The BESIII `FSFilter` Package (version 00-00-00, tag dev)

Ryan Mitchell

October 6, 2017

Abstract

This document describes how to use the `FSFilter` package to simultaneously reconstruct multiple final states at BESIII. Final states can be reconstructed either exclusively or inclusively. The output of the `FSFilter` package is a set of root trees that can then be used as the basis for further analysis.

Contents

1 Purpose

FSFilter is a package that can be used to convert BESIII data into a format that can be more easily analyzed. FSFilter "**Filter**s" information from many "**F**inal **S**tates" into root trees. Any number of final states can be selected using input parameters in the job options file (a file to control programs running in the BESIII BOSS framework). The final states can be reconstructed either exclusively (in which case a kinematic fit to the initial four-momentum is performed) or inclusively (in which case a range of missing masses can be specified or a kinematic fit to missing mass can be performed). Only

very loose cuts are applied at this stage – it is assumed that most of the analysis will be done using the output root trees.

2 Overview

The `FSFilter` algorithm follows four basic steps:

1. Lists of particles are made according to the loose selection criteria described in section 3. The particles considered are Λ (decaying to $p\pi^-$), $\overline{\Lambda}$ (decaying to $\overline{p}\pi^+$), e^+, e^-, μ^+, μ^-, p, $\overline{p}$, η (decaying to $\gamma\gamma$), γ, K^+, K^-, K^0_S (decaying to $\pi^+\pi^-$), π^+, π^-, and π^0 (decaying to $\gamma\gamma$). Tracks and showers can appear in multiple lists at this stage.
2. The particle lists are combined to form a specified final state. All combinations are considered. Here, tracks and showers can only be used once per combination to avoid double-counting.
3. If a final state is being exclusively reconstructed, a kinematic fit to the initial four-momentum is performed. For inclusive reconstruction, missing mass squared cuts and fits can be specified. This is described further in section 4.
4. Finally, the resulting information is written out to a series of root trees, described in section 8.

In the following documentation, the particle selection criteria (section 3) and event selection criteria (section 4) are first discussed. Then a number of details are given regarding the final state numbering scheme (section 5) and the job options files (sections 6 and 7). The contents of the output root trees are listed in section 8. Finally, different versions of the code, and where they are located, are documented in section 9.

3 Particle Selection

The `FSFilter` algorithm begins by creating lists of final state particles according to the selection criteria discussed in this section.

3.1 Tracks ($e^\pm, \mu^\pm, \pi^\pm, K^\pm, p^\pm$)

Tracks are selected using a combination of the `RecMdcKalTrack` and `ParticleID` packages. No additional requirements are made on the track quality, momentum or angles beyond those made within the `RecMdcKalTrack` package. For particle identification, the `ParticleID` package is set up as follows:

```
ParticleID* pid = ParticleID::instance();
```

```
pid->init();
pid->setMethod(pid->methodProbability());
pid->setRecTrack(*iTrk);
pid->usePidSys(pid->useDedx()  |
               pid->useTof1()  |
               pid->useTof2());
pid->identify(pid->onlyPion()     |
              pid->onlyKaon()     |
              pid->onlyProton()   |
              pid->onlyElectron() |
              pid->onlyMuon());
pid->calculate();
```

Very loose cuts are then placed on the probabilities of different hypotheses. For a track to be counted as a $(e^{\pm}, \mu^{\pm}, \pi^{\pm}, K^{\pm}, p^{\pm})$ the probability of the $(e^{\pm}, \mu^{\pm}, \pi^{\pm}, K^{\pm}, p^{\pm})$ hypothesis must be greater than 10^{-5} (but this requirement can be dropped using the `trackStudies` parameter, section 6.11). Note that tracks that satisfy more than one hypothesis are included in multiple lists. All possible combinations are considered when forming events (section 4).

3.2 Showers (γ)

Showers are reconstructed using the `RecEmcShower` package. In addition to the requirements imposed by the `RecEmcShower` package, the following standard selection criteria must be satisfied:

```
(1) 0 < shower time < 14
(2) |cos(theta)| < 0.80 (barrel photons)
       or 0.86 < |cos(theta)| < 0.92 (endcap photons)
(3) for barrel photons:  E > 25 MeV
(4) for endcap photons:  E > 50 MeV
```

Note that these additional requirements can be turned off using the `neutralStudies` parameter (see section 6.14).

3.3 $\pi^0 \to \gamma\gamma$ and $\eta \to \gamma\gamma$

Lists of $\pi^0 \to \gamma\gamma$ and $\eta \to \gamma\gamma$ decays are made using the `EvtRecPi0` and `EvtRecEtaToGG` packages, respectively. The following additional cuts are imposed:

```
(1) chi2 < 2500 (from the fit to the eta or pi0 mass)
(2) for pi0:  0.107 < mass(gamma gamma) < 0.163 GeV/c2
    for eta:  0.400 < mass(gamma gamma) < 0.700 GeV/c2
```

These default cuts can be removed using the `pi0Studies` or `etaStudies` job options parameters (sections 6.15 and 6.16). Both photons from the π^0 and η are also required to pass the shower cuts listed in section 3.2.

