D Tagging in CLEO-c

Laura Fields - Cornell
(Alex Smith - University of Minnesota)

- What is D Tagging?
- Example DTag analysis Suez job
- D Skims
  - D Skim “Grades”
  - Creating DTag information: DTagProd
- Tagging both D Mesons: DDoubleTag
- Exercises

CLEO 101
June 16, 2005
What is D Tagging?

- As you’ve probably noticed, we reconstruct D’s A LOT!!

- To make all our lives a bit easier, a class “DTag” has been written. It is based on CDDecay, so that it has all of the functionality of DChain objects you learned about yesterday. It also has other fun stuff particularly useful when dealing with D decays:
  - beamConstrainedMass() : \( M_{bc} = \sqrt{E_{\text{beam}}^2 - p_D^2} \)
  - deltaE() : \( \Delta E = E_D - E_{\text{beam}} \)

- You Can Also Easily Access tracks on other (non-tagged) side of the event, and can use loops or DChain to play with these
What is DTagging?

- A lot of analyses at Cleo use DTag to reconstruct one D, and then look for some thing else on the other side. There are two options if you want to do this:
  - Use the Dskimmed Data and Monte Carlo
    - This is a subset of Data that contains only events where at least one D has been reconstructed
    - The DTag Objects are already in the Frame!
  - To run on Data that has not been Dskimmed, use the producer “DTagProd” to insert DTag objects into the frame. I won’t say much more about this here, but you can find an example script at ~asmith/CLEO101/DTag.tcl. This will be particularly useful if you want to generate your own Monte Carlo, which you will hopefully learn about next week.
What is DTagging?

• In other, somewhat rarer cases, you might want to fully reconstruct both D’s. In this case, you want to use “DDoubleTagProd” instead.

• Further Information:
  - The DTagging website:
    http://www.lns.cornell.edu/restricted/CLEO/analysis/cleocHadronic/dtag/
Getting Started Coding

• You can find a helpful example of a processor that uses DTags by using mkproc with the argument -dtag:

```
cd
cd src
mkproc -dtag DTagProc
``` 

• You should now find the code is the directory ~/src/DTagProc/Class

• Now let's go through the code that you'll find in this directory:
An Example Suez Processor using \textit{D} Tagging

Get track and shower lists

Select \textit{D}Tag decay modes—Add some others!

```cpp
// methods bound to streams

ActionBase::ActionResult
DTagProc::event( Frame& iFrame ) // anal3 equiv.
{
  report( DEBUG, kFacilityString ) << "here in event()" << endl;

  // *** Fetch track list
  FATable< NavTrack > trackTable;
  extract( iFrame.record( Stream::kEvent ), trackTable );

  // *** Fetch shower list
  FATable< NavShower > showerTable;
  extract( iFrame.record( Stream::kEvent ), showerTable );

  // *** Fetch beam energy
  FAItem<LabNet4Momentum> iLabNet4Momentum;
  extract(iFrame.record( Stream::kStartRun ), iLabNet4Momentum);

  // *** Create a list of requested modes.
  //
  // In this example, we only extract D0 \to K^- \pi^+ and
  // D^+ \to K^- \pi^+ \pi^+ (add whatever modes you want)
  //
  STL_VECTOR(DTag::DecayMode) vectorModes;
  vectorModes.push_back(DTag::kD0toKPi);
  vectorModes.push_back(DTag::kDptoKPiPi);

  FAItem<DTagList> iDTagList;
```
An Example Suez Processor using D Tagging

// *** Loop over decay modes. Add all tags of this type to a
// *** D tag list

for (DTagDecayModes::const_iterator itrMode = DTag::modes().begin();
itrMode != DTag::modes().end(); ++itrMode) {
DTag::DecayMode mode = (DTag::DecayMode)itrMode->first;
STL_VECTOR(DTag::DecayMode)::iterator itrVector =
_find(vectorModes.begin(), vectorModes.end(), mode);
if (itrVector == vectorModes.end()) continue;

// If you want to extract DTagList with a production tag,
// e.g. "NoPID", use the commented sentence instead:
//extract(iFrame.record(Stream::kEvent), iDTagList,
//itrMode->second.name(), "NoPID");
extract(iFrame.record(Stream::kEvent), iDTagList,
itrMode->second.name());

