CBX 06-11
Peter Onyisi
Werner Sun
March 9, 2006

Developments in $D_{(s)}$-Tagging

Abstract

We describe a major revision of the D-Tagging code and provide examples to illustrate the new user interface.

1 Introduction

To coincide with the arrival of high-energy ($\sqrt{s} \sim 4$ GeV) CLEO-c data, we have implemented major upgrades to the D-Tagging code. We now support D_s^+ reconstruction, and the procedure for applying non-standard cuts at analysis time has been simplified. Also, we have eliminated storage of double tag candidates; users are now encouraged to form double tag lists by hand using the stored single tag lists. As much as possible, the original functionality of the code has been retained, and the user interface is backwards-compatible, so that only minimal changes to user analysis code are required.

2 Changes to `DTag` and `DTagProd`

2.1 New daughter selection criteria and changes to the output

In the new "version 2" of the D skims, several cuts have been tweaked:

- The K_S^0 mass cut window is now ± 30 MeV ("7.5σ") for both standard and loose skims (previously was ± 12 MeV for standard and ± 20 MeV for loose).
- For the loose skim only, the π^0 and η pull mass cut is now $\pm 6\sigma$ instead of $\pm 5\sigma$.
- Diphoton decay candidates (i.e. π^0, η, and $\eta' \to \gamma\gamma$) are now remade in the skimming process. These differ from the standard post-pass2 objects in that no E9/E25 cut is applied. These new objects are written out in the D skim. *Note that the π^0, η, and η' lists are now different in the eventstore* `physics` *and* `dtag` *grades!*

The D skim output has also changed in the following ways:

- The NoPID skim is no longer made.
- Double tags are not written out.
- There are now η' objects in the output.

These changes are discussed in more detail later in this note.

Access to the standard and loose skims has not changed; existing code and scripts will not have to be altered if they only use those two skims. Code that used the NoPID skim can still be made to work by using `DTagSelectionProd` as shown below.

2.2 `DTagSelectionProd`

We no longer write out the NoPID skim. Instead, we provide a new producer, `DTagSelectionProd`, that can create arbitrary subsets of the loose skim in a straightforward manner. The producer is configured with the production tag of the parent skim (usually `LOOSE`) and the usage and production tags for lists of the fundamental objects of which `DTag`s are composed. Whenever a request is made for a `DTagList` with the producer's production tag, `DTagSelectionProd` picks up the corresponding list of tags from the parent skim, and for each tag compares each of the daughters against the candidates in the paricle lists. If all the daughters are found in the lists, the tag is passed. The subset of the parent tags which pass are returned.

2.2.1 Recommended usage

Any user-specified cuts are made by producing particle candidate lists and placing them in the Frame. For changing the cut values of the standard DTag cuts, the easiest way to do this is to use the standard DTag daughter selectors (`TagD*`) with different production tags.

A new suez command, `setup_dtag_selection`, is available to make configuration of `DTagSelectionProd` more straightforward. It currently has three subcommands:

- `setup_dtag_selection general` with no options gives the loose skim, and is intended to tighten from the loose skim.
- `setup_dtag_selection standard` with no options tightens the loose skim to the equivalent of the standard skim, and is intended to loosen from the standard skim.
- `setup_dtag_selection nopid` removes particle ID requirements relative to the standard skim. An option is provided to tighten the K_S^0 mass cut.

All subcommands accept the `-prodtag` argument, which allows the user to change the production tag that the selected subset will appear as. The defaults are "`DSelection`" for `general` and `standard` and "`NoPID`" for `nopid`.

The `nopid` subcommand accepts the `-kshortsigma` argument, which allows the user to request a specific mass window for the K^0_S candidates ("1σ" = 4 MeV).

The `general` and `standard` subcommands take the `-parentskim` argument, which allows the user to change the skim of which a subset will be made. This can be used to tighten the standard skim, for example, or chain different instances of `DTagSelectionProd`.

The `general` and `standard` subcommands both take the following daughter selection arguments:

- `-pions`
- `-kaons`
- `-pi0s`
- `-kshorts`
- `-etas`
- `-ep2pipieta`
- `-ep2rhogam`

These arguments take the usage tag and production tag for the list you wish to use for that daughter type; both should be specified. If you do not use a particular argument, it will take the default for that subcommand type, which is the standard DTag list for `standard` and the parent skim's list (the loose skim) for `general`.

