# Table of Contents

# Table of Contents

# Recommended LoKi::Hybrid Functors

This lists the filters recommended for use in the HLT and the selections. See DaVinciTutorial4 for a hands-on tutorial. A longer, but not necessarily up-to-date, list can be found at LoKiParticleFunctions. It also contains examples on how to use these functors in C++ code. As neither are guaranteed to be a complete list, you can always see the DOxygen for the `LoKi::Cuts` namespace here.

This page refers to DaVinci v22r1.

# Particle Functors

Some mnemonic rules:

- The functors with the prefix `BPV` deal with "the best primary vertex". The best primary vertex is extracted from the desktop-tool using the method `IPhysDesktop::relatedVertex`
- The functors with the suffix `DV` get some even-data (e.g. the list of all vertices) through desktop tool
- The functors with the suffix `TES` get the event data from the Transient Event Store
- The `VD` as a part of the functor name means that the functor deals with "vertex distance"
- The `VDCHI2` as a part of the functor name means that the functor deals with "vertex $\chi^2$-distance"
- The `SIGN` as a part of the functor name means that functor return the value of $\chi^2$ or distance artificially signed according to some criteria.
- The `IP` stands for "impact parameter" and `MIP` means "the minimal (with reswpect to some set of vertices) value of impact parameter"
- Practically all "vertex functors" ( the functors which accept `const LHCb::VertexBase*` as arguments) have their names starting from `V`, the most evident exception if the functor `PRIMARY`
- All indices, e.g. for `CHILDCUT` or `CHILDFUN` functors, starts from `1. 0` always means the particle itself.
- For many Particle functors there is an equivalent Particle array functor starting with an `A`. A notable exception is `ADAMASS` for `ADMASS`.

## `ALL`: All

Takes all particles. This is required if one wants to apply no cut.

```
FilterDesktop.Filter.Code = "ALL"
```

## `ABSID`: Absolute value of PID.

Returns the absolute value of the PID. The following lines are equivalent:

```
FilterDesktop.Filter.Code = "ABSID==211"
FilterDesktop.Filter.Code = "ABSID=='pi+' "
FilterDesktop.Filter.Code = "ABSID=='pi-' "
```

Note the last line! The comparison `(ABSID=='pi-')` takes the absolute value on **both sides**. This avoid having to remember that the $\pi^+$ has a positive pid (211) while the $\mu^+$ has a negative pid (-13).

The pid can also be used without the absolute value. See `ID`.

## `ACHILD`: Compute a function on a particle in an array

The functor ACHILD acts as "meta-functor", it gets the daughter particle number "N" (the second parameter of constructor), apply the functor (the first parameter of contructor) and return the result.

e.g. ACHILD(PT,1) return the transverse momentum of the first particle in the combination, ACHILD(PT,2) return the transverse momentum of the second particle in the combination.

For example:

```
XXX.CombinationCut = "ACHILD(PT,1)*ACHILD(PT,2)>1500000"
```

See the function AMAXCHILD or AMINCHILD, to get the daughter particle by name.

## `ADMASS`: The absolute mass difference to the reference value

Calculates the absolute difference between the measured mass and the PDG reference value. It takes the pid of the reference particle as argument.

```
FilterDesktop.Filter.Code = "ADMASS('KS0')<50*MeV"
```

The mass difference can also be used without the absolute value. See `DMASS`.

## `ADPDGM` and `ADPDGMASS`: The absolute mass difference to the nominal mass

Returns the difference with the PDG value of a particle mass. Useful to compute the difference with the measured mass. **This calls the `ParticlePropertySvc` at very call and is therefore slow**. Use `ADMASS('B0')` rather than `ADPDGM` if you know the PID.

## `AMAXCHILD or AMINCHILD`: cut an array and return the max/min value of a functor.

Similar to MINTREE and MAXTREE, however can be performed on a particle array.

e.g. the minimal PT for all negative kaons: AMINCHILD(PT,"K-"==ID)

```
XXX.CombinationCut = "AMINCHILD(PT,"K-"==ID)*ACHILD(PT,"K+"==ID)>1500000"
```

This is computationally expensive, using ACHILD is much quicker if possible.

