KFParticle

User Instructions

Cameron Dean

11/10/2020

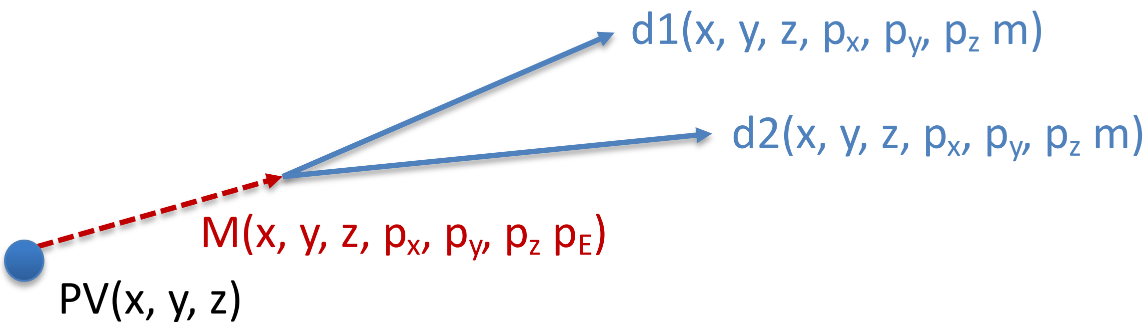
Introduction

- 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)
- These instructions:

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



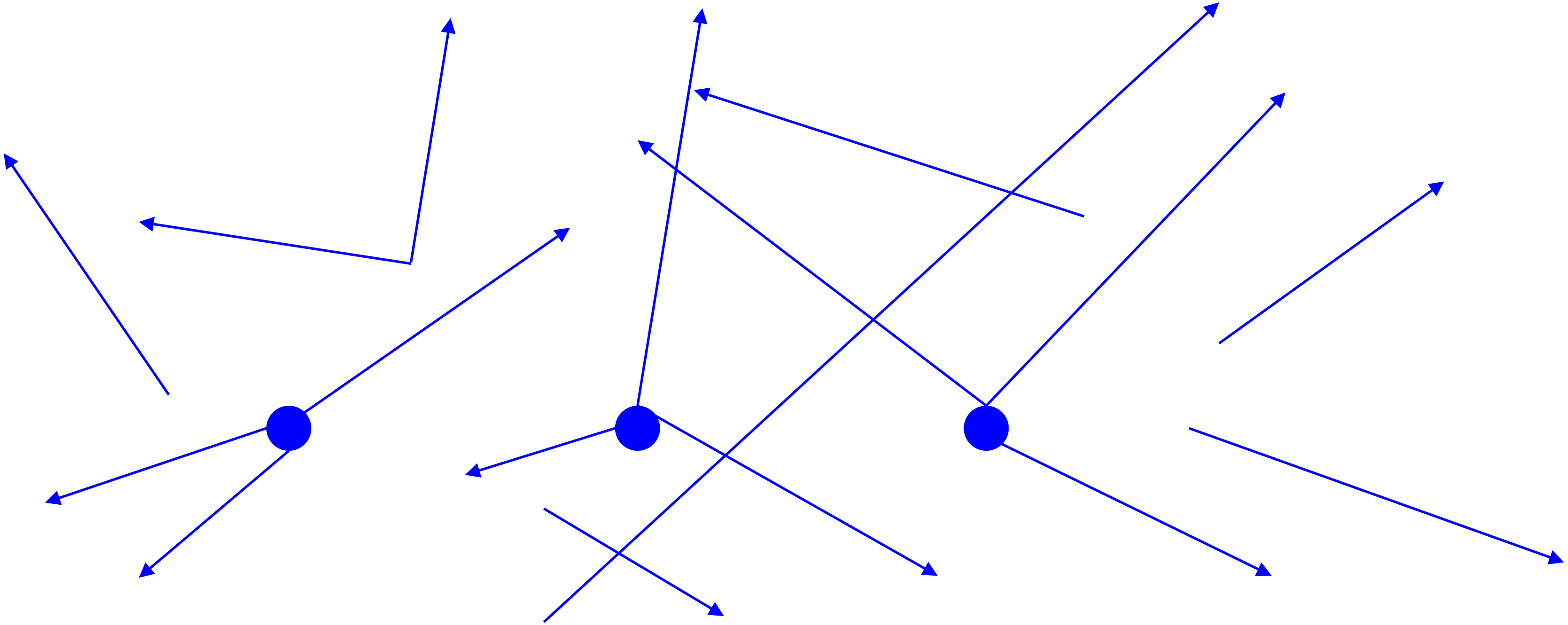
How is it implemented in Fun4All

- 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 $p_T\chi^2$, track χ^2 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 n-good tracks if needed to each prong) then apply vertex χ^2 requirement
4. Obtain list of unique PID combinations of the tracks based on user requirements
   - (Optional) Construct n-intermediate resonances
   - (Optional) Append n-tracks to intermediates from subset of 2.
5. Apply PID combinations to each decay product 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, mother FD χ^2 and mother IP χ^2
7. If multiple candidate mothers exist, select mother with the lowest mass uncertainty