A user's processor can subset lists to the user's desire and inject those to the Frame, and those can be used by `DTagSelectionProd`; this is an advanced topic and the user is referred to the CLEO software pages for information on doing this.

An example tcl snippet follows:

```
# Tighten pi0 mass cut
# other cuts copied from setup_dtag_command.tcl
prod sel TagDPi0Prod production DSelection
param TagDPi0Prod@DSelection PullMass 4.0
param TagDPi0Prod@DSelection NumberSigmasMax 1000.
param TagDPi0Prod@DSelection RejectECPi0 false

# Get the loose skim except with our pi0s
# will be called DSelection_loose
setup_dtag_selection general -pi0s TagDPi0 DSelection \
  -prodtag DSelection_loose

# Get the standard skim except with our pi0s
# will be called DSelection_standard
setup_dtag_selection standard -pi0s TagDPi0 DSelection \
  -prodtag DSelection_standard

# Get a nopid skim, with new wide kshort cuts
# will be called NoPID
setup_dtag_selection nopid
```

2.3 Tagging at high energy

`DTag` and `DTagProd` were initially written for running at $E_{cm} \approx 3.77$ GeV, for D^0 and D^+ decays. The kinematic selections used, ΔE and m_{BC}, are optimized assuming the pair production of equally massive particles from the $\psi(3770)$.

High energy running introduces new complications. Above 3.94 GeV, the D_s can be produced, and its hadronic decays need to be tagged. This can be handled by adding new decay modes to the `DTag` object, and new production helpers to `DTagProd`. A new complication is the fact that a large fraction of D_s decays contain an η' (over 10% from initial indications [1]), a narrow particle which is not explicitly searched for by the original `DTag` code.

The other major new issue is the production of higher-mass charmed states — the D^{*+}, D^{*+}, and D_s^{*+}. The high energy scan has demonstrated that most of the open charm cross-section goes through the highest-mass pairs that are available at any given energy. Decays are thus frequently asymmetric (two unequal mass parents), and the final D or D_s is often boosted. The variable m_{BC} must be interpreted as primarily a momentum discriminator (albeit an almost energy-independent one) and ΔE is shifted arbitrarily far from zero.

2.3.1 New modes

We have added two new D^0 and 25 D_s decay modes to `DTag`, which are listed in Table 1. The new D^0 tags are for the $K_S^0\eta'$ decay for CP studies. The D_s tags include the critical normalizing mode $\phi\pi^+ \to K^-K^+\pi^+$, many Cabibbo-allowed modes (including η and η' decays), and also five Cabibbo-suppressed modes. They are accessed exactly like any of the old `DTag` modes.

`CDDecay` objects representing η' decays are now stored in the Frame, and are given as daughters of the `DTag` objects. We reconstruct the η' in $\pi^+\pi^-\eta$ and $\rho^0\gamma$; both are placed in `CDDecay` objects which have no direct way to distinguish them (you can ask for the η' daughter characteristics). The daughter order is $\pi^+\pi^-\eta$ for the first mode, and $\rho^0\gamma$ for the second, where the ρ^0 itself is a `CDDecay` with order $\pi^+\pi^-$. Should you wish to access the full list of η' decays output by the new `TagDEtaPrimeProd`, you