## `BPVDIRA`: Direction angle

Computes the cosine of the angle between the momentum of the particle and the direction fo flight from the best PV to the decay vertex.

```
CombineParticles.MotherCut = "BPVDIRA()>0.9999"
```

## `BPVIPCHI2()`: IP on related PV

Computes the $\chi^2$-IP on the related PV.

```
CombineParticles.MotherCut = "BPVIPCHI2()<25"
```

**TODO** : So far it needs the ().

## `BPVLTFITCHI2`: the $\chi^2$ of the proper lifetime fit.

The functor evaluates the $\chi^2$ of the proper lifetime fit of the particle using `ILifetimeFitter` tool. This $\chi^2$ is probably the best measure of the consistency of the hypothesis that the particle originates from the given primary vertex. It is also probably the best "pointing" variable.

```
CombineParticles.MotherCut = "BPVLTFITCHI2()<16"
```

The related primary vertex is extracted from the desktop, the fitter itself is extracted using `DVAlgorithm::lifetimeFitter("nick-name")`. Please note that it is completely different functor from `BPVLTCHI2`!

## `BPVLTIME`: the proper lifetime of the particle

The functor evaluates the proper lifetime of the particle using `ILifetimeFitter` tool. Unfortunately due to very sad conventions adopted for LHCb, the proper time is measured in `ns` instead of the natural units: $\left[ c\tau \right] = \mathrm{mm}$.

```
CombineParticles.MotherCut = "BPVLTIME()>1.5"
```

The related primary vertex is extracted from the desktop, the fitter itself is extracted using `DVAlgorithm::lifetimeFitter("nick-name")`.

## `BPVLTCHI2`: the $\chi^2$-significance of the proper lifetime of the particle

The functor evaluates the $\chi^2$-significance of the proper lifetime of the particle using `ILifetimeFitter` tool.

```
CombineParticles.MotherCut = "BPVLTCHI2()>9"
```

The related primary vertex is extracted from the desktop, the fitter itself is extracted using `DVAlgorithm::lifetimeFitter("nick-name")`. Please note that it is completely different functor from `BPVLTFITCHI2`!

## `BPVLTSIGNCHI2`: the signed $\chi^2$-significance of the proper lifetime of the particle

The functor evaluates the signed $\chi^2$-significance of the proper lifetime of the particle using `ILifetimeFitter` tool.

```
CombineParticles.MotherCut = "BPVLTSIGNCHI2()>-4"
```

The related primary vertex is extracted from the desktop, the fitter itself is extracted using `DVAlgorithm::lifetimeFitter("nick-name")`.

## `BPVVDCHI2`: $\chi^2$-separation from related PV

Computes the $\chi^2$-distance from the related PV: (Pos(bestPV) - Pos(EndVertex))^T ( cov_(bestPV) + cov(EndVertex) )^-1 (Pos(bestPV) - Pos(EndVertex))

```
CombineParticles.MotherCut = "BPVVDCHI2()>100"
```

## `BPVVDZ`: $\delta z$-distance from the end vertex of the particle and the related primary vertex.

The functor computes the $\delta z$-distance from the end vertex of the particle and the related primary vertex.

```
CombineParticles.MotherCut = "0<BPVVDZ"
```

The concept and the name come from Sean Brisbane. The functor is available starting from Phys/LoKiPhys version >= v7r2.

## `BPVVDR`: $\rho$-distance from the end vertex of the particle and the related primary vertex.

The functor computes the $\rho$-distance(cylindrical) from the end vertex of the particle and the related primary vertex.

```
CombineParticles.MotherCut = "0.1<BPVVDR"
```

The concept and the name come from Sean Brisbane. The functor is available starting from Phys/LoKiPhys version >= v7r2.

## `BPVVDRHO`: $\rho$-distance from the end vertex of the particle and the related primary vertex.

The functor computes the $\rho$-distance(cylindrical) from the end vertex of the particle and the related primary vertex.

```
CombineParticles.MotherCut = "0.1<BPVVDRHO"
```

The concept and the name come from Sean Brisbane. The functor is available starting from Phys/LoKiPhys version >= v7r2.

## `CHI2PDGM` and `CHI2PDGMASS`: The mass difference to the nominal mass in units of chi2

Returns the difference with the PDG value of a particle mass. **This calls the `ParticlePropertySvc` at very call and is therefore slow**. Use `CHI2M('B0')` rather than `CHI2PDGM` if you know the PID.

