MC matching

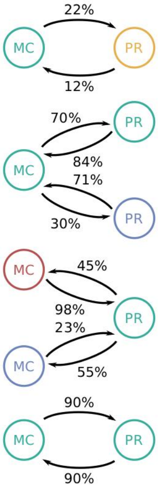
Frank Meier

2020 Belle II Physics Week
30 November 2020

Introduction

- MC matching useful to
  - optimize selection requirements, *e.g.* label training data for MVAs
  - calculate signal efficiencies
  - study background sources
- two steps
  1. relate mdst dataobjects (tracks, ECL cluster, KLM cluster) with MC particles (with weights)
  2. relate reconstructed particles with MC particles
- interpretation of MC matches necessary
- MC matching in a nut shell
  - `modularAnalysis.matchMCTruth('Upsilon4S', path=path)`
  - use **`isSignal==1`** to identify signal

MC matching for tracks

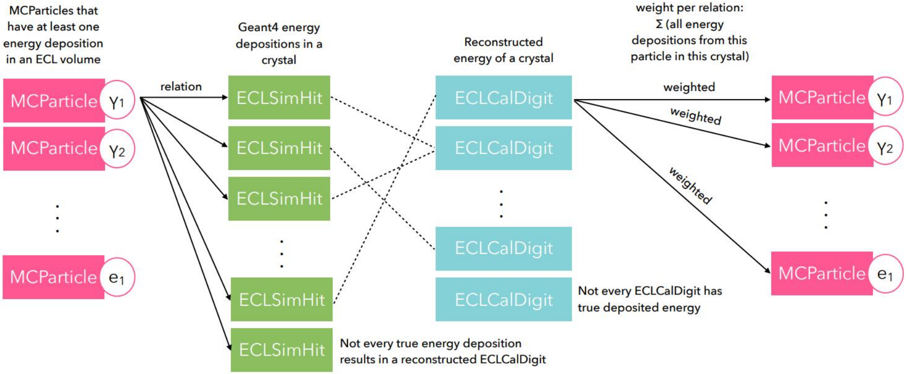


- compare overlap of hits of MC particle and hits used for pattern recognition of track
- track might mainly consist of background hits (case 1)
- true hits might be split between two or more track candidates (case 2)
- one track might use hits of two or more MC particles (case 3)
- certain level of efficiency and purity
  ⇒ relation set between MC particle and track

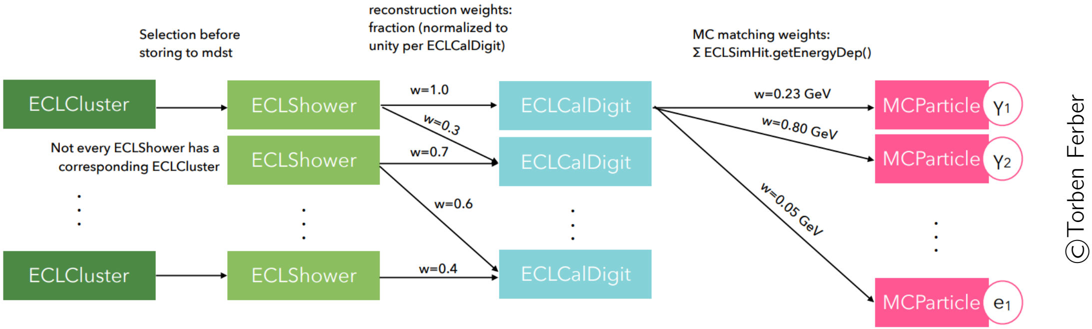
- alternative approach based on kinematics under development

MC matching for photons

- comparison of true deposited energy (from GEANT) in ECL cluster with energy of MC particle and reconstructed energy



MC matching for photons



- relation between ECL cluster based particle (γ, neutron, K_L^0) and MC particle set if
  - weight (energy of ECL cluster) $> 20\,\%$ of reconstructed particle energy
  - weight $> 30\,\%$ of MC particle energy
- `clusterMCMatchWeight` returns weight between ECL cluster and MC particle match of reconstructed particle
- highest weighted relation used for MC matching