3.4 $K_S^0 \to \pi^+\pi^-$, $\Lambda \to p\pi^-$ and $\overline{\Lambda} \to \overline{p}\pi^+$

$K_S^0 \to \pi^+\pi^-$, $\Lambda \to p\pi^-$ and $\overline{\Lambda} \to \overline{p}\pi^+$ reconstruction is done with the `EvtRecVeeVertex` package with the following additional cuts:

```
(1) chi2 < 100 (from the fit to the Ks or Lambda vertex)
(2) for Ks:      0.471 < mass(pi+ pi-) < 0.524 GeV/c2
    for Lambda:  1.100 < mass(p pi) < 1.300 GeV/c2
```

These default cuts can be removed with the `veeStudies` job options parameter (section 6.17).

4 Event Selection

Once the lists of particles are made, they are combined to form different final states. All combinations are considered. To avoid double-counting, tracks and showers (including tracks and showers from π^0, η, K_S^0, and Λ decays) are used only once per combination. Parameters can be specified in the job options file (see section 6) to reduce the combinatorics.

For both inclusive and exclusive final states, a primary vertex fit (using the `VertexFit` package) is performed using tracks originating from the primary vertex. The primary vertex and the resulting track parameters are saved. If the final state contains no tracks from the primary vertex, the beam spot is used as the primary vertex and no fit is performed. Secondary vertices due to K_S^0 and Λ decays are then created with the `SecondVertexFit` package.

For inclusive final states, bounds on the missing mass or a kinematic fit to the missing mass (using the `KalmanKinematicFit` package) can be specified in the job options file (see section 6). For exclusive final states, the `KalmanKinematicFit` package is used to fit the four-momenta of the final state particles to the initial four-momentum of the e^+e^- interaction. In either case, additional constraints are placed on the π^0, η, K_S^0, and Λ masses. A loose cut (that can be tuned in the job options file) is placed on the resulting χ^2. Multiple combinations can be recorded per event.

5 Final State Numbering

`FSFilter` can reconstruct any final state composed of a combination of Λ (called `Lambda`,

decaying to $p\pi^-$), $\overline{\Lambda}$ (called `ALambda`, decaying to $\overline{p}\pi^+$), e^+, e^-, μ^+, μ^-, p^+ (proton), p^- (anti-proton), $\eta(\rightarrow \gamma\gamma)$, γ, K^+, K^-, $K^0_S(\rightarrow \pi^+\pi^-)$, π^+, π^-, and $\pi^0(\rightarrow \gamma\gamma)$. Final states are designated using two integers, "code1" and "code2". The digits of each integer are used to specify the number of different particle types in the final state:

```
code1 = abcdefg
        a = number of gamma
        b = number of K+
        c = number of K-
        d = number of Ks  ( --> pi+ pi- )
        e = number of pi+
        f = number of pi-
        g = number of pi0 ( --> gamma gamma )

code2 = hijklmnop
        h = number of Lambda (--> p+ pi-)
        i = number of ALambda (--> p- pi+)
        j = number of e+
        k = number of e-
        l = number of mu+
        m = number of mu-
        n = number of p+
        o = number of p-
        p = number of eta  ( --> gamma gamma )
```

These integers are sometimes combined into a single string of the form:

```
"code2_code1"
```

Here are a few examples:

```
"0_111":      pi+ pi- pi0
"0_1000002":  gamma pi0 pi0
"1_220000":   eta K+ K+ K- K-
"11000_110":  mu+ mu- pi+ pi-
```

6 Job Options Parameters

6.1 FSFilter.FS(N) = "(EXC/INC)(tag)(code2)_(code1)";

Use a list of these commands to specify which final states to reconstruct.

```
(N): a unique integer between 0 and 9999 for each
```

```
     reconstructed final state (this is only used to
     differentiate FS(N) commands, and isn't used
     elsewhere)

(EXC/INC): use EXC to reconstruct the final state
           exclusively (using a kinematic fit);
           use INC for inclusive reconstruction

(tag):  an optional string to help further distinguish
        final states (for example, use this if you want
        to apply distinct sets of cuts to the same
        final state)

(code2) and (code1):  the final state numbering integers
                      described in the "final state
                      numbering" section
```

Examples:

```
Reconstruct eta K+ K- exclusively:

    FSFilter.FS100 = "EXC1_110000";

Reconstruct pi+ pi- inclusively:

    FSFilter.FS200 = "INC0_110";
```