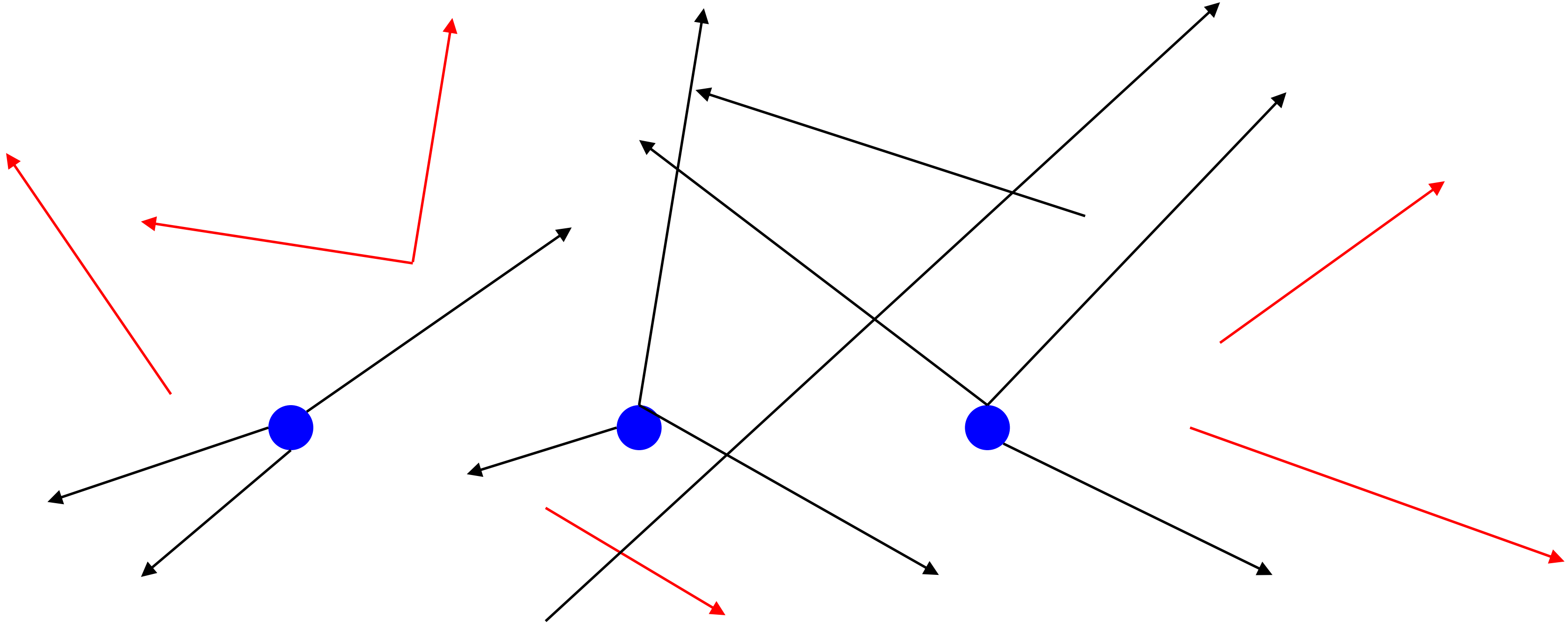
Step 1

- Unpack vertices and tracks



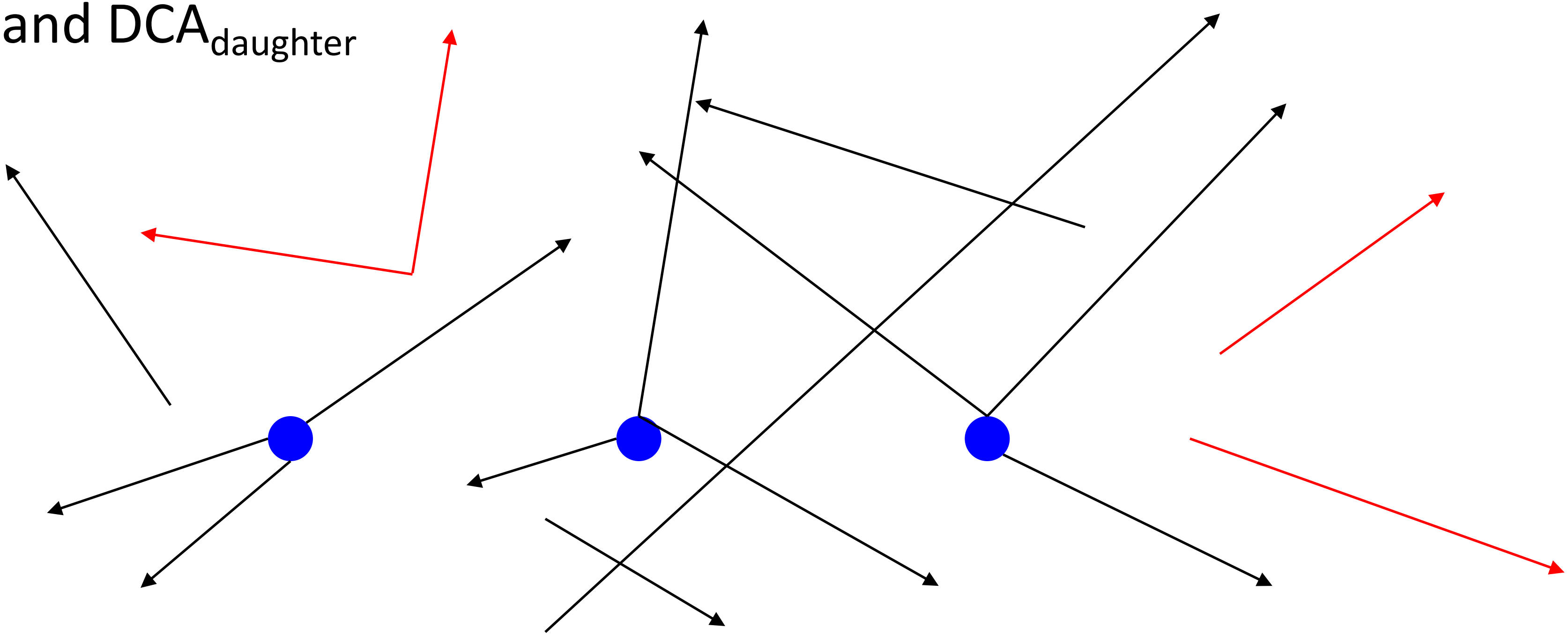
Step 2

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



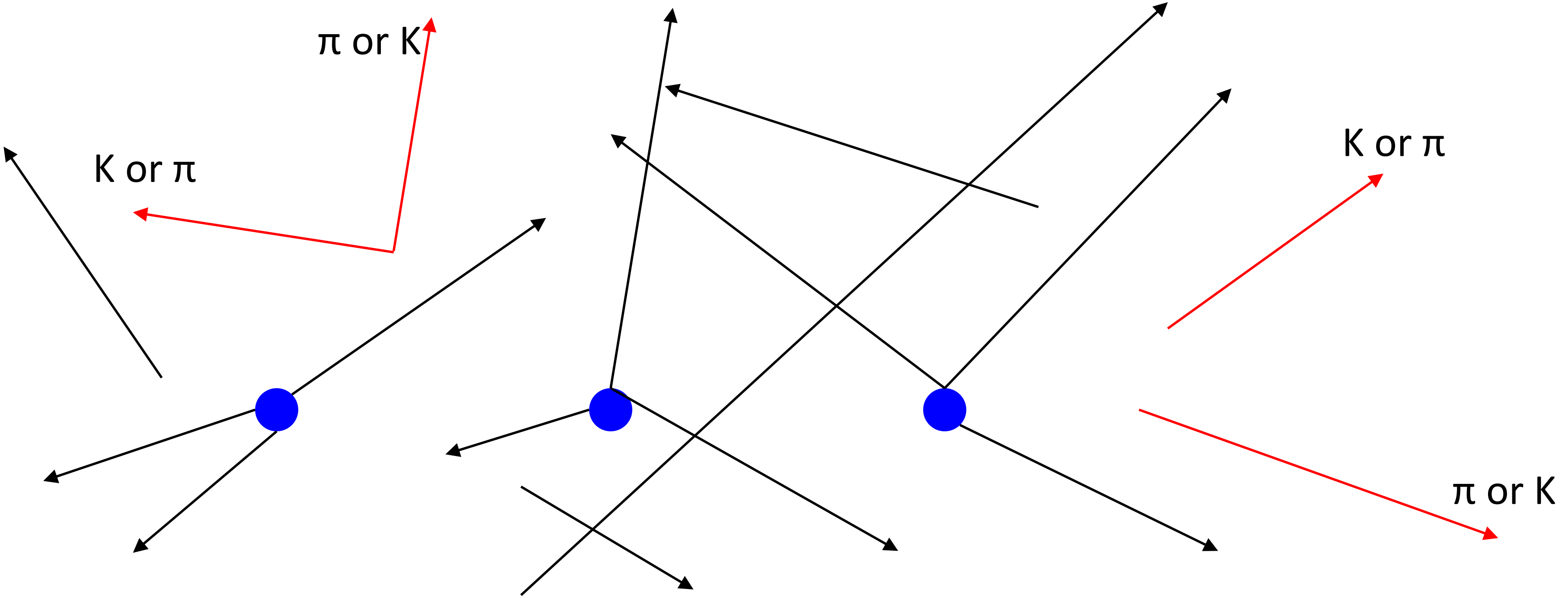
