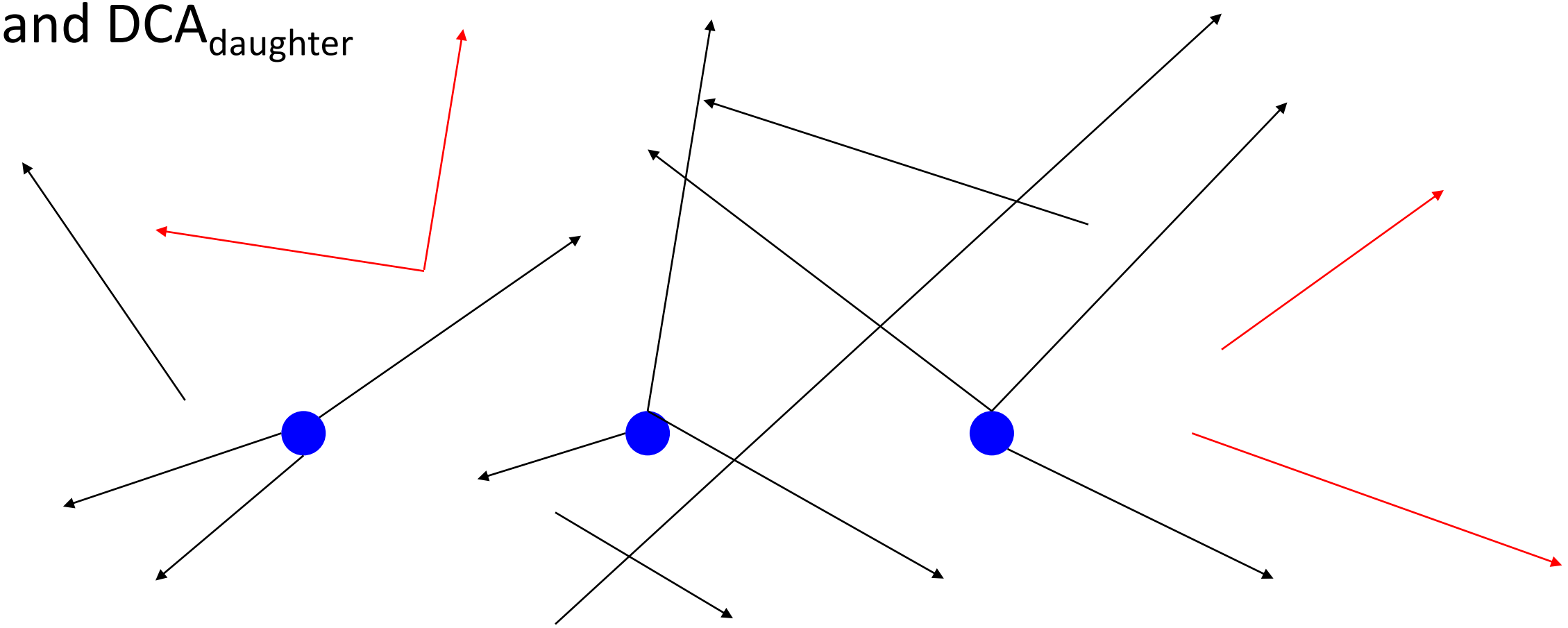
Step 2

- Select good tracks based on p_T, p_T χ^2, track χ^2 and DCA_{PV} χ^2



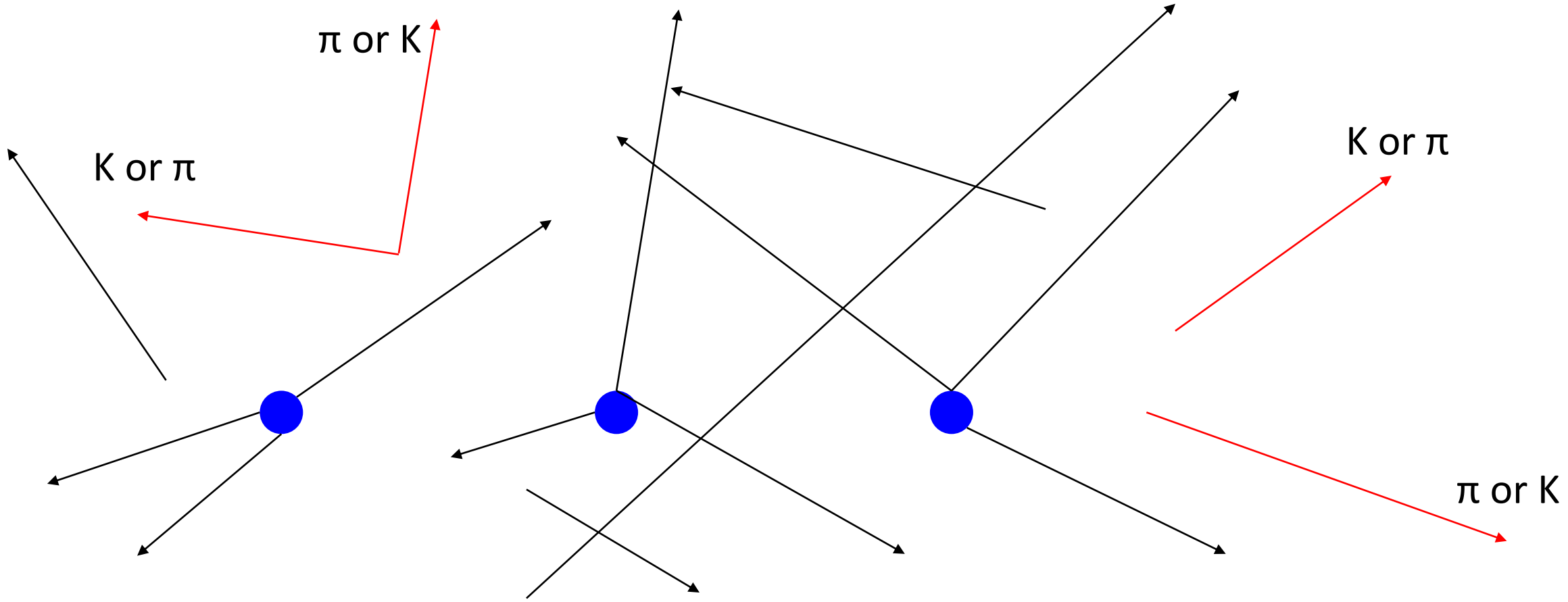
Step 3

- Select good vertices based on number of required tracks, vertex χ^2 and $DCA_{daughter}$



