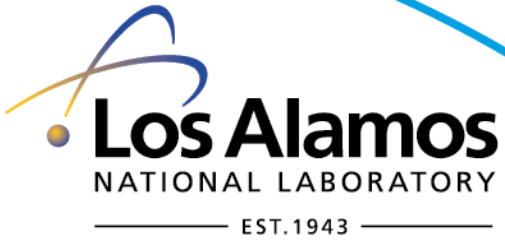


KFParticle: User Instructions

C. Dean

06/17/20

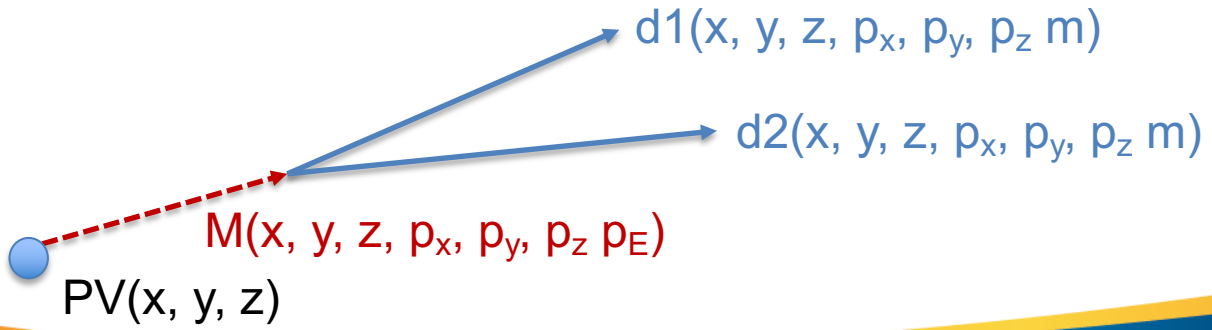
Introduction

- KFParticle is a decay reconstruction package, based around a Kalman Filter
- Implemented at ALICE, CBM and STAR
- Yuanjing has previously shown that KFParticle can be applied to sPHENIX data (https://indico.bnl.gov/event/7635/contributions/35077/attachments/26643/40449/KF_sphenix.pdf)
- This presentation:

1. What is KFParticle
2. How can it be used within Fun4All
3. Test example and null hypotheses
4. Next steps

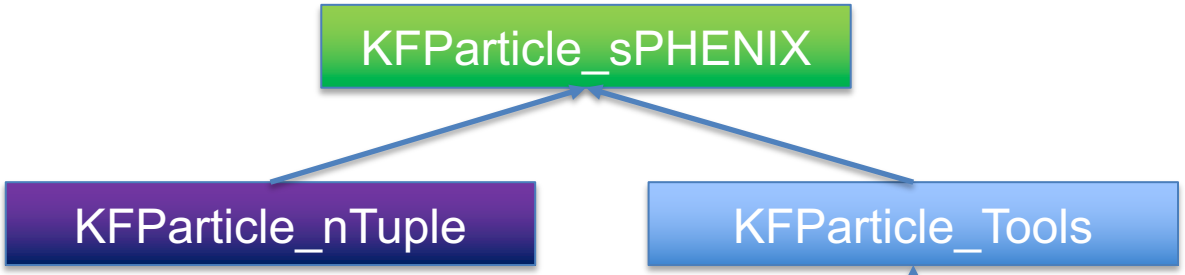
What is KFParticle

- A reconstruction package for tracks and vertices
- Based around a Kalman Filter
- Thus requires 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



Inheritance Diagram

- The project is separated into several sub-modules to aid in development



- User Interface
- Output
- Calculations

(KFParticle_nTuple and KFParticle_MVA contain KFParticle_Tools objects)

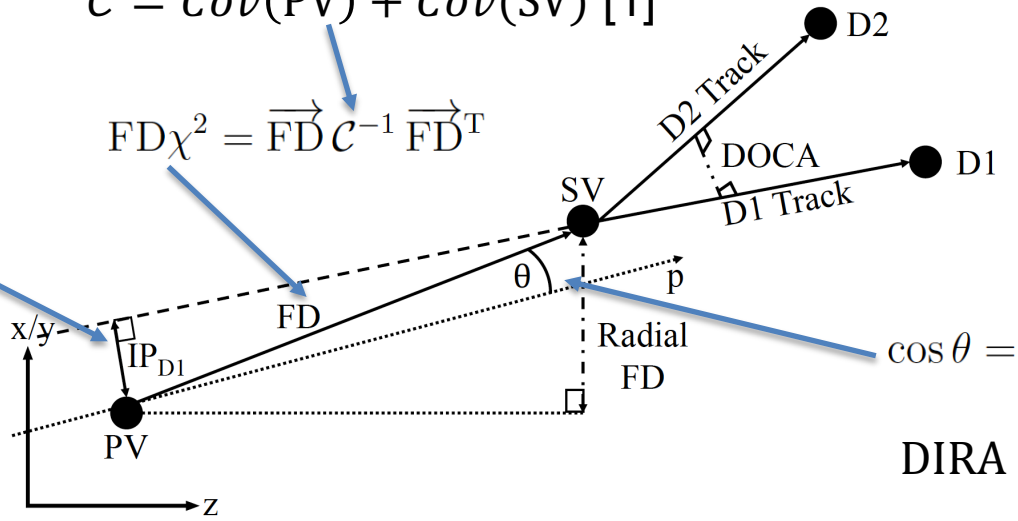
[1] = https://www-cdf.fnal.gov/physics/statistics/notes/cdf8661_chi2fit_w_corr_syst.pdf