MC matching failures

- clone tracks
  - `isCloneTrack` and `isOrHasCloneTrack`
- fake tracks
- wrong charge
  - analysis variable `isWrongCharge`
- overlapping cluster
  - can be studied via `nMCMatches`
- cluster split-offs
- beam background

MC matching errors

Duke UNIVERSITY

- `matchMCTruth('Upsilon4S', path=path)`
  - sets relations between **`Particles`** and **`MCParticles`**
  - recursively matches all daughter particles (since light-2002-janus and release-05-00-00)
  - bit-wise error flags indicate what went wrong in MC matching (variable **`mcErrors`**)

c_Correct = 0	This Particle and all its daughters are perfectly reconstructed.
c_MissFSR = 1	A Final State Radiation (FSR) photon is not reconstructed (based on MCParticle::c_IsFSRPhoton).
c_MissingResonance = 2	The associated MCParticle decay contained additional non-final-state particles (e.g. a rho) that weren't reconstructed. This is probably O.K. in most cases.
c_DecayInFlight = 4	A Particle was reconstructed from the secondary decay product of the actual particle. This means that a wrong hypothesis was used to reconstruct it, which e.g. for tracks might mean a pion hypothesis was used for a secondary electron.
c_MissNeutrino = 8	A neutrino is missing (not reconstructed).
c_MissGamma = 16	A photon (not FSR) is missing (not reconstructed).
c_MissMassiveParticle = 32	A generated massive FSP is missing (not reconstructed).
c_MissKlong = 64	A Klong is missing (not reconstructed).
c_MisID = 128	One of the charged final state particles is mis-identified (wrong signed PDG code).
c_AddedWrongParticle = 256	A non-FSP Particle has wrong PDG code, meaning one of the daughters (or their daughters) belongs to another Particle.
c_InternalError = 512	There was an error in MC matching. Not a valid match. Might indicate fake/background track or cluster.
c_MissPHOTOS = 1024	A photon created by PHOTOS was not reconstructed (based on MCParticle::c_IsPHOTOSPhoton).
c_AddedRecoBremsPhoton = 2048	A photon added with the bremsstrahlung recovery tools (correctBrems or correctBremsBelle) has no MC particle assigned, or it doesn't belong to the decay chain of the corrected lepton mother

isSignal variables

- convenience aliases for `mcErrors` variable
- `isSignal`: `mcErrors == 0`
- `isSignalAcceptMissingNeutrino`: `mcErrors == 0 or mcErrors == 8`
- `isSignalAcceptMissingGamma`: `mcErrors == 0 or mcErrors == 16`
- `isSignalAcceptBremsPhotons`: `mcErrors == 0 or mcErrors == 2048`
- `isSignalAcceptMissingMassive`: `mcErrors` ∈ [0, 32, 64, 96]
- `isSignalAcceptWrongFSPs`: `mcErrors` ∈ [0, 128, 256, 384]
- `isSignalAcceptMissing`: `mcErrors` ∈ [0, 8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96, 104, 112, 120]
- `create_isSignal_alias(aliasName, flags)`
  - `create_isSignal_alias("isSignalAcceptMissingNeutrinoAndMissingGamma", [8, 16])`: `mcErrors` ∈ [0, 8, 16, 24]
- since light-1911-heracles and release-05-00-00 three possible return values: 0, 1, and NaN
  - isSignal variables can no longer be used as booleans
  - NaN if no MC partner found (running on data or `matchMCTruth` not run or fake tracks / cluster)