6.2 FSFilter.FSCut(N) = "(FS) (submode) (type) (x1) (x2)";

The FSCut parameters can be used to place cuts on intermediate particle masses or recoil masses. This can dramatically decrease file sizes when studying final states with intermediate resonances.

```
(N): a unique integer between 0 and 9999 for each
     cut (this is only used to differentiate cuts,
     and isn't used elsewhere)

(FS): the name of the final state in which this cut
      should be applied (this corresponds to the
      "(EXC/INC)(tag)(code2)_(code1)" name of the
      previous section)

(submode): the particle subsystem that will be used
           in the cut, with the format "(code2)_(code1)"
           where the codes are described in the
           "final state numbering" section
```

```
(type): the type of cut -- options are:
        "RawMass", "IntMass", "FitMass",
           (for mass selections using raw, intermediate
             or fitted four-vectors)
        "RawRecoil", "IntRecoil", "FitRecoil",
           (for recoil mass selections)
        or any of the above with "Squared" appended
           (for mass squared or recoil mass squared)

(x1):  the lower limit of the cut in GeV or GeV^2

(x2):  the upper limit of the cut in GeV or GeV^2
```

Examples:

```
To select phi eta from the K+ K- pi+ pi- pi0 final state:

    FSFilter.FS1 = "EXC0_110111";
    FSFilter.FSCut11 = "EXC0_110111 0_110000 FitMass 0.9 1.1";
    FSFilter.FSCut21 = "EXC0_110111 0_111    FitMass 0.4 0.7";

To look for psi(2S) --> pi+ pi- J/psi inclusively:

    FSFilter.FS2 = "INC0_110";
    FSFilter.FSCut12 = "INC0_110 0_110 RawRecoil 3.0 3.2";
```

In cases where multiple intermediate particle combinations can be formed in a given final state, all combinations are checked, and the event passes if any of the combinations pass the cut.

6.3 FSFilter.FSMMFit(N) = "(FS) (mass)";

(INCLUSIVE RECONSTRUCTION ONLY) For inclusive final states, perform a kinematic fit to the missing mass (MM) using the `FSMMFit` parameters.

```
(N): a unique integer between 0 and 9999 for each
     cut (this is only used to differentiate parameters,
     and isn't used elsewhere)

(FS): the name of the final state in which this cut
      should be applied (this corresponds to the
      "(EXC/INC)(tag)(code2)_(code1)" name of the
      FS(N) parameter)
```

```
(mass): the mass to which the missing mass will be
        constrained (for example, the neutron mass)
```

Example:

```
To reconstruct J/psi --> p+ anti-n pi- with a missing
anti-neutron use:

    FSFilter.FS1 = "INC100_10";
    FSFilter.FSMMFit1 = "INC100_10 0.9396";
```

6.4 FSFilter.maxShowers = (n);

Only consider events with less than or equal to `(n)` showers. The default is 50. The purpose of this requirement is just to speed up reconstruction. When there are more than 50 showers in an event, the possible combinations of showers used to form photons or π^0 or η can become very large.

6.5 FSFilter.maxExtraTracks = (n);

(EXCLUSIVE RECONSTRUCTION ONLY) For exclusive reconstruction, only allow `(n)` extra tracks (tracks beyond those required in the final state) in the event. The default is 2. This requirement is ignored for inclusively reconstructed final states.

6.6 FSFilter.cmEnergy = (E);

Specify the center of mass energy `(E)` in units of GeV. The default is the $\psi(2S)$ mass, 3.686093 GeV. (Eventually the default will come from an external `BeamEnergy` package.) If this is set to -1, the run number is used to set the center of mass energy, but this currently only works for a limited number of run ranges.

6.7 FSFilter.crossingAngle = (theta);

Set the crossing angle in radians. The default is 0.011 radians.

6.8 FSFilter.energyTolerance = (deltaE);

(EXCLUSIVE RECONSTRUCTION ONLY) For exclusive reconstruction, only combinations of final state particles that have a total energy within `(deltaE)` of the initial

total energy will be processed. `(deltaE)` is given in GeV. This reduces the number of kinematic fits that are performed and helps speed processing. The default is 0.250 GeV.

6.9 `FSFilter.momentumTolerance = (deltaP);`

(EXCLUSIVE RECONSTRUCTION ONLY) Like `energyTolerance`, this command requires the total momentum of final state particles be within `(deltaP)` (in GeV/c) of the initial total momentum. The default is 0.250 GeV/c.

6.10 `FSFilter.maxKinematicFitChi2DOF = (chi2DOF);`

For exclusive reconstruction, or inclusive reconstruction that includes a kinematic fit, only record events that have a kinematic fit χ^2/dof less than `(chi2DOF)`. Using this can reduce file sizes. The default is 100.

6.11 `FSFilter.trackStudies = (true/false);`

If `true`, remove all of the default track selection criteria listed in section 3.1. The default is `false`.

6.12 `FSFilter.extraKaonPID = (true/false);`

If `true`, require kaons have PID probability greater than 10^{-5} and require the kaon hypothesis probability be greater than the pion probability. The default is `false`.

6.13 `FSFilter.pidStudies = (true/false);`

If `true`, include additional PID information in the root tree. The default is `false`.

6.14 `FSFilter.neutralStudies = (true/false);`

If `true`, remove all of the default photon selection criteria listed in section 3.2. The default is `false`.