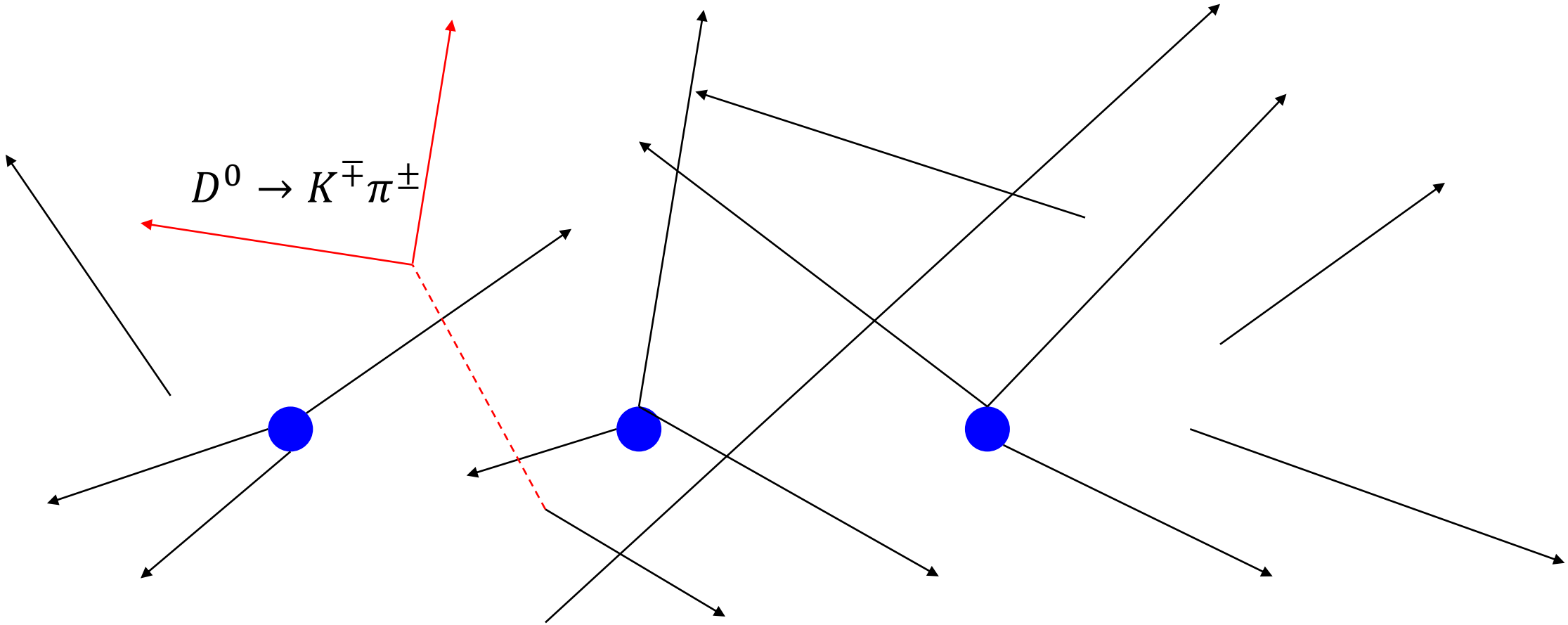
Step 4

- Assign PID based on unique combinations



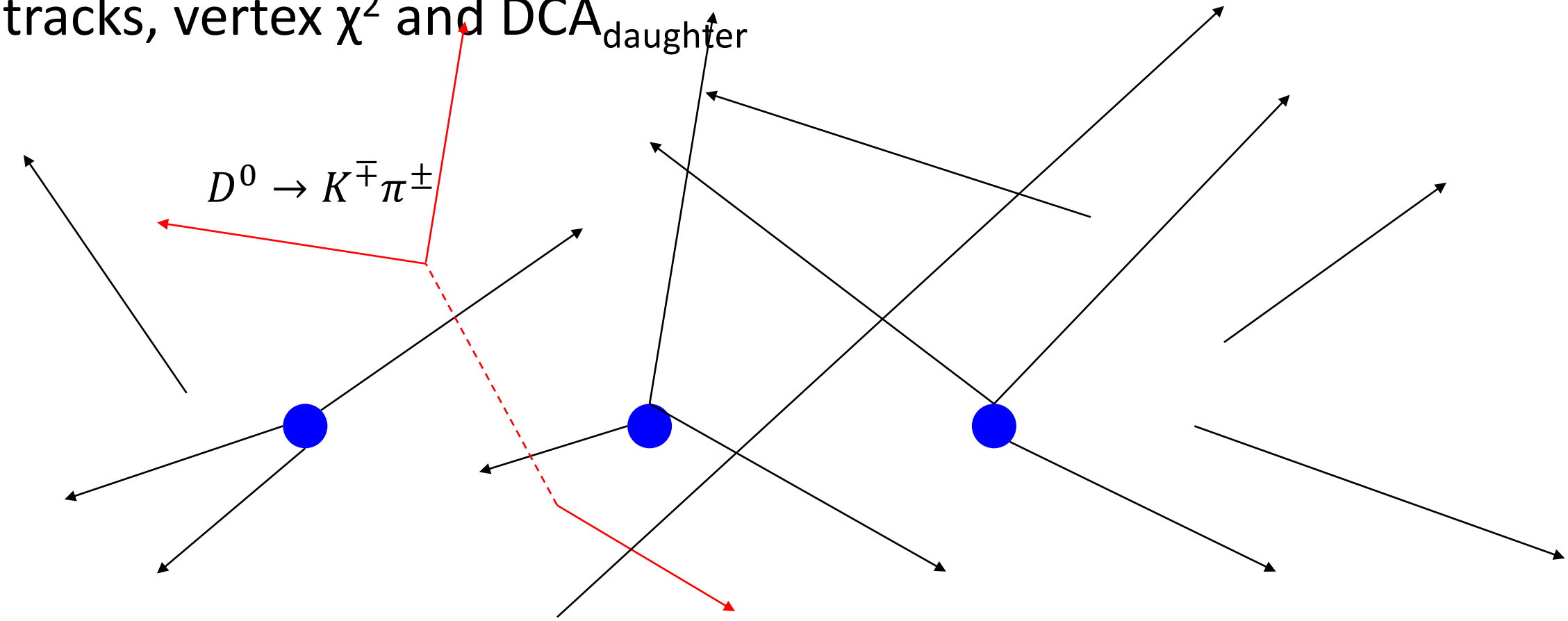
Step 4a (optional)

- Reconstruct intermediate decays based on selection and PID