Kinematic Cuts Available to sPHENIX

$$\mathcal{C} = Cov(\mathrm{PV}) + Cov(\mathrm{SV})\ [1]$$

$$\mathrm{FD}\chi^2 = \overrightarrow{\mathrm{FD}}\,\mathcal{C}^{-1}\,\overrightarrow{\mathrm{FD}}^{\mathrm{T}}$$

$$\mathrm{IP}\chi^2 = \left(\frac{\mathrm{IP}}{\sigma_{\mathrm{IP}}}\right)^2$$



$$\cos\theta = \frac{\overrightarrow{p}\cdot\overrightarrow{\mathrm{FD}}}{|\overrightarrow{p}||\overrightarrow{\mathrm{FD}}|}$$

$$\mathrm{DIRA} = \cos^{-1}\theta$$

Flight Distance χ^2 and DIRA are NOT in standard KFParticle packages

How is it implemented in Fun4All

- Make as user friendly as possible (do tricky parts behind the scenes)

1. Unpack all tracks and vertices then transform to KFParticle or KFPVertex
2. Search for good tracks (p_T and minimum IPχ^2 wrt all vertices)
3. Search for n-pronged decay based on DCA of tracks (find 2-prong from good tracks, add third or fourth good track if needed to each prong)
4. Obtain list of unique PID combinations of the tracks based on user requirements
5. Apply PID combinations to each prong and each PV combination to create mother
6. Accept or reject potential mothers based on invariant mass, mother p_T, angle between flight direction and momentum and mother FD χ^2
7. If multiple candidate mothers exist, select mother with the lowest mass uncertainty

Available particles

- The package currently handles 2- 3- and 4- body decays but no intermediate states (yet)
- The user specifies particles as a string. This is then checked against a map to return the required mass
- Only particles in the KFParticle package currently implemented
- New particles only require a string and an associated float to be used

```
std::map<std::string, float> particleMasses =
{
  { "electron", kfpDatabase.GetMass( 11 ) },
  { "muon",     kfpDatabase.GetMass( 13 ) },
  { "pion",     kfpDatabase.GetMass( 211 ) },
  { "kaon",     kfpDatabase.GetMass( 321 ) },
  { "proton",   kfpDatabase.GetMass( 2212 ) },
  { "pi0",      kfpDatabase.GetPi0Mass() },
  { "D0",       kfpDatabase.GetD0Mass() },
  { "Dplus",    kfpDatabase.GetDPlusMass() }
};
```

User Interface

- The UI is written to be as user friendly as possible
- Everything can be declared in the user top script
- There are several options to define the decay, set user cuts and set the output
- The next slide details the default options

Default options (Init and output)

```
void setNumberOfTracks( int num_tracks )
```
[Default is 2]

```
void setMotherCharge( int mother_charge )
```
[Default is uncharged]

```
void setFirstDaughter( std::string name )
```
[All daughters default to a pion]

```
void setSecondDaughter( std::string name )
void setThirdDaughter( std::string name )
void setForthDaughter( std::string name )
void saveOutput ( bool save )
```
[Default is true]

```
void setOutputName( std::string name )
```
[Default is outputData.root]

(saveOutput would be false if the user wishes to reuse the reconstructed event in a further analysis step)

Default options (Analysis)

`void setMinimumMass( float min_mass )` [Default is 0 GeV]
`void setMaximumMass( float max_mass )` [Default is 10 GeV]
`void setMinimumLifetime( float min_lifetime )` [Default is 0 ps]
`void setMaximumLifetime( float max_lifetime )` [Default is 10 ps]
`void setMinimumTrackPT( float pt)` [Default is 0.25 GeV]
`void setMinimumTrackIPchi2( float ipchi2 )` [Default is 10]
`void setMaximumDaughterDCA( float dca )` [Default is 0.2 mm (?)]
`void setFlightDistancechi2( float fdchi2 )` [Default is >50]
`void setMinDIRA( float dira_min )` [Default is 0.95]
`void setMaxDIRA( float dira_max )` [Default is 1.01 (i.e. no cut)]
`void setMotherPT( float mother_pt )` [Default is 0 GeV]

Default options (MVA, work in progress)

```
void useMVA( bool require_mva ) [Default is false]
void setNumMVAPars( unsigned int nPars )
void setMVAVarList( std::string mva_variable_list[ 99 ] )
void setMVAType( std::string mva_type )
void setMVAWeightsPath( std::string mva_weights_path )
void setMVACutValue( float cut_value )
```

A module exists to apply an MVA to your analysis. This module runs through ROOTs TMVA but is currently untested as I have no access to an MVA weight file. The methodology in the file has previously been tested on another experiment so the only issue I can imagine arising could be an out-of-scope issue when evaluating the MVA response to the variables but can easily be fixed if it arises, it will just make the packages more unseemly.

MVA (work in progress)

- If the analyst has an MVA weight file, this can be passed to KFParticle
- KFParticle_MVA will create an MVA reader and evaluate the events response
- The user needs to specify the path to the weight file and the MVA type (boosted decision tree, neural net etc.)
- The analyst also needs to pass an array of weight variables (in the same order/naming convention as the weight file!)
- These strings are then checked against a map to find the corresponding calculation of that variable before the event response is calculated
- If the user specifies a response cut value then this can be used to select events
- The calculated response for selected candidates should be written to a branch in the output file
- This module compiles and the initialization has been written into KFParticle_sPHENIX.cxx but is untested due to a lack of weight files for local testing

How to get/build/run tests

- The project is currently available at:
  https://github.com/cdean-github/KFParticle_sPHENIX
- To build, do:

```
cd KFParticle_sPHENIX/src/build
../autogen.sh --prefix=$MYINSTALL
make
make install
```

- To run, from build do:

```
cd ../..
root -l -q -b Fun4All_G4_Readback.C
```

- Inside Fun4All_G4_Readback.C you can specify the number of events to run over, modify cuts or decay channels. The default is 10k events, this can take a while to see an output.
- You should have a file called `outputData.root` with your selected events
- As an example, my environment variables are:

```
export SPHENIX=/sphenix/u/cdean/sPHENIX
export MYINSTALL=$SPHENIX/install
```