Step 3

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



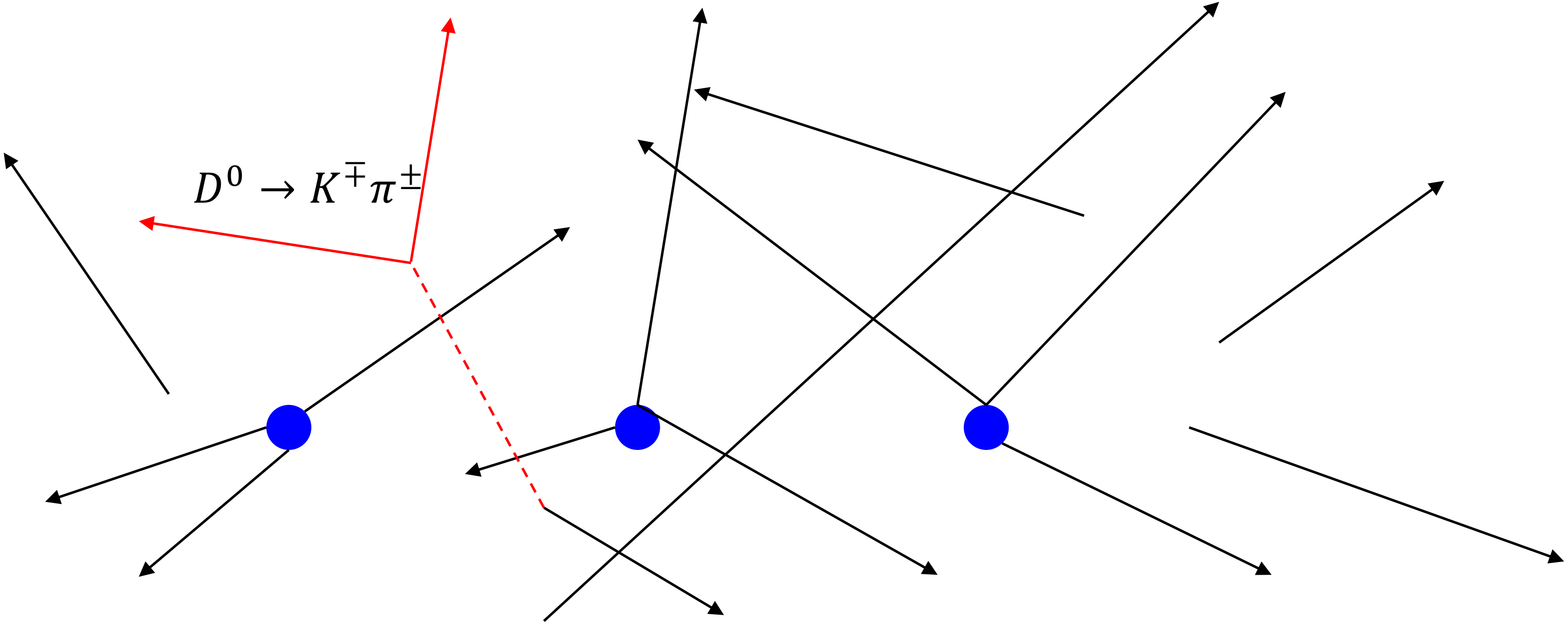
Step 4

- Assign PID based on unique combinations



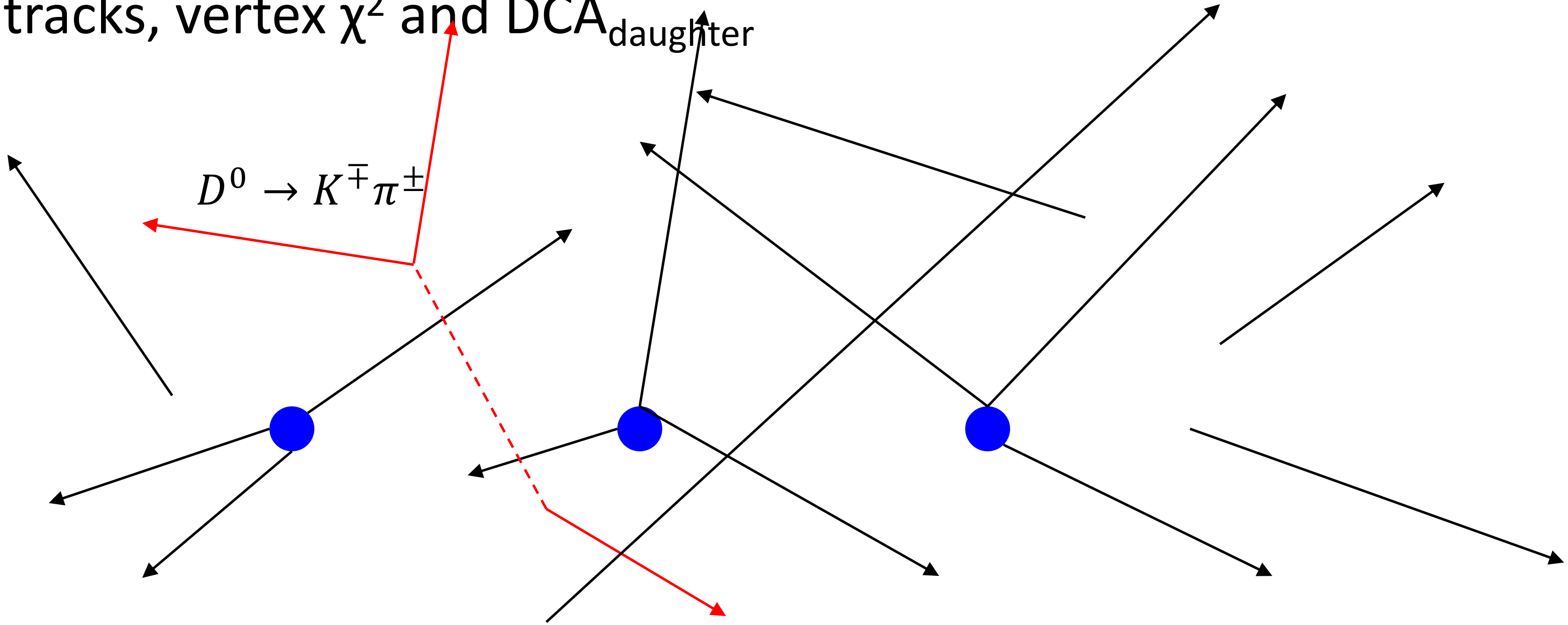
Step 4a (optional)