// *** Loop over list of D tags for this decay mode

for (DTagList::const_iterator itrDTag = iDTagList->particle_begin();
itrDTag != iDTagList->particle_end(); ++itrDTag) {
const DTag& id = (*itrDTag).particle();

// Loop over modes
Selecting a different production tag or “grade”

// Loop over DTag list for this mode
An Example Suez Processor using D Tagging

Get $\Delta E$, $M_{bc}$, and decay mode

Get "other-side" tracks and showers

Example loop over "other-side" tracks. Could also use DChain here to build you own particles

```cpp
/// *** Example of access to useful info
/// report(INFO, kFacilityString) << iD << endl;
mode = iD.decayMode();
double mBC = iD.beamConstrainedMass();
double delAE = iD.deltaE();

/// *** Get tracks and showers from the "other side" D
/// FACConstPtrTable<NavTrack> otherTracks;
FACConstPtrTable<NavShower> otherShowers;
iD.otherSideTracks(trackTable, otherTracks);
iD.otherSideShowers(showerTable, otherShowers);

/// *** Example loop over other-side tracks
/// report(INFO, kFacilityString) << "Number of other side tracks: "
   << otherTracks.size() << " out of "
   << trackTable.size() << endl;
for (FACConstPtrTable<NavTrack>::const_iterator
     iNT = otherTracks.begin(); iNT != otherTracks.end(); ++iNT) {
    report(INFO, kFacilityString) << "otherside track id: "
       << iNT->identifier() << endl;
}

/// *** Example loop over other-side showers
/// report(INFO, kFacilityString) << "Number of other side showers: "
   << otherShowers.size() << " out of "
   << showerTable.size() << endl;
for (FACConstPtrTable<NavShower>::const_iterator
     iNS = otherShowers.begin(); iNS != otherShowers.end(); ++iNS) {
    report(INFO, kFacilityString) << "otherside shower id: "
       << iNS->identifier() << endl;
}
An Example Suez Processor using **D** Tagging

```cpp
// *** Extract kinematic info of DTag. The DTag is a
// *** CDDecay, so you can also extract the usual kinematic
// *** variables.
//
// double charge = iD.charge();
// double mass = iD.mass();
// double energy = iD.energy();
// const HepVector3D& p3 = iD.momentum();
// HepLorentzVector p4 = iD.kinematicData().lorentzMomentum();
// Count nChildren = iD.numberChildren();

// *** Check for match to Monte Carlo particle
//
// if (iD.hasMatchedMCParticle()) {
//   const MCParticle& mcD = iD.matchedMCParticle();
// }

// *** You can also use DChain to build particles from the
// *** other-side track and shower lists, just as you did
// *** in earlier sessions.

//

} } return ActionBase::kPassed;
D Skims

- **D Skims are provided as a service to the user**
  - **A D Skim is a subset of the data containing only events with a D Tag**
    - Reduces the time required to rerun on all of the data or Monte Carlo samples
    - DTags already saved in the frame
  - Three levels of cuts applied to select different D Tag “grades”:
    - Default
    - LOOSE
    - No PID
- **Where can I find out which data and MC samples have been D Skimmed?**
  - [http://www.lns.cornell.edu/restricted/CLEO/analysis/cleocHadronic/dtag/](http://www.lns.cornell.edu/restricted/CLEO/analysis/cleocHadronic/dtag/)
Tagging Both D’s

• In some analyses, you may want to tag both D mesons
• This can be done by tagging one side and using DChain to reconstruct the event on the other side
• One can also use DDoubleTagProd to reconstruct both D mesons
• One can create an example using mkproc:

```bash
cd
cd src
mkproc --ddoubletag DDoubleTagProc
```