To do

- Improve charge conservation check. Currently ensures that the mother has the correct charge but not that the daughters are correct
  (e.g. $D^+ \to K^-\pi^+\pi^+ \neq D^+ \to K^+\pi^-\pi^+$)
- Implement intermediate decays (e.g. $B^+ \to \overline{D}^0(\to K^+\pi^-)\pi^+$)
- Test the multi-variate analysis module
- Figure out why lifetime calculation fails (lifetime error works though)
- Improve createDecay in KFParticle_Tools. it's currently messy and returns 4 daughters even if it's a 2-prong decay (the extra daughters are just a blank vector)
- Maybe alter in KFParticle_Tools. A map or pair could be used to check all the cuts, it would be neater and easier to update later on
- Implement an option to write truth variables in KFParticle_nTuple. Could possible do something with a track and vertex array to get the correct raw objects?
- Optimize code for memory usage

Examples

Example, $D^0 \rightarrow K^+\pi^-$

Green = non-default values
Blue = default values

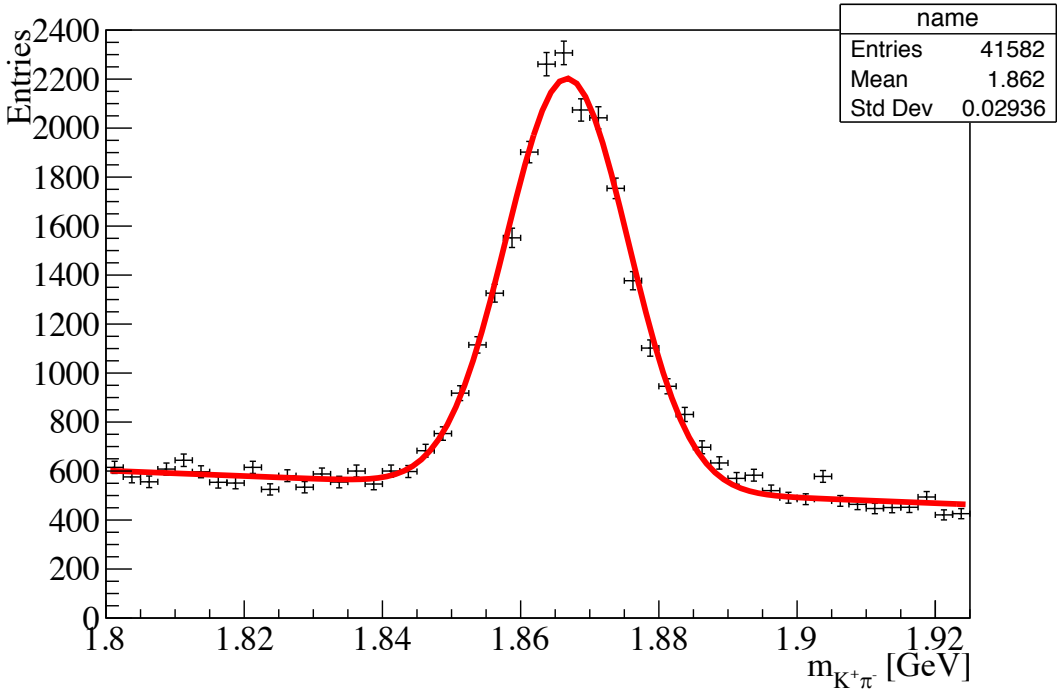
- The following steps can be taken to search for candidates
- All parameters have defaults
- Lifetime calculation is currently not available (returns 0)

- Include KFParticle header
- Create an object to find a decay
- Set number of daughter tracks (default is 2)
- Set mass range (default is 0 to 10 GeV)
- Set lifetime range (default is 0 to 10 ps)
- Set minimum track p_T (default is 0.25 GeV)
- Set minimum track IPχ^2 (default is > 10)
- Set maximum track DCA (default is 0.2 mm(?))
- Set minimum flight distance χ^2 (default is 50)
- Set DIRA (default is 0.95)
- Set minimum mother p_T (default is 0 GeV)
- Set mother charge (default is 0, likely to be removed soon)
- Declare daughter flavors (default is a pion)

```
#include <kfparticle_sphenix/KFParticle_sPHENIX.h>

 KFParticle_sPHENIX *kfparticle = new KFParticle_sPHENIX();

  kfparticle->setNumberOfTracks(2);
  kfparticle->setMinimumMass(1.8);
  kfparticle->setMaximumMass(1.925);
  kfparticle->setMinimumTrackPT(0.25);
  kfparticle->setMinimumTrackIPchi2(10);
  kfparticle->setMaximumDaughterDCA(0.2);
  kfparticle->setFlightDistancechi2(30);
  kfparticle->setDIRA(0.9);
  kfparticle->setMotherPT(0);
  kfparticle->setMotherCharge(0);
  kfparticle->setFirstDaughter("kaon");
  kfparticle->setSecondDaughter("pion");

  se->registerSubsystem(kfparticle);
```

