CBX 07-30
June 13, 2005
Roy A. Briere
Guang-Pei Chen
(for the CLEO-c DTag Group)
Title updated on Oct. 19, 2007,
by Pete Zweber

CLEO-c D Tagging: "2005" Version (aka Version 1 DTags)

Abstract

During the more useful phases of its history, this document existed as a working draft in the DTag group web page. It is being posted now largely as a historical reference.

This self-contained document describes the D skims made for CLEO-c of the Fall, 2004 re-pass2 of db31-34 and the first pass2 of db35-37 in 2005. The older April 2004 and subsequent "DFix" skims used for ICHEP 2004 preliminary results and subsequent publications are documented elsewhere (CBX04-32).

Significant changes included the expansion to 55 modes, the addition of two easily-selected looser skims, and some PID cut changes (RICH momentum threshold increased).

Future work (nearly complete) involves preparation for the "D_s energies", where $D^*\bar{D}, D\bar{D}^*, D_s^+ D_s^-$, etc. are produced. In addition, cut changes and an updated means of dealing with loose skims and double-tags are in the works. These will be documented elsewhere.

Contents

List of Tables

List of Figures

1 Introduction

The goal is to document the hopefully more stable CLEO-c D skims generated Winter 2004-5 from db31-37. This represents the second wave of D skims: the first concerned the pilot-run data of about 56 pb^{-1} from db31-34. The current skims contain all of db31-37, comprising about 275 pb^{-1} and will be used for summer 2005 results and beyond. Running conditions are essentially identical, expect for increasing the number of wigglers from 6 to 12 before db35 was taken.

Up-to-date information is generally available on the D Tag web page:
`https://www.lns.cornell.edu/restricted/CLEO/analysis/cleocHadronic/dtag/`

I have tried to infuse this note with the locations of lots of pieces of source code so you can see how things are done for yourselves, and find the typos and outright errors that have no doubt crept into this document. As always, it is wise to talk to experts to avoid studying in detail source code or scripts that are not in fact in use due to changes or errors on my part.

1.1 Old Skims

A previous note, CBX04-32, describes the following skims. The first was made around April, 2004 from the then recently finished pass2 of data31-34. The second was made in late June, 2004 after a re-pass2 of a subset of events designed to fix a bug with the tracking bunch-finder.

We will not discuss these any further here, except to occasionally highlight changes made for the current skims.

2 Getting Started Fast: mkproc Skeletons

You can use `mkproc` to create sample processor to use `DTag` and/or `DDoubleTag`.

```
mkproc -dtag DTagProc
mkproc -ddoubletag DDoubleTagProc
mkproc -dtag -ddoubletag BothProc
```

(Try `mkproc -list` to show all currently available skeletons.)

The processor show how to:
• Extract the `LabNet4Momentum`
• Extract `DTag` (You can specify the modes you're interested in.)
• Extract `DDoubleTag`
The code also accesses some of the `DTag` and `DDoubleTag` data needed for analysis, such as ΔE, M_{bc}, unused tracks and showers, ...

In the skeleton processor's `Test` directory, there is an example `load*.tcl` script to start with. There may also be interesting scripts on the D Tag web page mentioned above.

3 New 55-mode *D* Skims

As of this writing, these have not quite happened yet...we are waiting to get post-pass2 constants and related issues more integrated into EventStore.

3.1 Pass2 Code and Constants

The Pass2 release is `WhoKnows_04`. ***

The constants tag for the pass2 is `WhoKnows2` ***

3.2 DTag Skims

Public D skimming for datasets db31-37 (including the first D skim of db35,36,37) was done with a "new" frozen code release so that things are fully reproducible. The release used is `20050316_FULL` .
In particular, the `TagD*Prod` selectors containing important daughter particle cuts are preserved in this release.

To re-produce a D skim, most likely to skim private signal Monte Carlo samples, see directions in the Monte Carlo section below.

The following is retained for completeness; please see the paragraph above re: how to do your own skimming. The scripts that the official skimmer uses to make a D skim are collected in the package:
`/nfs/cleo3/Offline/scripts/DSkimPackage/`
There is a very nice (perhaps now a bit dated?) description in the file `A_ReadMe_File.txt` in that directory. This area is independent of code release, however. Therefore, the most critical `.tcl` scripts are preserved in `$C3_SCRIPTS/DTagScripts/` which *is* release-dependent.

In practice, the important scripts, with cut parameters and other important items, are kept in the release-dependent `$C3_SCRIPTS/DTagScripts/` area. The `DSkimPackage` contains scripts which control the mechanics of job submission. These often need to be tweaked by the skimmer; sometimes local copies in the `dskim` account are used.

3.3 Overview of Recent Changes

One major change is the increase from 24 to 55 total $D^{)}$ and D^{+} decay modes.

Some cuts are changed relative to the most recent public (DFix) skims. The momentum point above which the RICH might be used is changed fro 550 MeV to 700 MeV.

Finally, three production tags are now available:

- Standard default skim (most like previous public skims)
- "NoPID" skims
- "LOOSE" skims

3.4 Monte Carlo

Generic public Monte Carlo of several types has been generated and skimmed for D Tags already (See "MC Issues" below). Directions are linked from the main DTag web page. Currently, instructions are:

```
Guide to dskim your own mc:
---------------------------

1) Copy ~dskim/skimDSignalMC.csh in your area.

2) Define your output directory, file name and tcl script in
   skimDSignalMC.csh. You should use the official dskim releases.
   The latest dskim release 20050316_FULL.

#your output directory
setenv OUTPUTDIR /home/batbold/
#your pds file name
setenv FILENAME yourpdsfilename
#your tcl script
setenv FILELIST /home/batbold/yourtcl.tcl
```

Note that "latest" re: dskim release as mentioned above is not really the issue; the point is that `20050316_FULL` was used for public skims; your signal MC skims should match this. When new skims are made in the future, use the library that matches them, etc.

In the future, fancier "command.tcl" files may be provided to facilitate this process.

4 DTag Cuts

Many cuts are controlled with parameters. Below, we give the parameters, default values, and values explicitly provided (sometimes same as defaults) in parameter statements for each of the three production tags.

These are, conceptually,:

- Standard – same general idea as past skims
- noPID: standard, except no particle ID cuts
- Loose: even looser tracking, shower, etc. cuts

See `$C3_SCRIPTS/DTagScripts/DTagDSkim.tcl` for the creation of these production tags with altered cut parameters.

4.1 Skim Types: Default, NoPID, LOOSE

There are three different skim types; the detailed cuts for each are tabulated below. Briefly, they are a "Default" version, a "NoPID" version with neither RICH or dE/dx not used; and a "LOOSE" version with NoPID plus looser tracking, π^0, and K_S cuts.

The format of extract calls is demonstrated in the skeleton processor:

```
// 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());
```

The example mentions "NoPID", but "LOOSE" may be used as well.

4.2 Software

The relevant software which creates the D tag skims, and supplies some utilities for analysis resides in these libraries:
`DTag`; `DTagProd`
plus the selector utility libraries for daughters:
`TagDKShortProd`, `TagDKaonProd`, `TagDPi0Prod`, `TagDPionProd`, `TagDEtaProd` .

Particularly important are the routines which apply selection cuts, defining the $K^{\pm}, \pi^{\pm}, \pi^0$, and K_S candidates from which D are reconstructed:
`TagD*Prod/TagD*Prod/TD*Selector.h`
`TagD*Prod/Class/TD*Selector.cc`
`TagD*Prod/Class/TagD*Prod.cc`
Here, `* = Kaon, Pion, Pi0, KShort, Eta`. In addition, the charged Kaon and Pion selection use routines:

`TagDKaonProd/Class/TDKaonCut.cc`
and similarly for Kaon → Pion.

The `Selector.h` routines are largely uninteresting re: cuts, the `Prod.cc` routines (or `TD*Cut.cc`) have constructors with parameter default values and register the proxies to make them available with usage tags, and the `Selector.cc` actually apply the criteria.

What follows is an attempt to summarize these cut criteria. Each section also lists any parameter setting from the `.tcl` script which may over-ride default settings.

4.3 Event Selection

For the final pass2 of DB31,32,33, output goes to .pds files. We run on all events processed by pass2, which processes all events saved at data collection time.

For the cuts below, the `Selector` code has many switches and possible cuts; we list the ones employed and also mention some of the interesting ones that are available but not used at this time.

4.4 Cut Overview

A summary of cuts is provided in Tables 1, 2, 3, and 4.

A blank entry means the default cut is not changed for a given skim. One should distiguish "default" cuts, meaning defaults in source code that may be changed in scripts, from what *might* be called the "default" skim. To avoid confusion, we call the usually-used skim "Std" (standard) in the tables. The point is that scripts may over-ride default cuts, and this "Std" skim may not have "default" cuts...while this is not the case currently, it could occur in principle.

4.5 Track Quality

Track quality and PID are done together in the $K^{\pm}$ and $\pi^{\pm}$ selectors. We separate them here for only for ease of presentation.

The default skim requires:
- $0.050 \leq p \leq 2.0$ GeV
- $|db| \leq 0.005$ m
- $|z0| \leq 0.050$ m
- $\chi^2 \leq 100000$
- $hitfrac \geq 0.5$
- $|\cot\theta| \leq 2.53$ (same as $|\cos\theta| \leq 0.930$)

The hit fraction cut is now calculated with a member function:

```
*     // Hit fraction cut
     float frac = piQuality->numberHitsExpected() > 0?
       piQuality->ratioNumberHitsToExpected() : 0.0;
```

Table 1: Charged Pion and Kaon cut parameters in Pion and Kaon selector code.

Parameter	Default	Std	NoPID	LOOSE	Comment
UseDBCD	true				db or d0
SoftTrkPMax	0.0			0.120	low p? → next 7 cuts off!
pMin	0.05				
pMax	2.0				
dxcdWindow	0.005			0.020	
z0cdWindow	0.05			0.200	
chi2Max	100000.0				
cotThetaMax	2.53				
HitFractionMin	0.5			0	
UseTrackMan	false				
UsePID	true		false	false	
DedxSigmaMax	3.0				
RichValidMomMin	0.70				
RichValidCotMax	1.0/0.75				i.e., 4/3
RichNumGammaMin	3				
CombinedDelLogMax	0.0				

and similarly for kaons. (apparently this function needs external divide-by-zero protection? There is indeed no internal protection in `TrackRoot/TrackRoot/TRTrackFitQuality.h`)

Note that this variable was formerly calculated as:
`piQuality().numberHits()/piQuality().numberHitsExpected()`
and similarly for kaons.

We have now switched from $d0$ to db for all skims; a flag is provided to toggle back to $d0$ if desired for study.

We do not use "zb", but remain with $z0$. The mean z *by run* is quite close to zero, especially compared to the physical luminous region of Gaussian width about 1 cm, which dominates over the resolution as well. This makes the difference less relevant than for db, where motions can be and have been large compared to resolution. One could do better in principle by cutting on $z0$ minus an *event* vertex z, but one would have to be very careful not to generate any bad event vertex fits...

The "TrackMan" package is not used.

4.6 Particle ID

Note that the PID momentum above which the RICH is used has been increased from 550 MeV to 700 MeV.

To use the RICH, we require:
• that RICH info is valid
• that both π, K hypotheses analyzed

Table 2: Pi0 and Eta cut parameters; first two are in PhotonDecaysProd; others in Pi0 and Eta selectors.

Parameter	Default	Std	NoPID	Loose	Comment
EnergyMin	***				
E925Scheme	***				
PullMass	3.			5	
NumberSigmasMax	1000.			1000	
UnconMassMax	1000.				
UnconMassMin	0.0				
ChisqMax	10000.				
EMinGB	0.0				
EMinNGB	0.0				
RejectECPi0	true			false	means *both* γ's in EndCap
UseE9o25HiSh	false				
UseE9o25LoSh	false				
UseSplitOff	false				

Let's call the 'and' of these two criteria "RichDone"

We combine the RICH and dE/dx for Kaon ID as:
$\mathcal{L} = \sigma_\pi^2 - \sigma_K^2 + L_\pi - L_K$
Requirements for charged pion ID are the same, with $\pi \leftrightarrow K$ everywhere.
The default skim requires:
• 3σ dE/dx consistency (if dE/dx valid)
• Combined log-likelihood $\mathcal{L} \geq 0$
• $n_\gamma > 2$ for the kaon hypothesis
• If no useful information, we use the track

We use dE/dx only in $\mathcal{L}$ and omit the n_γ cut if:
• RichDone is false, or $p < 0.70$ GeV, or $|\cos\theta| > 0.8$
The extra cut on $\cos\theta$ avoids an inefficiency near the edge of the RICH for kaons (present in the April 2004 skim). Near threshold, the number of photons may be small; this, together with being at the edge, resulted in a large fraction (most) of the kaons failing the RICH ID cut for a narrow range of $\cos\theta$ just past 0.8.

There is currently no cut on the number of dE/dx hits.

(Note: the logic here is arranged a bit different than the actual code: positive cut logic, and the 'valid' and 'done' are combined. The net effect is the same. If in doubt, read the code!)

Table 3: KShort cut parameters in KShort selector code.

Parameter	Default	Std	NoPID	Loose	Comment
ChisqMax	1000.				
NumberMassSigmaCut	4.5			5	
ProbMin	1000.				
Flv0	1000.				
Fvsgnf	1000.				
Nhtvtx1	1000				
Nhtotx1	1000				
Nhtvtx2	1000				
Nhtotx2	1000				
Rbmtx	1000.				
Rmstx	1000.				
Drmstx	1000.				
Chvdau1	1000.				
Chvdau2	1000.				
CleanV0	false				
RareB	false				
MassMax	1000.				
MassMin	0.0				

Table 4: Miscellaneous cut parameters; global D cuts.

Parameter	Default	Std	NoPID	Loose	Comment
BCM_Cut	1.83				
DE_Cut	0.1				

4.7 Neutral Pion Selection

We use the default setting of `PhotonDecaysProd` for all current skims. Parameters are provided to change the defauls, such as `EnergyMin` and `E925Scheme`, but these are unused now[1] See `www.lns.cornell.edu/~cleo3/current/src/PhotonDecays/Doc/PhotonDecays.html` for more information on this processor.
The default skim requires:
- $|pi0().pullMass()| \leq 3$
- $|pi0().numberSigma()| \leq 1000$
- $0 \leq pi0().unconMass() \leq 1000$ GeV
- $\chi^2 \leq 10000$

Here are things sometimes done in CLEO (external to π^0 finder) which we *do not* do:

[1]Note that they *were* changed for the April 2004, but not the June, 2004 "DFix", skims.

