Decay Reconstruction

$\pi^0$’s and $K_S$’s and $D$’s, oh my!


CLEO 101 Day 4
12 June 2005
We want to learn about particles produced in the $e^+e^-$ collision or shortly after

- Focus on $D$ mesons produced as $e^+e^- \rightarrow \psi(3770) \rightarrow D\bar{D}$

What we detect is basically tracks and showers

- Tracks: charged particles passing through the tracking chambers, DR and ZD. Additional information on the identity of the tracks comes from the RICH, a “Čerenkov detector.”
- Showers: energy deposits in the crystal calorimeter, the CC. Photons can be found here.

We need to get from what we see (the final state) to what actually happened

By the end of today, you should have done precisely this for several decays.
An interaction

\[ \psi(3770) \]

Directly observe these (maybe)

Infer from momentum imbalance
Reconstruction (in this context) means putting together the things we detect to see if they are consistent with having some parent particle.

The classic way to do this is to add together the four-vectors of detected particles to get a candidate four-vector, then compute the invariant mass: $E^2 - \vec{p}^2$ should have the right value.\(^1\)

- This is not the only way. For example, for some long-ish lived neutral particles, we can see decays far from the origin; for $D$'s, we also use somewhat different variables.

\(^1\)Like most of the particle physics community, but unlike CLEO publications, I always set $c = 1$. 
Combinatorics

Say you are looking for the decay $\pi^0 \rightarrow \gamma\gamma$. (Hey, we will be!)

- There is nothing that tells you in advance that a given shower is the daughter of a $\pi^0$, or even that it is a photon. You also don’t know which pairs to combine to form the $\pi^0$.

- You have to just try every combination!

- Don’t you get this wrong often? Yes!
  - How is this any good then? You rely on the signal (what you’re looking for) having a different distribution in your variables than the wrong combinations (combinatoric background). For example, you hope that randomly joined photons don’t have a peak in their invariant mass at the $\pi^0$ mass.
  - OK, you don’t just hope this; you have to check it.

- Sometimes you just can’t see the signal above the background. Tough luck.
We look for the “stable” (i.e. long-enough lived to be detected) particles. For *hadronic* physics, these are charged kaons, pions, and photons. For *leptonic* and *semileptonic* physics, these also include electrons and muons.

We find the $\pi^0$, the neutral pion, through the decay $\pi^0 \rightarrow \gamma \gamma$.

We find the $K^0_S$, one of the neutral kaons, generally through $K^0_S \rightarrow \pi^+ \pi^-$.

There’s another neutral kaon, the $K^0_L$, which we generally can’t observe, although some people try very hard.

We can “find” a neutrino by seeing that there is a momentum imbalance and that the total energy is less than we expect, and seeing that this is compatible with a zero-mass particle.
How can you tell a kaon from a pion?

This is generically called “Particle Identification” (PID). CLEO-c has two main methods, both of which measure particle velocity.

\[ \frac{dE}{dx} \]

The energy loss rate of charged particles in our tracking chambers depends only on velocity (usually written as a function of \( \beta\gamma \) called the Bethe-Bloch formula). This energy loss is in ionizing the gas, so if we measure the charge a particle deposits we measure its velocity.

\( \frac{dE}{dx} \) information has a very simple interface and is easy to understand, so we will use it today.

RICH

The RICH measures the angle of the cone of Čerenkov light that particles emit as they pass through a medium.

The RICH is very powerful but a little tricky to use; it also doesn’t work for low-momentum tracks (exercise for the student: why?). Therefore we are not going to use it today.
You know what the momentum of the track is (this is what the tracking system measures), and you can also find \( \frac{dE}{dx} \), so you can test consistency with different hypotheses.

Notice you can’t separate \( \pi \) and \( \mu \) since they have very similar masses.

Separation between \( K \) and \( \pi/\mu \) better at lower momentum.
And now, let’s actually find something!

In Day4Proc (/home/ljf26/workshop/Day4Proc/), I’ve implemented code to find candidates for two decays:

- $\pi^0 \rightarrow \gamma \gamma$ and
- $D^0 \rightarrow K^- \pi^+$.\(^2\)

\(^2\)We almost always search for a decay and its charge conjugate, i.e. both $D^0 \rightarrow K^- \pi^+$ and $\overline{D}^0 \rightarrow K^+ \pi^-$. 
$\pi^0 \rightarrow \gamma\gamma$
Extracting showers and applying cuts

In the Event method of your processor, you extract whatever navigation objects you are interested in. Here, we want to extract showers:

```cpp
FATable< NavShower > showerTable;
extract( iFrame.record( Stream::kEvent ), showerTable );
```

Function for cuts:

```cpp
DABoolean isGoodShower(const NavShower& shower) {
    const CcShowerAttributes& atts = shower.attributes();
    return (0.05 < atts.energy() && atts.energy() < 2.0
            && !atts.hot()
            && atts.e9oe25UnfOK()
            && shower.noTrackMatch()
            && (atts.goodBarrel() || atts.goodEndcap()));
}
```
Extracting showers and applying cuts