Decay string arrows

- MC matching interpretation configured via decay arrow, markers, and keywords
- four different allowed arrow types
  - `->` decays via intermediate resonances and / or with radiative photons are counted as signal
  - `=norad=>` if in actual decay photon was radiated it is **not** counted as signal; intermediate resonances do not have to be specified
  - `=direct=>` if actual decay proceeds via intermediate resonance it is **not** counted as signal; decays with radiative photons are counted as signal
  - `=exact=>` intermediate resonances and radiative photons have to be specified explicitly, otherwise decay is not counted as signal
- example:
  - generated decay: $B^+ \to J/\psi K^+$ with $J/\psi \to e^+e^-$ and final state photon
  - `B+:default -> K+ e+ e-` $\Rightarrow$ `isSignal == 1`
  - `B+:noFSR =norad=> K+ e+ e-` $\Rightarrow$ `isSignal == 0` and `mcErrors == 1`
  - `B+:nores =direct=> K+ e+ e-` $\Rightarrow$ `isSignal == 0` and `mcErrors == 2`
  - `B+:exact =exact=> K+ e+ e-` $\Rightarrow$ `isSignal == 0` and `mcErrors == 3`

Decay string marker and keywords

- markers for inclusive decays
  - `@` marked (composite) particle can be any particle and decay is still counted as signal
  - `(misID)` marked (final state) particle can be other particle type and decay is still counted as signal
  - `...` even if there are more massive final state particles than specified, decay is counted as signal
    - missing π^0 are not counted as signal
- `(decay)` decay in flight of marked particle counted as signal, *e.g.* $\pi \to \mu\nu_\mu$
- `?nu` neutrinos are ignored in MC matching (**isSignal** works like **isSignalAcceptMissingNeutrino**)
- `?addbrems` Bremsstrahlung correction photons are ignored in MC matching (**isSignal** works like **isSignalAcceptBremsPhotons**)
- `?gamma` missing photon(s) are ignored in MC matching (**isSignal** works like **isSignalAcceptMissingGamma**)
- `@`, `(misID)`, and `(decay)` have to be placed in front of particle
- `...`, `?nu`, `?addbrems`, and `?gamma` have to be placed at the end of the decay string

MC reconstruction

- determine how many events of certain type have been generated, especially for inclusive decays
- create particle lists of generated particles using `fillParticleListFromMC(decayString, cut, path)`
  - useful to select only primary MC particles using **`mcPrimary`** in cut string
  - add argument **`"addDaughters=True"`** to recursively create particles for all daughters and set relation to mother MC particle
- define decay chain using `reconstructMCDecay(decayString, cut, path)` (supersedes outdated `findMCDecay`)

MC matching examples

- sample with
  a) $D^0 \to K^- \pi^+ \pi^0$
  b) $D^0 \to K^{*-} \pi^+$ with $K^{*-} \to K^- \pi^0$
  c) $D^0 \to K^- \pi^+ \pi^0 \gamma_{\mathsf{FSR}}$
  d) $D^0 \to K^- \pi^+ \pi^0$ with $\pi^+ \to \mu^+ \nu_\mu$
  e) $D^0 \to K^{*-} \mu^+ \nu_\mu$

decay string	isSignal == 1
1. `D0 -> K- pi+ pi0`	
2. `D0 =direct=> K- pi+ pi0`	
3. `D0 =exact=> K- pi+ pi0`	
4. `D0 -> K- (decay)pi+ pi0`	
5. `D0 -> K- (misID)pi+ pi0 ?nu`	
6. `D0 -> K- (misID)(decay)pi+ pi0 ?nu`	

MC matching examples

- sample with
  a) $D^0 \to K^- \pi^+ \pi^0$
  b) $D^0 \to K^{*-} \pi^+$ with $K^{*-} \to K^- \pi^0$
  c) $D^0 \to K^- \pi^+ \pi^0 \gamma_{\text{FSR}}$
  d) $D^0 \to K^- \pi^+ \pi^0$ with $\pi^+ \to \mu^+ \nu_\mu$
  e) $D^0 \to K^{*-} \mu^+ \nu_\mu$

decay string	`isSignal == 1`
1. `D0 -> K- pi+ pi0`	a), b), c)
2. `D0 =direct=> K- pi+ pi0`	
3. `D0 =exact=> K- pi+ pi0`	
4. `D0 -> K- (decay)pi+ pi0`	
5. `D0 -> K- (misID)pi+ pi0 ?nu`	
6. `D0 -> K- (misID)(decay)pi+ pi0 ?nu`	