6.15 `FSFilter.pi0Studies = (true/false);`

If `true`, remove all of the default π^0 selection criteria listed in section 3.3. The default is `false`.

6.16 FSFilter.etaStudies = (true/false);

If `true`, remove all of the default η selection criteria listed in section 3.3. The default is `false`.

6.17 FSFilter.veeStudies = (true/false);

If `true`, remove all of the default K_S and Λ selection criteria listed in section 3.4. The default is `false`.

6.18 FSFilter.bypassVertexDB = (true/false);

If `true`, zero the beam vertex information. This is useful for testing data before vertex information is available. The default is `false`.

6.19 FSFilter.YUPING = (true/false);

If `true`, use the kaon and pion tracking corrections developed by Yuping Guo. The default is `false`.

6.20 FSFilter.writeNTGen = (true/false);

If `true`, write the global truth information tree (`ntGEN`, described in section 8). The default is `true`. If you have already run over MC, setting this to `false` can reduce file sizes.

6.21 FSFilter.writeNTGenCode = (true/false);

If `true`, write the truth information associated with every mode (the tree named `ntGEN(tag)(code2)_(code1)`, described in section 8). The default is `true`.

6.22 Parameters for Debugging

6.22.1 FSFilter.printTruthInformation = (true/false);

If `true`, print MC truth information to the screen. This can be useful for debugging. The default is `false`.

6.22.2 `FSFilter.isolateRunLow = (run);`

Only process runs with run number greater than or equal to `run`. This also turns on some debugging output. The default is -1, which means it is not used.

6.22.3 `FSFilter.isolateRunHigh = (run);`

Only process runs with run number less than or equal to `run`.

6.22.4 `FSFilter.isolateEventLow = (event);`

Only process events with event number greater than or equal to `event`.

6.22.5 `FSFilter.isolateEventHigh = (event);`

Only process events with event number less than or equal to `event`.

7 Example Job Options File

Here is a job options file to exclusively reconstruct $\psi(2S) \to \gamma X_i$, where X_i is:

1. $\pi^+\pi^+\pi^-\pi^-$
2. $\pi^+\pi^+\pi^-\pi^-\pi^0\pi^0$
3. $\pi^+\pi^+\pi^+\pi^-\pi^-\pi^-$
4. $K^-K^0_S\pi^+$
5. $K^+K^0_S\pi^-$
6. $K^+K^-\pi^0$
7. $K^+K^-\pi^+\pi^-$
8. $K^-K^0_S\pi^+\pi^+\pi^-$
9. $K^+K^0_S\pi^+\pi^-\pi^-$
10. $K^+K^-\pi^+\pi^-\pi^0$
11. $K^+K^+\pi^+\pi^+\pi^-\pi^-$
12. $K^+K^+K^-K^-$
13. $\eta\pi^+\pi^-$
14. $\eta\pi^+\pi^+\pi^-\pi^-$

and to search for $\psi(2S) \to \gamma\eta_c(1S)$ using inclusive photons:

```
// ************************************************
//      EXAMPLE FSFILTER JOB OPTIONS FILE
// ************************************************

#include "$ROOTIOROOT/share/jobOptions_ReadRec.txt"
#include "$VERTEXFITROOT/share/jobOptions_VertexDbSvc.txt"
#include "$MAGNETICFIELDROOT/share/MagneticField.txt"
#include "$ABSCORROOT/share/jobOptions_AbsCor.txt"
#include "$PI0ETATOGGRECALGROOT/share/jobOptions_Pi0EtaToGGRec.txt"
#include "$FSFILTERROOT/share/jobOptions_FSFilter.txt"

FSFilter.cmEnergy = 3.686093;

FSFilter.FS1  = "EXC0_1000220";
FSFilter.FS2  = "EXC0_1000222";
FSFilter.FS3  = "EXC0_1000330";
FSFilter.FS4  = "EXC0_1011100";
FSFilter.FS5  = "EXC0_1101010";
FSFilter.FS6  = "EXC0_1110001";
FSFilter.FS7  = "EXC0_1110110";
FSFilter.FS8  = "EXC0_1011210";
FSFilter.FS9  = "EXC0_1101120";
FSFilter.FS10 = "EXC0_1110111";
FSFilter.FS11 = "EXC0_1110220";
FSFilter.FS12 = "EXC0_1220000";
FSFilter.FS13 = "EXC1_1000110";
FSFilter.FS14 = "EXC1_1000220";

FSFilter.FS100 =    "INC0_1000000";
FSFilter.FSCut101 = "INC0_1000000 0_1000000 RawRecoil 2.6 3.2";

// Input REC or DST file name
EventCnvSvc.digiRootInputFile = {"/bes3fs/offline/...."};

// Set output level threshold
// (2=DEBUG, 3=INFO, 4=WARNING, 5=ERROR, 6=FATAL )
MessageSvc.OutputLevel = 5;

// Number of events to be processed (default is 10)
ApplicationMgr.EvtMax = 500;

ApplicationMgr.HistogramPersistency = "ROOT";
NTupleSvc.Output = { "FILE1 DATAFILE='PsiPrimeGammaX.root'
                     OPT='NEW' TYP='ROOT'"};
```