## `CHI2M` and `CHI2MASS`: The mass difference to some reference mass in units of chi2

Returns the difference with the PDG value of a particle mass. It takes an argument like `ADMASS`

```
CombineParticles.MotherCut = "CHI2M('B0')<5"
```

## `CHILDCUT`: Applies a cut to a given child

```
FilterDesktop.Filter.Code = "CHILDCUT ( MIPCHI2DV ( PRIMARY ) > 1 , 1 )"
```

In this example one applies an IP cut on the **first** daughter of the input particle. This requires to know which is the first, second, etc daughter. Can be useful when `(N)INTREE` won't work. Like here for the slow pion in a $D^*$ where searching for a pion in the tree would also return the daughters of the $D^0$. Use the safer `INTREE` and `NINTREE` instead.

**TODO** : Is there a way of applying a cut to the daughters only, without navigating the whole tree? From next version `NINGENERATION` and `INGENERATION` will do this.

## `DECTREE`: Check if a particle satisfies a certain decay descriptor

```
FilterDesktop.Filter.Code = "DECTREE ('B0 -> (D- -> K+ pi- pi-) pi+')"
```

The example selects the decay of B0 to D- pi+ followed by D- to K+ pi- pi-. This can be useful for example to select one or more decay channels from a stripping line that contains many decay channels

## `DMASS`: The mass difference to the reference value

Calculates the difference between the measured mass and the PDG reference value. It takes the pid of the reference particle as argument. In most reasonable cases one will use the absolute mass difference (see `ADMASS`) or the mass itself (see `MM`).

## `DPDGM` and `DPDGMASS`: The mass difference to the nominal mass

Returns the difference with the PDG value of a particle mass. Useful to compute the difference with the measured mass. **This calls the `ParticlePropertySvc` at very call and is therefore slow**. Use `DMASS('B0')` rather than `DPDGM` if you know the PID.

## `ID`: Particle ID

Like `ABSID` but without the absolute value.

## `INGENERATION`: "in generation"

The predicate which checks the existence of the particle satisfying the requirements in the decay tree at the given depth

```
FilterDesktop.Code = "INGENERATION ( ( 'mu+'==ABSID)  & ( PT > 2 * GeV ) , 2 ) "
```

Requires the presence of at least one granddaughter $\mu^{\pm}$, which the transverse momentum $p_T > 2\mathrm{GeV}/c$. The generations are defined as: 0 - the particle itself, 1 - the direct children, 2 - grandchildren , etc. The concept and the name come from Patrick Koppenburg. The functor is available starting from Phys/LoKiPhys version >= v7r2.

## `INTREE`: In tree

Requires there is a particle in the decay tree satisfying the requirements.

```
FilterDesktop.Filter.Code = "INTREE( (ID=='J/psi(1S)') & (BPVVDCHI2>25) ) "
```

Requires there is a $J/\psi$ in the tree more than $5\sigma$ away from the best vertex.

## `MAXTREE`: Maximum value in the decay tree

Returns the maximum value of some functor in the decay tree. Useful to apply cuts on all particles, or to extract some value from the tree.

```
CombineParticles.MotherCut = "M-MAXTREE('D0'==ABSID,M)<165.5"
```

This example looks for all $D^0$ in the decay, returns the largest mass (which is the mass of the $D^0$ as there's only one) and computes the mass difference with the mother.

## `MINTREE`: Minimal value in the decay tree

Takes a functor as agrument and returns its minimal value searching through the decay tree.

```
FilterDesktop.Filter.Code = "MINTREE(ABSID=='K+',PT)>1400*MeV"
```

This would look for all particles kaons in the decay tree and find their $P_T$ and return the minimum. The cut then requires that all kaon descendents have a $P_T > 1.4\,\mathrm{GeV}$.

## `MINVDDV`: minimum distance between the particle's end-vertex and any other vertex

The functor evaluates the minimal 3D-distance between the particle's end-vertex and the vertices form "list". The list of vertices is extracted from the desktop:

```
CombineParticles.MotherCut = "MINVDDV(PRIMARY)>1"
```

Requires that the particle's end-vertex has at least one millimeter separation with respect to any primary vertex.