- Reconstruct intermediate decays based on selection and PID



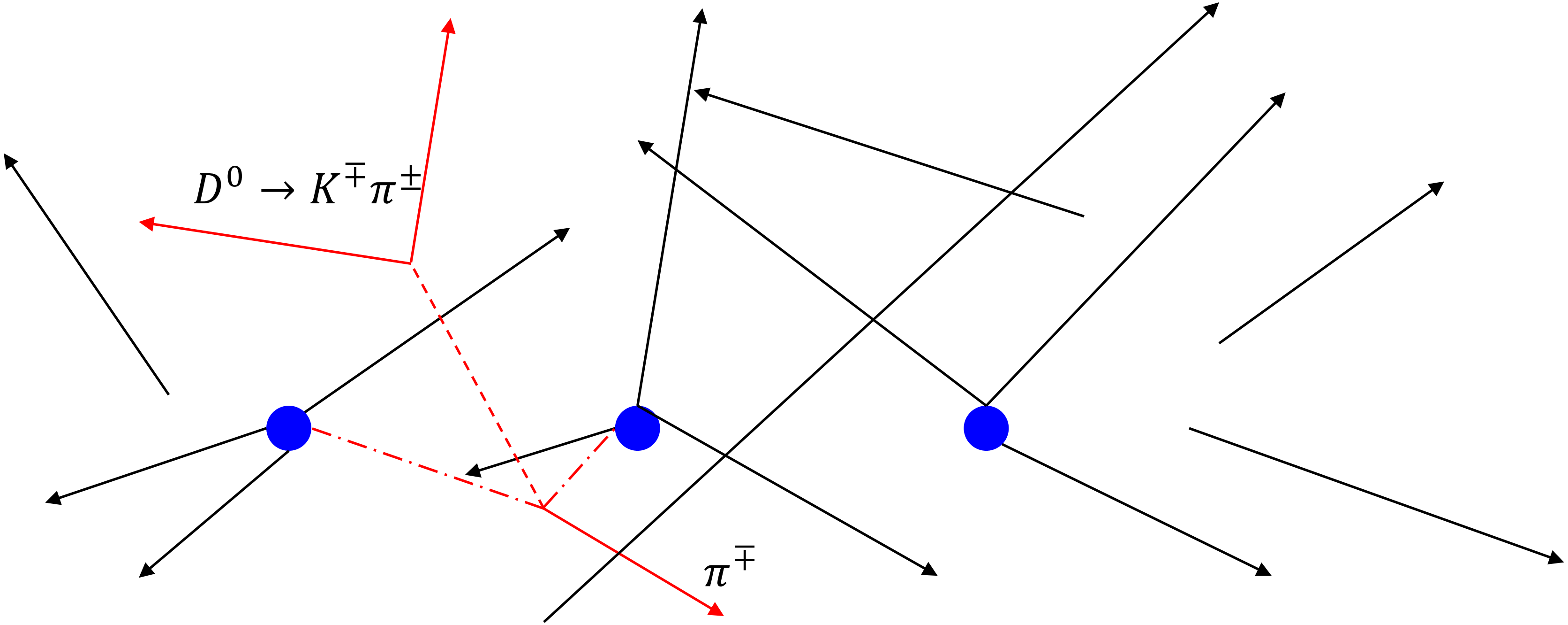
Step 4b (optional)