Example, $D^0 \to K^+\pi^-$

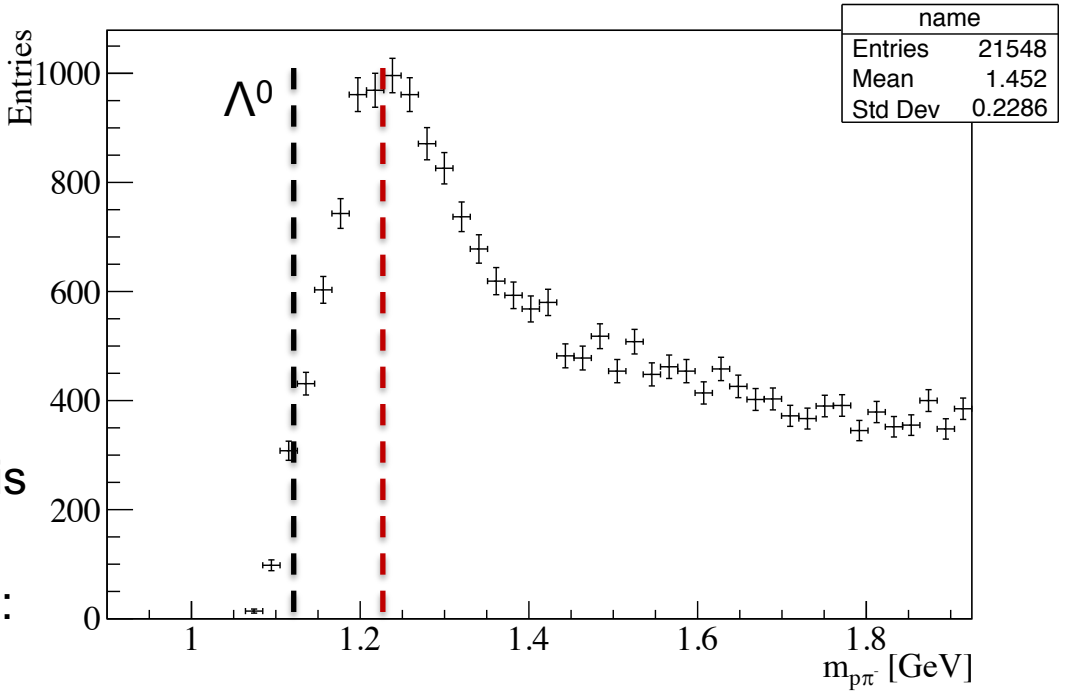
- Jin produced a large sample of D^0 events
- This has been the default for studying the implementation
- The previous slide details the cuts but several were the default value (only shown as an example)
- The reconstructed mass was: m(Kπ) = 1866.8 ± 0.1 MeV (PDG value is 1864.8 ± 0.1 MeV)
- The width was 8.8 ± 0.1 MeV



Example, null hypothesis

- 2nd test, reconstruct 2-prong vertex as a proton and pion
- MC was D^0 decays so this should return background and misID peaks.
- $\Lambda^0 \rightarrow p\pi^-$ is a common decay, extend mass range to cover this area
- Fun4All alterations wrt slide 12:

```
kfparticle->setMinimumMass(0.9);

kfparticle->setFirstDaughter("proton");
```

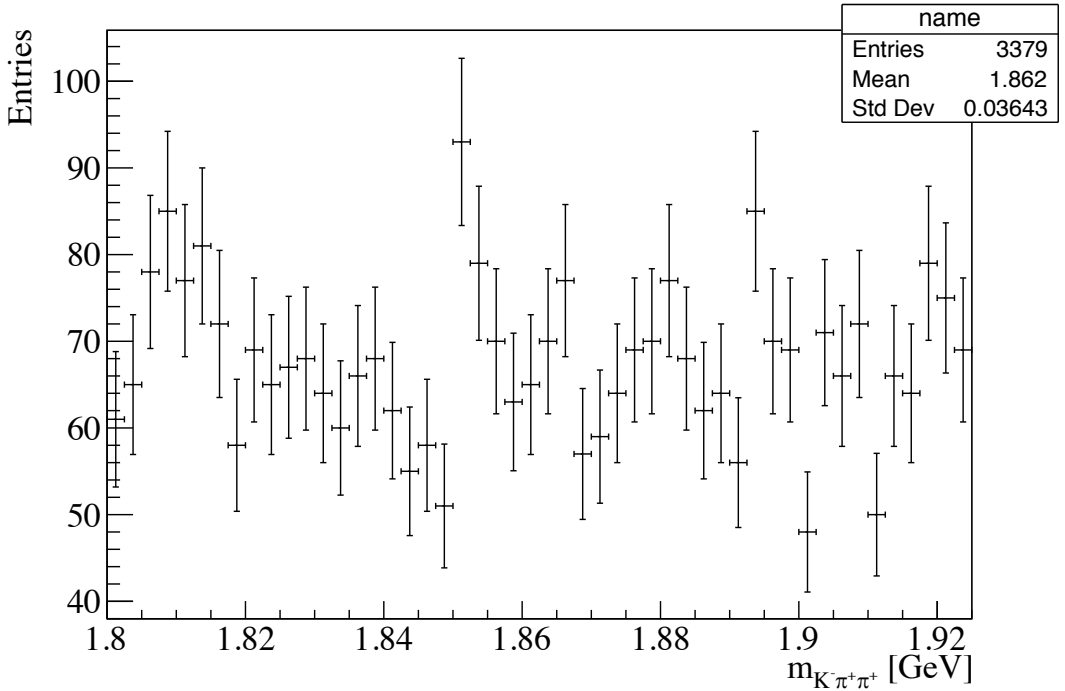


Example, null hypothesis

- 3rd test, reconstruct 3-prong vertex as a kaon and 2 pions.
- $D^+ \rightarrow K^-\pi^+\pi^+$ is a common decay, switch mother to charged and increase no. tracks to 3

Fun4All alterations wrt slide 12:

```
kfparticle->setNumberOfTracks(3);
kfparticle->setMotherCharge(1);
```