MC matching examples

Duke UNIVERSITY

- sample with
  a) $D^0 \to K^-\pi^+\pi^0$
  b) $D^0 \to K^{*-}\pi^+$ with $K^{*-} \to K^-\pi^0$
  c) $D^0 \to K^-\pi^+\pi^0\gamma_{\mathsf{FSR}}$
  d) $D^0 \to K^-\pi^+\pi^0$ with $\pi^+ \to \mu^+\nu_\mu$
  e) $D^0 \to K^{*-}\mu^+\nu_\mu$

decay string	`isSignal == 1`
1. `D0 -> K- pi+ pi0`	a), b), c)
2. `D0 =direct=> K- pi+ pi0`	a) and c)
3. `D0 =exact=> K- pi+ pi0`	
4. `D0 -> K- (decay)pi+ pi0`	
5. `D0 -> K- (misID)pi+ pi0 ?nu`	
6. `D0 -> K- (misID)(decay)pi+ pi0 ?nu`	

MC matching examples

- sample with
  a) $D^0 \to K^- \pi^+ \pi^0$
  b) $D^0 \to K^{*-} \pi^+$ with $K^{*-} \to K^- \pi^0$
  c) $D^0 \to K^- \pi^+ \pi^0 \gamma_{\mathsf{FSR}}$
  d) $D^0 \to K^- \pi^+ \pi^0$ with $\pi^+ \to \mu^+ \nu_\mu$
  e) $D^0 \to K^{*-} \mu^+ \nu_\mu$

decay string	`isSignal == 1`
1. `D0 -> K- pi+ pi0`	a), b), c)
2. `D0 =direct=> K- pi+ pi0`	a) and c)
3. `D0 =exact=> K- pi+ pi0`	only a)
4. `D0 -> K- (decay)pi+ pi0`	
5. `D0 -> K- (misID)pi+ pi0 ?nu`	
6. `D0 -> K- (misID)(decay)pi+ pi0 ?nu`	

MC matching examples

- sample with
  a) $D^0 \to K^- \pi^+ \pi^0$
  b) $D^0 \to K^{*-} \pi^+$ with $K^{*-} \to K^- \pi^0$
  c) $D^0 \to K^- \pi^+ \pi^0 \gamma_{\text{FSR}}$
  d) $D^0 \to K^- \pi^+ \pi^0$ with $\pi^+ \to \mu^+ \nu_\mu$
  e) $D^0 \to K^{*-} \mu^+ \nu_\mu$

decay string	`isSignal == 1`
1. `D0 -> K- pi+ pi0`	a), b), c)
2. `D0 =direct=> K- pi+ pi0`	a) and c)
3. `D0 =exact=> K- pi+ pi0`	only a)
4. `D0 -> K- (decay)pi+ pi0`	a), b), c), and d)
5. `D0 -> K- (misID)pi+ pi0 ?nu`	
6. `D0 -> K- (misID)(decay)pi+ pi0 ?nu`	

MC matching examples

- sample with
  a) $D^0 \to K^- \pi^+ \pi^0$
  b) $D^0 \to K^{*-} \pi^+$ with $K^{*-} \to K^- \pi^0$
  c) $D^0 \to K^- \pi^+ \pi^0 \gamma_{\text{FSR}}$
  d) $D^0 \to K^- \pi^+ \pi^0$ with $\pi^+ \to \mu^+ \nu_\mu$
  e) $D^0 \to K^{*-} \mu^+ \nu_\mu$

decay string	`isSignal == 1`
1. `D0 -> K- pi+ pi0`	a), b), c)
2. `D0 =direct=> K- pi+ pi0`	a) and c)
3. `D0 =exact=> K- pi+ pi0`	only a)
4. `D0 -> K- (decay)pi+ pi0`	a), b), c), and d)
5. `D0 -> K- (misID)pi+ pi0 ?nu`	a), b), c), and e)
6. `D0 -> K- (misID)(decay)pi+ pi0 ?nu`	