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



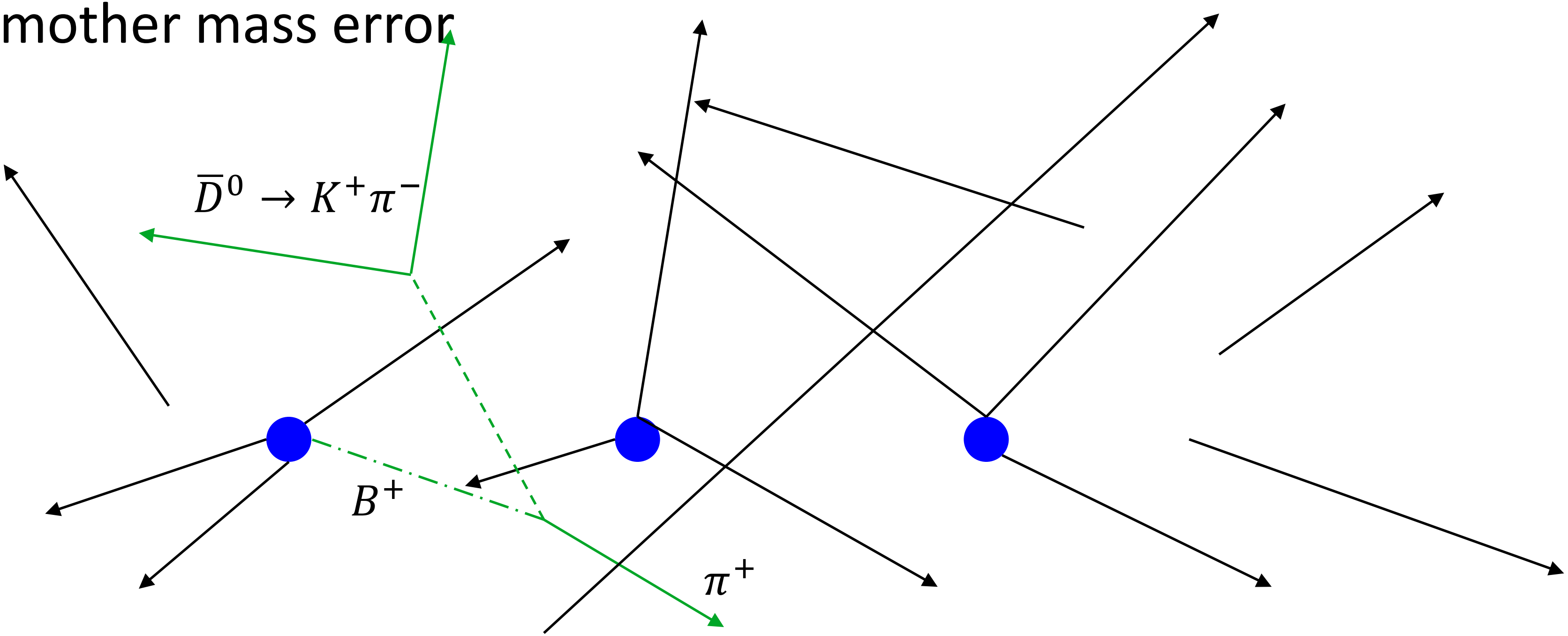
Step 5

- Reconstruct mother candidates based on selection and PID



Step 6

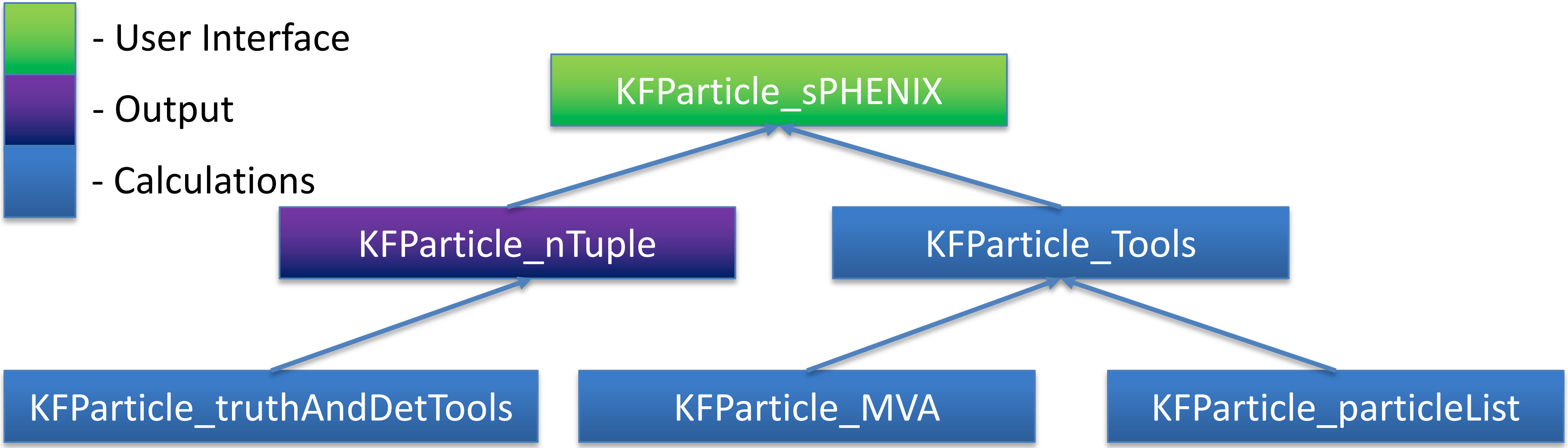
- If end vertex has more than 1 candidate, select based on lowest mother mass error



*Constraint to PV is also optional

Inheritance Diagram

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



(KFParticle_nTuple and KFParticle_MVA contain KFParticle_Tools objects)

How to get/build/run tests

- The project is currently available at:

  https://github.com/sPHENIX-Collaboration/analysis/blob/master/HF-Particle/KFParticle_sPHENIX/

- To build, do:

```
cd analysis/HF-Particle/KFParticle_sPHENIX/src/build
git checkout KFParticle
../autogen.sh --prefix=$MYINSTALL
make
make install
```

- To run, from build do:

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

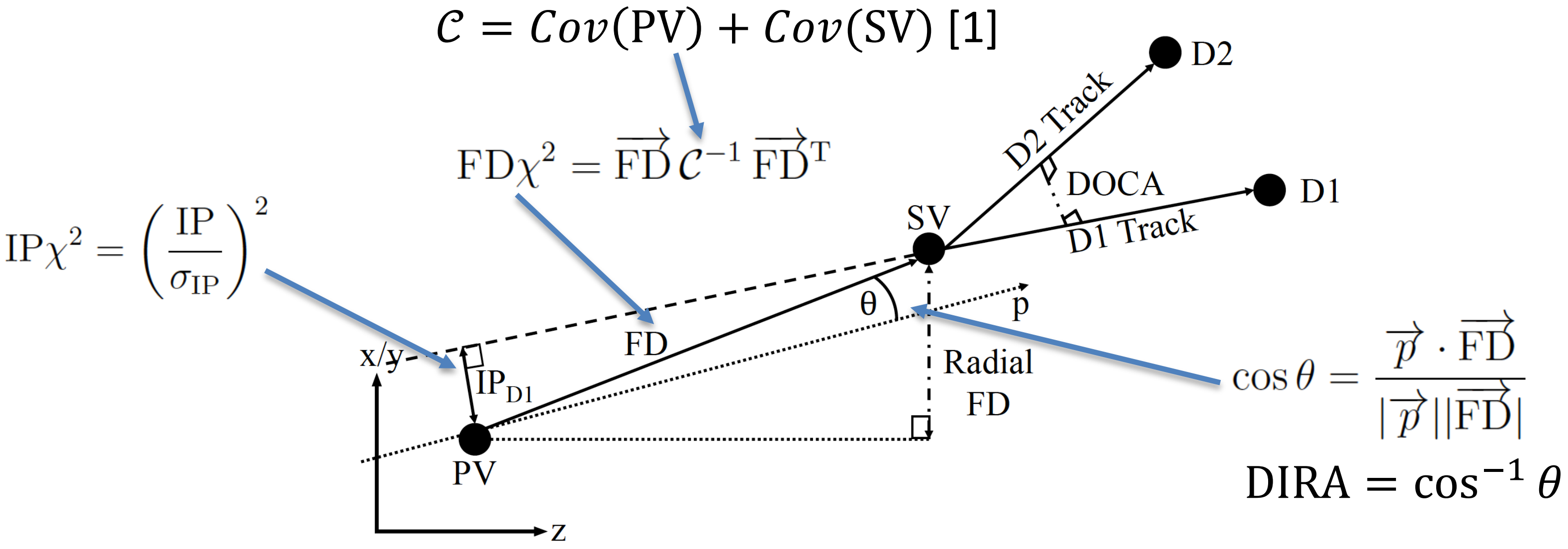
- As an example, my environment variables are:

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

Example macro

- The Fun4All example in the repository has been set up to perform 4 different reconstructions:
  1. $D^0 \rightarrow K^- \pi^+$
  2. $B_s^0 \rightarrow J/\psi(\rightarrow e^+ e^-)\phi(\rightarrow K^+ K^-)$
  3. $B^0 \rightarrow D^-(\rightarrow K^+ \pi^- \pi^-)\pi^+$
  4. $\Upsilon(nS) \rightarrow B^0\ (\rightarrow K^+ \pi^- \pi^- \pi^+)\ \bar{B}^0(\rightarrow K^- \pi^+ \pi^+ \pi^-)$
- The reconstruction is set from a map and will not allow 2 reconstructions at once (only one KFParticle object is created in the example)
- There's also a test space for general understanding
- N.B. The charm sample has many events, it will take a long time to process all of them so it is recommended to set this to 1k events first to get a feel for the reconstruction efficiency