8 Output Root Trees

`FSFilter` generates three types of root trees:

1. Reconstructed information is stored in trees named using the convention described in section 6.1 (with an `nt` prepended): `nt(EXC/INC)(tag)(code2)_(code1)`, where `(EXC/INC)` is either `EXC` or `INC` depending on whether the reconstruction was done exclusively or inclusively, and `(code2)` and `(code1)` are the numbering codes described in section 5, etc. There is one entry for each combination of particles that passes the loose cuts described in sections 3 and 4. Thus there may be more than one entry per event. This is the main tree for physics analysis.
2. MC generated information for each exclusive final state is stored in trees named `ntGEN(tag)(code2)_(code1)`. This type of tree contains exactly one entry per each event generated with this final state, regardless of whether or not the event was reconstructed. It is only used for exclusive final states. This tree can be used to look at generator-level distributions.
3. MC generated information for all final states is stored in one tree named `ntGEN`. This tree contains exactly one entry per event, independent of final state, and independent of reconstruction. Use this tree to count the number of events generated for different processes.

Each of these trees contains some subset of the blocks of information described in the following subsections. Only the reconstructed tree, `nt(EXC/INC)(tag)(code2)_(code1)`, for example, contains shower or track information.

By convention particles are listed in trees in the following order:

```
Lambda ALambda e+ e- mu+ mu- p+ p- eta gamma K+ K- Ks pi+ pi- pi0
```

For example, in the process $\psi(2S) \to \pi^+\pi^- J/\psi; J/\psi \to \mu^+\mu^-$, the μ^+ is particle 1, the μ^- is particle 2, the π^+ is particle 3, and the π^- is particle 4. Or as another example, in the process $\psi(2S) \to \gamma\chi_{c1}; \chi_{c1} \to \eta\pi^0\pi^0$, the η is particle 1, the γ is particle 2, one π^0 is particle 3, and the other π^0 is particle 4. In cases like this where there are identical particles, no ordering is assumed.

8.1 Event Information

The "Event Information" block contains information about the event as a whole.

The following information is contained in all three tree types:

```
Run:          run number
Event:        event number
BeamEnergy:   beam energy (ACTUALLY BEAM ENERGY,
```

```
                              NEED CROSSING ANGLE TO GET SQRT(S))
NTracks:      total number of tracks in the event
NShowers:     total number of showers in the event
```

The following information is only contained in `nt(EXC/INC)(tag)(code2)_(code1)`, where the total energies and momenta are calculated using the raw particle four-momenta (i.e., before any vertex or kinematic fitting):

```
TotalEnergy:    total energy of all final state particles
TotalPx:        total px of all final state particles
TotalPy:        total py of all final state particles
TotalPz:        total pz of all final state particles
TotalP:         total momentum of all final state particles
MissingMass2:   missing mass squared of the event
VChi2:          chi2 of the vertex fit if there was a
                  vertex fit (-1 otherwise)
```

Finally, this information is only contained in `nt(EXC/INC)(tag)(code2)_(code1)` and only when a kinematic fit is performed:

```
Chi2:         chi2 of the kinematic fit
Chi2DOF:      chi2/dof of the kinematic fit
```

8.2 Vertex Information

The "Vertex Information" block contains the beam spot and the primary vertex. It is only written out for the reconstructed tree, `nt(EXC/INC)(tag)(code2)_(code1)`.

```
BeamVx:      x-position of the beam spot
BeamVy:      y-position of the beam spot
BeamVz:      z-position of the beam spot
PrimaryVx:   x-position of the primary vertex
PrimaryVy:   y-position of the primary vertex
PrimaryVz:   z-position of the primary vertex
```

8.3 Particle Four-Momenta

The "Particle Four-Momenta" block contains the four-momentum for each particle in the final state:

```
(prefix)PxP(n):  x momentum of particle (n)
(prefix)PyP(n):  y momentum of particle (n)
(prefix)PzP(n):  z momentum of particle (n)
```

```
(prefix)EnP(n):  energy of particle (n)
```

Different types of four-momenta are distinguished using prefixes. MC generated four-momenta have a prefix `MC`; raw four-momenta have a prefix `R`; vertex-constrained four-momenta or four-momenta that include 1C mass fits (e.g. constraining the π^0 or η masses) have a prefix `I` (for "Intermediate"); and the final four-momenta (the fully-constrained four-momenta resulting from the kinematic fit in the exclusive case, or the vertex-constrained and 1C mass-constrained four-momenta in the inclusive case) have no prefix.