Begin loop and check that showers pass quality cuts:

```cpp
FATable< NavShower >::const_iterator showerBegin = showerTable.begin();
FATable< NavShower >::const_iterator showerEnd = showerTable.end();
for (FATable< NavShower >::const_iterator showerItr1 = showerBegin;
     showerItr1 != showerEnd ; ++showerItr1 ) {
    if (isGoodShower(*showerItr1)) {
        for (FATable< NavShower >::const_iterator showerItr2 = showerItr1+1;
             showerItr2 != showerEnd ; ++showerItr2 ) {
            if (isGoodShower(*showerItr2)) {
...
```

For each set of showers that have passed the cuts, form a four-vector and fill histogram with invariant mass:

```cpp
const HepLorentzVector& v1 = (*showerItr1).photon().lorentzMomentum();
const HepLorentzVector& v2 = (*showerItr2).photon().lorentzMomentum();
m_pi0mass->fill((v1+v2).m());
```
Result

pi0 candidate mass (GeV)


Decay Reconstruction

CLEO 101 Day 412 June 2005

Entries 219752
Mean 0.1293
RMS 0.05337
A note on Navigation objects

You’ve already encountered NavTrack and NavShower. The next slide mentions some other Nav* objects. Navigation (https://www.lns.cornell.edu/restricted/CLEO/CLEO3/soft/LibraryDocs/Navigation/Navigation.html) objects are intended to simplify and unify the interface to commonly used information.

If you want information on a high-level detected object, it’s probably in Navigation. Take a look at it first.

The exception being (heh heh) $D$ tags. These are accessed through a different interface that you will learn about in a couple of days.

BEWARE: Extracting a certain Navigation Item may require additional header files or alterations to your make file. Using the appropriate argument to mkproc can help you figure this out if you have problems!
How do you extract four-vectors?

- The software objects that hold the momentum information are of the class `KTKinematicData`.³

- You can get a `LorentzVector` from a `KTKinematicData` object by calling `lorentzMomentum()`.

- How do you get `KTKinematicData`? Extract the appropriate Navigation item, then:
  
  - Tracks: `NavTrack::pionFit()` or `kaonFit()` or `xxxxFit()`
  - Photons: `NavShower::photon()`
  - $\pi^0$'s: `NavPi0ToGG::pi0()`
  - $K^0_S$'s: `NavKs::kShort()`

³They may *inherit* from `KTKinematicData`, though.
We have a standard $\pi^0$ and $K^0_S$ reconstruction that is better than the simple invariant mass peak we’ve looked for.

- $\pi^0$’s are reconstructed from pairs of photons, as we do, but they are “fit” to the known $\pi^0$ mass, so when you use their four-vectors, you always have $E^2 - \vec{p}^2 = (0.135 \text{ GeV})^2$.

- $K^0_S$ ($\rightarrow \pi^+\pi^-$) have the complication that they have non-trivial lifetimes; they frequently decay far from the interaction region, so the daughter pions don’t come from the interaction point.
  - The invariant mass must be calculated using the momenta at the decay point, not the IP.
  - In our software this is implemented with VFinderProd, which can find tracks intersecting away from the origin.

Ready-made $\pi^0$ and $K^0_S$ objects, with default cuts, are available for the data in EventStore. If you ever need to remake them, you need PhotonDecaysProd and VFinderProd.
$D^0 \rightarrow K^- \pi^+$
Reconstructing $D$'s

You can reconstruct $D$ mesons using just invariant mass. However the peaks are very broad.

- When we form the invariant mass, we use $m^2 = E^2 - \vec{p}^2$. But how do we get $E$? By summing up the energies of the decay particles, which we get from $E^2 = m^2 + \vec{p}^2$!

- Errors in the momenta of the decay particles can cancel in the momentum sum because they are vectors, but the energies are scalars so their errors sum together linearly.

We can do better, by separating these two uncertainties into different variables.

- We know that the $D$ should have had energy equal to the beam energy (= half the center of mass energy).\(^4\)

- We use this to obtain two variables, the “beam-constrained mass” $m_{BC}$ and “Delta-E” $\Delta E$.

\(^4\)Not 100% true, but ask your advisor to explain about crossing angles and initial state radiation.
Say we have
\[
\vec{p}_D = \sum_{\text{decay particles}} \vec{p}_i
\]
\[
E_D = \sum_{\text{decay particles}} E_i
\]
Then
\[
m_{BC}(D)^2 = \left(\frac{E_{cm}}{2}\right)^2 - \vec{p}_D^2
\]
and
\[
\Delta E = E(D) - \frac{E_{cm}}{2}
\]
- \(m_{BC}\) tests that the momentum of the \(D\) candidate is right.
- \(\Delta E\) tests that we have the particle assignments right and are not missing anything.
DABoolean isGoodTrack(const NavTrack& track) {
    const TRHelixFit& helix = *track.pionHelix();
    const TRTrackFitQuality& fitqual = *track.pionQuality();
    return (0.05 < (*track.pionFit()).pmag() && (*track.pionFit()).pmag() < 2.0
            && fabs(helix.d0()) < 0.005
            && fabs(helix.z0()) < 0.05
            && fitqual.chiSquare() < 100000
            && fitqual.ratioNumberHitsToExpected() > 0.5
            && ! fitqual.fitAbort());
}
Particle identification using $dE/dx$:

```cpp
daBoolean isKaon(const NavTrack& track) {
    const FAItem<DedxInfo>& dedxinfo = track.dedxInfo();
    return (dedxinfo.valid() && (*dedxinfo).valid() && fabs((*dedxinfo).kSigma()) < 3.0);
}
ndaBoolean isPion(const NavTrack& track) {
    const FAItem<DedxInfo>& dedxinfo = track.dedxInfo();
    return (dedxinfo.valid() && (*dedxinfo).valid() && fabs((*dedxinfo).piSigma()) < 3.0);
}
```