• We do *not* look at the number of endcap photons.
• There are no further energy cuts based on photon location.
• There are no split-off cuts.

Note that since we write out the found π^0's, the `***` described below makes sure that `PhotonDecaysProd`, usually run at analysis time, is deselected.

4.8 Eta Selection

Cuts are the same as the π^0 cuts, except for the nominal mass.

4.9 K_S Selection

See Table 3 for a summary of K_S cut parameters and values.
The only cuts in addition to standard Vee-finding are • Daughter π's must have `TRTrackFitQuality->fitA`
false.
• 3σ mass cut with fixed $\sigma = 0.004$ MeV, relative to $M_{KSs} = 0.4977$ GeV

Note that the 0.004 MeV constant σ is hard-wired still in `TagDKShortProd/Class/TDKShortSelector.c`
The *number* of sigma for the cut is a parameter. In principle, this still does allow for any symmetric cut window.

There are many other cuts "turned off" by setting loose cut values; see Table 3.

The popular "Clean V0" cuts are also *not* applied by default, but may be selected with a parameter. In particular, the default cuts are loose, such that prompt $\pi^+\pi^-$ pairs with the correct mass may fake K_S's. This is mostly a problem for a Cabibbo-allowed mode feeding into a Cabibbo-suppressed one since the accepted $\pi\pi$ mass window is relatively narrow; see Section 9.4.

4.10 D Selection

In addition to selecting `DTagProd`, one must specify the names of the Usage Tags from the selector Prod's that supply the list of candidate π, K, π^0, K_S, η:

```
prod sel DTagProd
param DTagProd Kaons "TagDKaon"
param DTagProd Pions "TagDPion"
param DTagProd Pi0s  "TagDPi0"
param DTagProd Pi0s  "TagDEta"
param DTagProd Kshorts "TagDKShort"
```

Given a set of daughters which can combine to form a given D mode (without re-use of tracks or showers: this is enforced by CDDecay machinery), there are only two additional cuts:
• $|\Delta E| \equiv E_{cand} - E_{bm} < 0.1$ GeV
• $M_{bc} \equiv \sqrt{E_{bm}^2 - p_{cand}^2} > 1.83$
Note that M_{bc} is corrected for the crossing angle effect.

4.11 More on M_{bc} Cut

In the latest scripts, the ΔE and M_{bc} cuts are done with parameters (passed as environment variables). See `$C3_SCRIPTS/DTagScripts/DTagDSkim.tcl` . For M_{bc} the lower limit is shifted for below-resonance running; this is done in: [2] `create_submissions.awk`,
which lives in `/nfs/cleo3/Offline/scripts/DSkimPackage/scripts/` .
The cut is lowered to $M_{bc} > 1.78$ if $E_{bm} < 1.86$. The value of M_{bc} written out is *not* affected. The endpoint of M_{bc} is clearly E_{bm} for any run. The on-resonance data endpoints cluster around some value, call it E_0. To use continuum data as a background shape for on-resonance data, we can either *shift*: $M_{bc} \longrightarrow M_{bc} + (E_0 - E_{bm})$ or *stretch*: $M_{bc} \longrightarrow M_{bc} \times (E_0/E_{bm}) = M_{bc} + (E_0 - E_{bm})(M_{bc}/E_{bm})$. The difference is small. And the amount of continuum background is small for $D\bar{D}$ compared to, say, rare B decays.

Also, beware of ψ' resonance data, which is not normal continuum, but is also lower in energy than the $\psi(3770)$. The large effective cross-section leads to lots of combinatorics for D tag candidates per unit luminosity. You may want to skip these to save ntuple space.

4.12 $K^{\mp}\pi^{\pm}$ Swaps

All candidates passing the above cuts are kept. Thus, K-π swaps from $D^0 \to K^-\pi^+$ are present. The D boost means that the momenta in 2-body decays are not equal in general, but this is clearly still a very special type of background.

One solution is to *require* RICH information on one or both tracks for $D^0 \to K^-\pi^+$. However, in practice, the effect is fairly small, even given the sizable fraction of solid angle outside the RICH, since dE/dx and ΔE kinematics can both help.

5 Running on D Skim Output

The skim writes out:
- A copy of the "hot-store" pass2 output
- A copy of the frozen postpass2 analysis objects (see below)
- The usage tag for TagDKaon, TagDPion, TagDKShort, TagDPi0, TagDEta
- The DTag-specific information (like M_{bc}, etc.)

Running on the D skims is much like running on any Pass2 output. use `$C3_SCRIPTS/setup_analysis`, not the older `$C3_SCRIPTS/runOnNewDSkim*.tcl`, This new method for recent pass2 takes advantage of the fact that we write out objects formerly created at analysis time, like pi0's, beamspot, and dE/dx. This guarantees that users get a stable set of data to analyze. The main difference between the new and old scripts is that many producers are now omitted since more items are written with the data. Currently, items to freeze postpass2 are listed in:
`~pass2/p2/Pass2Scripts/extract_p2postinfo/extract_postp2info.tcl`
But it is best to talk to experts if you have a detailed question.

[2] Note that some scripts have an incorrect comment that it is lowered only to 1.80 . Also, the "DSMassWinLoose" and "DSEWinLoose" parameters for M_{bc} and ΔE cut windows in the .awk file are not used anywhere currently, but were for the loose skim needed to do the fixing for the 2004 "DFIx" skim.

We have also saved the particle lists for each of the daughter particle selectors. You should *not* load the TagD selector Prods (i.e., `TagDXxxProd` with Xxx = Kaon, Pion, Pi0, KShort, Eta). Simply extract a given object with the proper Usage Tag, and you will get exactly which tracks were used as charged kaons, etc., from this saved information. Perhaps this will be clearest if you see the objects and Usage Tags in context as in the script:
`$C3_SCRIPTS/DTagScripts/DTagOut.tcl` :

```
# save TagD Tags as well
set TAGD "TagDKaon"
lappend additionToStore [list FATable<NavTrack> $TAGD]
set TAGD "TagDPion"
lappend additionToStore [list FATable<NavTrack> $TAGD]
set TAGD "TagDPi0"
lappend additionToStore [list FATable<NavPi0ToGG> $TAGD]
set TAGD "TagDEta"
lappend additionToStore [list FATable<NavEtaToGG> $TAGD]
set TAGD "TagDKShort"
lappend additionToStore [list FATable<NavKs> $TAGD]
```

There are only four added data members of the `DTag` class currently:
• DecayMode decayMode() const;
• double beamEnergy() const;
• double beamConstrainedMass() const;
• double deltaE() const
You can see for yourself by looking at `DTag/DTag/DTag.h` . In addition, the usual information from the `CDDecay` class is available for each DTag candidate.

Note that there is an outdated version of `DTagOut.tcl` in :
`/nfs/cleo3/Offline/scripts/DSkimPackage/tcl/DTagOut.tcl` ; it is missing the command to wrote out `DedxInfo`; avoid this one. One other curiousity is that the `DedxEventFlag` object, used to classify events for dE/dx calibration work, is also written out. This is not necessary (as far as I understand it).

5.1 Mode Numbering and Grouping

Here is the comment header from `DTag/DTag/DTagDecayModes.h` :

You can get the Mode of the decay mode by using the [] operator and passing the DTag enum as an argument. Once you have the Mode you can call the `name()` method to get the name (which corresponds to the Usage tag needed to extract the DTagList). For convenience you can use the `DTagDecayModes.name(...)` method to just directly get the name for the mode ID.

5.2 Simple Looping

Some `const\_iterators` are provided to make looping over certain subsets of tags easier. (They are implemented in `DTag/DTag/DTagDecayModes.h`).

To loop over all modes:
• use `begin()` and `end()`
To loop over only the $D^{\pm}$'s:
• use `Dp_begin()` and `Dp_end()`
To loop over only the flavored D^0's:
• use `D0_flavor_begin()` and `D0_flavor_end()`
To loop over only the un-flavored (self-conjugate) D^0's:
• use `D0_no_flavor_sc_begin()` and `D0_no_flavor_sc_end()`
To loop over only the un-flavored (non-self-conjugate) D^0's:
• use `D0_no_flavor_nsc_begin()` and `D0_no_flavor_nsc_end()`
(Note that the old names `D0_no_flavor_begin()` and `D0_no_flavor_end()` where discontinued to avoid confusion between the two types of neutral non-flavored modes...)

E.g., to print out the mode number and name for all D^+ modes available:

```
for(DTagDecayModes::const_iterator itMode = DTag::modes().Dp_begin();
    itMode != DTag::modes().Dp_end();
    ++itMode ) {
    cout <<" mode number : "<<itMode->first
         << " is " <<itMode->second.name()<<endl;
}
```

The iterator `itMode` is a C++ map of `<unsigned int, DTagDecayModes::Mode >`. The first "int" is the mode number (from the enum), and the second is a `DTagDecayModes::Mode`.

5.3 Looping over Selected Modes

It is also easy to build a list of tags chosen to include only the users' preferred modes. This is a wise idea if you wish to avoid newly-added modes creeping into your stable analysis by surprise. Also, as we add more and more modes, it is unlikely that there is any analysis that actually needs to look at all of the modes. It is most likely not a smart default!

```
DTagDecayModes modes;
vector<const char*> modesILike;
modesILike.push_back(modes.name(DTag::kD02KPi) );
modesILike.push_back(modes.name(DTag::kD02KPiPi) );

DTagList myD0List;
for( vector<const char*>::iterator itModeName = modesILike.begin();
     itModeName != modesILike.end(); ++itModeName ) {
   FAItem<DTagList> thisMode;
   extract( iFrame.record(Stream::kEvent), thisMode, *itModeName );
   myD0List += thisMode;
}
```

5.4 LabNet4Momentum

As a user, you shouldn't ever need to access the CESR crossing angle directly. What you usually need has been packaged in `LabNet4Momentum` for you. This is the four-momentum of the initial state, based on the beam energy and crossing angle.

The current skims wrote out the `LabNet4Momentum` object (unlike the older "April" and "DFix" skims from 2004).

Note that M_{bc} is corrected for the CESR crossing-angle. This is currently done with Lorentz boosts in the DTag code. The code gets the crossing angle in:
`DTagProd/Class/DTagListHolder.cc`;
it is passed via call to constructor of `DTagSelector` class in:
`DTagProd/DTagProd/DTagSelector.h`

Getting the crossing angle in a script that uses `runOnNewPass2.tcl` (e.g., the DTag skim script) takes some care and effort. The `runOnNewPass2.tcl` script uses runinfo.runinfo as a source of information to replace RunStatistics (which was used in `runOnPass2.tcl`), but only RunStatistics has the crossing angle. This is why at analysis time, we've provided the convenient alternative of `LabNet4Momentum` .

5.5 Accessing Beam Energy

```
#include "BeamEnergy/BeamEnergy.h"
...
   FAItem< BeamEnergy > beamEnergyItem;
   extract( iRecord.frame().record(Stream::kStartRun), beamEnergyItem );
   double beamE = beamEnergyItem->value();
```

5.6 Extracting Charm/Anti-charm Tag

In `DTag/Class/DTag.cc`, the function `int DTag::charm() const` is defined.

Return 0 for N- and s-type. For f-type, we return the charge of the D for $D^{\pm}$. In the case of $D^0/\bar{D}^0$, we return $-Q(K)$ where K represents the first daughter. The convention of putting the tag Kaon first most be enforced when adding new modes... Also, the ordering of the different types of D^0 modes must be maintained; the implementation assumes that $D^0 \to K_S K^- \pi^+$ is the first non-flavored mode.

Note that using properties of `CDDecay` list heads and tails didn't seem to be viable for non-self-conjugate, non-flavored (n-type, see below) decays.

5.7 Nominal D Mass

In `DTag/Class/DTag.cc`, the function `double DTag::nominalMass() const` is defined; it gives the relevant D^0 or D^+ nominal mass.

This uses another function implemented in `DTag/Class/DTagDecayModes.cc`, where much of this type of decay-mode book-keeping is done.

5.8 Nominal Observed Widths of M_{bc} and ΔE

The constants entity is named `DTagModeWidth`. It can be viewed easily via the interface at: `https://www.lns.cornell.edu/restricted/CLEO/CLEO3/soft/Constants/`

For purpose of DTag monitoring or physics analysis, we record the DTag $M_{\rm BC}$ and ΔE widths in constant database. This features versioning and makes it flexible to use different values for different datasets. Therefore, we discarded the old hard-wiring method.

We name the constants for DTag $M_{\rm BC}$ and ΔE Widths as `DTagModeWidth`. The constants have the following format:

```
Type: Constants
Name: DTagModeWidth
{
      UInt16 Index
      UInt16 DecayMode
      double SigmaMBCLeft
      double SigmaMBCRight
      double SigmaDeltaELeft
      double SigmaDeltaERight
}
```

From above we see that, for each decay mode, we have four widths associated with it: the widths for DTag $M_{\rm BC}$ and ΔE from both left and right. For now, we set the values equal from left and right. The idea to have four widths here is that, the fits of the $M_{\rm BC}$ and ΔE peaks are complicated, we might want to use Crystal Ball function or bifurcated Gaussian in the future, if it is necessary.

To use these widths in the analysis code:

- In the C++ code,

```
// Including the header files
#include "C3DTag/DTagStream.h"
#include "DTag/DTagModeWidth.h"

  // Extracting the constants
  FAItem<DTagModeWidth> aDTagModeWidth;
  extract(iFrame.record(DTagStream::kModeWidth), aDTagModeWidth);

  // Using the values
  for(DTagDecayModes::const_iterator itMode = DTag::modes().begin();
      itMode != DTag::modes().end(); ++itMode ) {
    unsigned int decayMode = itMode->first;
    report(INFO, kFacilityString) << itMode->second.name() << endl;
    report(INFO, kFacilityString)
      << "simgaMBCLeft    :"
      << aDTagModeWidth->sigmaMBCLeft(decayMode) << endl;
```

```
    report(INFO, kFacilityString)
      << "simgaMBCRight   :"
      << aDTagModeWidth->sigmaMBCRight(decayMode) << endl;
    report(INFO, kFacilityString)
      << "simgaDeltaELeft :"
      << aDTagModeWidth->sigmaDeltaELeft(decayMode) << endl;
    report(INFO, kFacilityString)
      << "simgaDeltaERight:"
      << aDTagModeWidth->sigmaDeltaERight(decayMode) << endl;
  }
```

- In `Makefile`
  Uncomment the following line:

  ```
  #COMM_INCS =     $(CONSTANTS_INCS)
  % $ to fix emacs colors...
  ```

  and add `C3DTag` into `CLEO3_LIBS`.