Different particles are differentiated using the postfix `P(n)`, where `(n)` is the number of the particle in the ordered list. Four-momenta for secondaries originating from particle `(n)`, such as the two γ's from a π^0, are recorded using `P(n)a` and `P(n)b`, where the ordering follows the same conventions as above, or, in the case of identical daughter particles, ordering is from low to high energy. As two examples: in the process $\psi(2S) \to \pi^+\pi^- J/\psi; J/\psi \to \mu^+\mu^-$, the raw energy of the π^+ is given by `REnP3`; and in the process $\psi(2S) \to \pi^+\pi^- J/\psi; J/\psi \to \pi^+\pi^-\pi^0$, the y-momentum of the high-energy photon from the π^0, after a 1C fit to the π^0 mass, is given by `IPyP5b`.

Particle four-momenta are included in both the `nt(EXC/INC)(tag)(code2)_(code1)` and the `ntGEN(tag)(code2)_(code1)` trees. In the `ntGEN(tag)(code2)_(code1)` tree, the generated four-momenta are actually written out twice, once with the `MC` prefix and once with no prefix. This makes it easier to treat the generated MC more like data in some applications.

In the reconstructed tree, `nt(EXC/INC)(tag)(code2)_(code1)`, MC generated information is also included for events in which the generated and reconstructed final states are the same. When there are identical particles, however, no attempt is made to ensure that the particles are ordered consistently, i.e., no attempt is made to match reconstructed particles with generated particles.

8.4 Shower Information

Shower information is written out for every reconstructed photon that is part of a final state. As in the four-momenta case (section 8.3), showers from different particles are differentiated using the postfix `P(n)`, where `(n)` is the number of the particle in the ordered list. As examples: the energy of the low energy photon from the π^0 decay in $\psi(2S) \to \pi^+\pi^-\pi^0$ is called `ShEnergyP3a`; and the timing of the radiated photon in $\psi(2S) \to \gamma\eta$ is called `ShTimeP2`.

```
ShTimeP(n):     timing information
ShEnergyP(n):   shower energy
ShCosThetaP(n): cos(theta) in the lab
ShE925P(n):     E9/E25 (the energy in a 3x3 array
                  over the energy in a 5x5 array)
ShPi0PullP(n):  the pull of the best pi0 formed
```

```
                  with this shower
ShDangP(n):     smallest angle between this shower
                  and tracks projected to the EMC
ShMatchP(n):    1.0 if this shower has an associated
                  charged track; -1.0 otherwise
```

8.5 Track Information

Track information is written out for every reconstructed track that is part of a final state. The postfix `P(n)` follows the same convention as for the four-momenta (section 8.3).

```
TkProbPiP(n):   probability track is a pion
TkProbKP(n):    probability track is a kaon
TkProbPP(n):    probability track is a proton
TkProbMuP(n):   probability track is a muon
TkProbEP(n):    probability track is an electron
TkRVtxP(n):     radial distance of closest approach
                  to the primary vertex (in cm)
TkZVtxP(n):     longitudinal distance
                  to the primary vertex (in cm)
TkCosThetaP(n): cos(theta) in the lab
TkEPP(n):       E/p (shower energy over
                  track momentum)
TkMucDepthP(n): penetration depth in muon chamber
                  (in cm)
```

If the `pidStudies` flag is set to true, also include these variables:

```
TkProbPHP(n):   a dE/dx variable in arbitrary units
                  (550 for Bhabha electrons)
TkNormPHP(n):   (not sure)
TkErrorPHP(n):  (not sure)
TkIndexP(n):    track index in the track collection
```

8.6 $\pi^0 \to \gamma\gamma$ Information

The following information for each $\pi^0 \to \gamma\gamma$ decay is recorded:

```
Pi0MassP(n):    the unconstrained gamma gamma mass
Pi0Chi2P(n):    the chi2 of the 1C fit to the pi0 mass
```

Note that the shower information for each γ from the π^0 is recorded along with the other showers (section 8.4) and the four-momentum for each photon is recorded along with the other four-momenta (section 8.3).

8.7 $\eta \to \gamma\gamma$ Information

The decay $\eta \to \gamma\gamma$ is treated in the same way as $\pi^0 \to \gamma\gamma$ (section 8.6):

```
EtaMassP(n):    the unconstrained gamma gamma mass
EtaChi2P(n):    the chi2 of the 1C fit to the eta mass
```

8.8 $K^0_S \to \pi^+\pi^-$, $\Lambda \to p\pi^-$ and $\overline{\Lambda} \to \overline{p}\pi^+$ Information

This information is recorded for each $K^0_S \to \pi^+\pi^-$, $\Lambda \to p\pi^-$ and $\overline{\Lambda} \to \overline{p}\pi^+$ decay:

```
VeeMassP(n):     the unconstrained mass of the daughters
VeeChi2P(n):     the chi2 of the initial Ks vertex fit
Vee2ndChi2P(n):  the chi2 of the secondary vertex fit
VeeLSigmaP(n):   the separation between the primary and
                  secondary vertex (L) over its error (sigma)
VeeSigmaP(n):    the error of the decay length (L)
VeeVxP(n):       the x-position of the secondary vertex
VeeVyP(n):       the y-position of the secondary vertex
VeeVzP(n):       the z-position of the secondary vertex
```

Note that track information and four-momenta for the daughter tracks are also recorded as in sections 8.5 and 8.3, respectively.