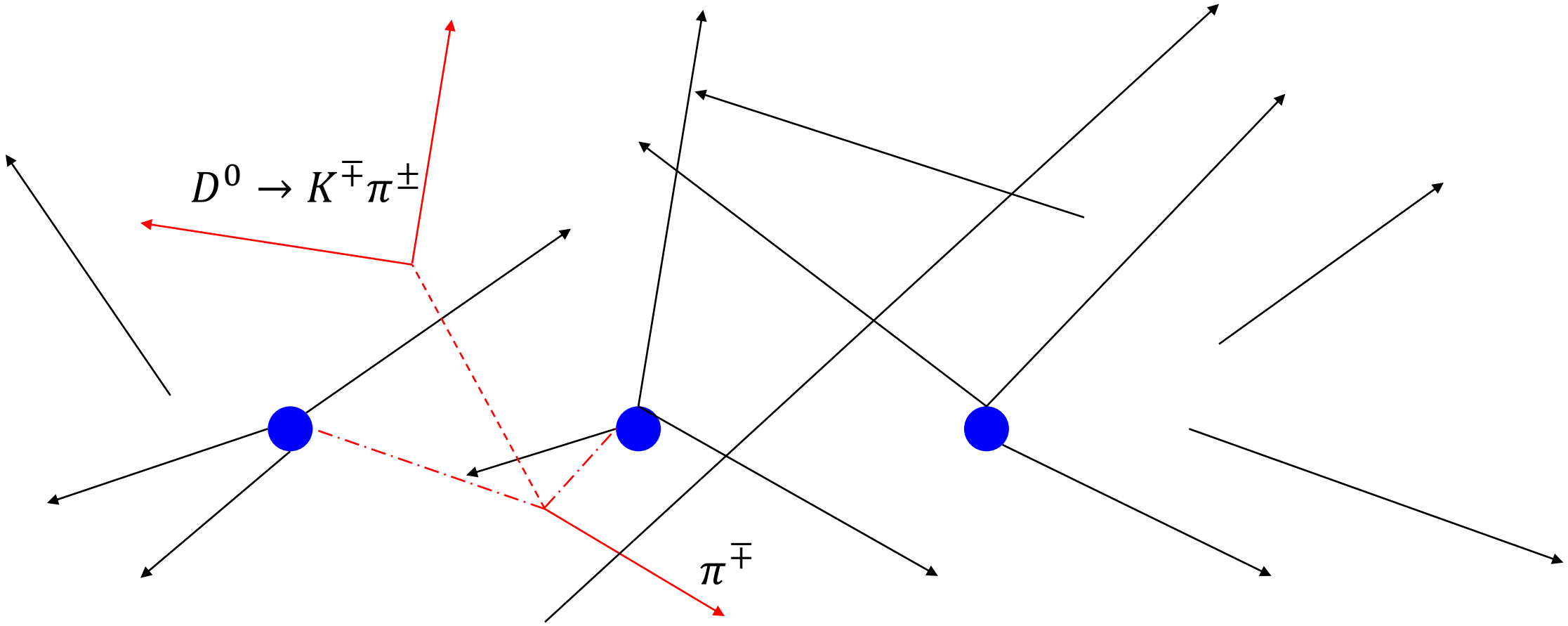
Step 4b (optional)

- Append extra tracks to intermediates based on number of extra tracks, vertex χ^2 and $DCA_{daughter}$