Table 1: New `DTag` modes

Mode #	Decay
124	$D^0 \to K^0_S\eta'\ (\eta' \to \pi^+\pi^-\eta,\ \eta \to \gamma\gamma)$
125	$D^0 \to K^0_S\eta'\ (\eta' \to \rho^0\gamma)$
400	$D_s^+ \to K^0_S K^+$
401	$D_s^+ \to K^- K^+ \pi^+$
402	$D_s^+ \to K^0_S K^+ \pi^0$
403	$D_s^+ \to K^0_S K^0_S \pi^+$
404	$D_s^+ \to K^- K^+ \pi^+ \pi^0$
405	$D_s^+ \to K^0_S K^+ \pi^+ \pi^-$
406	$D_s^+ \to K^0_S K^- \pi^+ \pi^+$
407	$D_s^+ \to K^- K^+ \pi^+ \pi^+ \pi^-$
420	$D_s^+ \to \pi^+ \pi^0$
421	$D_s^+ \to \pi^+ \pi^+ \pi^-$
422	$D_s^+ \to \pi^+ \pi^+ \pi^- \pi^0$
423	$D_s^+ \to \pi^+ \pi^+ \pi^+ \pi^- \pi^-$
424	$D_s^+ \to \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^0$
440	$D_s^+ \to \pi^+ \eta\ (\eta \to \gamma\gamma)$
441	$D_s^+ \to \pi^+ \pi^0 \eta\ (\eta \to \gamma\gamma)$
442	$D_s^+ \to \pi^+ \pi^+ \pi^- \eta\ (\eta \to \gamma\gamma)$
460	$D_s^+ \to \pi^+ \eta'\ (\eta' \to \pi^+\pi^-\eta,\ \eta \to \gamma\gamma)$
461	$D_s^+ \to \pi^+ \pi^0 \eta'\ (\eta' \to \pi^+\pi^-\eta,\ \eta \to \gamma\gamma)$
480	$D_s^+ \to \pi^+ \eta'\ (\eta' \to \rho^0\gamma)$
481	$D_s^+ \to \pi^+ \pi^0 \eta'\ (\eta' \to \rho^0\gamma)$
500	$D_s^+ \to K^0_S \pi^+$
501	$D_s^+ \to K^0_S \pi^+ \pi^0$
502	$D_s^+ \to K^+ \pi^- \pi^+$
503	$D_s^+ \to K^+ \pi^- \pi^+ \pi^0$
504	$D_s^+ \to K^- K^+ K^+$

can `extract` a `CDDecayList` from the frame, with usage tags `TagDEtaPrime2PiPiEta` and `TagDEtaPrime2RhoGam`.

The cuts used for η' candidates when skimming are listed in Table 2.

The new modes will always be available when running with the new `DTagProd` or new skims — in particular accessing D_s modes will not cause an error when running on $\psi(3770)$ data. However, with the default cuts, these lists will be empty.

Table 2: Cuts for η' candidates in skims

Default	Loose
$\eta' \to \pi^+\pi^-\eta,\ \eta \to \gamma\gamma$	
Pions in **default** `TagDPion`	Pions in **loose** `TagDPion`
Eta in **default** `TagDEta`	Eta in **loose** `TagDEta`
$0.9478 < m_{\eta'}$ (GeV) < 0.9678	$0.9328 < m_{\eta'}$ (GeV) < 0.9828
$\eta' \to \rho^0\gamma$	
Pions in **default** `TagDPion`	Pions in **loose** `TagDPion`
$0.5 < m_{\pi^+\pi^-}$ (GeV) < 1.0	$0 < m_{\pi^+\pi^-}$ (GeV) < 1.05
30 MeV $< E_\gamma <$ 2 GeV	
Shower has no hot crystals	
Shower has no track match	
Shower in good barrel or endcap	No detector restriction
$0.92 < m_{\eta'}$ (GeV) < 0.995	$0.865 < m_{\eta'}$ (GeV) < 1.05

2.3.2 New cuts at high energy

As mentioned earlier, the m_{BC} and ΔE variables lose the normal connection to mass and consistency with the event total energy when there are intermediate higher-mass charmed resonances. At the $\psi(3770)$, these two variables are essentially momentum and energy cuts, so we can replace them as necessary at higher energy by other variables that perform the same function.

We have decided to use m_{BC} and Δm_{inv} as the cut variables for high energy running. The normal definition of m_{BC} is used, and $\Delta m_{inv} \equiv m_{inv} - m_D$, for each species of D. The invariant mass cut is a "moral equivalent" of an energy cut, and backgrounds are more easily interpreted and removed by using such a physical variable. The m_{BC} variable is still useful as a proxy for momentum, since it is much less sensitive to E_{cm} than the raw momentum is. In particular, a fixed m_{BC} cut below the mass of the relevant D meson will be fully efficient at all energies, simplifying scripts.

The new Δm_{inv} cut is set at 85 MeV by default, to prevent D_s^+ candidates from showing up as D^+ candidates or vice versa.

For reference, here is the formula for the m_{BC} value reconstructed in the process $e^+e^- \to XY$:

$$m_{BC} = \sqrt{\frac{1}{2}\left[m_X^2 + m_Y^2 - \frac{(m_X^2 - m_Y^2)^2}{2E_{cm}^2}\right]}$$

The values for various processes involving starred mesons at 4170 MeV are listed in Table 3.