• Let’s have a look at the code produced by this...
Tagging Both D’s

Usage tag sets DTag grade, etc. (set in TCL script)

```c++
ActionBase::ActionResult
DDoubleTagProc::event( Frame& iFrame ) // anal3 equiv.
{
  report( DEBUG, kFacilityString ) << "here in event()" << endl;

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

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

  // Extract LabNet4Momentum
  FAItem<LabNet4Momentum> iLabNet4Momentum;
  extract(iFrame.record(Stream::kStartRun), iLabNet4Momentum);
  //report(INFO, kFacilityString)
  //  << "Lab net four momentum: " << *iLabNet4Momentum << endl;

  // For DDoubleTag
  for (SmallCount i = 0; i < DDoubleTag::kNumOfDDbarModes; ++i) {
    FAItem<DDoubleTagList> iDDDoubleTagListItem;
    extract(iFrame.record(Stream::kEvent), iDDDoubleTagListItem,
             DDoubleTag::kUsageTag[i].c_str());
    for (DDoubleTagList::const_iterator iDBarItr =
         iDDDoubleTagListItem->particle_begin();
         iDBarItr != iDDoubleTagListItem->particle_end();
         ++iDBarItr) {
```
Tagging Both D’s

Access info for D or Dbar

Access D or Dbar fit info

```cpp
const DDoubleTag& iDDbar = (*iDDBarItr).particle();
report(INFO, kFacilityString) << iDDbar << endl;

// Get D and Dbar
DTag iD = iDDbar.taggedD();
DTag iDbar = iDDbar.taggedDbar();

// DDbar features
// Angle between D and Dbar in rest frame of DDbar
double cosThetaFit = iDDbar.cosThetaFit();
// Chi2 of vertex constraint fit for D and Dbar individually
double chi2VertexD = iDDbar.chi2VertexD();
double chi2VertexDbar = iDDbar.chi2VertexDbar();
// Chi2 of energy constraint fit for D and Dbar individually
double chi2EnergyD = iDDbar.chi2EnergyD();
double chi2EnergyDbar = iDDbar.chi2EnergyDbar();
// Chi2 of mass constraint fit for D and Dbar individually
double chi2MassD = iDDbar.chi2MassD();
double chi2MassDbar = iDDbar.chi2MassDbar();
// Chi2 of vertex constraint fit for DDbar
double chi2Vertex = iDDbar.chi2VertexDDbar();
// Monte Carlo match
if (iDDbar.hasMatchedMCParticle()) {
    const MCParticle& mcDDbar = iDDbar.matchedMCParticle();
}
```
Tagging Both D's

Can access other-side track and shower info as in single DTags

```cpp
// To get other side tracks/showers
FAConstPtrTable<NavTrack> otherTracks;
FAConstPtrTable<NavShower> otherShowers;
IDDbar.otherSideTracks(trackTable, otherTracks);
IDDbar.otherSideShowers(showerTable, otherShowers);
report(INFO, kFacilityString) << "Number of other side tracks: "
  << otherTracks.size() << " out of "
  << trackTable.size() << endl;
report(INFO, kFacilityString) << "Number of other side showers: "
  << otherShowers.size() << " out of "
  << showerTable.size() << endl;

return ActionBase::kPassed;
```
Exercises

• Make and edit a DTag Processor to do the analysis shown on the next page:

```bash
cd
cd src
mkproc -dtag DTAGProc
cd DTAGProc/
cd Class
emacs DTAGProc.cc
```

• Run that code on DDbar Monte Carlo DSkim
• Use DSkim Data listed on the DTag Web page
  – http://www.lns.cornell.edu/restricted/CLEO/analysis/cleocHadronic/dtag/
  – Try Data Set 31, 32, or 33 DDBar
  – For Example,
    • Eventstore in 20050609 dtag dataset data31
    • Also see the sample tcl in the 'Test' subdirectory of your processors for help with your .tcl script
Exercise of DTag Usage: $D^0 \rightarrow K^-\pi^+$

- Reconstruct the D Tag side with DTag
  - Use several modes
    - $K\pi$, $K\pi\pi^0$, $K\pi\pi\pi$

- Reconstruct a $K^-\pi^+$ or $K^+\pi^-$ from "other-side" tracks using DChain

- Plot the $\Delta E$ and $M_{bc}$ of the "other side" D for each candidate

- Run the job on DDbar Data (see bottom of last page)
  - Does plot make sense?

- You can also try the same analysis using DDoubleTag