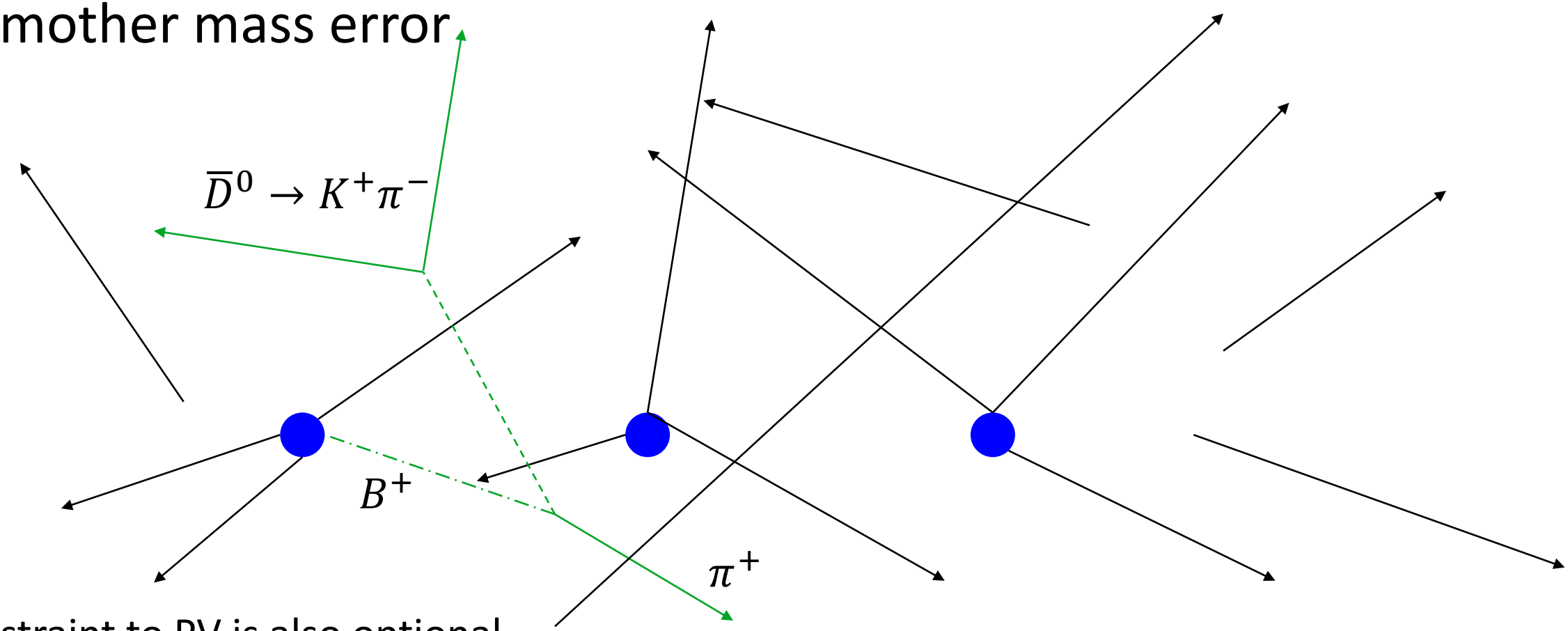
Step 5

- Reconstruct mother candidates based on selection and PID



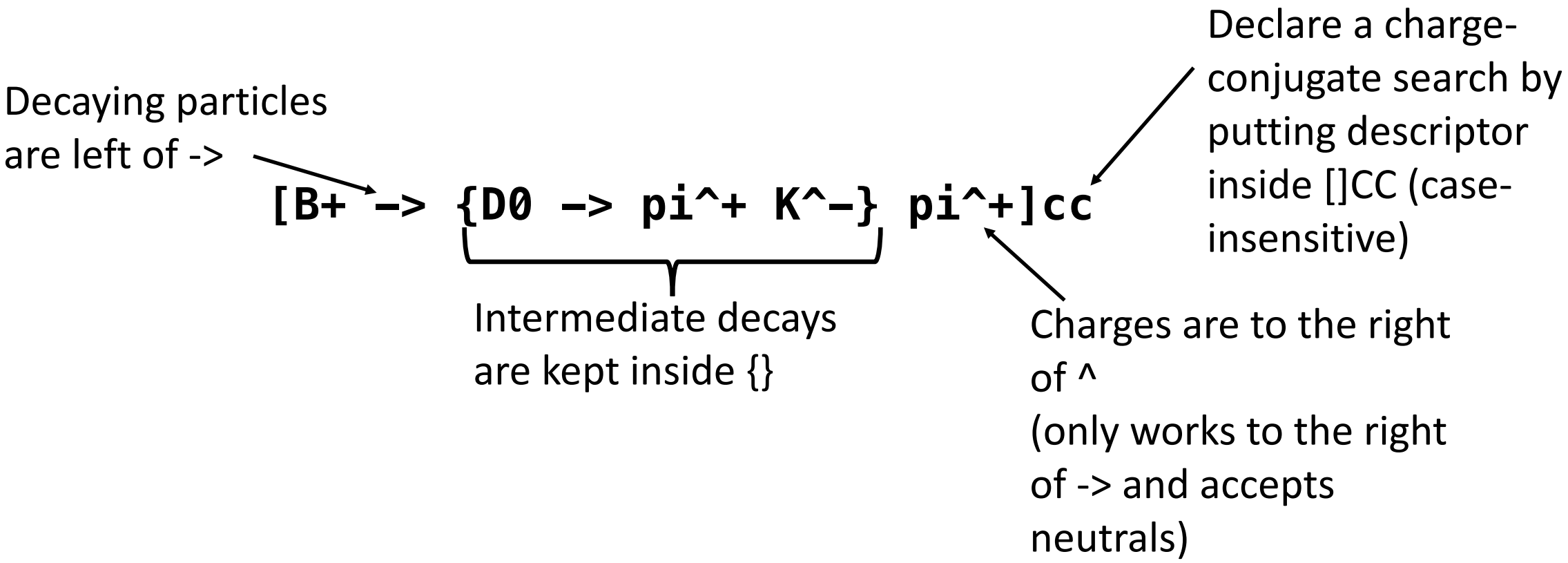
Step 6

- If end vertex has more than 1 candidate, select based on lowest mother mass error