- In the tcl script:

  ```
  prod sel DTagConstProd
  ```

  One can use local ASCII constants by adding:

  ```
  source_format sel DTagModeWidthFileSourceFormat
  file in my_private.dtagmodewidth
  ```

- `DTagConstProd`

- `DTagModeWidthFileSourceFormat`

As of this writing, the constants are identical for all runs. They are preliminary numbers based on db31-33. As a result, the values are only approximate for some of the dirtiest modes. Also, from data35 onward, the use of 12 (vs. 6) wigglers means that the M_{bc} width, generally dominated by the beam-energy spread, increases by about 0.1 MeV (the total width is of order 1.5 MeV, and is somewhat mode-dependent).

5.9 Other Member Functions

There are also functions for `otherSideTracks`, `otherSideShowers`;
see comments in `DTag/DTag/DTag.h` header file.

6 DDoubleTagProd

There is a double tag producer, `DDoubleTagProd`, which will combine the tag lists, make an object to insert in the frame, and add extra information. The extra information includes things like the angle between the D momenta, calculated after vertex and other fits are also performed. You can simply select this producer in your analysis .tcl scripts.

Creating a correct list of Double Tags involves some care and some choices as well. Our philosophy as currently implemented in the code is discussed below in Section 17.2. That algorithm is applied in:
`DDoubleTagProd/Class/DDoubleTagListProxy.cc`

After this list is made, further processing is done in :
`DDoubleTagProd/Class/DDoubleTagSelector.cc`
The implementation is as follows: for each of the D and $\bar{D}$ in the double tag, the following kinematic fits are applied:

- A vertex fit. The modified momentum is used as input in the next fit.
- An energy constraint (beam energy) fit. The modified momentum is used as input in the next fit.
- An mass constraint (D^0 or D^+ mass) fit.
- The modified momentum from the two fits is then used to calculate the angle (θ) between D and $\bar{D}$ after another fit. This angle in turn can be used to eliminate background.

The χ^2 values and the angle between the D's are saved as member data and accessible via member functions; see `DDoubleTag/DDoubleTag/DDoubleTag.h` .

There is provision for cut parameters in :
`/DDoubleTag/DDoubleTag/DDoubleTagCutCriteria.h` .
Currently, the only one provided is accessible in SUEZ as:
`param DDoubleTagProd DDbarThetaMin`
and is set to 0 by default (i.e., no cut, since we expect a peak at 180°).

Double Tags are now written out with the skims (unlike the older "April" and DFix" skims from 2004). You do not need to select the producer at run time.

7 DTagUtilities: Some Helpful Software Utilities

Here we describe various helpful bits of code that were contributed for public use. These are collected in `DTagUtilities`. Note that `DTagUtilities` is not a class, but a namespace.

Simply include `DTag/DTagUtilities.h"`, and any `DTagUtilities` function is globally available for calling as, for example:

```
DTagUtilities::trackPassesLeptonVeto(...)
```

Particular arguments are documented with each below; this information is easily found from the header files:
`$C3_CVSSRC/DTag/DTag/DTagUtilities.h`) while the implementation of criteria is in:
`$C3_CVSSRC/DTag/Class/DTagUtilities.cc`.

7.1 Missing Mass

```
double DTagUtilities::m2miss( const DTag& dtag,
                              const HepLorentzVector& p,
                              const LabNet4Momentum& lab)
```

This uses `LabNet4Momentum`, which seemed like the best way to store the initial state information (both beam energy and crossing angle).

7.2 Dalitz Variables

```
Triplet< double, double, double >
   DTagUtilities::dalitzVariables( const DTag& aDTag )
```

The variables are returned in the order $m^2_{12}, m^2_{13}, m^2_{23}$. The $1, 2, 3$ order of the daughters is the order (standardized for D Tags) used when the CDDecay is built; see mode tables in the Appendix.

The `dalitzVariables()` function throws an exception if given a non-three-body decay.

7.3 Kshort Quality Cuts

```
 DABoolean kshortPassesQualityCuts( const NavKs&,
                                    FATable< KsQuality >&,
                                    double aFlightSignificanceCut ) ;
```

The cuts are:

- Check `TRSeedTrackQuality` for each track; each must have:
  . `valid()` true and and `originUsed()` false
- Check `TRTrackFitQuality` for each track; each must have:
  . `valid()` true and and `fitAbort()` false
- Check `VXFitVeeKShort`; do not allow `fitChiSquare() < 0`
- Check `KsQuality`; cut on `flightDistanceSignificanceEnhanced()`
  cut is passed as an argument as indicated above.

7.4 Unmatched Showers above 50 MeV

```
int numberOfUnmatchedOtherSideShowersAbove50MeV( const DTag&,
                                                 FATable< NavShower >& )
```

The source code also notes that:

```
// In principle, the input shower list should be pruned of showers matched
// to tracks in the DTag candidate, since these will appear in the "other
// side" showers.  However, since track matching is checked later on, this
// step is unnecessary.
```

One example of a use for this function is presented in the next section on Lepton Vetoes.

7.5 Lepton Veteos

```
DABoolean trackPassesElectronVeto( const NavTrack& )
DABoolean trackPassesMuonVeto( const NavTrack& )
DABoolean trackPassesLeptonVeto( const NavTrack& )
DABoolean primaryTracksInDTagPassElectronVeto( const DTag& )
DABoolean primaryTracksInDTagPassMuonVeto( const DTag& )
DABoolean primaryTracksInDTagPassLeptonVeto( const DTag& )
```

For two-track modes, there can be background from junk events, believed to be cosmics. They occur across our entire skimmed ΔE range. Those with a ΔE inside our tighter analysis cuts, show up at $M_{bc} = E_{bm}$ due to the zero net momentum. They are a nuisance when fitting M_{bc} peaks with an Argus background.

Recommendation: in $D^0 \to K^-\pi^+$, when there are only two tracks in the event:
apply `DTagUtilities::primaryTracksInDTagPassLeptonVeto()`
and `DTagUtilities::numberOfUnmatchedOtherSideShowersAbove50MeV() > 0`

8 MC Issues

Various generic MC has been generated at Minnesota. The DTag web page is one place to find current information on available samples:
`http://www.lns.cornell.edu/restricted/CLEO/analysis/cleocHadronic/dtag/`

8.1 MC Beam Energy Spread

Stored in `BeamEnergySpread` constants; currently there is only one file for all CLEO-c runs with a value of "0.0015", or 1.5 MeV. This value dominated the M_{bc} resolution for most modes.

This value probably increased by about 0.1 MeV when we began using 12 wigglers, instead of 6, beginning with data35. So far this has not been studied carefully (though I can see the increase easily in the monitoring plots).

8.2 CDDecay MC tagging

See `http://www.lepp.cornell.edu/~chengp/DTag/mctag.ps`. and CBX 04-10.

Using MC matching for DTag is very easy:
If `iD` is a DTag, one first asks if there is a match at all:
`iD.hasMatchedMCParticle()`
If this is true, the following will pass a MCParticle to you:
`const MCParticle& iMCParticle = iD.matchedMCParticle();`

The code is more general than just DTagging; "DTag" above can be *any* `CDCandidate` (base class of `CDDecay`, `CDKs` etc.) and the tagging will work.

The original implementation (spring 2004) had the feature that final-state radiation (FSR) from a track would cause the tag to fail. This was fixed (tagged 30Jun04) by allowing for extra photons to be missing (when the mode with the photon is not in the decay tables).

You can check out the current `CleoDChain` and use this with the `20040622_FULL` release to get this last update.

8.3 Initial State Radiation

ISR has been implemented in the $D\bar{D}$ MC. However, the Spring, 2004 MC from Minnesota does *not* yet include ISR.

ISR was already present for the $q\bar{q}$ simulation, but not for resonant states due to subtleties. I believe it also stopped being generated for continuum when we switched to the Lund area-law method (CLEO-c vs. CLEOIII?); this will be fixed in the future.

The code is in: `EvtGenModels/Class/EvtVPHOtoVISR.cc`
To use it, you simply put this in your decay file:

```
Decay vpho
1.000    psi(3770)  gamma     VPHOTOVISR;
Enddecay
```

9 Data Features

Here I collect a laundry list of some data features.

9.1 M_{bc} and ΔE Anti-correlation

We use a run-averaged energy to calculate M_{bc} and ΔE. Individual collisions have a varying energy due to the beam-energy spread. This makes M_{bc} and ΔE correlated.

The $E_{bm,ave}$ written here is identical to the usual E_{bm} from the database; we are just emphasizing that it is run-averaged here to avoid confusion.

If the true energy has fluctuated higher, then:
- $\Delta E \equiv E_{cand} - E_{bm,ave}$ will increase
- $M_{bc} \equiv \sqrt{E^2_{bm,ave} - p^2_{cand}}$ will decrease.

Keep in mind that $E_{bm,ave}$ is unaware of the fluctuations in a given event. Thus, for M_{bc}, increased true energy means p_{cand} increases, leading to M_{bc} decreasing.

9.2 ISR and High Tail on M_{bc}

Initial state radiation leads to a high-side tail in M_{bc} due to the decrease in p_D.

9.3 Low Shoulder on M_{bc}

There is a low shoulder seen in high-statistics MC on M_{bc}. Current (unconfirmed) speculation centers on decays in flight.

9.4 Backgrounds from Cross-talk

Cross-talk is observed between modes that differ by replacing a K_S with $\pi^+\pi^-$ and vice-versa. Some pion pairs happen to have a mass near the K_S; some K_S daughters will pass prompt track cuts.

Consider $\pi^+\pi^-$ faking K_S; which mode is Cabibbo-suppressed is very relevant to the rates. For example, $K^-\pi^+\pi^+\pi^-$ faking $K^-K_S\pi^+$ would naively be a more serious issue than $\pi^+\pi^-\pi^0$ faking $K_S\pi^0$. We should remember that faking works in the other direction as well, with short-lived K_S showing up as $\pi^+\pi^-$ from the origin; these are easily cut with a mass cut, and show up as obvious bands in Dalitz plots.

One can also add a fake π^0 by combining an FSR photon with a soft accidental photon. E.g., $K^-\pi^+$ with radiation can become $K^-\pi^+\pi^0$, albeit with a slightly shifted ΔE.

9.5 Double Tags

The two M_{bc} values are correlated, since much of the width is due to the beam energy spread, which is in common.

A striking diagonal stripe where $M_{bc,1} = M_{bc,2}$ is often present due to "mis-partitioning". This refers to an event where (essentially) all showers and tracks are found, but more than one plausible way of dividing them into D vs. $\bar{D}$ daughters is found. The momenta of the D and $\bar{D}$ are always equal and opposite such that the M_{bc} values are equal even for incorrect partitions.

One needs to be aware of multiple candidates per event. Modes like $K3\pi$ vs. $K3\pi$ have enough combinatorics that mis-partitions may also pass tighter signal region cuts.

10 Appendix: Cross Sections and MC

We generating MC samples to be $N\times$ the data, where N varies by sample. To get the number of events from the integrated luminosity (or vice-versa) requires cross-sections.

Some approximate values are listed in the table below.

Table 5: Nominal Cross Sections (for MC)

MC Sample	Cross Section (nb)
$D\bar{D}$	5.7
Continuum	18
Rad. Return to ψ'	3.2
$\tau^+\tau^-$	3.0

The $D\bar{D}$ MC also requires a $D^0\bar{D}^0 : D^+D^-$ ratio. This is currently specified as 4:3 via:

```
Decay vpho
0.571429    D0  anti-D0                    VSS;
0.428571    D+  D-                         VSS;
Enddecay
```

11 Appendix: Mode Lists

In June 2004, 12 more modes were added to the initial set of 12 modes.

Before the February, 2005 skims, the number of modes was increased from 24 to 55.

Identifiers and name strings are added in: `DTag/Class/DTagDecayModes.cc` with commands like:

```
m_map.insert( value_type( DTag::kD0toKPi, "D02K-Pi+" ) );
```

Note that the name string include the signs of electric charge, while the integer enums `DTag:kXXX` do not. The enums are defined in `DTag/DTag/DTag.h` The one exception is that the enum names for $K_S K^+\pi^+\pi^-$ and $K_S K^-\pi^+\pi^+$ would be degenerate without some indication of charge; they are thus denoted `KsKpluspipi` and `KsKminuspipi`, respectively.

We have standardized the order of daughters to be:
$K_S, K^-, K^+, \pi^+, \pi^-, \pi^0, \eta$
Note that the K charges are opposite in order to the π (largely due to K^- being Cabibbo-favored). The idea is that, given a final state, there is a simple recipe to easily know the ordering of the daughters.

Modes are defined in `DTagProd/Class/DTagProd.cc` , with code like:

```
FILL_FUNC( DTag::kD0toKPi )
{
   DTagList tempList( listHolder().selector(iRecord,DTag::kD0toKPi) );
   tempList = chargedKaons(iRecord).minus()*
              chargedPions(iRecord).plus();

   iList = tempList;
}
```

Table 6: Flavored neutral modes; no new modes were added in June 2004.

Mode	Identifier	Name String
$K^-\pi^+$	DTag::kD0toKPi	"D02K-Pi+"
$K^-\pi^+\pi^0$	DTag::kD0toKPiPi0	"D02K-Pi+Pi0"
$K^-\pi^+\pi^0\pi^0$	DTag::kD0toKPiPi0Pi0	"D02K-Pi+Pi0Pi0"
$K^-\pi^+\pi^+\pi^-$	DTag::kD0toKPiPiPi	"D02K-Pi+Pi+Pi-"

Table 7: Unflavored neutral modes; modes below the line were added in June 2004.

Mode	Identifier	Name String
$K_S\pi^+\pi^-$	DTag::kD0toKsPiPi	"D02KsPi+Pi-"
$K_S\pi^+\pi^-\pi^0$	DTag::kD0toKsPiPiPi0	"D02KsPi+Pi-Pi0"
$K_S\pi^0$	DTag::kD0toKsPi0	"D02KsPi0"
$\pi^+\pi^-\pi^0$	DTag::kD0toPiPiPi0	"D02Pi+Pi-Pi0"
$\pi^+\pi^-$	DTag::kD0toPiPi	"D02Pi+Pi-"
K^-K^+	DTag::kD0toKK	"D02K-K+"
$K^-K^+\pi^0$	DTag::kD0toKKPi0	"D02K-K+Pi0"
$\pi^0\pi^0$	DTag::kD0toPi0Pi0	"D02Pi0Pi0"
K_SK_S	DTag::kD0toKsKs	"D02KsKs"
$K_SK_S\pi^0$	DTag::kD0toKsKsPi0	"D02KsKsPi0"
$K_S\pi^0\pi^0$	DTag::kD0toKsPi0Pi0	"D02KsPi0Pi0"
$K_SK^-K^+$	DTag::kD0toKsKK	"D02KsK-K+"
$K_S\eta$	DTag::kD0toKsEta	"D02KsEta"
$\pi^0\pi^0\pi^0$	DTag::kD0toPi0Pi0Pi0	"D02Pi0Pi0Pi0"

Table 8: Charged modes; modes below the line were added in June 2004.