8.9 Information to Tag $\psi(2S) \to XJ/\psi$

A few variables are included to try to identify $\psi(2S)$ transitions to J/ψ, sometimes useful for identifying J/ψ backgrounds. This information is currently included when running over all data sets, but it is only meaningful for $\psi(2S)$ data.

```
JPsiPiPiRecoil:  the smallest difference between the J/psi
                   mass and any combination of pi+ pi-
                   recoil mass (using all positive and
                   negative tracks with assumed pion
                   masses)
JPsiGGRecoil:    the smallest difference between the J/psi
                   mass and any combination of gamma gamma
                   recoil mass
```

8.10 Truth Information

When running over Monte Carlo, the truth information about a given event is included in all three tree types. These variables should allow one to separate and count signal and background events. The numbers in this block of information always refer to generator level information. For example, the "number of K_L^0" refers to the true number of generated K_L^0.

These variables label the generated final state:

```
MCDecayCode1:      the true code1, described in the
                     "final state numbering" section
MCDecayCode2:      the true code2, described in the
                     "final state numbering" section
MCExtras:          1000 * number of neutrinos +
                    100 * number of K_L +
                     10 * number of neutrons +
                      1 * number of antineutrons
MCTotalEnergy:     the total generated energy calculated
                     using particles in MCDecayCode(1,2)
                     and MCExtras:  use this to double-
                     check that no particles are missing
MCSignal:          1 if the reconstructed final state
                     matches the generated final state;
                   0 otherwise
```

These variables count open charm particles:

```
MCOpenCharm1:      1000 * number of D+ +
                    100 * number of D- +
                     10 * number of D0 +
                      1 * number of D0bar
MCOpenCharm2:      1000 * number of (D*+ to pi+ D0) +
                    100 * number of (D*- to pim D0bar) +
                     10 * number of (D*0 to gamma D0) +
                      1 * number of (D*0bar to gamma D0bar)
MCOpenCharm3:      1000 * number of (D*+ to pi0 D+) +
                    100 * number of (D*- to pi0 D-) +
                     10 * number of (D*0 to pi0 D0) +
                      1 * number of (D*0bar to pi0 D0bar)
MCOpenCharmTag:    abcd, where
                      a is for the D+
                          a = 0 for no D+
                          a = 1 for D+ to K- pi+ pi+
                          a = 2 for D+ to K- pi+ pi+ pi0
                          a = 3 for D+ to Ks pi+
                          a = 4 for D+ to Ks pi+ pi0
```

```
        a = 5 for D+ to Ks pi+ pi+ pi-
    b is for the D- (same modes as a)
    c is for the D0
        c = 0 for no D0
        c = 1 for D0 to K- pi+
        c = 2 for D0 to K- pi+ pi0
    d is for the D0bar (same modes as c)
```

These variables keep track of how various states were produced, where `X` stands for any particle:

```
MCChicProduction:  10 for X to gamma chi_c0
                   11 for X to gamma chi_c1
                   12 for X to gamma chi_c2
                  100 for any other production of chi_c0
                  101 for any other production of chi_c1
                  102 for any other production of chi_c2
                    0 for no produced chi_cJ
MCJPsiProduction:  1 for X to pi+ pi- J/psi
                   2 for X to pi0 pi0 J/psi
                   3 for X to eta J/psi
                   4 for X to gamma J/psi
                   5 for X to pi0 J/psi
                   6 for any other production of J/psi
                   0 for no produced J/psi
MCHcProduction:    1 for X to pi+ pi- h_c
                   2 for X to pi0 pi0 h_c
                   3 for X to eta h_c
                   4 for X to gamma h_c
                   5 for X to pi0 h_c
                   6 for any other production of h_c
                   0 for no produced h_c
MCEtacProduction:  1 for X to gamma eta_c
                   2 for any other production of eta_c
                   0 for no produced eta_c
```

These variables count the number of ways in which various particles decayed:

```
MCX3872Decay:      1 for X to rho J/psi
                   2 for X to pi+ pi- J/psi
                   3 for X to omega J/psi
                   4 for X to gamma J/psi
                   5 for X to gamma psi(2S)
                   6 for X to pi0 chi_c0
                   7 for X to pi0 chi_c1
                   8 for X to pi0 chi_c2
                   0 for no produced X
```

```
MCPi0Decay:          10 * number of gamma e+ e- +
                      1 * number of gamma gamma
MCEtaDecay:        10000 * number of gamma e+ e- +
                    1000 * number of gamma pi+ pi- +
                     100 * number of pi+ pi- pi0 +
                      10 * number of pi0 pi0 pi0 +
                       1 * number of gamma gamma
MCEtaprimeDecay:  1000000 * number of pi0 pi0 pi0 +
                   100000 * number of gamma gamma +
                    10000 * number of gamma omega +
                     1000 * number of gamma pi+ pi- +
                      100 * number of gamma rho +
                       10 * number of eta pi0 pi0 +
                        1 * number of eta pi+ pi-
MCPhiDecay:      10000 * number of gamma eta +
                  1000 * number of rho pi +
                   100 * number of pi+ pi- pi0 +
                    10 * number of Ks Kl +
                     1 * number of K+ K-
MCOmegaDecay:    100 * number of pi+ pi- +
                  10 * number of gamma pi0 +
                   1 * number of pi+ pi- pi0
MCKsDecay:       10 * number of pi0 pi0 +
                  1 * number of pi+ pi-
MCFSRGamma:    number of FSR gammas in this event
                (FSR gammas are ignored elsewhere)
```