Output

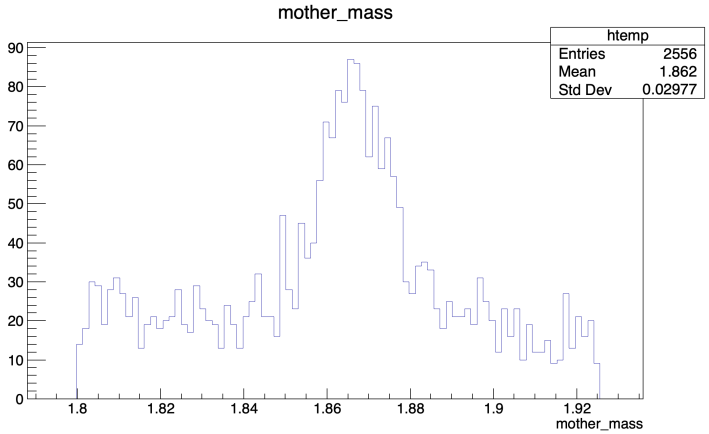
Output file

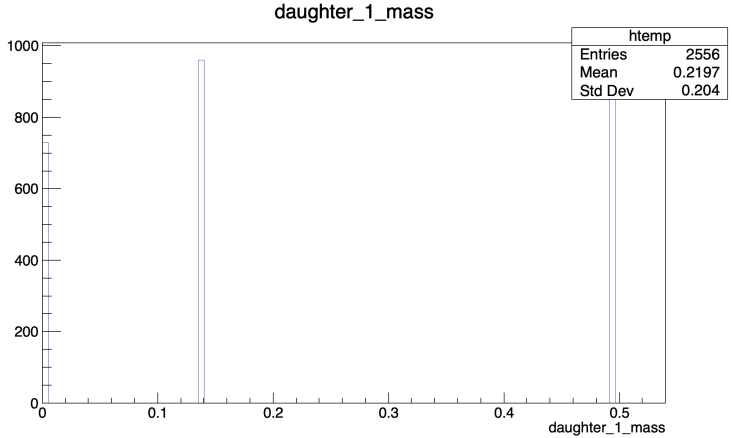
- The output file has several useful branches for analysts, eg. Mother and daughter masses, pT and event multiplicity (need to confirm)
- The user can further refine their cuts after using KFParticle (DIRA and FDchi2 are written to the file amongst other cuts)
- When MVA module has been tested, the MVA response should be written to the file for further offline analysis

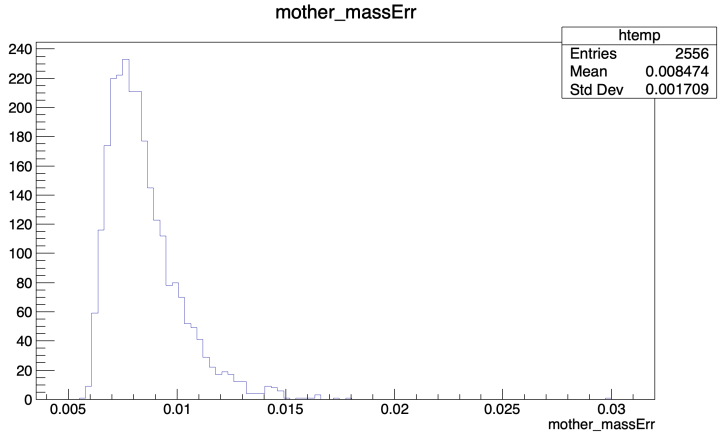
Mass

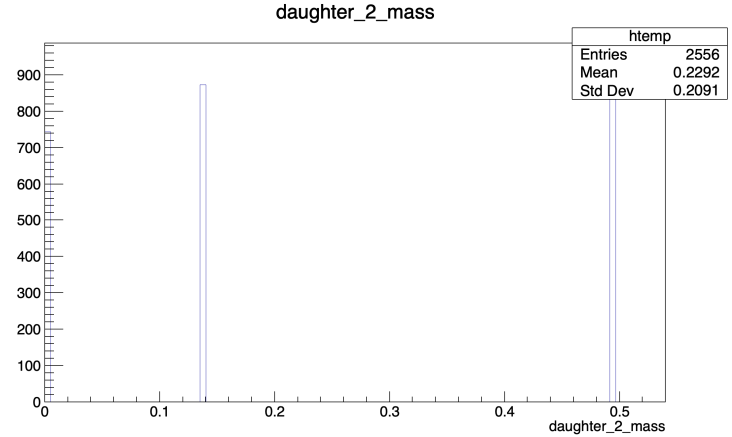
- The mass is written for each mother and daughter
- The daughter mass is stable however the branch is an admixture of different species
- This is currently due to no PID in sPHENIX, a more elegant solution could be a made at some stage to rearrange the daughters based on their flavor but you would need to be careful about charge and intermediate resonances…