Kinematic Cuts Available



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

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

Available particles

- The package currently handles n-body decays and up to 4 intermediate decays
  - **Output file is limited to 20 tracks while internal tool set is limited to 99 tracks (based on size of array)**
- The user specifies particles as pair of PID and charge. This is then checked against a map to return the required mass
- New particles only require a string and an associated float to be used
- Particles are defined in KFParticle_particleList.cxx
- Over 50 unique particles plus common alternatives and charge-conjugates

```
std::map<std::string, float> particleMasses;
  //Leptons
  particleMasses["electron"] = kfpDatabase.GetMass(11);
  particleMasses["muon"]     = kfpDatabase.GetMass(13);
  particleMasses["tau"]      = 1.77686;
  //B-hadrons
  particleMasses["B+"]       = 5.279;
  particleMasses["B-"]       = 5.279;
  particleMasses["B0"]       = 5.279;
```

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 slides detail the default options

Default options (tracks and vertices)

```
void setNumberOfTracks( int num_tracks ) [Default is 2]
void setMinimumTrackPT( float pt)  [Default is 0.25 GeV]
void setMaximumTrackPTchi2( float ptchi2 ) [Default is FLT_MAX]
void setMinimumTrackIPchi2( float ipchi2 ) [Default is 10]
void setMaximumTrackchi2nDOF( float trackchi2ndof ) [Default is 4]
void setMaximumDaughterDCA( float dca ) [Default is 0.05 mm]
void setMaximumVertexchi2nDOF( float vertexchi2nDOF )
void setDaughters( std::pair<std::string, int> daughter_list[99] )
```
[Default is $\pi^+\pi^-\pi^+\pi^-$]

Default options (output)

```
void saveOutput ( bool save ) [Default is true]
void setOutputName( std::string name ) [Default is outputData.root]
void doTruthMatching( bool truth ) [Default is false]
void getDetectorInfo( bool detinfo ) [Default is false]
```

saveOutput and setOutput name will write reconstructed candidates to an nTuple

doTruthMatching will write truth variables for the selected tracks such as true ID, momentum and decay vertex positions

getDetectorInfo will write hit locations in {x,y,z} and also which ladder/chip/TPC side registered the hit (this can make the nTuple very large). There is a map at the top of KFParticle_truthaAndDetTools.cxx where sub-detectors can be turned on/off

```
std::map<std::string, int> Use =
{
  { "MVTX",  1 },
  { "INTT",  1 },
  { "TPC",   1 },
  { "EMCAL", 0 },
  { "OHCAL", 0 },
  { "IHCAL", 0 }
};
```

Default options (mothers)

```
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] (not used)
void setMaximumLifetime( float max_lifetime ) [Default is 10 ps] (not used)
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]
void setMotherIPchi2( float mother_ipchi2 ) [Default is FLT_MAX]
void constrainToVertex( bool constrain_to_vertex ) [Default is false]
void getChargeConjugate( bool get_charge_conjugate ) [Default is true]
```

Default options (intermediates)

```
void hasIntermediateStates( bool has_intermediates )
void setNumberOfIntermediateStates( int n_intermediates )
void setNumberTracksFromIntermeditateState( int num_tracks[99])
```
(How many tracks are associated to each intermediate)
```
void constrainIntermediateMasses( bool constrain_int_mass )
```
(Constrain the intermediate decays to their PDG mass)
```
void setIntermediateMassRange( std::pair<float, float> intermediate_mass_range[99] )
```
(Set the range for each intermediates invariant mass)
```
void setIntermediateMinPT ( float intermediate_min_pt[99] )
```