The `MCDecayParticle` variables are an ordered list of the PDG ID numbers of the particles coming from the initial decay. For example, for $\psi(2S) \to \rho^+\pi^-$, `MCDecayParticle1` is -211 (for the π^-); `MCDecayParticle2` is 213 (for the ρ^+); and all the others are zero.

```
MCDecayParticle1:  the PDG ID of the 1st particle
MCDecayParticle2:  the PDG ID of the 2nd particle
MCDecayParticle3:  the PDG ID of the 3rd particle
MCDecayParticle4:  the PDG ID of the 4th particle
MCDecayParticle5:  the PDG ID of the 5th particle
MCDecayParticle6:  the PDG ID of the 6th particle
```

The generated four-vectors are also available for these same particles coming from the initial decay.

```
MCDecayParticleEnPn:  the generated energy of the nth MCDecayParticle
MCDecayParticlePxPn:  the generated x-momentum of the nth MCDecayParticle
MCDecayParticlePyPn:  the generated y-momentum of the nth MCDecayParticle
MCDecayParticlePzPn:  the generated z-momentum of the nth MCDecayParticle
```

9 Notes on Different Code Versions

The code for FSFilter is currently NOT checked into the BESIII repository. Instead, it resides in a CVS repository at IU on the stanley cluster. To check out the code, you can use:

```
> setenv CVSROOT (user)@stanley.physics.indiana.edu:/s4/remitche/cvsroot/
> setenv CVS_RSH ssh
> cvs co FSFilter
```

Since the code is not in the BESIII repository, the BESIII versioning number (the number that appears in the first subdirectory, for example, `FSFilter-00-00-00`) is currently being held fixed at `00-00-00`. Instead, different releases are being tagged in their current CVS repository using tags like "`v[DATE]`". The original release is `v20111104`. For any new release, the above document will be modified and then notes on those changes will be made in the sections below.

9.1 From v20111104 to v20120323

Changes to the job options parameters:

```
* added the neutralStudies parameter
```

Changes to the algorithm:

```
* user-transparent changes to MCTruthHelper that fix bugs in the
  MCDecayParticle[N] variables that occured when running over
  continuum MC

* calculate the initial 4-momentum without any small-angle or
  small-mass approximations, i.e. calculate it exactly
```

Changes to the root output:

```
* include vertex information for the beam spot, the primary vertex,
  and secondary vertices

* include 4-momenta for daughter particles (gammas from pi0,
  pions from Ks, etc.)

* change the names of the secondary tracks and showers (pions from Ks
  or gammas from pi0, etc.) in the root trees:  use postfixes P(n)a
  and P(n)b rather than the old prefixes KsTk1, etc.
```

```
* use the prefix "I" to mark "intermediate" four-momenta, i.e. four-
  momenta that result from vertex or 1C mass constraints

* use no prefix for the final four-momenta in inclusive final states

* added MCPi0Decay

* fixed bug in VChi2 variable
```

9.2 From v20120323 to v20120810

Changes to the job options parameters:

```
* added the writeNTGenCode parameter

* added the trackStudies, pi0Studies, etaStudies, and
  veeStudies parameters

* added the FSCut(n) parameters

* removed the lowerMissingMass2 and upperMissingMass2
  parameters (now redundant with FSCut(n) parameters)
```

9.3 From v20120810 to dev

Changes to output root tree:

```
* added muon chamber information

* added MCOpenCharm flags

* added ShMatch

* added VeeSigma

* added Vee2ndChi2

* added more info in MCChicProduction
```

Changes to the job options parameters:

```
* added the FSMMFit(n) parameters
```

```
* added the pidStudies parameter

* added bypassVertexDB parameter

* added the extraKaonPID parameter
```

Bugs:

```
* replaced the BOSS default maximum chi2 cut on kinematic
   fitting with the maxKinematicFitChi2DOF parameter

* fixed the PDG codes for e+, e-, mu+, and mu- (positive
   and negative charges were reversed)

* substantial rewrite of the MCTruthHelper class

* added a divide-by-zero check for Vee L/sigma

* check that the vertex chi2 is not nan
```

Other notes:

```
* use the MeasuredEcmsSvc package to get the beam energy

* track corrections are done for pions, kaons, electrons, and muons

* turn off corrections automatically if running over data

* track corrections are updated (up to December 2015)
```