Mass

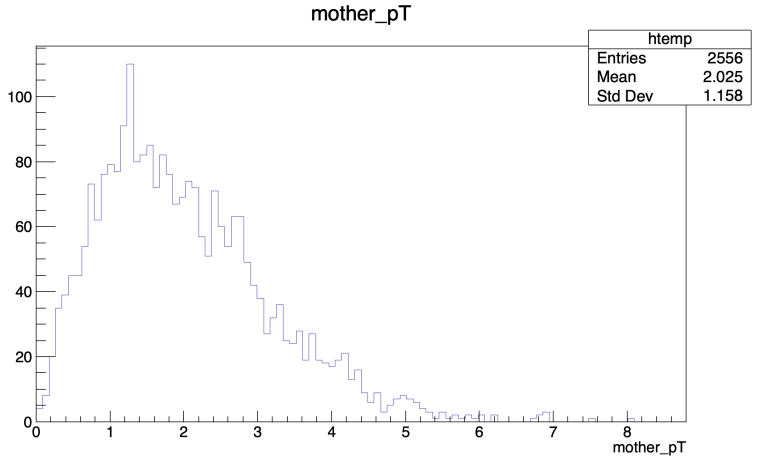


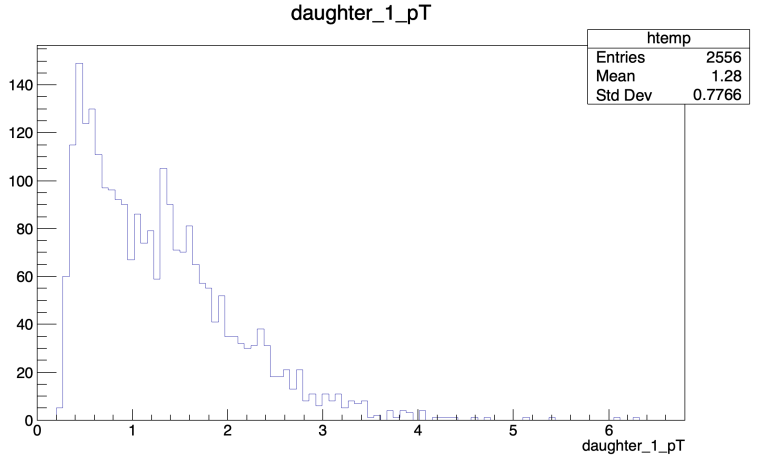


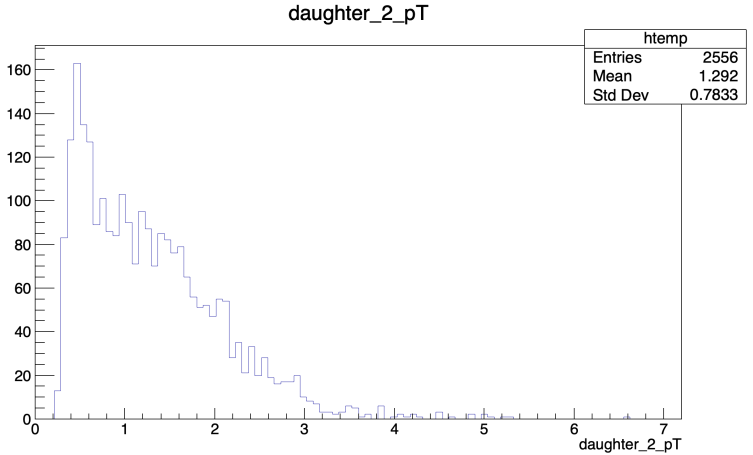




pT

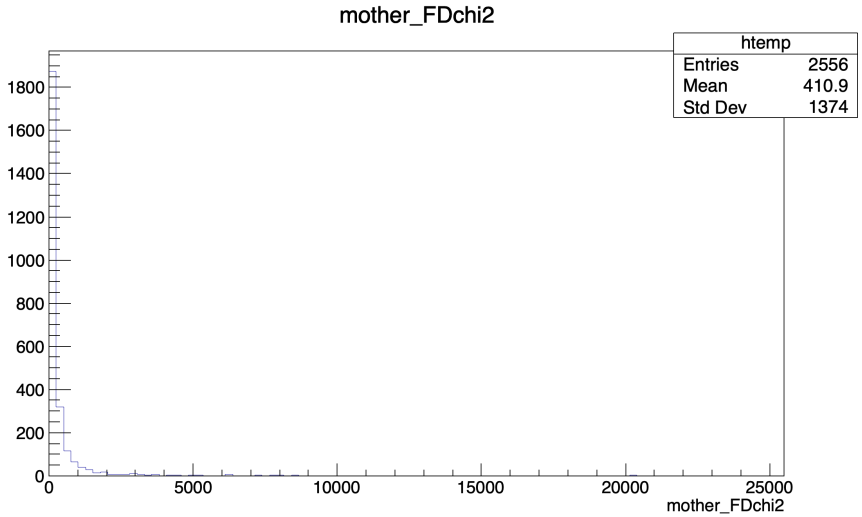


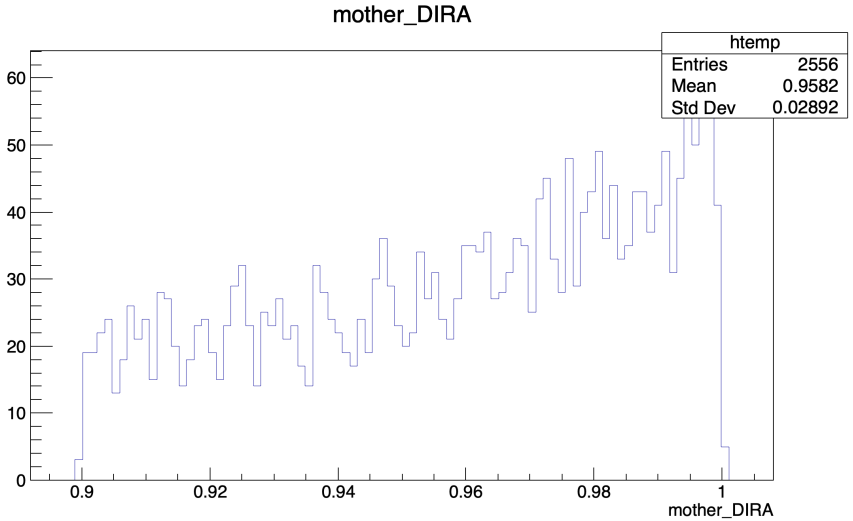




FDchi2 and DIRA

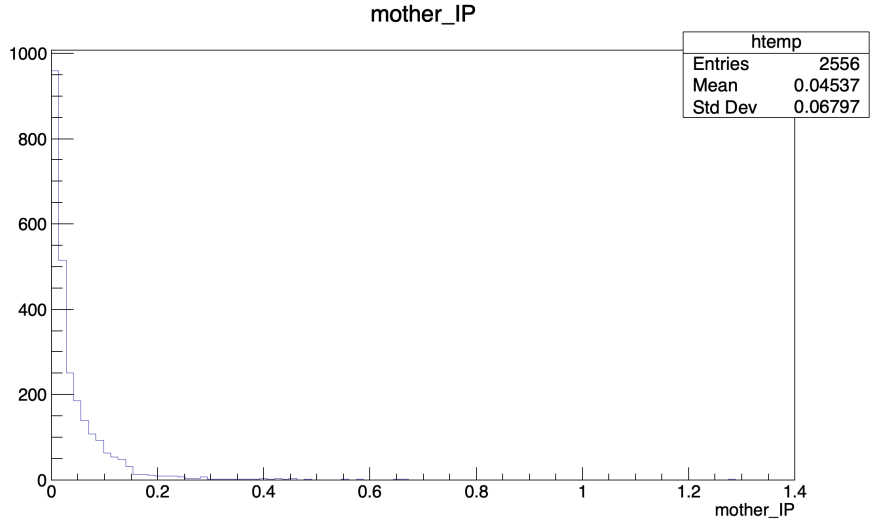
- Only written for the mother (doesn't make sense for the daughters just now)