Mode	Identifier	Name String
$K^-\pi^+\pi^+$	DTag::kDptoKPiPi	"D+2K-Pi+Pi+"
$K^-\pi^+\pi^+\pi^0$	DTag::kDptoKPiPiPi0	"D+2K-Pi+Pi+Pi0"
$K_S\pi^+$	DTag::kDptoKsPi	"D+2KsPi+"
$K_S\pi^+\pi^0$	DTag::kDptoKsPiPi0	"D+2KsPi+Pi0"
$K_S\pi^+\pi^+\pi^-$	DTag::kDptoKsPiPiPi	"D+2KsPi+Pi+Pi-"
$K^-K^+\pi^+$	DTag::kDptoKKPi	"D+2K-K+Pi+"

12 Appendix: Basic Parameters And Formulae

This is a very abbreviated version of stuff from the DTag Kinematics document. In particular, various correlations are discussed there.

Table 9: Nominal Parameter Values

Parameter	Value(error)	Units
$M_{\psi''}$	3769.9(2.5)	MeV
Γ	23.6(2.7)	MeV
Γ_{ee}	0.26(0.04)	keV
E_{bm}	1885.	MeV
M_{D^0}	1864.5(0.5)	MeV
M_{D^+}	1869.3(0.5)	MeV
$M_{D^+} - M_{D^0}$	4.78(0.10)	MeV
p^*_+	243	MeV
p^*_0	277	MeV
p^*	$260 \times (1.00 \pm 0.07)$	MeV
θ_X	0.004	rad
Beam E spread	1.5	MeV

Beam-constrained mass M_{bc}:
$M_{bc}^2 = E_{bm}^2 - |(\sum \vec{p}_i)|^2$
(note: assumes $E_D = E_{bm}$; violated by boost, which we correct for, and ISR).
Dominated by beam spread; independent of PID.
Energy Balance ΔE:
$\Delta E \equiv \sum E_i - E_{bm}$
Gross sensitivity to PID.

13 Appendix: Accessing Crossing Angle

You really should be using LabNet4Momentum; please see the section on this useful object above. But if you insist, here's some information on the crossing angle itself.

This requires that your processor include:

```
#include "CesrCleoComm/CesrCrossingAngle.h"
```

and then invoke:

```
FAItem< CesrCrossingAngle > crossingAngleItem;
extract( iFrame.record(Stream::kStartRun), crossingAngleItem );
crossingAngle = crossingAngleItem->value();
```

Since `runOnNewPass2.tcl` no longer uses RunStatistics, one must also have this in the tcl script:

```
# Standard Pass2 Script
run_file $env(C3_SCRIPTS)/runOnNewPass2.tcl
# if you want to get the crossing angle use this dangerous Kludge
# i.e. uncomment the following two lines
#module sel RunStatisticsSourceModule
#runstatistics in
```

Note that we have corrected for the crossing angle in calculating M_{bc}, in the missing-mass function, etc.

14 Appendix: D Tag Contributors

Here is a list (apologies to those I've omitted!) of those who actively contributed to the work described in this CBX:
Nadia Adam, David Asner, Chaouki Boulahouache, Guangpei Chen, Dan Cronin-Hennessy, Mikhail Dubrovin, Datao Gong, Chris Jones, Feng Liu, Peter Onyisi, Chulsu Park, Anders Ryd, Batbold Sanghi, Werner Sun, Gocha Tatishvili.

I would also like to thank Melissa Cravey, Norm Lowrey, Alexander Scott, and Mike Weinberger for additional feedback on using D tagging.

The Pass2 and EventStore crew, including Jean Duboscq and Valentin Kuznetsov also helped in making D skims accessible.

Finally, I thank Alex Smith, Brian Lang, and friends for taking over and carrying D Tagging forward into the D_s era.

15 Explanation of Mode Tables

*** note poor Mbc, DeltaE widths? (wait to check constants vs. new plots...) mention plots!

*** rewrite as explan. of full lists *and* 55 mode summary; use table ref's !!!

We list **all** final states of the type nK mπ where $n+m \leq 5$ and $n \leq 3$. The restriction on n is absolute, due to phase-space. That on $n+m$ is for sanity's sake.

We include all two- and three-body modes (except for three-body DCSD K^+ modes). [3] The same is true of four-body modes, except that a few modes with excessive numbers of π^0's and K_S are omitted. For five-body modes, only a few of the nicer ones are selected.

We also include a few $\eta \to \gamma\gamma$ modes. Note that resonances, like ρ or ω, that decay to all K, π final-states are included in corresponding inclusive modes.

15.1 DCSD Modes

For charged D's, we keep only the one two-body DCSD decay mode.

*** reword; avoid repeating later...***

DCSD decays are particularly interesting for neutral D's since the DCSD background to a $D^0 - \bar{D}^0$ mixing signal is absent due to the quantum mechanics of the $\psi(3770) \to D^0\bar{D}^0$ decay.

[3] DCSD D^0 modes are present as charge conjugates, see discussion later.

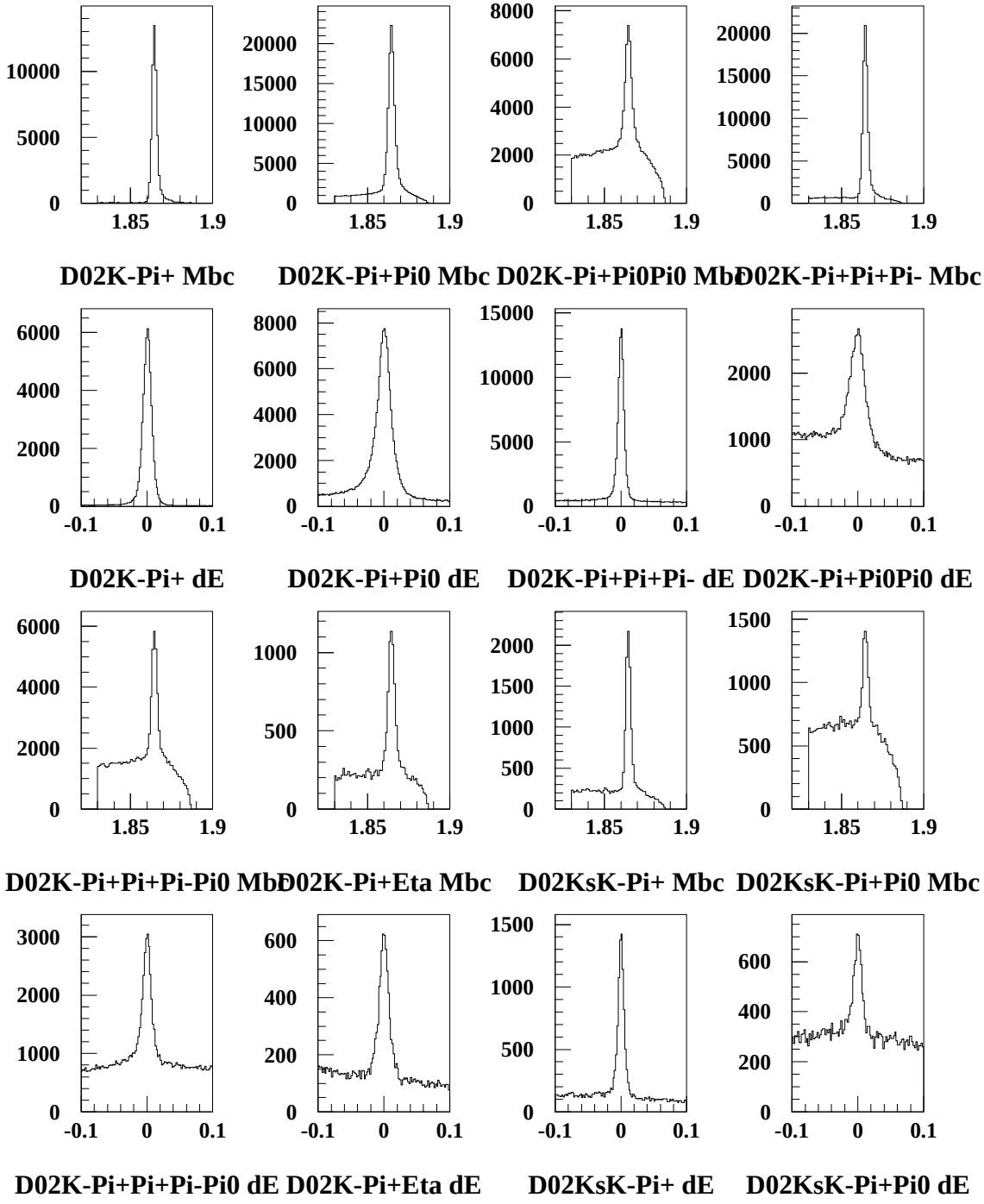

Figure 1: M_{bc} and ΔE plots for the six flavored and the two self-conjugate but non-flavored D^0 modes.

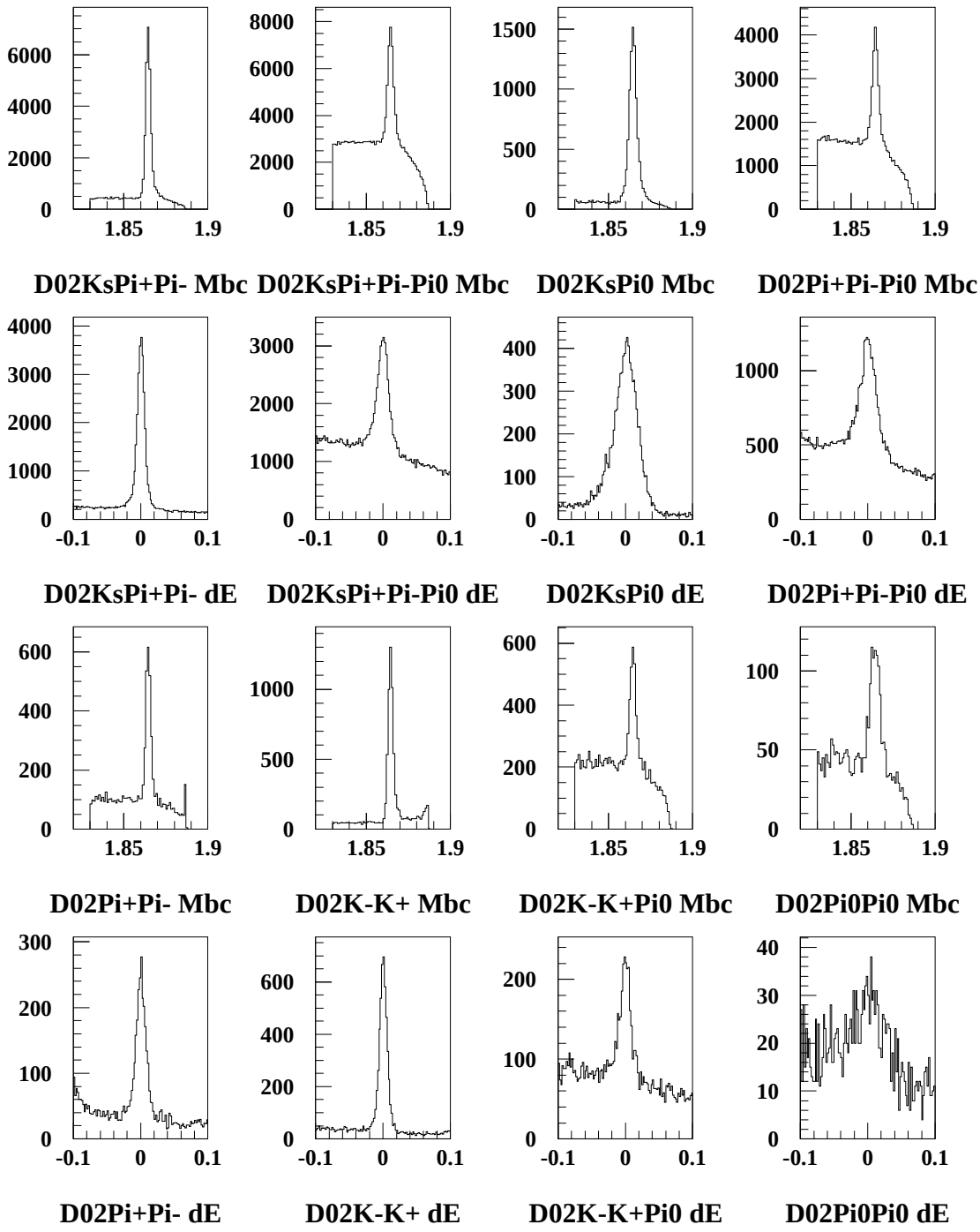

Figure 2: M_{bc} and ΔE plots for several self-conjugate D^0 modes.