`DTagProd` can be manually switched beweeen using old-style or high-energy cuts with a parameter. By default, it chooses mode automatically, with the threshold being $E_{cm} = 3.8$ GeV. Use the command

```
DTagProd TaggingMode help
```

for more information.

2.3.3 Selectors for ϕ, ρ^+, and K^{*0}

We have provided selectors to enable easy detection of certain resonances commonly seen in D_s decays. These selectors take a `DTag` as input, and merely check if the tag's daughters pass mass cuts (which can be set in the constructor).

```
#include "DChain/List/Template/DCDecayList.cc"
#include "DTag/DTagList.h"
#include "DTag/PhiSelector.h"
#include "DTag/NeutralKstarSelector.h"

using DTagUtilities::PhiSelector;
using DTagUtilities::NeutralKstarSelector;

...

FAItem<DTagList> kkpiTags;
extract( iFrame.record( Stream::kEvent ), kkpiTags, "Ds2K-K+Pi+" );

// Take default phi mass window
PhiSelector phisel;
DTagList phipiTags(phisel);
phipiTags = *kkpiTags;

// Setup our selector with non-default mass window
NeutralKstarSelector kstarsel(0.83, 0.97);
DTagList kstarkTags(kstarsel);
kstarkTags = *kkpiTags;

for (DTagList::iterator iTag = phipiTags.particle_begin();
```

Table 3: Central values of m_{BC} (in GeV) for different charm production modes. Boosts from decays of D^*, D_s^* smear the distribution.

$D^0\overline{D}^{*0}$	D^+D^{*-}	D^*D^*	$D_s^+D_s^{*-}$
1.9358	1.9398	~ 2.01	2.0406

```
      iTag != phipiTags.particle_end();
      ++iTag) {
      // Iterate over tags that passed phi selection
      ...
}
```

The default cuts and the modes the selectors accept are:

- `PhiSelector`: 1.0095 GeV $< m_{K^+K^-} <$ 1.0295 GeV; any `DTag` mode with opposite sign charged kaons
- `NeutralKstarSelector`: 0.8211 GeV $< m_{K^+\pi^-} <$ 0.9711 GeV; $D_s \to K^-K^+\pi^+$ only
- `ChargedRhoSelector`: 0.6211 GeV $< m_{\pi^+\pi^0} <$ 0.9211 GeV ; $D_s \to K^-K^+\pi^+\pi^0$, $\pi^+\pi^0\eta$, $\pi^+\pi^0\eta'(\eta' \to \pi^+\pi^-\eta)$[1]

3 Changes to `DDoubleTag` and `DDoubleTagProd`

For D-skims made before March 2006 and software releases prior to `20060224_FULL`, `DDoubleTagList` objects, which are lists of `DDoubleTag` objects, were constructed by `DDoubleTagProd` during D-skimming and were written out to EventStore. Based on observations of users' analysis code, we believe that `DDoubleTagProd` was written in such a way that its output `DDoubleTagList` objects are unnecessarily complicated and inefficient for the most common use case. Furthermore, when one needed to rerun `DDoubleTagProd` at analysis time, many CPU cycles were wasted on a series of seven kinematic fits whose results were rarely used.

The primary limitation of `DDoubleTagProd` is that its output is not ready-to-use. The user may only extract the following four lists of candidates:

- Self-conjugate (SC) $D^0\bar{D}^0$, which includes modes like $K^-\pi^+/K^+\pi^-$, $K^-\pi^+\pi^+\pi^-/K^+\pi^-$, $K^0_S\pi^0/\pi^+\pi^-\pi^0$, and $K^0_SK^+\pi^-/K^0_SK^-\pi^+$.
- Non-self-conjugate (NS) $D^0\bar{D}^0$, which includes modes like $K^-\pi^+/K^-\pi^+$, $K^-\pi^+/K^0_S\pi^0$, $K^0_SK^+\pi^-/K^0_S\pi^0$, and $K^0_SK^+\pi^-/K^0_SK^+\pi^-$.
- D^+D^-.
- $D^+_sD^-_s$.

These lists typically contain many more modes than the user is interested in. For instance, if one desires only $K^+\pi^-/K^-\pi^+$ double tags, then one extracts the SC $D^0\bar{D}^0$ list and loops through all the candidates to pare away the vast majority that

[1]$\pi^+\pi^0\eta'(\eta' \to \rho\gamma)$ will be available in a release after 20060224_FULL

come from extraneous modes. Very rarely does one ask for *all* SC $D^0\bar{D}^0$ double tag modes at once. A secondary limitation of `DDoubleTagProd` is its lack of flexibility; one cannot use `DDoubleTagProd` to form double tags with different cuts on the D and the $\bar{D}$.