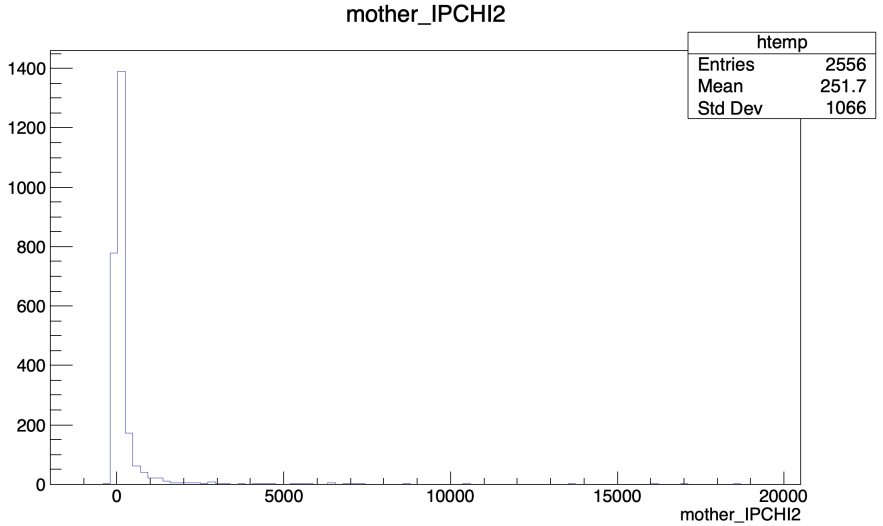




IP and IPchi2

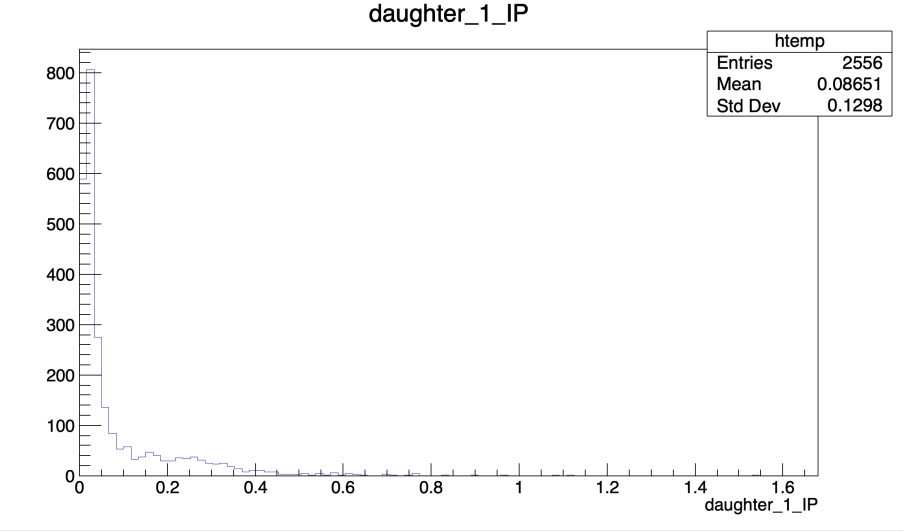
- Written for mothers and daughters
- Daughter IPchi2 is a useful cut (use greater than to improve displaced track yield)