Default options (MVA)

```
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.

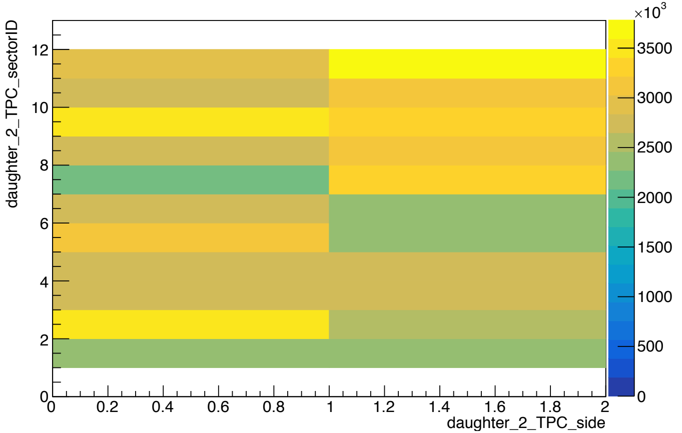
MVA (work in progress)

- 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

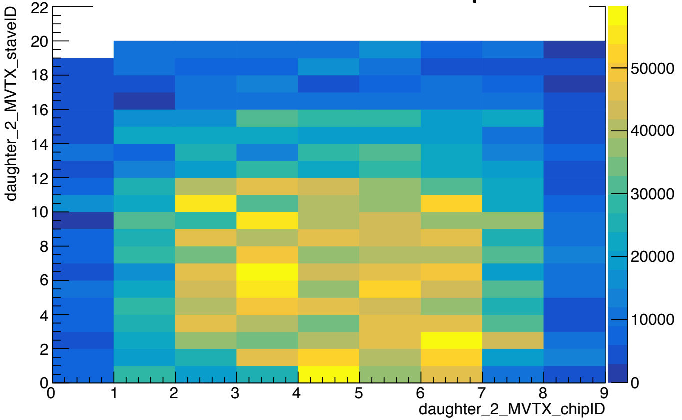
Cluster information

- daughter_1_local_x
- daughter_1_local_y
- daughter_1_local_z
- daughter_1_layer
- daughter_1_INTT_ladderZID
- daughter_1_INTT_ladderPhiID
- daughter_1_MVTX_staveID
- daughter_1_MVTX_chipID
- daughter_1_TPC_sectorID
- daughter_1_TPC_side

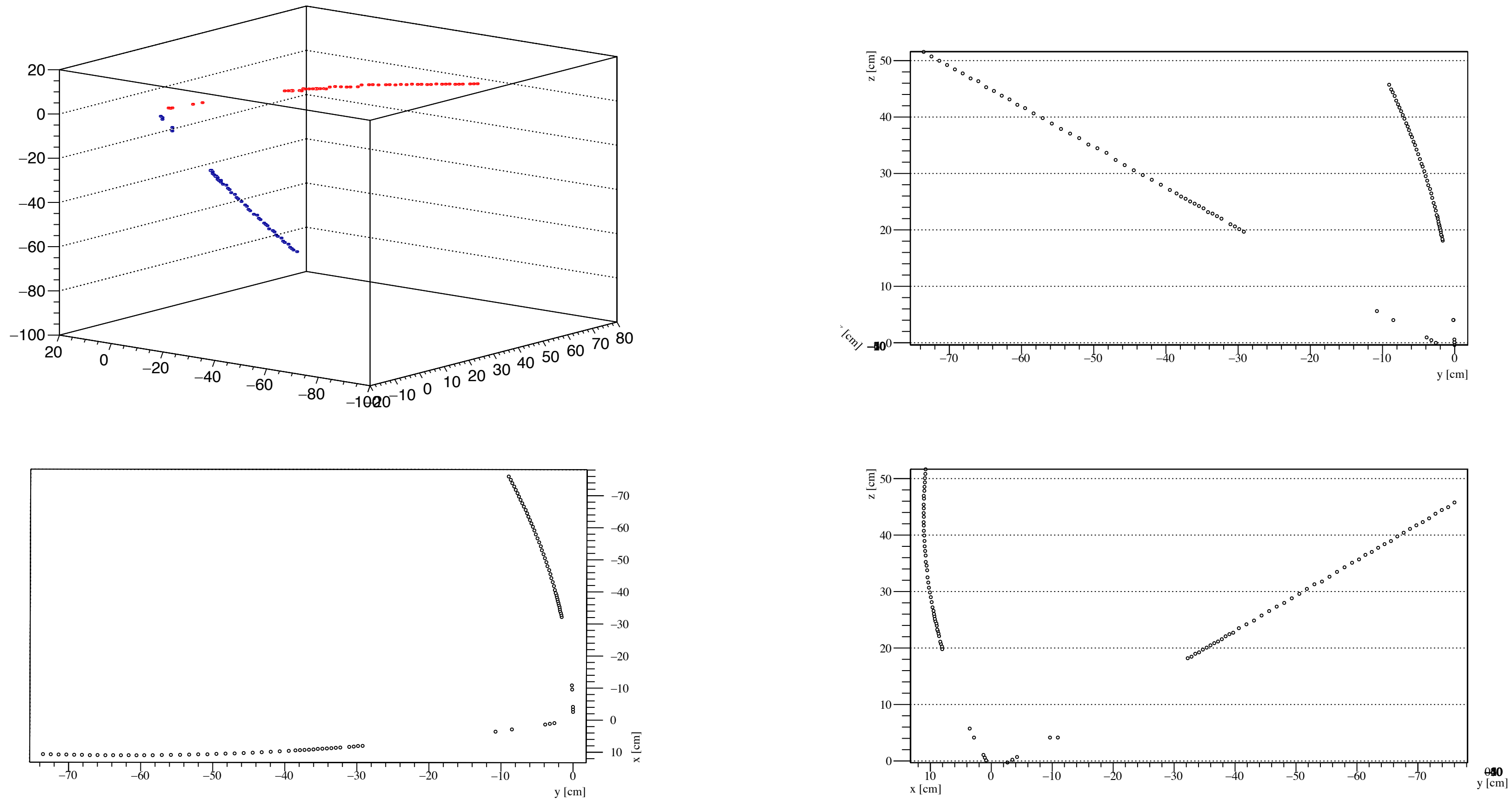
TPC Sector ID vs TPC Side



MVTX Stave ID vs Chip ID

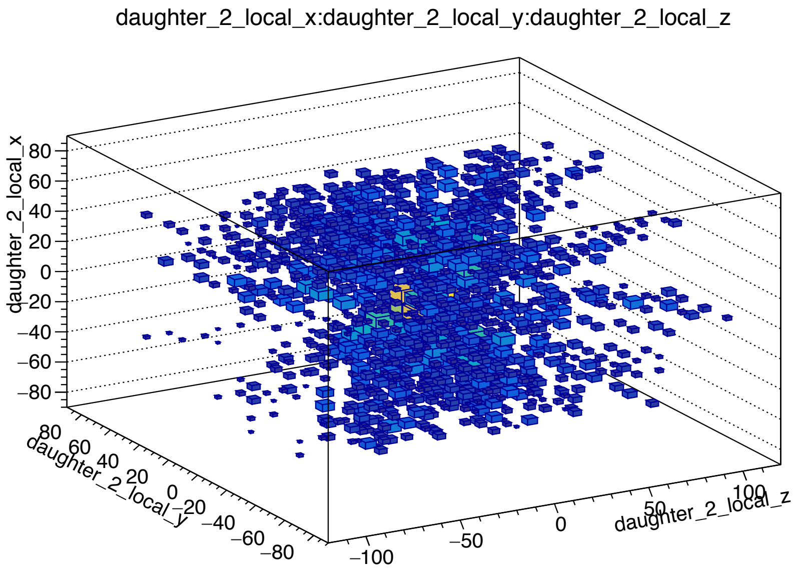


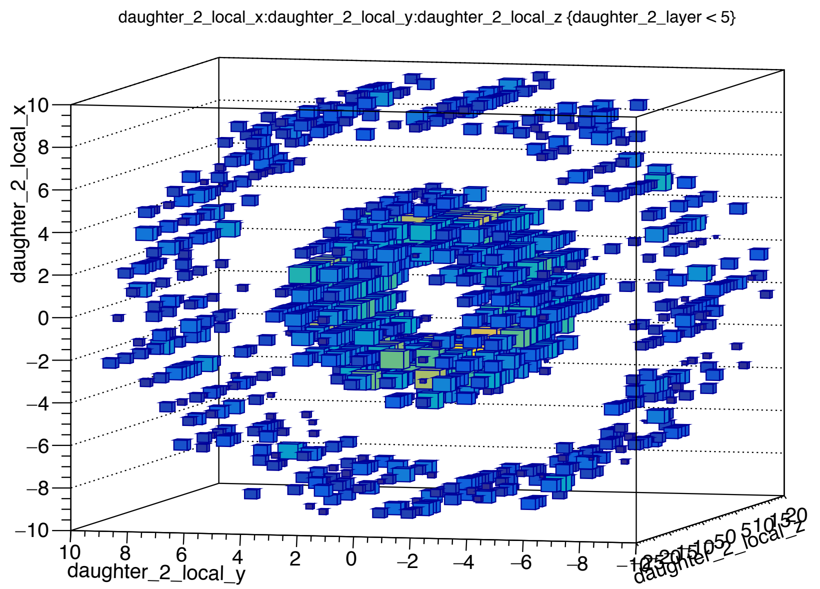
Bonus, decay display



Bonus, decay display

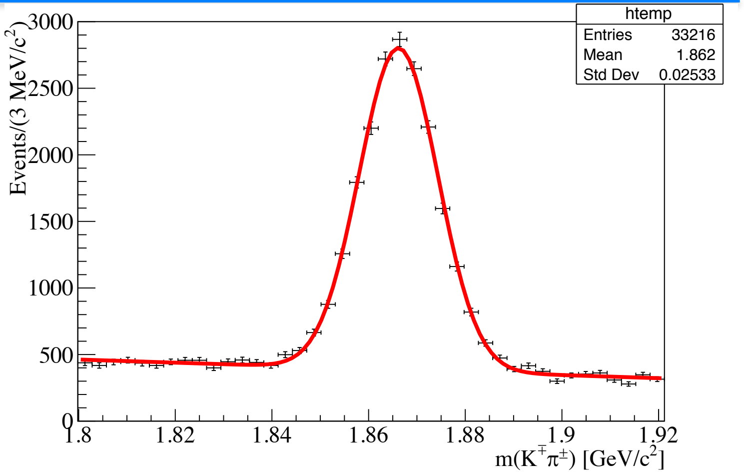
- 1754 selected $D^0 \rightarrow K^-\pi^+$ candidates
- Left – MVTX + INTT + TPC, right – MVTX + INTT

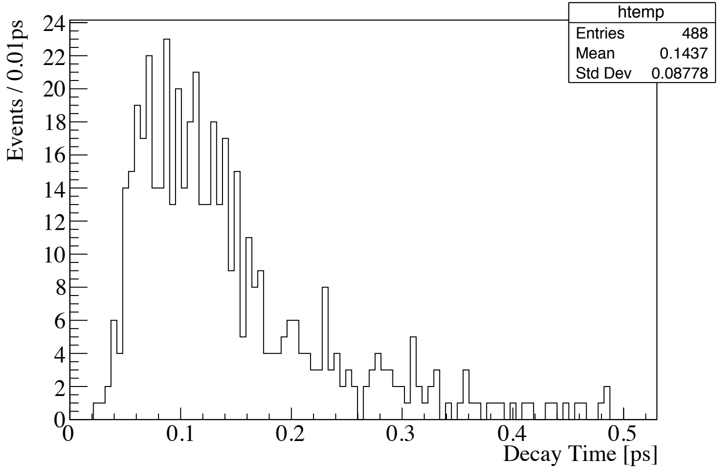




Conclusions

- Package has progressed to beta stage
- We can now reconstruct various heavy flavour decays; mothers to stable tracks, mothers to intermediates states, mixtures of these two and back-to-back reconstruction (quarkonia)
- Beta testing requires people to both make suggestions and try to break the package





Top - $D^0 \rightarrow K^-\pi^+$ invariant mass
Bottom - $D^0 \rightarrow K^-\pi^+$ decay time

Particle List

Particle list