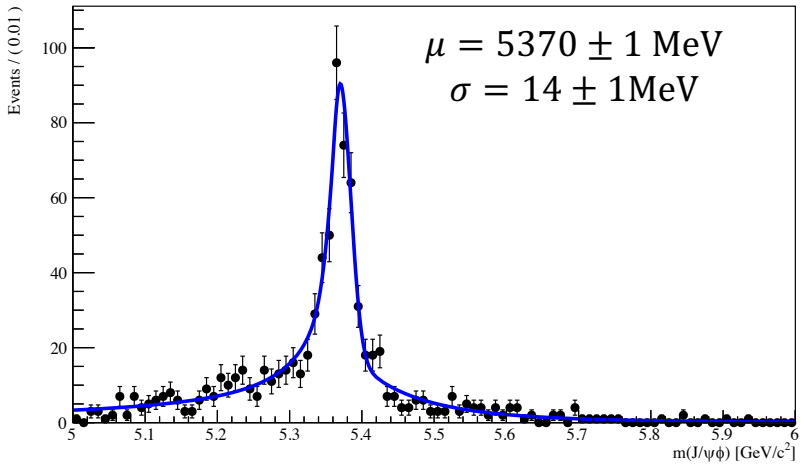
*Constraint to PV is also optional

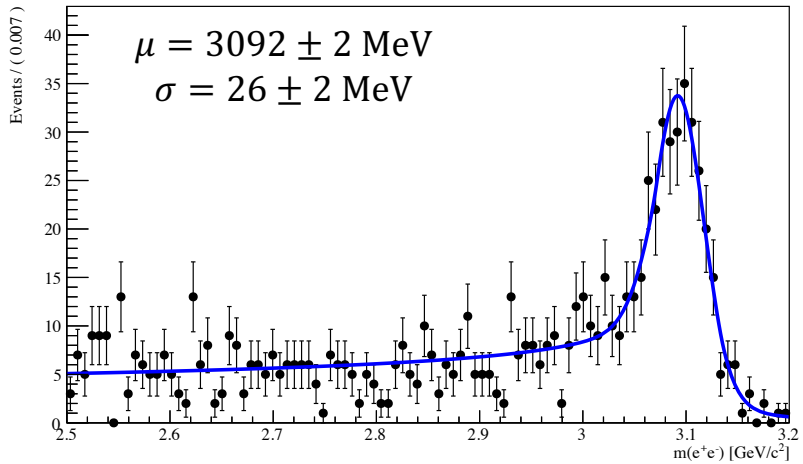
Setting the decay description

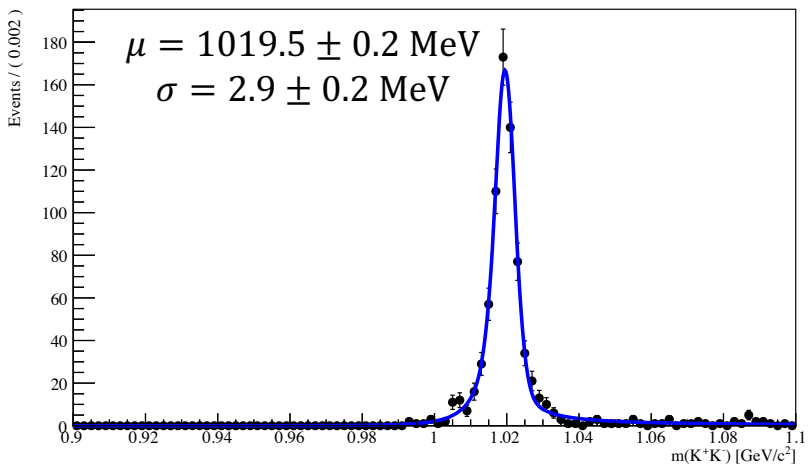


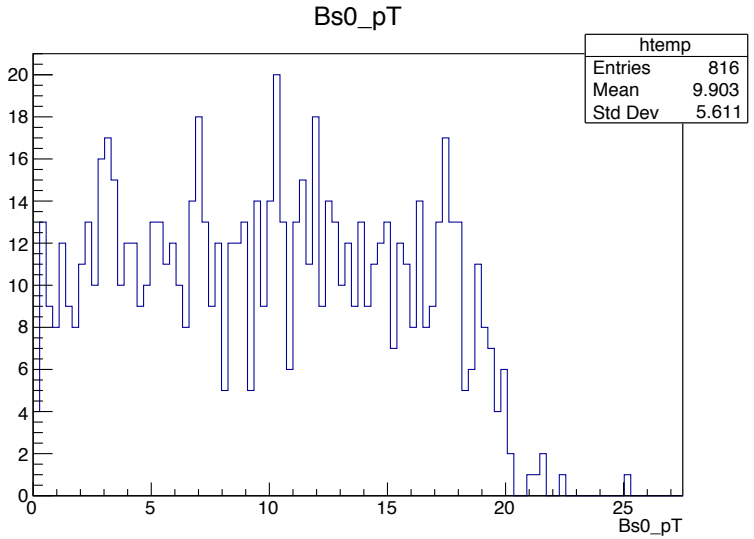
- As long as the particle is declared in KFParticle_particleList.cc, it can be used here