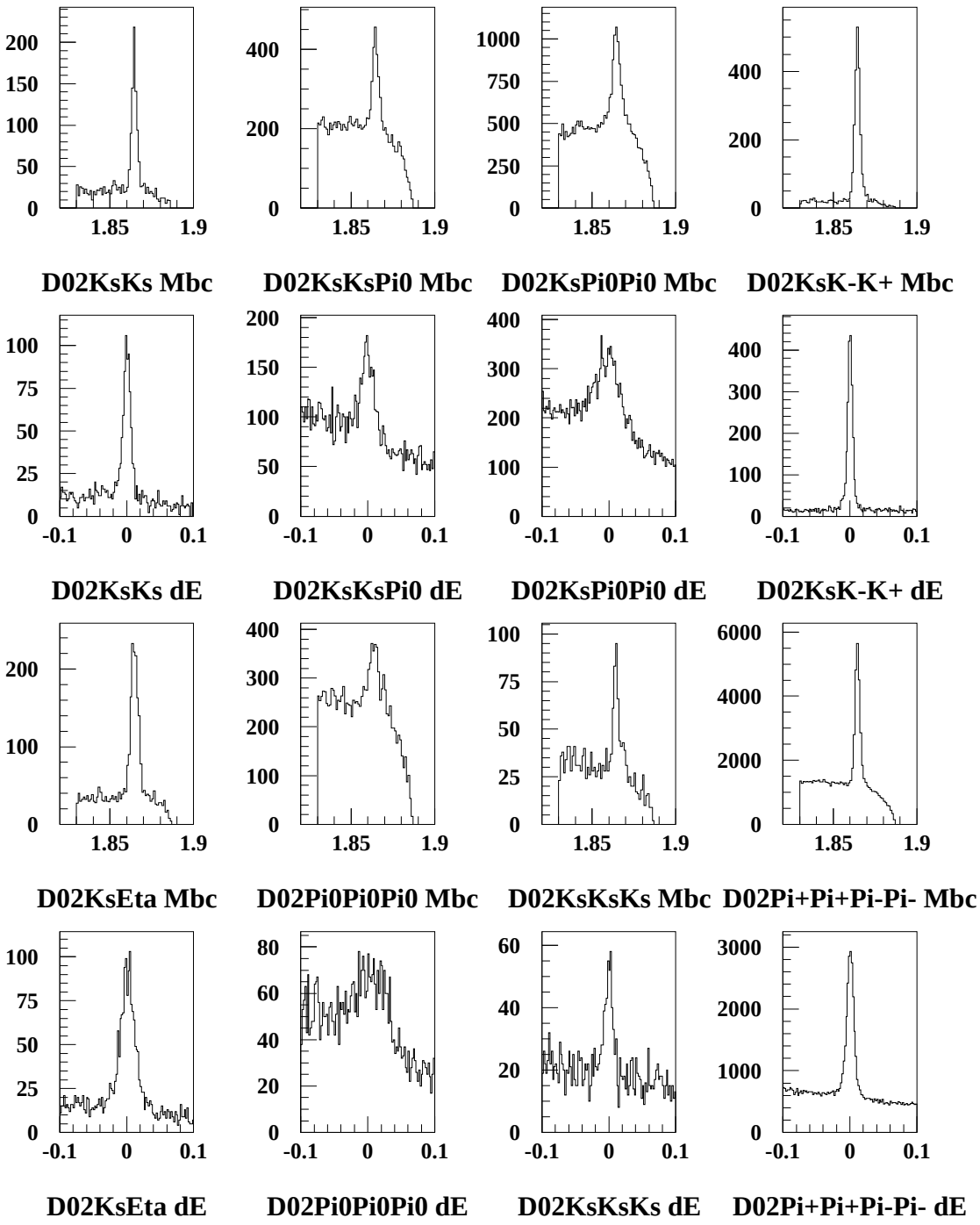

Figure 3: M_{bc} and ΔE plots for several self-conjugate D^0 modes.

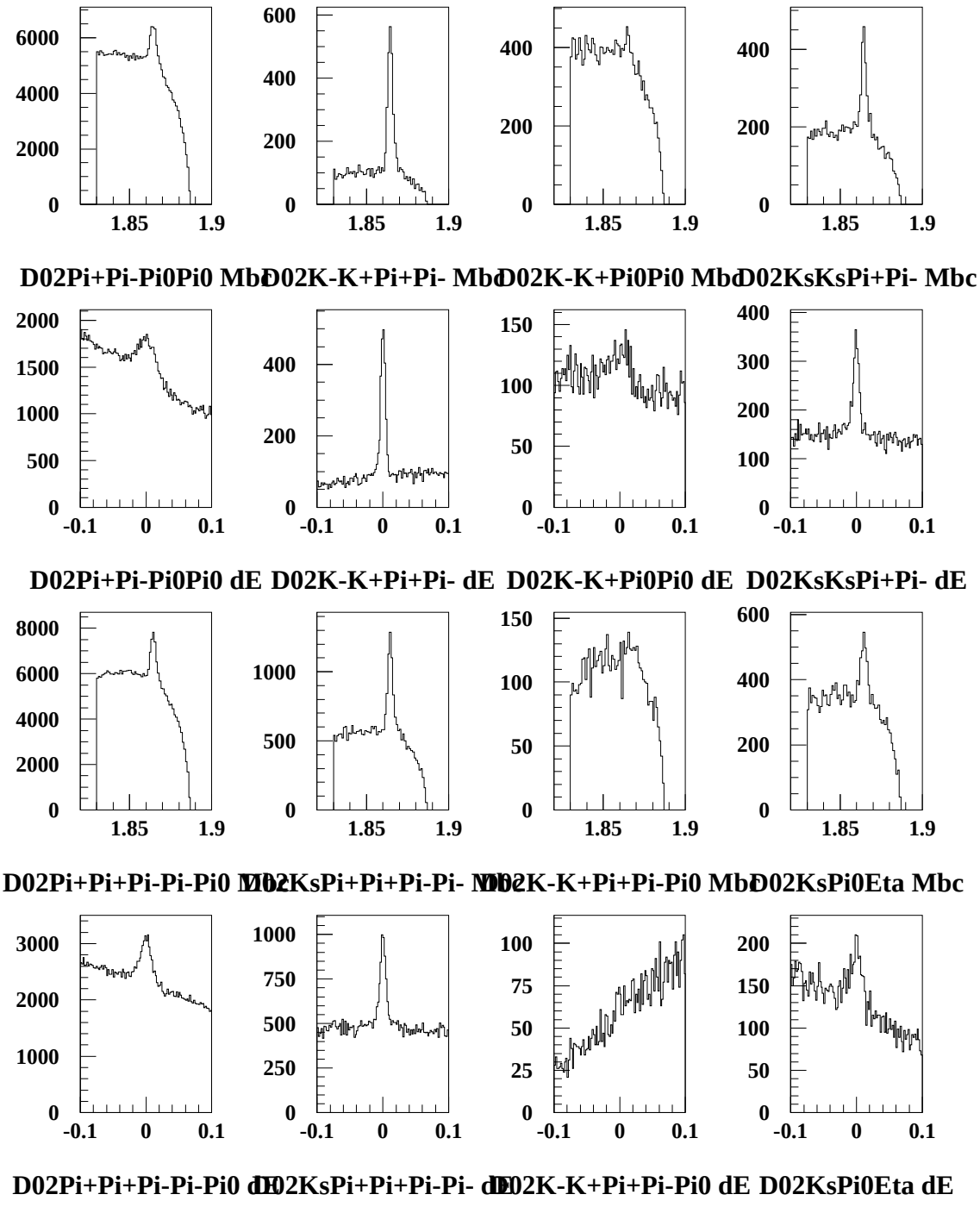

Figure 4: M_{bc} and ΔE plots for several self-conjugate D^0 modes.

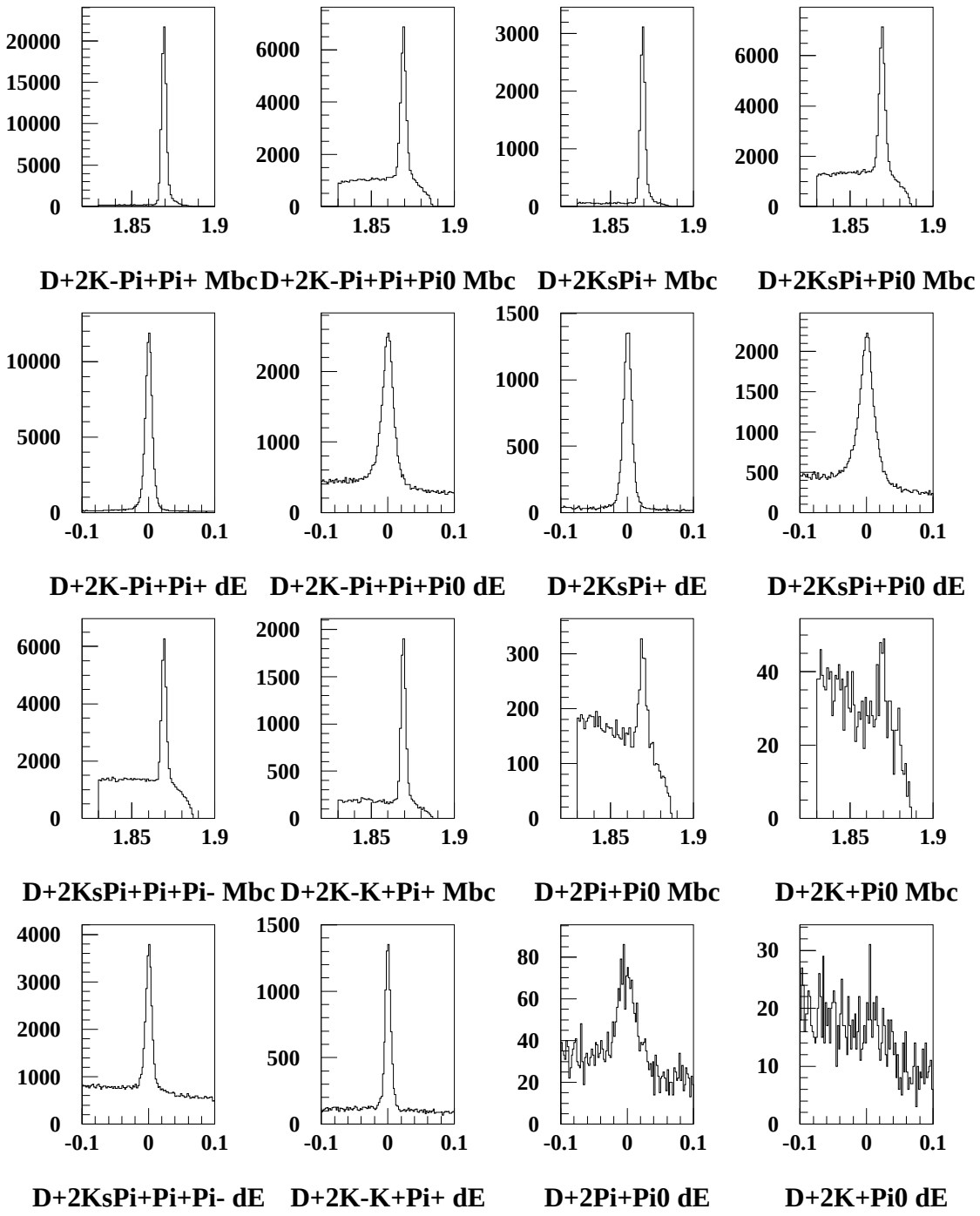

Figure 5: M_{bc} and ΔE plots for several D^+ modes.

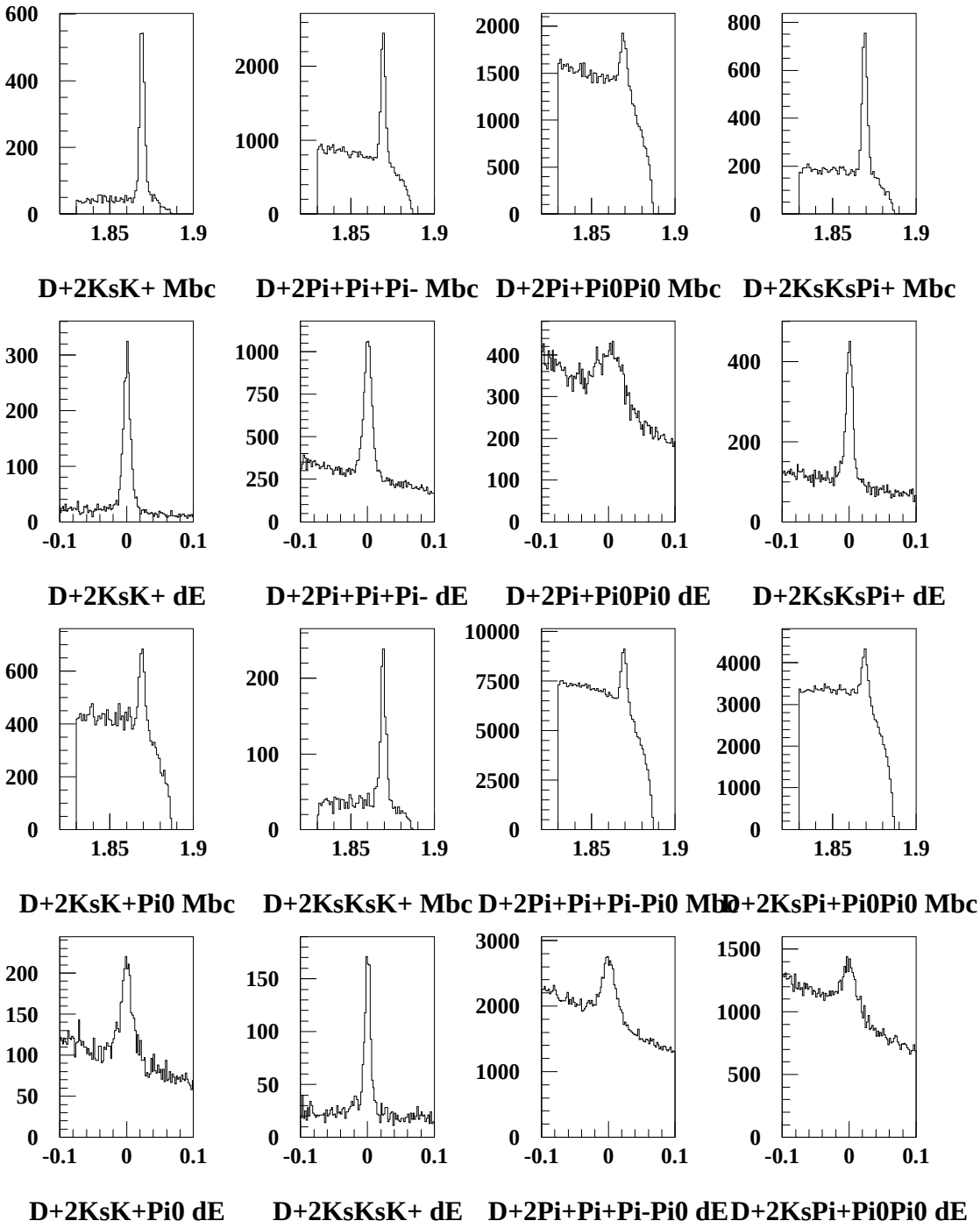

Figure 6: M_{bc} and ΔE plots for several D^+ modes.