```
//Leptons
particleMasses["electron"] = kfpDatabase.GetMass(11);
particleMasses["muon"]     = kfpDatabase.GetMass(13);
particleMasses["tau"]      = 1.77686;

//Gauge bosons and Higgs
particleMasses["W+"]    = 80.379;
particleMasses["W-"]    = 80.379;
particleMasses["Z"]     = 91.1876;
particleMasses["Higgs"] = 125.10;
```

```
//Light, unflavoured mesons
particleMasses["pion"]       = kfpDatabase.GetMas
particleMasses["pi+"]        = kfpDatabase.GetMas
particleMasses["pi-"]        = kfpDatabase.GetMas
particleMasses["pi0"]        = kfpDatabase.GetPi0
particleMasses["eta"]        = 0.547862;
particleMasses["f0(500)"]    = 0.5;
particleMasses["rho"]        = 0.77526;
particleMasses["rho(770)"]   = 0.77526;
particleMasses["f0(980)"]    = 0.990;
particleMasses["phi"]        = 1.019461;
particleMasses["phi(1020)"]  = 1.019461;
```

Particle list

```
//Strange mesons
particleMasses["kaon"]     = kfpDatabase.GetMass(321);
particleMasses["K+"]       = kfpDatabase.GetMass(321);
particleMasses["K-"]       = kfpDatabase.GetMass(321);
particleMasses["K0"]       = 0.497611;
particleMasses["KS0"]      = 0.497611;
particleMasses["KL0"]      = 0.497611;
particleMasses["K*(892)"] = 0.89166;

//Light baryons
particleMasses["proton"]  = kfpDatabase.GetMass(2212);
particleMasses["neutron"] = 0.93957;
particleMasses["Lambda"]  = 1.11568;
particleMasses["Sigma+"]  = kfpDatabase.GetMass(3222);
particleMasses["Sigma0"]  = 1.192642;
particleMasses["Sigma-"]  = kfpDatabase.GetMass(3112);
particleMasses["Xi0"]     = 1.31486;
particleMasses["Xi+"]     = 1.32171;
particleMasses["Xi-"]     = 1.32171;
```

Particle list

```
//Charm-hadrons
particleMasses["D0"]       = 1.86483;
particleMasses["D+"]       = 1.86965;
particleMasses["D-"]       = 1.86965;
particleMasses["Ds+"]      = 1.96834;
particleMasses["Ds-"]      = 1.96834;
particleMasses["D*0"]      = 2.00685;
particleMasses["D*+"]      = 2.01026;
particleMasses["D*-"]      = 2.01026;
particleMasses["Ds*+"]     = 2.1122;
particleMasses["Ds*-"]     = 2.1122;
particleMasses["Lc+"]      = 2.28646;
particleMasses["Lambdac+"] = 2.28646;
particleMasses["Xic0"]     = 2.47090;
particleMasses["Xic+"]     = 2.46794;
particleMasses["Xic-"]     = 2.46794;
particleMasses["Omegac"]   = 2.6952;
particleMasses["Xicc++"]   = 3.6212;
```

```
//B-hadrons
particleMasses["B+"]       = 5.279;
particleMasses["B-"]       = 5.279;
particleMasses["B0"]       = 5.279;
particleMasses["Bs0"]      = 5.366;
particleMasses["Bc+"]      = 6.2749;
particleMasses["Bc-"]      = 6.2749;
particleMasses["Bc"]       = 6.2749;
particleMasses["Bc(2S)"]   = 6.8716;
particleMasses["Lambdab0"] = 5.61960;
particleMasses["Sigmab+"]  = 5.81056;
particleMasses["Sigmab-"]  = 5.81056;
particleMasses["Xib0"]     = 5.7919;
particleMasses["Xib+"]     = 5.7970;
particleMasses["Xib-"]     = 5.7970;
particleMasses["Omegab+"]  = 6.0461;
particleMasses["Omegab-"]  = 6.0461;
```

Particle list

```
//Quarkonia
//c-cbar
particleMasses["J/psi"]        = 3.09690;
particleMasses["psi(2S)"]      = 3.68610;
particleMasses["X(3872)"]      = 3.87169;
particleMasses["chic1(3872)"] = 3.87169;
//b-bbar
particleMasses["Upsilon(1S)"] = 9.46030;
particleMasses["Upsilon(2S)"] = 10.02326;
particleMasses["Upsilon(3S)"] = 10.3552;
particleMasses["Upsilon(4S)"] = 10.5794;
particleMasses["Upsilon(5S)"] = 10.8852;
```