$B_s^0 \rightarrow J/\psi\phi$ example

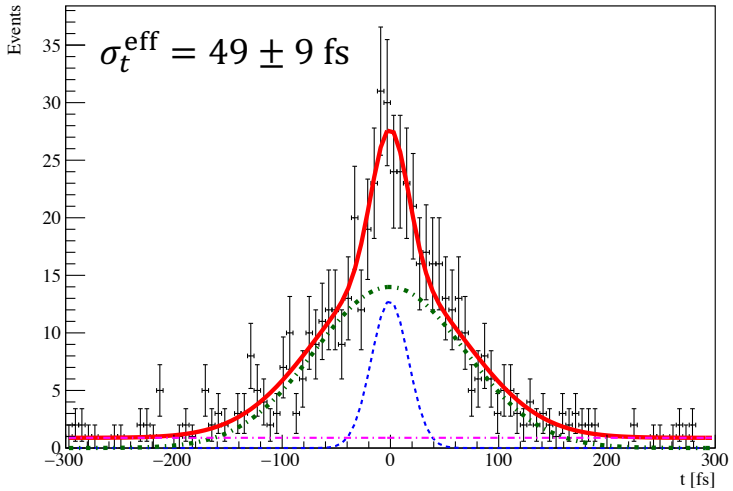








J/ψ decay time



Large amount of variables available to analysts:
pT, η, θ, ϕ of all particles
Reconstruction χ^2 and DCA
Event multiplicity and number of vertices
Truth matching
Detector information
and many more

jT Calculation

- Several variables are available by default, suchasthe track pT relative to parent (jT)
- Method taken from T. Snellman's thesis

4.2 Definition of j_T

The reconstructed jet axis is used for j_T reference. Any charged track within a fixed cone with radius R is taken as a jet constituent, as opposed to using the constituent list provided by the jet algorithm. Anti-k_T produces jets that are very circular in shape. Thus this doesn't change the constituent list considerably. Calorimeter clusters are used only in jet reconstruction.

The jet fragmentation transverse momentum, $\vec{j}_T$, is defined as the component of the constituent track momentum, $\vec{p}_{track}$, transverse to the jet momentum, $\vec{p}_{jet}$. It represents the transverse kick with respect to the initial hard parton momentum that a fragmenting particle receives during the fragmentation process, which is a measure of the momentum spread of the jet fragments.

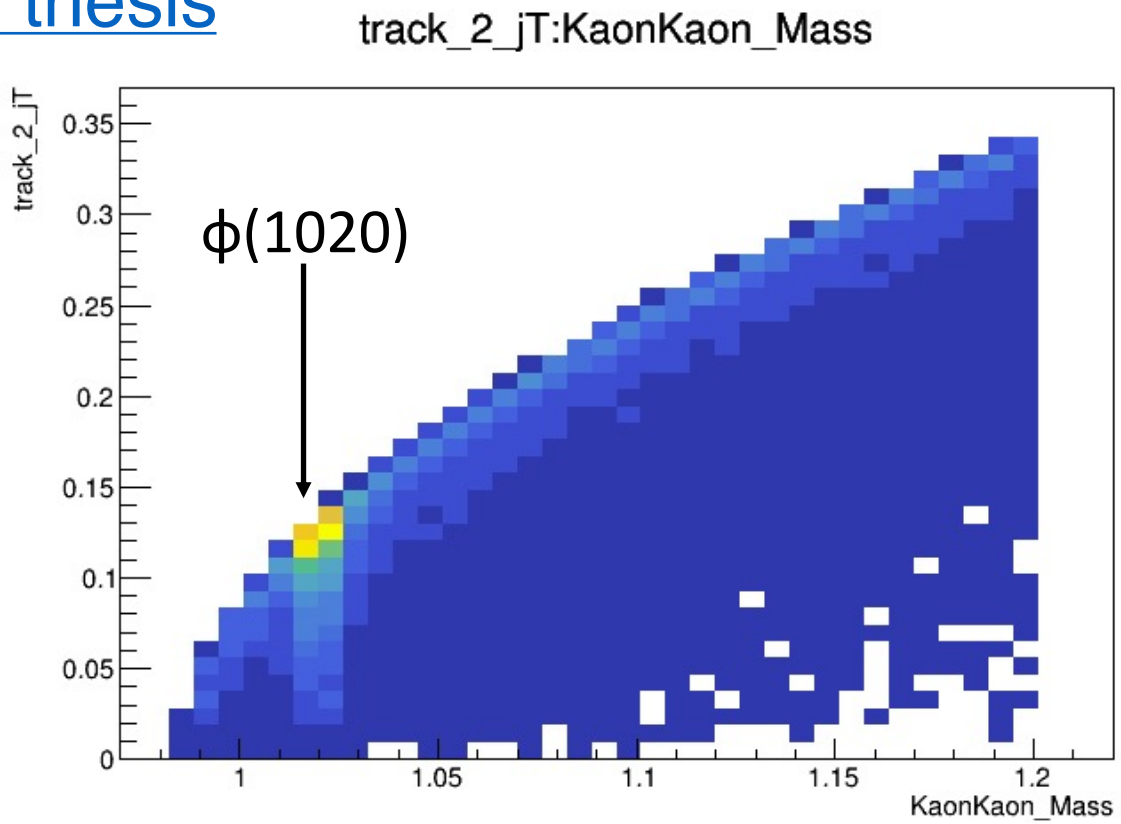
The resulting $\vec{j}_T$ is illustrated in Fig. 4.2. The length of the $\vec{j}_T$ vector depends on the jet and track momentum vectors $\vec{p}_{jet}$ and $\vec{p}_{track}$

$$j_T = \frac{|\vec{p}_{jet} \times \vec{p}_{track}|}{|\vec{p}_{jet}|}. \quad (4.4)$$