Requires $3\sigma$ consistency between the pion/kaon hypothesis and the observed $dE/dx$.

You should do checks that 1) the FAItem is pointing to something, and 2) the DedxInfo is itself valid.

(Incidentally, for $D^0 \to K^-\pi^+$ one should really use the RICH. This is just to show you how to use $dE/dx$.)
Extract objects we need

```cpp
FAItem<LabNet4Momentum> labMomentum;
extract(iFrame.record(Stream::kStartRun), labMomentum);

FATable< NavTrack > trackTable;
extract(iFrame.record(Stream::kEvent), trackTable);
```

Set up loop and apply cuts

```cpp
FATable< NavTrack >::const_iterator trackBegin = trackTable.begin();
FATable< NavTrack >::const_iterator trackEnd = trackTable.end();

if ((*labMomentum).e() > 3.7*k_GeV) {
    for (FATable< NavTrack >::const_iterator trackItr1 = trackBegin;
         trackItr1 != trackEnd ; ++trackItr1 ) {
        if (isGoodTrack(*trackItr1) && isKaon(*trackItr1)) {
            const TDKinematicFit& kineData1 = *(*trackItr1).kaonFit();
            for (FATable< NavTrack >::const_iterator trackItr2 = trackBegin;
                 trackItr2 != trackEnd ; ++trackItr2 ) {
                const TDKinematicFit& kineData2 = *(*trackItr2).pionFit();
                if (isGoodTrack(*trackItr2) && isPion(*trackItr2)
                     && kineData2.charge() == -kineData1.charge()) {
```

Require good tracks, opposite sign, $dE/dx$ hypotheses pass. We loop through the whole track list twice since first and second tracks are not equivalent (first is “kaon” and second is “pion”).
Compute $m_{BC}$ and $\Delta E$, and fill histogram:

```cpp
const HepLorentzVector& v1 = kineData1.lorentzMomentum();
const HepLorentzVector& v2 = kineData2.lorentzMomentum();
HepLorentzVector tot(v1+v2);
HepDouble totenergy = tot.e();
tot.setE((*labMomentum).e()/2);
m_kpi->fill(tot.m(), totenergy - tot.e());
```

Result:

Exercise for the audience: what are the vertical bands?
Required exercises

All the exercises can be done using only the $\psi(3770)$ data in data33, i.e.

```
eventstore in 20060214 dtag dataset data33 energy psi(3770)
```

You can use “physics qcd” instead of “dtag all”, but you want to use “dtag all” to debug your code.

- Reconstruct the decay $D^+ \rightarrow K^- \pi^+ \pi^+$.
- Reconstruct the decay $D^0 \rightarrow K^- \pi^+ \pi^0$. You can either make your own $\pi^0$'s or use those from NavPi0ToGG.
- Reconstruct the decay $D^0 \rightarrow K_S \pi^0$. You will need to use NavKs and NavPi0ToGG to get good enough resolution to see this mode. Notice that you don’t directly detect either of the $D$ decay products!
Optional exercises

The Particle Data Group’s *Review of Particle Physics* will be helpful.

- Reconstruct the decay $\phi \rightarrow K^- K^+$. The $\phi$ is produced in some decays of $D$’s, but you don’t need to find the $D$. Just look for the $\phi$ peak in the $K^- K^+$ invariant mass. (Use a $dE/dx$ cut. The $\phi$ is very narrow, so choose your histogram width and binning carefully. You might want to check where to look first …)

- Reconstruct $\eta \rightarrow \pi^+ \pi^- \pi^0$ and $\eta \rightarrow \gamma \gamma$, and compare the yield and resolution in the two modes.

- Plot the momentum distribution of kaons and pions in $D^0 \rightarrow K^- \pi^+$. Why is there a range of momenta? Can you predict what this range should be?

- Justify to yourself that the $\Delta E$ resolution should improve as the number of tracks in the $D$ decay increases.

- Find the mass distribution of $\pi^0$ candidates for different cuts on the minimum total energy of the $\pi^0$ (200 MeV, 400 MeV, ...). How does the relative size of signal and background vary with the minimum energy?

- Correct the $m_{BC}$ calculation for crossing angle. The *HepLorentzRotation* class may be helpful.