Therefore, we will no longer run `DDoubleTagProd` nor store the `DDoubleTagList` objects as part of D-skimming. Instead, we recommend that `DDoubleTagList` objects be generated at analysis time via `DChain`, using as input the `DTagList` objects (lists of single tags) extracted from EventStore. The `DDoubleTag` code has been modified so that kinematic fitting will be performed only on demand. In this way, we retain the functionality of the `DDoubleTag` class while increasing flexibility and ease of use without affecting job speed too adversely. One disadvantage of this plan is the possible loss of persistence (not storing double tags means they might conceivably change the next time one regenerates the lists). However, this risk is non-existant if no double tag cuts are applied because the lists of daughter single tags *are* stored. Below, we discuss these changes in more detail and provide code examples to illustrate our recommendations.

3.1 Code Changes

The changes to the `DDoubleTag` and `DDoubleTagProd` packages involve shifting the kinematic fitting from the Producer to the `DDoubleTag` class itself, thus allowing the fits to be performed on demand, rather than at the time the `DDoubleTag`s are created. In addition, one may still run `DDoubleTagProd` (so that `DDoubleTagList`s are made available through the `Frame`). However, the Producer has been modified such that the extracted `DDoubleTag`s will not have their kinematic constraints pre-evaluated. One consequence of this change is that, should one choose to write out the `DDoubleTagList`s, the fit results cannot be stored because these data were not present at creation time. Thus, even with stored double tags, the kinematic fits (if desired) must be performed at analysis time. Any fit results that are currently in EventStore (such as for D-skims made before March 2006) will be ignored by the new version of `DDoubleTagStorageHelper`.

The files `DDoubleTag/DDoubleTagCutCriteria.h (.cc)` and `DDoubleTagProd/DDoubleTagSelector.h (.cc)` used to contain the kinematic fitting code used by the Producer. Since these files are no longer needed, they have been removed from their corresponding packages.

Because the kinematic fits require information external to the `DDoubleTag` class, the corresponding accessor functions in `DDoubleTag`, which previously required no arguments, now have a different signature:

```
double cosThetaFit( const MagneticField* aMagField,
                    const LabNet4Momentum* a4Momentum ) ;
double chi2VertexD( const MagneticField* aMagField,
                    const LabNet4Momentum* a4Momentum ) ;
```

```
double chi2VertexDbar( const MagneticField* aMagField,
                       const LabNet4Momentum* a4Momentum ) ;
double chi2EnergyD( const MagneticField* aMagField,
                    const LabNet4Momentum* a4Momentum ) ;
double chi2EnergyDbar( const MagneticField* aMagField,
                       const LabNet4Momentum* a4Momentum ) ;
double chi2MassD( const MagneticField* aMagField,
                  const LabNet4Momentum* a4Momentum ) ;
double chi2MassDbar( const MagneticField* aMagField,
                     const LabNet4Momentum* a4Momentum ) ;
double chi2VertexDDbar( const MagneticField* aMagField,
                        const LabNet4Momentum* a4Momentum ) ;
```

The `chi2XXX(...)` function returns the χ^2 values of various kinematic constraints, and `cosThetaFit(...)` returns the cosine of the angle between D and $\bar{D}$ in the center-of-mass frame after the above constraints have been applied. Note that the constrained 4-momenta of the D and $\bar{D}$ are not saved; just the angle between them. The first time any one of these functions is called, `DDoubleTag` performs the seven kinematic fits and caches the results internally (to save time on subsequent calls). We have also introduced two new functions that compute angular information without performing these (time-consuming) fits.

```
// Cosine of angle between D and Dbar in center-of-mass frame,
// without any vertex, energy, or mass constraints.
double cosThetaCoM( const LabNet4Momentum* a4Momentum ) const ;

// Cosine of angle between D and Dbar in the lab frame, without
// any vertex, energy, or mass constraints.
double cosThetaLab() const ;
```