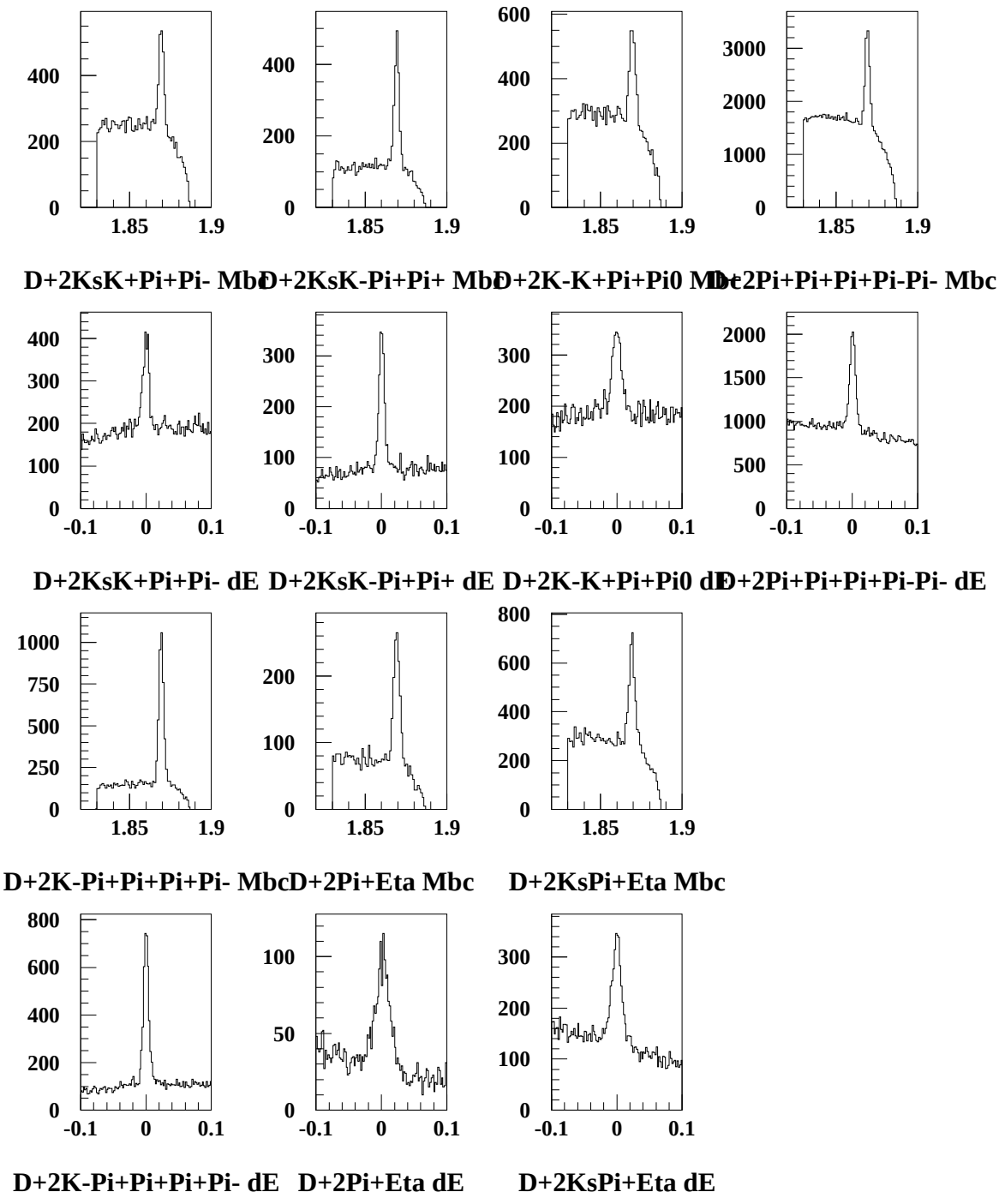

Figure 7: M_{bc} and ΔE plots for several D^+ modes.

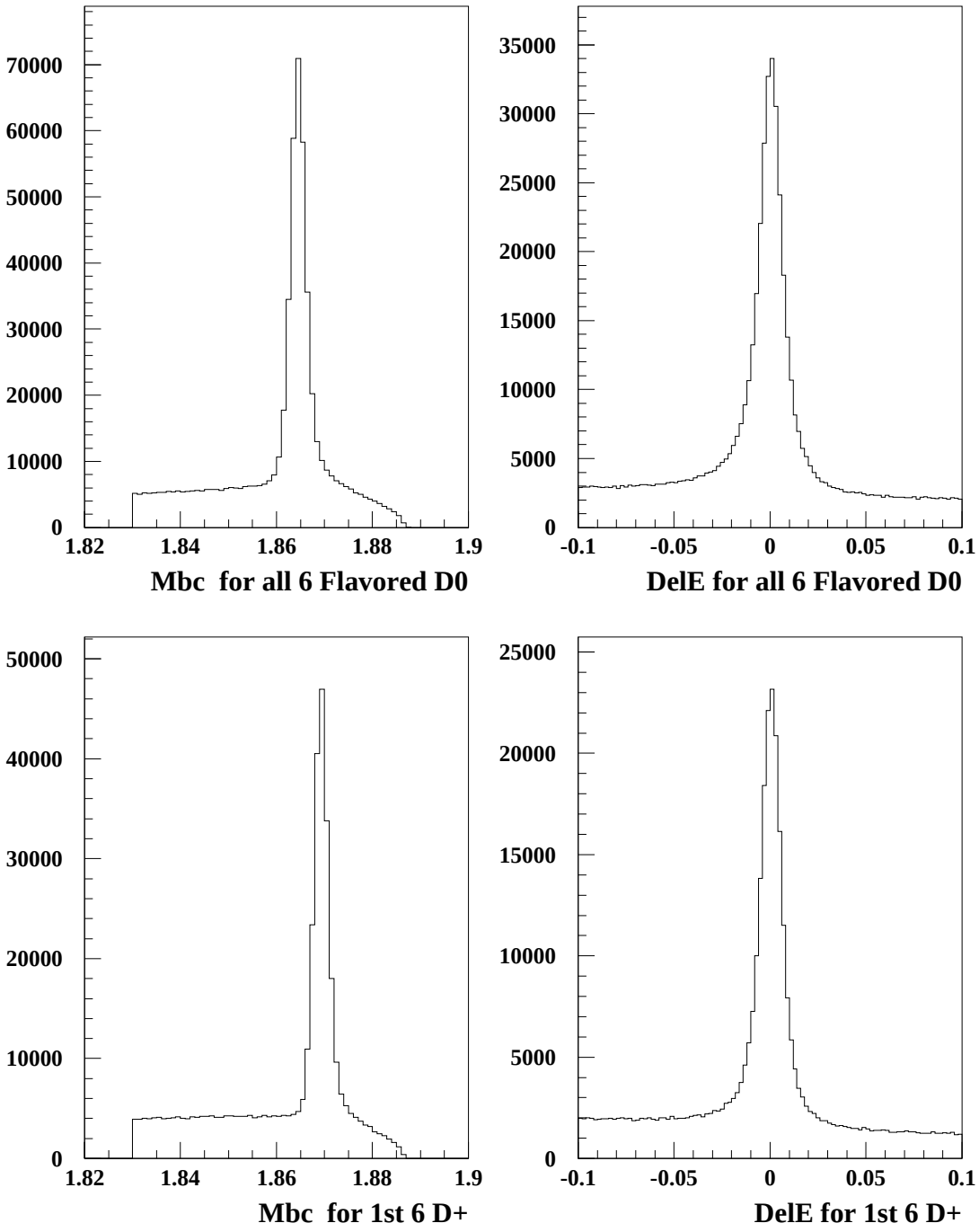

Figure 8: Summary of beam-constrained mass (top row) and delta E (bottom row) for the sum of six flavored D^0 modes and the first six D^+ modes.

That is, the CP of the state leads to canceling amplitudes for a $(K^-\pi^+)$ vs. $(K^+\pi^-)$ final state, for example.

15.2 Type Column

The "Type" column in the tables lists:
- "A" means Cabibbo-Allowed.
- "S" means Singly-Cabibbo-suppressed.
- "D" means Doubly-Cabibbo-suppressed.

The "Type" column often lists a second character for D^0, after a ",", for D^0 modes only. This indicates:
- "f" means flavor-tagged (therefore, not self-conjugate)
- "s" means self-conjugate (therefore, not flavor-tagged)
- "n" means neither self-conjugate nor flavor-tagged
- "w" means wrongly-flavor-tagged (i.e., DCSD)

See the section below for more details on the decay processes.

15.3 Status Column

The "Status" column of the table lists:
- "12" for mode in the original set of 12 used by DTag
- "24" means it is part of the expansion from 12 to 24 modes
- "55" means it is part of the expansion from 24 to 55 modes
- "XX" means intentionally omitted.
- "?" means consider adding in future

Intentionally omitted includes neutral DCSD and conjugates of included neutral "n"-type decays. In each case the intent is to avoid a complete double- counting of the mode in the single-tag list since the conjugate mode is already listed. Implications for the DDoubleTagProd are discussed below. That discussion is a bit technical re: innards of the software; if you aren't interested you can probably just trust us.

15.4 $\mathcal{B} \times 10^4$ Column (current PDG br. frac.)

I've also listed current (2004) PDG branching fractions where available. Generally, the PDG lists K^0 and $\bar{K}^0$; I multiply by 0.5^N for modes with N K_S's. (In rare cases, they do list K_S; they should always do this in my opinion...see comments below re: interference.)

Entries that say $> XX(YY)$ are generally cases were some resonant mode is listed. For example, the PDG lists
$B(D^0 \to K^*(892)^0\eta) \times B(K^*(892)^0 \to K^-\pi^+) \times B(\eta \to \pi^+\pi^-\pi^0)$.
I divide by the η branching ratio, and then list the rate as ">", since this on only one of the possible contributions.

16 Decay Processes: Physics

Let's concentrate first on the physics of the D decays. For non-strange D mesons, the initial state has $S = 0$, and hence ΔS and the final-state S are equal. Note that we by default discuss decays of D^0 and D^+ mesons with a c quark, and not those of the conjugate states.

Note that the standard convention is that an s ($\bar{S}$) quark has $S = -1$ ($S = +1$).

The most common decays yield states with an s quark and $S = -1$. Cabibbo-suppressed decays yield $S = 0$, and may or may not have an $s\bar{s}$ pair. Rare DCSD processes have a lone $\bar{s}$ quark, and hence $S = +1$.

Final states with $|S| > 1$ are not allowed.[4] But remember that K_S are part $S = +1$ and part $S = -1$. This leads to multiple decay diagrams for modes with K_S; interference between these can make the usual factor of two between K^0 or $\bar{K}^0$ and K_S invalid. It also means that states with K_S's have indefinite strangeness ($S = \pm 1$); one should look for the dominant process; i.e., interpret as $S = -1$ if possible; if not, then $S = 0$, etc.

16.1 Flavor Tags

All $D^\pm$ decays are flavor-tagged (i.e., we know if the initial state contained c or $\bar{c}$) simply from the net charge. The rest of this discussion therefore concerns only neutral D's.

The central idea is to tag with the strangeness of the final state, since the dominant Cabibbo-allowed decay process is $c \to sW^+ \to c(u\bar{d})$. If there are no kaons in the final state, we cannot tag. The strangeness can come from either of the two W vertices: at W emission or W decay. One can make an s for $S = -1$, an $\bar{s}$ for $S = +1$, no strange quarks for $S = 0$, or both an s and $\bar{s}$ for $S = 0$ as well. Tagging works since the "make an s for $S = -1$" dominates the rate due to the CKM factors which lead to the dominance of $c \to sW^+; W^+ \to u\bar{D}$.

K_S have strangeness of either $S = -1$ or $S = +1$. This causes confusion when, depending how one or more K_S are assigned, both $S = -1$ and $S = +1$ are possible for the final state.

If there is one kaon, and it is a $K^\pm$, then we assume a K^- indicates D^0 and a K^+ indicates a $\bar{D}^0$. We will see below that certain rare decay processes (e.g., DCSD decays) make this not quite 100% correct. If there is one kaon, and it's a K_S, we cannot tag.

If there are two kaons, the net strangeness must be 0 (never ± 2: this would require a negligibly small second-order weak process), and we then have no way of flavor-tagging.

If there are three kaons, the net strangeness must be ± 1. If there are 2, or 3 K_S, we cannot tell which it is. If there is 1 K_S sometimes we can tell (e.g., $K_S K^- K^-$), but other times we cannot (e.g., $K_S K^+ K^-$). If there are no K_S, then we can tell (e.g., $K^\pm K^- K^+$). Once again, rarer processes can fool us, but normally, if we have $S = -1$, then that indicates a D^0, and vice-versa.

16.2 Cabibbo-allowed Decays

$c\bar{q} \to s\bar{q}u\bar{d}$
This always gives $S = -1$: a K^- or $\bar{K}^0$ (detected as a K_S). Any other strangeness must be

[4] They are possible from second-order weak interactions, but the rates are utterly negligible. The only evidence for such second-order interactions is in meson-mixing, which is a uniquely sensitive interferometer.

created pairwise by $s\bar{s}$ popping.[5]

Note that the separation, familiar to many of us from B physics, into color-allowed and color-suppressed seems to not be very relevant at the lower charm energies, due to large final-state interaction effects.

16.3 Singly-Cabibbo-Suppressed Decay: W-emission

Suppressed by $\mathcal{O}(\tan^2\theta_C) \sim 0.05$ relative to Cabibbo-favored modes.

$c\bar{q} \to d\bar{q}u\bar{d}$

$S = 0$; no net strangeness.

$c\bar{q} \to s\bar{q}u\bar{s}$

Also $S = 0$; no net strangeness, but with an $s\bar{s}$ pair present, without quark-pair popping.

16.4 Doubly-Cabibbo-Suppressed Decay

Suppressed by $\mathcal{O}(\tan^4\theta_C) \sim 0.002$ relative to Cabibbo-favored modes.

DCSD:

$c\bar{q} \to d\bar{q}u\bar{s}$

This always gives $S = +1$: a K^+ or K^0 (detected as a K_S). Any other strangeness must be created pairwise by $s\bar{s}$ popping. DCSD processes cause two interesting effects which we now catalog.

First, they lead to flavor mis-identification of neutral D decays with $K^\pm$ modes. The rate of mis-tagging is of order $\tan^4\theta_c \simeq (0.2 - 0.3)\%$.

In the absence of $D^0 - \bar{D}^0$ mixing, a final state such as $K^-\pi^+$ is an incoherent sum of neutrals from Cabibbo-allowed and DCSD decays. If we add mixing, then mixing followed by a Cabibbo-allowed decay interferes coherently with the DSCD process.

Adding the twist that K_S may be K^0 or $\bar{K}^0$ leads to more twists. There is interference with Cabibbo-allowed decays for $D^0 \to K_S X$ modes. The latter effect invalidates the usual factor-of-two correction between K_S and $K^0(\bar{K}^0)$ decay rates.[6]

Second, Cabibbo-allowed and DCSD decays interfere even for *charged* D decays. Although it doesn't lead to wrong-tagging since we have net charge always, it can affect various games one might play with rates and amplitudes. As an example, consider the decay $D^+ \to K_S pi+$. It may occur as:

- $\bar{K}^0\pi^+$ via $c \to s(u\bar{d})$ plus initial $\bar{d}$ (Cabibbo-allowed)
- $K^0\pi^+$ via $c \to d(u\bar{s})$ plus initial $\bar{d}$ (DCSD)

16.5 Weak Annihilation and W-Exchange

Suppressed by short-distance nature of weak interaction: quarks "need to overlap":
This gives a suppression of order $\sim (f_D/m_D)^2$ (?)
Also have even rarer Cabibbo-suppressed versions.