MC matching examples

- sample with
  a) $D^0 \to K^- \pi^+ \pi^0$
  b) $D^0 \to K^{*-} \pi^+$ with $K^{*-} \to K^- \pi^0$
  c) $D^0 \to K^- \pi^+ \pi^0 \gamma_{\text{FSR}}$
  d) $D^0 \to K^- \pi^+ \pi^0$ with $\pi^+ \to \mu^+ \nu_\mu$
  e) $D^0 \to K^{*-} \mu^+ \nu_\mu$

decay string	`isSignal == 1`
1. `D0 -> K- pi+ pi0`	a), b), c)
2. `D0 =direct=> K- pi+ pi0`	a) and c)
3. `D0 =exact=> K- pi+ pi0`	only a)
4. `D0 -> K- (decay)pi+ pi0`	a), b), c), and d)
5. `D0 -> K- (misID)pi+ pi0 ?nu`	a), b), c), and e)
6. `D0 -> K- (misID)(decay)pi+ pi0 ?nu`	a), b), c), d), and e)

MC classification

- TopoAna tool
  - requires PDG code and mother-daughter relations of all generated particles of an event
    tool **`mc_gen_topo()`** automatically creates list of necessary variables / aliases
  - example called **`usingMCGenTopo`** in examples folder of analysis package
  - tutorial at February 2020 B2GM (slides)
- TauDecayMarker module
  - assigns identifier to tau particles
  - use wrapper function `labelTauPairMC` to call the module
  - identifier stored in analysis variables **`tauPlusMCMode`**, **`tauMinusMCMode`**, **`tauPlusMCProng`**, and **`tauMinusMCProng`**
- GenMCTag tool
  - generalization of TauDecayMarker: identifier for each particle decay in an event
  - presentation in analysis software meeting (slides)
  - open pull request here (any help appreciated)

Fix of hierarchic MC matching

Duke UNIVERSITY

- problem was present in release-04-02-08 and light-2002-ichep, is fixed since release-05-00-00
- less restrictive MC exception flags of mother were propagated to daughters
  - mother was classified as signal even if daughter itself classified as background
    - `D0:dir =direct=> K- pi+ pi0` $\Rightarrow D^0 \to K^- \rho^+$ should not be counted as signal
    - `B- -> D0:dir pi- pi0` $\Rightarrow$ decays via intermediate resonance like $B^- \to D^{*0} \pi^-$ or $B^- \to D^0 \rho^-$ should be counted as signal
    - B^- with decay chain `D0 -> K- rho+` would have been accepted as signal
- MC acceptance flags of daughters were ignored by mother
  - `D0 -> K- pi+ ...` (missing massive particle, *e.g.* $\pi^+\pi^-$ accepted)
  - `B- -> D0 pi-`
    - B^- with sub-decay `D0 -> K- pi+ pi+ pi-` would have been classified as background
- now arrow types and MC acceptance flags are properly propagated

Known MC issues

Duke UNIVERSITY

- MC matching of radiative photons not working in continuum samples
  - error in generation and not in analysis $\Rightarrow$ fixed for MC14
- NaN returned for photons matched to beam background cluster (fake cluster) BII-7223
- MC matching for decays with only one (non-neutrino) daughter
  - accepting missing resonance implemented for cases like $\tau^+ \to \rho^+ \nu_\tau$
  - $B^+ \to K^+ \nu\nu$ causes issue

Summary

- MC matching not an exact science
- matching between tracks and cluster with simulated MC particles in reconstruction
- interpretation of MC matching needed:
  Define signal via decay string grammar and / or use dedicated isSignal variable
- **`isSignal`** is not a boolean (NaN cases)
- store **`mcErrors`** in your ntuple
- several bug fixes in release-05 $\Rightarrow$ use latest software release