3.2 Recommended Usage

The new recommended usage is to form double tag lists by hand, which can be done in a few short lines of code in one's Processor. The following example is for $K\pi/K\pi$ double tags:

```
#include "DChain/Iterator/MuteWholeItr.h"
#include "DChain/List/CombinatoricList.h"
#include "DTag/DTagList.h"
#include "DDoubleTag/DDoubleTagList.h"

FAItem< DTagList > kpiList ;
extract( aFrame.record( Stream::kEvent ), kpiList, "D02K-Pi+" ) ;
```

```
// K-Pi+ vs. K+Pi- (self-conjugate final state)
DDoubleTagList rightSignDoubleTags ;
rightSignDoubleTags = ( *kpiList ) * ( kpiList->bar() ) ;

// K-Pi+ vs. K-Pi+  AND  K+Pi- vs. K+Pi-
DDoubleTagList wrongSignDoubleTags ;
wrongSignDoubleTags = ( *kpiList ) * ( *kpiList ) ;
```

Note that the wrong-sign list automatically contains both charge-conjugate final states. One should *not* write:

```
// BAD!!!  DOUBLE-COUNTING!!!
wrongSignDoubleTags = ( *kpiList ) * ( *kpiList ) +
                      ( *kpiList->bar() ) * ( *kpiList->bar() ) ;
```

as this would result in double-counting. This caveat also applies to double tag modes where at least one side is self-conjugate because, as with any self-conjugate list in `DChain`, the `bar()` version is the same as the original list.

To combine single tag lists with different cuts (*e.g.*, default D vs. LOOSE $\bar{D}$) one writes:

```
FAItem< DTagList > looseKpiList ;
extract( aFrame.record( Stream::kEvent ), looseKpiList,
   "D02K-Pi+", "LOOSE" ) ;

// Default K-Pi+ (= kpiList from above) vs. LOOSE K+Pi-
DDoubleTagList mixedDoubleTags ;
mixedoubleTags = ( *kpiList ) * ( looseKpiList->bar() ) ;
```

To perform the kinematic fits for each candidate, one needs to have extracted `MagField` and `LabNet4Momentum`:

```
#include "LabNet4Momentum/LabNet4Momentum.h"
#include "MagField/MagneticField.h"

FAItem< MagneticField > magField ;
extract( aFrame.record( Stream::kStartRun ), magField ) ;

FAItem< LabNet4Momentum > net4P ;
extract( aFrame.record( Stream::kStartRun ), net4P ) ;

DDoubleTagList::iterator tagEnd = doubleTagList.particle_end() ;
DDoubleTagList::iterator tagItr = doubleTagList.particle_begin() ;
```

```
for( ; tagItr != tagEnd ; ++tagItr )
{
   // No kinematic fitting for these two functions.
   double costhLab = ( *tagItr ).particle().cosThetaLab() ;
   double costhCoM = ( *tagItr ).particle().cosThetaCoM( &*net4P ) ;

   // A non-const copy.
   DDoubleTag ddtag = ( *tagItr ).particle() ;

   // Calling cosThetaFit(...) performs the seven kinematic fits.
   double costhFit = ddtag.cosThetaFit( &*magField, &*net4P ) ;
}
```

As long as one uses the `LabNet4Momentum` and `MagneticField` objects that were written out to EventStore (*i.e.*, as long as `LabNet4MomentumFromCrossingAngleProd` and `MagFieldProd` are not loaded in one's `suez` job), these fit results will not change from day to day.

Finally, one can still run `DDoubleTagProd` as before (`prod sel DDoubleTagProd`) to produce the multi-mode lists of SC $D^0\bar{D}^0$, NS $D^0\bar{D}^0$, D^+D^-, and $D_s^+D_s^-$ double tags:

```
FAItem< DDoubleTagList > scList ;
extract( aFrame.record( Stream::kEvent ), scList, "D0D0barSC" ) ;

DDoubleTagList::const_iterator tagEnd = scList->particle_end() ;
DDoubleTagList::const_iterator tagItr = scList->particle_begin() ;
for( ; tagItr != tagEnd ; ++tagItr )
{
   // A non-const copy.
   DDoubleTag ddtag = ( *tagItr ).particle() ;
   double costhFit = ddtag.cosThetaFit( &*magField, &*net4P ) ;
}
```

Because the kinematic fits are now performed only on demand, one can save processing time by only constraining the candidates of interest.

For more examples, one can issue the command

```
mkproc -ddoubletag DDoubleTagTestProd,
```

which generates a working skeleton Processor that exercises the `DDoubleTag` class.

References

[1] R. Sia and S. Stone, CBX 06-04