Resulting j_T distributions are shown as

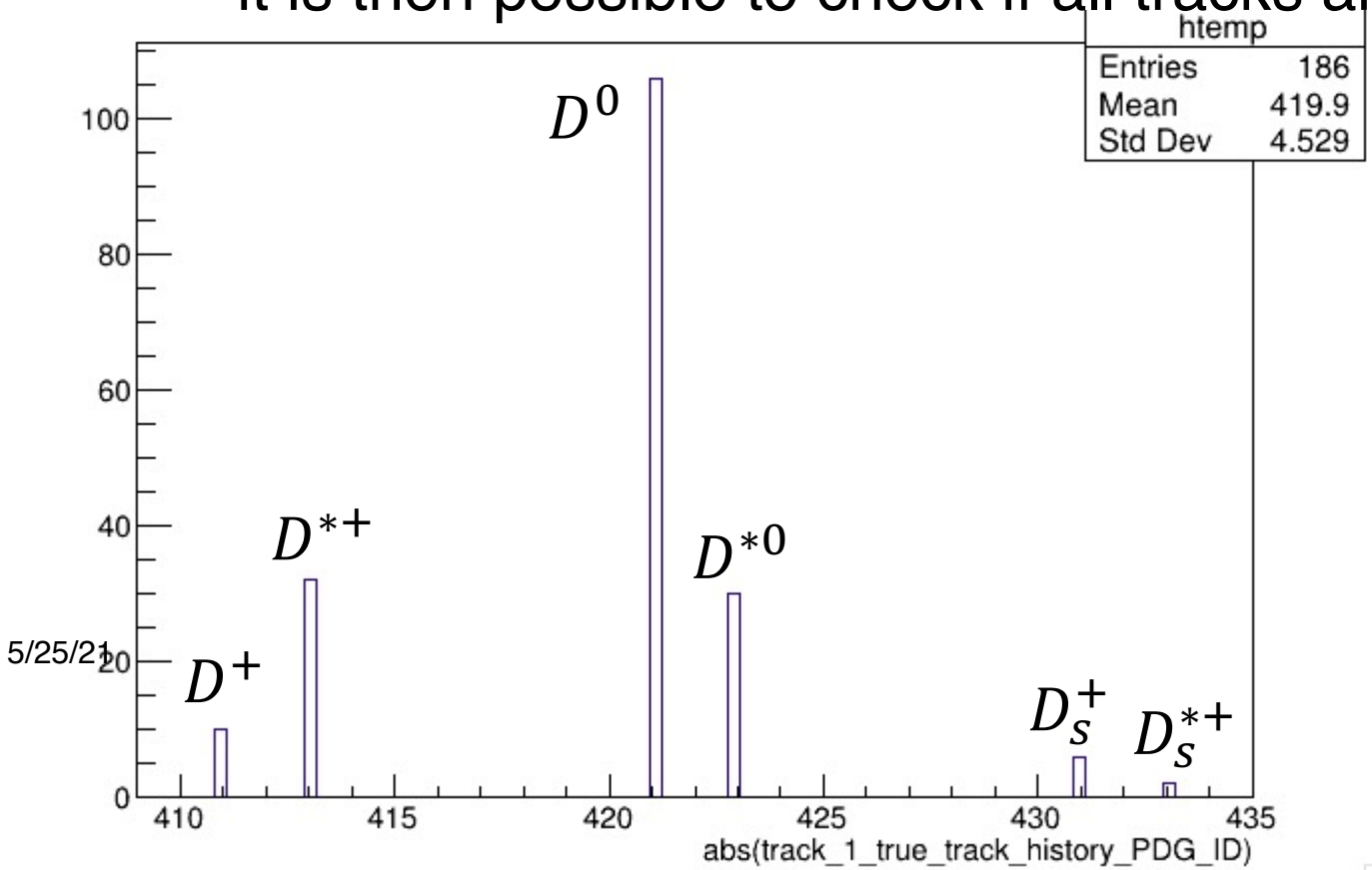
$$\frac{1}{j_T}\frac{dN}{dj_T} \quad (4.5)$$

distributions. The logic behind this is that j_T is inherently a two-dimensional



Truth MC history

- Truth matching can be enabled for any reconstruction
  - Implemented separately for each reconstructed track, it is then possible to check if all tracks are from same parent



N.B. The information here is a vector so we can see potential parents to the D^0. D^+ and D_s^+ are probably partially reconstructed

Using KFParticle

- KFParticle is kept in coresoftware
  - This means everyone has access to it as a standard tool
- You can create as many KFParticle instances as you like per analysis
  - Each KFParticle object can be named to avoid confusion in Fun4All
- You can set the class verbosity to 1 or greater to see reconstruction output during processing
- Reconstructed particles can be written to an nTuple or back onto the node tree and reused
  - You can specify the new node name AND the input track map
  - This means you can chain KFParticle objects ad infinitum