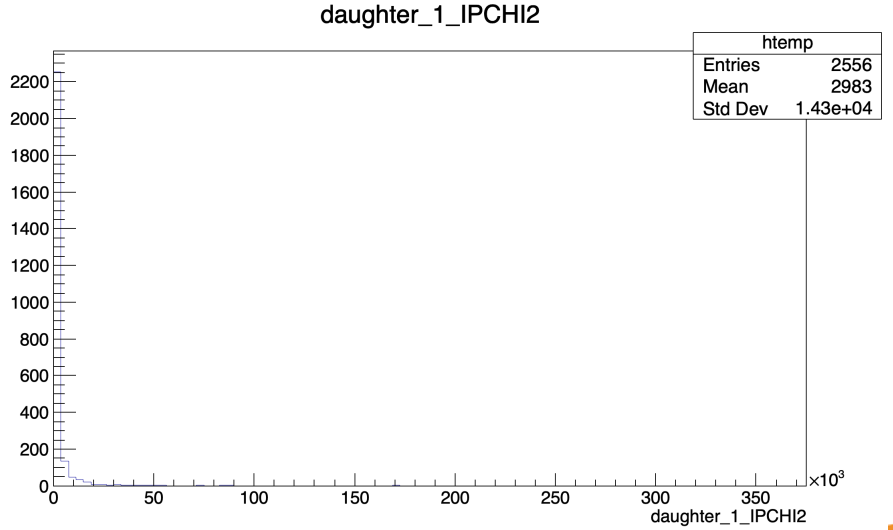




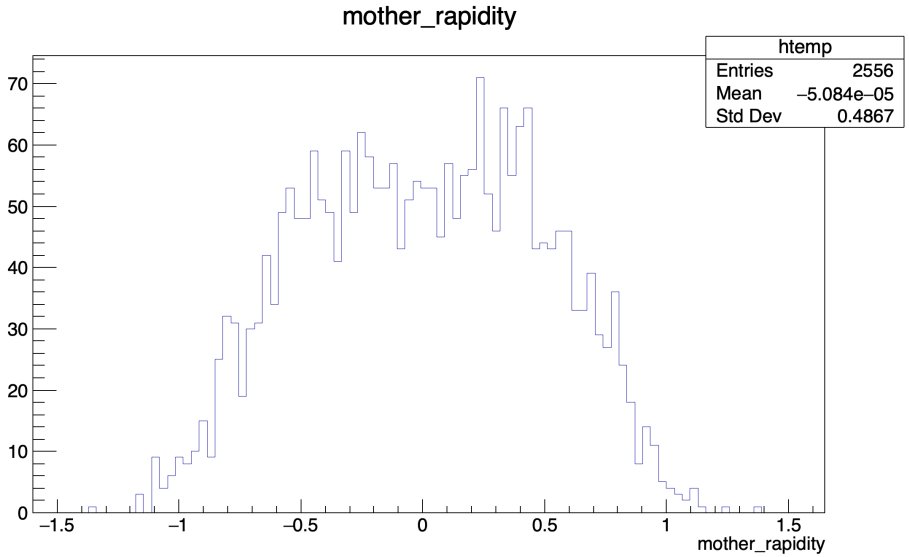
IP and IPchi2

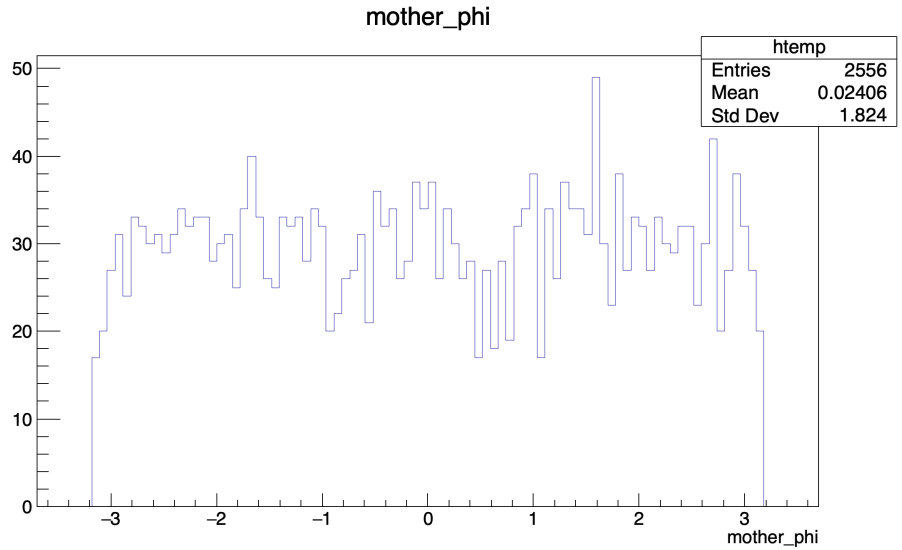
- Written for mothers and daughters
- Daughter IPchi2 is a useful cut (use greater than to improve displaced track yield)



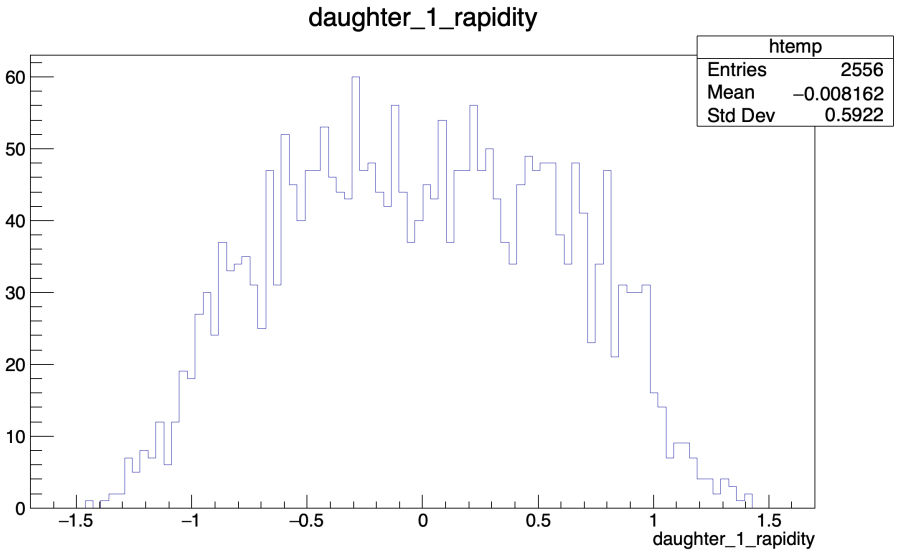


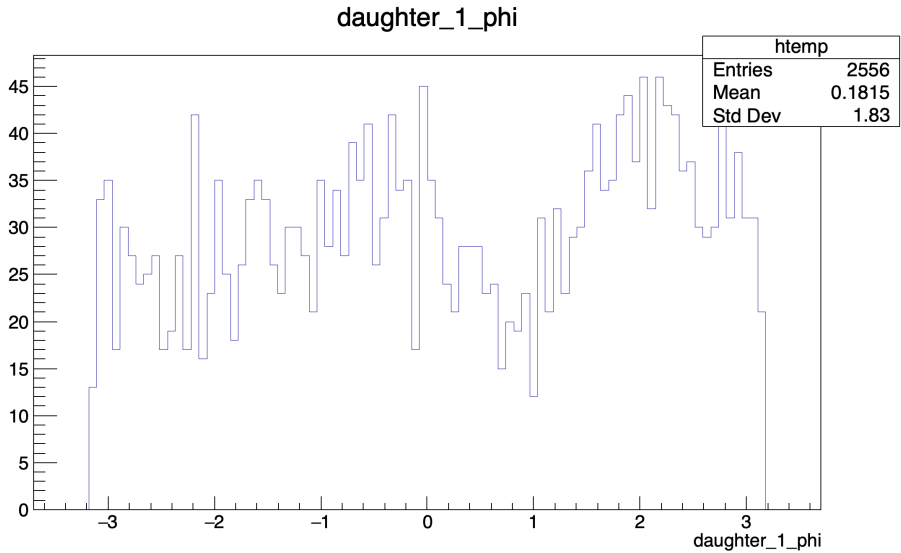
Rapidity and phi





Rapidity and phi





No. PVs and event multiplicity

- This can be useful for multiplicity-dependent measurements
- Need to confirm that the number of tracks in the DST is the same as the event multiplicity