[5] "Popping" refers to creation of a $q\bar{q}$ pair from the strong interaction; no *net* flavor is created.

[6] Ikaros I.Y. Bigi and H. Yamamoto, Phys. Lett. **B349** 363, (1995). See also CLEO-c analysis work of Steve Stroiney and Chul-su Park.

Weak Annihilation (D^+ only; Cabibbo-allowed version):
$c\bar{d} \to u\bar{d}q\bar{q}$
$S = 0$; no net strangeness.

Cabibbo-suppressed weak annihilation: $c\bar{d} \to u\bar{s}q\bar{q}$
$S = +1$.

W-Exchange (D^0 only; Cabibbo-allowed version):
$c\bar{u} \to s\bar{d}q\bar{q}$
$S = -1$.

Singly-Cabibbo-suppressed W-Exchange:
$c\bar{u} \to d\bar{d}q\bar{q}$ or $c\bar{u} \to d\bar{d}q\bar{q}$
$S = 0$

Doubly-Cabibbo-suppressed W-Exchange:
$c\bar{u} \to d\bar{s}q\bar{q}$
$S = +1$.

16.6 Penguin and Penguin Annihilation

This is more for some sort of obsessive but not so relevant completeness. In B decays, penguins are dominated by local operators obtained from integrating out the top loop. Here, the b almost decouples due to small CKM factors, and the s loops are not really describable by naive short-distance diagrams. Also, B penguins are $\mathcal{O}(\lambda^2)$, the same as the dominant $b \to c$ decays in terms of CKM factors. D "penguins" are $\mathcal{O}(\lambda)$, while the favored $c \to s$ decays are $\mathcal{O}(1)$ (in amplitude).

Penguin:
$c\bar{q} \to u\bar{q}q'\bar{q}'$
$S = 0$, suppressed by $\tan^2\theta_C$, loop factors, α_s, and GIM cancellations.

Penguin annihilation: (D^0 only)
$c\bar{u} \to q\bar{q}q'\bar{q}'$
$S = 0$, same suppression as above.

16.7 Other Oddities

The modes $K_S K^-\pi^+$ and $K_S K^+\pi^-$ are both accessible from both D^0 and $\bar{D}^0$. If we list both explicitly, then the single-tag lists double-count every combination. On the other hand, one would also consider the effect on the double-tag lists.

As discussed above, we will only list one, since this keeps single tags simple. The algorithm for generating double tags will be updated to accommodate this new class of decays.

17 Decay Processes: Software

Let's consider some general issues first, fix terminology, and point out at least one interesting subtlety.

All modes are either *flavor-tagged (FT)* or *non-flavor-tagged (NFT)*. In addition, all modes are either *self-conjugate (SC)* or *non-self-conjugate (NSC)*. Each is a pair of non-overlapping

categories that includes every mode. From a logical standpoint, they are separate, and four combinations exist. Charge D's are always both FT and NSC due to the net electric charge; the other three combinations are not possible for them. Only the neutral D's can get non-trivial: the question is what the physics allows. To good approximation, neutral decays are mostly *FT, NSC* or *NFT, SC*. But the correspondence of these two different partitioning of decay modes is not exact.

The first, and more interesting issue, is if all four $(FT, NFT) \times (SC, NSC)$ combinations exist for neutral D's. Let's do it with examples:
• FT, NSC: Exists, like $K^-\pi^+$, $K^-\pi^-\pi^+\pi^+$, etc.
• NFT, SC: Exists, like K^-K^+, $\pi^+\pi^-\pi^0$, etc.
• NFT, NSC Exists! $K_SK^-\pi^+$, for example.
• FT, SC: Impossible!
We later abbreviate the first three cases more concisely as 'f', 's', and 'n', respectively. We conclude that while SC implies NFT, NFT *does not* imply SC. The two categorizations are almost the same, except for these curious modes.[7]

Second, we note that the existence of Doubly-Cabibbo-suppressed decays means that sometimes the flavor-tagging for neutral modes is incorrect. Any neutral flavor-tagged decay has a DCSD counterpart. This leads to a small (and known, in the more common modes) mis-tag rate. There are also implications for Double Tags discussed below. Charged DCSD decays do not cause such confusion and may be treated as normal decays.

17.1 DChain and Conjugation

For the next two sections, it is helpful to look at what `DDoubleTagProd` does in `$C3_CVSSRC/DDoubleTagProd/Class/DDoubleTagListProxy.cc`.

Let's start with a simple example. Imagine we've defined exactly two decay modes:
• $D^0 \to K^-\pi^+$
• $D^0 \to K^-\pi^+\pi^0$
The list of candidates for each mode will come in two pieces: a sublist for the mode, as defined, and then a second sublist for its charge-conjugate. (If the mode is self-conjugate, DChain notices, and the second part of the list is empty.)

We can make a list of all D^0 decays by adding the list of candidates for $K\pi$ to the $K\pi\pi^0$ using the over-loaded + operation. Let's call that list `D0List`. It's best to think of it still as two sublists: the first one is D^0 decays in both modes; the second sublist is the conjugate decays in both modes.

We can then look for $D^0\bar{D}^0$ pairs by using the over-loaded $*$ operator:
```
DoubleTagList = D0List*D0List.bar
```
Note the ".bar"; this gives us $D^0\bar{D}^0$ combinations; without it, we would get D^0D^0 instead.

What the $*$ does is to pair all members of the first sublist with all members of the second sublist, and keep those without duplicated tracks and showers. That is, we find:
• $K^-\pi^+$ vs $K^+\pi^-\pi^0$
• $K^-\pi^+\pi^0$ vs $K^+\pi^-$

[7] If you draw the diagrams, the difference is whether a quark pair is popped when the W hadronizes (i.e., $W- > K^*$), or if it is popped between the spectator and the line from the decaying c quark. I.e., it is not obvious that one dominates enough to treat it like DCSD and claim it can be mostly tagged correctly...

• $K^-\pi^+$ vs $K^+\pi^-$
• $K^-\pi^+\pi^0$ vs $K^+\pi^-\pi^0$

For single tags, we do not list DCSD decays separately, since that would cause double-counting if one was not careful, and be swamped by Cabibbo-allowed decays of the other flavor of meson. Similar considerations apply for'n' modes like $K_S K^-\pi^+$: adding its charge-conjugate as a separate mode of D^0 would cause double-counting. And neither is useful for flavor-tagging. The more subtle issues arise when considering double tags.

The relevant thing for double tags is the possibility of decays like
• $D^0 \to K^-\pi^+, \bar{D}^0 \to K^-\pi^+$ (one DCSD)
• $D^0 \to K^-\pi^+, D^0 \to K^-\pi^+$ (mixing)
In reality, the quantum mechanics of the $\psi(3770)$ system forbids this to be a DCSD process, but rather it would be a mixing signal.

17.2 Correct Creation of Double Tags

Recall our short-hand for decay types:
f = flavor-tagged, and hence non-self-conjugate, modes
s = self-conjugate, and hence non-flavor-tagged, modes
n = non-flavor-tagged, non-self-conjugate modes.

Let's ignore at first n-type decays and DCSD for simplicity. We then have only f and s types, and we create DoubleTags as:[8]

$$f * f.bar + s * s.bar + f * s.bar$$

Since $f * s$ is not self-conjugate, DChain will generate $f.bar * s$ as well. There is no need for $s * f$; this would double-count.

For completeness, we note that adding flavored to non-flavored modes is not handled by the software; it may compile, but it crashes. So it's not relevant to speculate what combination operations on previously summed lists would do; e.g., $(f + s) * (f + s).bar$.

Now, let's add in the n-type decays and DCSD. They act like s in the sense that they can be used as D^0 or $\bar{D}^0$ modes, but they are like f in that they have a 2-part list which includes conjugates. Recall that we list, for example, $D^0 \to K_S K^-\pi^+$, but not $D^0 \to K_S K^+\pi^-$ as a single-tag to avoid double-counting in the those lists. The correct way to add n type decays is then:

$$\begin{aligned}&(s * s.bar + f * f.bar + n * n.bar)\\ &+(s * f.bar + s * n.bar + f * n.bar + f * f + f * n + n * n)\end{aligned}$$

where we have separated self-conjugate *pairs* from non-self-conjugate *pairs*. Note that for s and n decays, we cannot tell D^0 from $\bar{D}^0$, even though the logic above may make it look like the decay is a specific flavor! Thus, the $f * f$ term is either mixing or DCSD. The $f * n + n * n$ look like $D^0 D^0$ but are most likely just $D^0\bar{D}^0$ decays confused by the fact that we don't list both conjugations of n as D^0 or $\bar{D}^0$ decay, but rather one as each, to avoid double-counting the single-tags.

For handling DCSD/mixed Double Tags, the new term, $f * f$ will give $K^-\pi^+$ vs. $K^-\pi^+$, etc., as well as the conjugate. This probably gives us more combinations than will be useful,

[8]Note that s and $s.bar$ are the same final state; the interpretation of the parent does change from D^0 to $\bar{D}^0$, however.

but it is logically correct and hence includes everything of interest. *Users should be careful that the understand how the DDoubleTag lists are made and be sure they use them correctly!*

As coded, cases that appear to be the same flavor on both sides (due to one DCSD decay or mixing) will appear in the DDoubleTag lists as D^0 vs. D^0, or $\bar{D}^0$ vs. $\bar{D}^0$. That is, they are *named* as if they are due to $D^0 - \bar{D}^0$ mixing, due to the methods of DDoubleTag creation as described above. One can alternatively interpret them as a DCSD decay vs. a Cabibbo- allowed decay.[9] Every common Cabibbo-allowed vs. Cabibbo-allowed decay may also be interpreted as DCSD vs. DCSD, but this is both very unlikely and indistinguishable [10]. We also note that terms like $f*s$ or $f*n$ (included above) cannot be used to separate DCSD from Cabibbo-allowed decays since the other side is not tagged.

18 K_S, π^0, η, Wide and Narrow Resonances, ...

Common wide resonances include: K^*, ρ, a_1. These all decay to K's and π's and are included in the appropriate modes with the K, π explicitly listed.[11]

The decays $K_S \to \pi^+\pi^-$; $\pi^0 \to \gamma\gamma$ are special, and are the only ones used for these particles. The only caveat is to be aware of cross-feed and double-counting for modes that differ only in replacing K_S with $\pi^+\pi^-$.

Finally, there are narrow resonances with hadronic final states, like η, ω, η', ϕ. Once again, these show up mostly in $(nK)(m\pi)$ final states; e.g., $\phi\pi^+$ appears as a subset of $K^-K^+\pi^+$, etc.

The common (39%) $\eta \to \gamma\gamma$ decay is treated separately, since it adds appreciable statistics to the $\eta \to 3\pi$ decay mode. This is done only for large or interesting channels.

The other significant non-$(nK)(m\pi)$ decays of narrow resonances are few enough to enumerate explicitly, to see what we are missing by not using these channels:

- $\phi \to K_L K_S$ 34%
- $\eta' \to \pi^+\pi^-\gamma$ 30%
- $\omega \to \pi^0\gamma$ 9%
- $\eta \to \pi^+\pi^-\gamma$ 5%

Given that K_L are hard to detect, the only really significant loss is the exclusion of the η' decay listed.

[9] If the final state is the same on both sides, then the $C = -1$ initial state of $\psi(3770)$ cannot produce it via DCSD, and it must be mixing. This is not true of the "off-diagonal" combinations of different modes: i.e., $D^0 \to K^+\pi^-$ vs. $\bar{D}^0 \to K^+\pi^-\pi^0$ is allowed, while$D^0 \to K^+\pi^-$ vs. $\bar{D}^0 \to K^+\pi^-$ is not.

[10] But since they interfere, the effect of this rare process on the rates is $\mathcal{O}\tan^2\theta_C$, and not $\mathcal{O}\tan^4\theta_C$

[11] One could argue that certain high-multiplicity modes that are omitted might be more interesting if restricted to cases where some subset of particles is resonant...

Table 10: Quark-Level Decay Processes. For cases where only a D^0 or a D^+ participates, the flavor of the spectator quark is explicit.

Type	Quark Process	S	Suppression
Cabibbo-allowed	$c\bar{q} \to s\bar{q}ud$	-1	–
SCSD	$c\bar{q} \to d\bar{q}ud$	0	$\tan^2\theta_C$
SCSD	$c\bar{q} \to s\bar{q}u\bar{s}$	0	$\tan^2\theta_C$
DCSD	$c\bar{q} \to d\bar{q}u\bar{s}$	$+1$	$\tan^4\theta_C$
Weak Annih.	$cd \to udq\bar{q}$	0	$(f_D/m_D)^2$
W-Exchange	$c\bar{u} \to s\bar{d}q\bar{q}$	-1	$(f_D/m_D)^2$
SCS Weak Annih.	$cd \to u\bar{s}q\bar{q}$	$+1$	$(f_D/m_D)^2\tan^2\theta_C$
SCS W-Exchange	$c\bar{u} \to d\bar{d}q\bar{q}$	0	$(f_D/m_D)^2\tan^2\theta_C$
SCS W-Exchange	$c\bar{u} \to s\bar{s}q\bar{q}$	0	$(f_D/m_D)^2\tan^2\theta_C$
DCS W-Exchange	$c\bar{u} \to d\bar{s}q\bar{q}$	$+1$	$(f_D/m_D)^2\tan^4\theta_C$
Penguin	$c\bar{q} \to u\bar{q}q'\bar{q}'$	0	$\alpha_s\tan^2\theta_C/16\pi^2$,
Penguin Annih.	$c\bar{u} \to q\bar{q}q'\bar{q}'$	0	$\alpha_s\tan^2\theta_C/16\pi^2$,

Table 11: Two-Body Modes; sorted by decay.