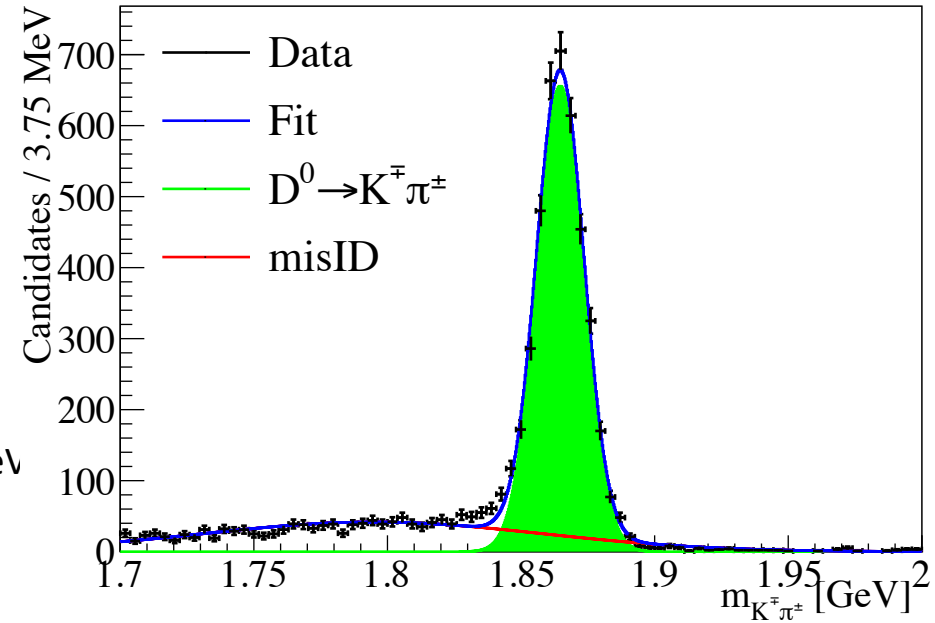
D^0 example reconstruction

----------KFParticle candidate information----------
Mother information:
Track ID: -1
PDG ID: 421, charge: 0, mass: 1.86592 GeV
(px,py,pz) = (0.840637 +/- 0.0122078, -3.90266 +/- 0.0347824, 0.000426173 +/- 0.00238134) GeV
(x,y,z) = (0.00286134 +/- 0.00206705, -0.0111923 +/- 0.00426421, 4.81138 +/- 0.00161587) cm

Final track information:
Track ID: 0
PDG ID: 211, charge: 1, mass: 0.13957 GeV
(px,py,pz) = (-0.391714 +/- 0.00196463, -0.243336 +/- 0.00125698, -0.168335 +/- 0.000992688) GeV
(x,y,z) = (0.00229592 +/- 0.00227791, -0.0115516 +/- 0.00366684, 4.81108 +/- 0.00459086) cm

Track ID: 1
PDG ID: -321, charge: -1, mass: 0.493666 GeV
(px,py,pz) = (1.23236 +/- 0.0124612, -3.65934 +/- 0.0353158, 0.168762 +/- 0.00227103) GeV
(x,y,z) = (0.00149067 +/- 0.00163804, -0.00713227 +/- 0.000551643, 4.8112 +/- 0.00172057) cm

The number of primary vertices is: 1
The number of tracks in the event is: 2



- I used Fun4All's $D^0 \to K^-\pi^+$ particle gun
- 6000 event generations
- I will give a demo now
- Example code is in backup

$\mu[\text{PDG}] = 1864.83 \pm 0.05$ MeV
$\mu = 1864.7 \pm 0.2$ MeV
$\sigma = 8.8 \pm 0.1$ MeV

Summary

- KFParticle is a heavy flavor reconstruction package, originally for CBM
- It was adapted to work in Fun4All
- The package is a part of our daily build, no need to compile anything
- Users only need to specify a decay and give selection options, package does all the reco and writing for you
- There is a KFParticle manual that details setup, input options and nTuple branches
- There are also example reconstructions in the sPHENIX repo
  - These files are dynamic as the package matures
- Please get in contact with me if you encounter any issues

Backup

$D^0 \rightarrow K^-\pi^+$ example (from G4_User.C)

```
#include <kfparticle_sphenix/KFParticle_sPHENIX.h>
R__LOAD_LIBRARY(libkfparticle_sphenix.so)

void UserAnalysisInit()
{
  Fun4AllServer* se = Fun4AllServer::instance();

  KFParticle_sPHENIX *kfparticle = new KFParticle_sPHENIX("D0reco");
  kfparticle->Verbosity(Enable::USER_VERBOSITY);

  kfparticle->setDecayDescriptor("[D0 -> K^- pi^+]cc");

  kfparticle->doTruthMatching(true);
  kfparticle->setTrackMapNodeName("TrackMap");
  kfparticle->useFakePrimaryVertex(true);
  kfparticle->constrainToPrimaryVertex(false);
  kfparticle->allowZeroMassTracks(true);
  kfparticle->setOutputName("myD0Reconstruction.root");
```

Libraries and headers

Give an optional name to the object

Write the decay descriptor

sPHENIX and ECCE have different default track map names!

ECCE PV map may not be filled, disable the readback here

Get reco. info. wrt. the PV

Specify an optional output nTuple name

$D^0 \rightarrow K^-\pi^+$ example (from G4_User.C)

```
  //Track parameters
  kfparticle->setMinimumTrackPT(0.0);
  kfparticle->setMinimumTrackIPchi2(0);
  kfparticle->setMinimumTrackIP(0.00);
  kfparticle->setMaximumTrackchi2nDOF(4000);

  //Vertex parameters
  kfparticle->setMaximumVertexchi2nDOF(4000);
  kfparticle->setMaximumDaughterDCA(5);

  //Parent parameters
  kfparticle->setMotherPT(0.0);
  kfparticle->setMinimumMass(1.7);
  kfparticle->setMaximumMass(2.0);

  se->registerSubsystem(kfparticle);

  return;
}
```

- Note, our default units are GeV and cm
- See our manual for more variables and explanations

Don't forget to register the module with Fun4All