$\boldsymbol{D^0}$	**Type**	**Status**	$\boldsymbol{\mathcal{B}\times 10^4}$
$K^-\pi^+$	A,f	12	380(9)
$K_S\pi^0$	A+D,s	12	115(11)
$K^+\pi^-$	D	XX	1.38(0.11)
$\pi^+\pi^-$	S,s	24	13.8(0.5)
$\pi^0\pi^0$	S,s	24	8.4(2.2)
K^-K^+	S,s	24	38.9(1.5)
K_SK_S	S,s	24	1.8(0.5)
$\boldsymbol{D^+}$	**Type**	**Status**	$\boldsymbol{\mathcal{B}\times 10^4}$
$K_S\pi^+$	A	12	141(10)
$K^+\pi^0$	D	55	N/A
$\pi^+\pi^0$	S	55	26(7)
K_SK^+	S	55	30(30)

Table 12: Three-Body Modes; sorted by decay.

$\boldsymbol{D^0}$	**Type**	**Status**	$\boldsymbol{\mathcal{B} \times 10^4}$
$K^-\pi^+\pi^0$	A,f	12	1300(80)
$K_S\pi^+\pi^-$	A+D,s	12	299(18)
$K_S\pi^0\pi^0$	A+D,s	24	
$K^+\pi^-\pi^0$	D	XX	5.6(1.7)
$\pi^+\pi^-\pi^0$	S,s	24	110(40)
$\pi^0\pi^0\pi^0$	S,s	24	
$K_SK_S\pi^0$	S,s	24	< 5.9
$K^-K^+\pi^0$	S,s	24	12.4(3.5)
$K_SK^-\pi^+$	S,n	55	30(5) (chg avg)
$K_SK^+\pi^-$	S,n	XX	30(5) (chg avg)
$K_SK_SK_S$	A,s	55	
$K_SK^-K^+$	A,s	24	
$\boldsymbol{D^+}$	**Type**	**Status**	$\boldsymbol{\mathcal{B} \times 10^4}$
$K^-\pi^+\pi^+$	A	12	920(60)
$K_S\pi^+\pi^0$	A	12	485(150)
$K^+\pi^+\pi^-$	D	?	7.0(1.5)
$K^+\pi^0\pi^0$	D	?	
$\pi^+\pi^+\pi^-$	S	55	31(4)
$\pi^+\pi^0\pi^0$	S	55	
$K^-K^+\pi^+$	S	24	
$K_SK_S\pi^+$	S	55	
$K_SK^+\pi^0$	S	55	
$K_SK_SK^+$	A+D	55	45(20)
$K^-K^+K^+$	D	?	0.9(0.1)

Table 13: Four-Body Modes; sorted by decay.

D^0	Type	Status	$\mathcal{B} \times 10^4$
$K^-\pi^+\pi^+\pi^-$	A,f	12	746(31)
$K^-\pi^+\pi^0\pi^0$	A,f	12	
$K_S\pi^+\pi^-\pi^0$	A+D,s	12	
$K_S\pi^0\pi^0\pi^0$	A+D,s		
$K^+\pi^+\pi^-\pi^-$	D	XX	3.1(1.0)
$K^+\pi^-\pi^0\pi^0$	D	XX	
$\pi^+\pi^+\pi^-\pi^-$	S,s	55	73(5)
$\pi^+\pi^-\pi^0\pi^0$	S,s	55	
$\pi^0\pi^0\pi^0\pi^0$	S,s		
$K_SK^-\pi^+\pi^0$	S,n	55	
$K_SK^+\pi^-\pi^0$	S,n	XX	
$K^-K^+\pi^+\pi^-$	S,s	55	24.9(2.3)
$K^-K^+\pi^0\pi^0$	S,s	55	
$K_SK_S\pi^+\pi^-$	S,s	55	
$K_SK_S\pi^0\pi^0$	S,s		
$K^-K^-K^+\pi^+$	A,f	?	0.31(0.14) (non-res)
$K_SK_SK^-\pi^+$	A+D,n	?	
$K_SK_SK^+\pi^-$	A+D,n	?	
$K_SK^-K^+\pi^0$	A+D,s	?	
$K_SK_SK_S\pi^0$	A+D,s		
$K^-K^+K^+\pi^-$	D		
D^+	Type	Status	$\mathcal{B} \times 10^4$
$K^-\pi^+\pi^+\pi^0$	A	12	650(110)
$K_S\pi^+\pi^+\pi^-$	A+D	12	355(50)
$K_S\pi^+\pi^0\pi^0$	A+D	55	
$K^+\pi^+\pi^-\pi^0$	D	?	
$K^+\pi^0\pi^0\pi^0$	D		
$\pi^+\pi^+\pi^-\pi^0$	S	55	6.8(1.4) (η only)
$\pi^+\pi^0\pi^0\pi^0$	S		
$K^-K^+\pi^+\pi^0$	S	55	
$K_SK^-\pi^+\pi^+$	S	55	
$K_SK^+\pi^+\pi^-$	S	55	
$K_SK^+\pi^0\pi^0$	S		
$K_SK_S\pi^+\pi^0$	S		
$K_SK_SK_S\pi^+$	A+D		
$K_SK_SK^+\pi^0$	A+D		
$K_SK^-K^+\pi^+$	A+D	?	2.8(0.7)
$K_SK^+K^+\pi^-$	D		
$K^-K^+K^+\pi^0$	D		

Table 14: Five-Body Modes; sorted by decay.

D^0	Type	Status	$\mathcal{B} \times 10^4$
$K^-\pi^+\pi^+\pi^-\pi^0$	A,f	55	400(40)
$K^-\pi^+\pi^0\pi^0\pi^0$	A,f		
$K_S\pi^+\pi^+\pi^-\pi^-$	A+D,s	55	32(9)
$K_S\pi^+\pi^-\pi^0\pi^0$	A+D,s		
$K_S\pi^0\pi^0\pi^0\pi^0$	A+D,s		
$K^+\pi^+\pi^-\pi^-\pi^0$	D		
$K^+\pi^-\pi^0\pi^0\pi^0$	D		
$\pi^+\pi^+\pi^-\pi^-\pi^0$	S,s	55	
$\pi^+\pi^-\pi^0\pi^0\pi^0$	S,s		
$\pi^0\pi^0\pi^0\pi^0\pi^0$	S,s		
$K^-K^+\pi^+\pi^-\pi^0$	S,s	55	31(20)
$K^-K^+\pi^0\pi^0\pi^0$	S,s		
$K_SK_S\pi^+\pi^-\pi^0$	S,s		
$K_SK_S\pi^0\pi^0\pi^0$	S,s		
$K_SK^-\pi^+\pi^+\pi^-$	S,n	?	
$K_SK^+\pi^+\pi^-\pi^-$	S,n	?	
$K_SK^-\pi^+\pi^0\pi^0$	S,n		
$K_SK^+\pi^-\pi^0\pi^0$	S,n		
$KKK\pi\pi$			
D^+	Type	Status	$\mathcal{B} \times 10^4$
$K^-\pi^+\pi^+\pi^+\pi^-$	A	55	62(8)
$K^-\pi^+\pi^+\pi^0\pi^0$	A	?	
$K_S\pi^+\pi^+\pi^-\pi^0$	A+D	?	
$K_S\pi^+\pi^0\pi^0\pi^0$	A+D		
$K^+\pi^+\pi^+\pi^-\pi^-$	D		
$K^+\pi^+\pi^-\pi^0\pi^0$	D		
$K^+\pi^0\pi^0\pi^0\pi^0$	D		
$\pi^+\pi^+\pi^+\pi^-\pi^-$	S	55	18.2(2.5)
$\pi^+\pi^+\pi^-\pi^0\pi^0$	S	?	
$\pi^+\pi^0\pi^0\pi^0\pi^0$	S		
$K^-K^+\pi^+\pi^+\pi^-$	S	?	2.5(1.3)
$K^-K^+\pi^+\pi^0\pi^0$	S	?	
$K_SK_S\pi^+\pi^+\pi^-$	S	?	
$K_SK_S\pi^+\pi^0\pi^0$	S		
$K_SK^+\pi^+\pi^-\pi^0$	S	?	
$K_SK^+\pi^0\pi^0\pi^0$	S		
$K_SK^-\pi^+\pi^+\pi^0$	S	?	
$KKK\pi\pi$			

Table 15: Select Six-Body Modes; sorted by decay (BR's from FOCUS hep-ex/0401019).

$\boldsymbol{D^0}$	**Type**	**Status**	$\boldsymbol{\mathcal{B} \times 10^4}$
$K^-\pi^+\pi^+\pi^+\pi^-\pi^-$	A,f	?	2.0(0.5)
$\pi^+\pi^+\pi^+\pi^-\pi^-\pi^-$	S,s	?	3.9(1.1)

Table 16: Select Eta Modes; sorted by decay.

$\boldsymbol{D^0}$	**Type**	**Status**	$\boldsymbol{\mathcal{B} \times 10^4}$
$K_S\eta$	A+D,s	24	39(6)
$K^-\pi^+\eta$	A,f	55	
$K_S\pi^0\eta$	A+D,s	55	
$\boldsymbol{D^+}$	**Type**	**Status**	$\boldsymbol{\mathcal{B} \times 10^4}$
$K_S\pi^+\eta$	A	55	
$K^+\eta$	D	?	
$\pi + \eta$	S	55	30(6)

Table 17: Analyzed Neutral Flavored Modes; sorted by enumeration code.

$\boldsymbol{D^0}$	**Type**	**Status**	$\boldsymbol{\mathcal{B} \times 10^4}$	**enum**	**widths (MeV)**
$K^-\pi^+$	A,f	12	380(9)	0	1.6/7.5
$K^-\pi^+\pi^0$	A,f	12	1300(80)	1	2.0/12
$K^-\pi^+\pi^0\pi^0$	A,f	12		2	2.4/14
$K^-\pi^+\pi^+\pi^-$	A,f	12	746(31)	3	1.6/5.0
$K^-\pi^+\pi^+\pi^-\pi^0$	A,f	55	400(40)	4	1.7/6.6
$K^-\pi^+\eta$	A,f	55	$> 119(27)$	5	2.1/8.1

Table 18: Analyzed Neutral Unflavored, non-Self-Conjugate Modes; sorted by enumeration code.

$\boldsymbol{D^0}$	**Type**	**Status**	$\boldsymbol{\mathcal{B} \times 10^4}$	**enum**	**widths (MeV)**
$K_SK^-\pi^+$	S,n	55	30(5) (chg avg)	50	1.5/4.9
$K_SK^-\pi^+\pi^0$	S,n	55		51	1.7/7.0

Table 19: Analyzed Neutral Unflavored, Self-Conjugate Modes; sorted by enumeration code.

D^0	Type	Status	$\mathcal{B} \times 10^4$	enum	widths (MeV)
$K_S\pi^+\pi^-$	A,s	12	299(18)	100	1.5/6.2
$K_S\pi^+\pi^-\pi^0$	A,s	12	540(65)	101	1.9/8.8
$K_S\pi^0$	A,s	12	115(11)	102	2.2/16
$\pi^+\pi^-\pi^0$	S,s	24	110(40)	103	1.9/13
$\pi^+\pi^-$	S,s	24	13.8(0.5)	104	1.4/8.0
K^-K^+	S,s	24	38.9(1.5)	105	1.5/5.9
$K^-K^+\pi^0$	S,s	24	12.4(3.5)	106	1.5/13 ?
$\pi^0\pi^0$	S,s	24	8.4(2.2)	107	4/20 ?
K_SK_S	S,s	24	1.8(0.5)	108	1.5/9 ?
$K_SK_S\pi^0$	S,s	24	< 5.9	109	2.4/10 ?
$K_S\pi^0\pi^0$	A,s	24	$> 46(7)$	110	3.2/19
$K_SK^-K^+$	A+ss,s	24	51(10)	111	2.0/3.7
$K_S\eta$	A,s	24	39(6)	112	2.1/12 ?
$\pi^0\pi^0\pi^0$	S,s	24		113	2/20 ?
$K_SK_SK_S$	A+ss,s	55	9.2(1.6)	114	1.0/12 *
$\pi^+\pi^+\pi^-\pi^-$	S,s	55	73(5)	115	1.5/6.0
$\pi^+\pi^-\pi^0\pi^0$	S,s	55		116	2.4/15
$K^-K^+\pi^+\pi^-$	S,s	55	24.9(2.3)	117	1.5/3.5
$K^-K^+\pi^0\pi^0$	S,s	55		118	1.6/15 *
$K_SK_S\pi^+\pi^-$	S,s	55	19(7)	119	1.5/4.0
$\pi^+\pi^+\pi^-\pi^-\pi^0$	S	55		120	1.6/12 *
$K_S\pi^+\pi^+\pi^-\pi^-$	A,s	55	32(9)	121	2/5
$K^-K^+\pi^+\pi^-\pi^0$	S,s	55	31(20)	122	1.6/12 *
$K_S\pi^0\eta$	A,s	55		123	2/10

Table 20: Analyzed Charged Modes; sorted by enumeration code.

D^+	Type	Status	$\mathcal{B} \times 10^4$	enum	widths (MeV)
$K^-\pi^+\pi^+$	A	12	920(60)	200	1.6/6.1
$K^-\pi^+\pi^+\pi^0$	A	12	650(110)	201	1.8/9.6
$K_S\pi^+$	A	12	141(10)	202	1.5/6.7
$K_S\pi^+\pi^0$	A	12	485(150)	203	1.8/13
$K_S\pi^+\pi^+\pi^-$	A	12	355(50)	204	1.5/4.9
$K^-K^+\pi^+$	S	24		205	1.5/4.8
$\pi^+\pi^0$	S	55	26(7)	206	2/12 *
$K^+\pi^0$	D	55	N/A	207	2/12 *
K_SK^+	S	55	30(30)	208	1.2/6.0
$\pi^+\pi^+\pi^-$	S	55	31(4)	209	1.3/6.0
$\pi^+\pi^0\pi^0$	S	55		210	2.5/15 *
$K_SK_S\pi^+$	S	55		211	1.4/5
$K_SK^+\pi^0$	S	55		212	2.1/20?
$K_SK_SK^+$	A+ss	55	45(20)	213	1.6/4
$\pi^+\pi^+\pi^-\pi^0$	S	55	6.8(1.4) (η only)	214	1.5/11
$K_S\pi^+\pi^0\pi^0$	A	55		215	1.9/12
$K_SK^+\pi^+\pi^-$	S	55		216	1.6/5 *
$K_SK^-\pi^+\pi^+$	S	55		217	1.7/4.0
$K^-K^+\pi^+\pi^0$	S	55		218	2/10 *
$\pi^+\pi^+\pi^+\pi^-\pi^-$	S	55	18.2(2.5)	219	1.4/4.7
$K^-\pi^+\pi^+\pi^+\pi^-$	A	55	62(8)	220	1.4/3.9
$\pi^+\eta$	S	55	30(6)	221	1.6/6 *
$K_S\pi^+\eta$	A	55		222	1.6/6