## `MINVDSOURCE`: minimum distance between the particle's end-vertex and any other vertex

The functor evaluates the minimal 3D-distance between the particle's end-vertex and the vertices form "list". The list of vertices is extracted from the "source":

```
CombineParticles.MotherCut = "MINVDSOURCE( VSOURCEDV ( PRIMARY ) )>1"
```

Requires that the particle's end-vertex has at least one millimeter separation with respect to any primary vertex.

## `MINVDTES`: minimum distance between the particle's end-vertex and any other vertex

The functor evaluates the minimal 3D-distance between the particle's end-vertex and the vertices form "list". The list of vertices is extracted from the TES:

```
CombineParticles.MotherCut = "MINVDTES('SomeLocationInTES',PRIMARY)>1"
```

Requires that the particle's end-vertex has at least one millimeter separation with respect to any primary vertex.

## `MIPCHI2DV`: Minimum IP-$\chi^2$

Returns the minimum $\chi^2$ distance of a particles's trajectory to any set of vertices. Most useful in conjunction with the `PRIMARY` argument that looks at PVs.

```
FilterDesktop.Filter.Code = "MIPCHI2DV(PRIMARY)>9"
FilterDesktop.Filter.Code = "MIPCHI2DV()>9"   ## ditto
```

This requires that the particle is $3\sigma$ away from any PV.

## `MIPDV`: Minimum IP

Returns the minimum distance of a particles's trajectory to any set of vertices. Most useful in conjunction with the `PRIMARY` argument that looks at PVs.

```
FilterDesktop.Filter.Code = "MIPDV(PRIMARY)>0.15"
```

This requires that the particle is 0.15 mm away from any PV.

## `M`: Mass

Returns $\sqrt{E^2 - p^2}$. Uses the function `LHCb::Particle::momentum().M()`. For many cases it should be practically the same as `MM`.

## `MM`: Measured Mass

Returns the measured mass of the particle. The function `LHCb::Particle::measuredMass()` is used for evaluation.

```
HltSharedDiMuon.CombinationCuts = "AM<11*GeV"
```

This selects dimuons up to a mass of 11 GeV.

## `NINGENERATION`: "Number of particles in generation"

The function which counts the particles satisfying the requirements in the decay tree at the given depth:

```
FilterDesktop.Filter.Code = " 2 == NINGENERATION ( ( 'mu+'==ABSID)  & ( PT > 2 * GeV ) , 2 ) "
```

Requires the presence of exactly two granddaughter $\mu^{\pm}$ with the transverse momentum $p_T > 2\mathrm{GeV}/c$. The generations are defined as: 0 - the particle itself, 1 - the direct children, 2 - grandchildren , etc. The concept and the name come from Patrick Koppenburg. The functor is available starting from Phys/LoKiPhys version >= v7r2.

## `NINTREE`: Number of particles in tree

Returns the number of particles in a tree satisfying some selection criteria.

```
FilterDesktop.Filter.Code = "2 == NINTREE( (ABSID=='e-') & (PT>1*GeV))"
```

Requires that there are two electrons in the tree with $P_T > 2\ \mathrm{GeV}$.

## `NMASS`: The nominal mass of a particle

Returns the PDG value of a particle mass. Useful to compute the difference with the measured mass. **This calls the `ParticlePropertySvc` at very call and is therefore slow**. Use `ADMASS('B0')` rather than `abs(MM-NMASS)` if you know the PID.

## `P`: Momentum

Gets the momentum of the particle.

```
FilterDesktop.Filter.Code = "P>2*GeV"
```

Gets particles with $P > 2\,\mathrm{GeV}$.

## `PIDe`, `PIDmu`, `PIDK`, `PIDp`, `PIDpi` : PID $\Delta\log\mathcal{L}_{x-\pi}$

Gets the combined delta-log-likelihood for the given hypothesis (wrt the pion), $\Delta\log\mathcal{L} = \log\frac{\mathcal{L}_x}{\mathcal{L}_\pi}$:

```
FilterDesktop.Filter.Code = "PIDe-PIDpi>6"
```

## `PT`: Transverse momentum

Gets the transverse momentum of the particle. Note that this is evaluated at the first measurement of the particle, which might not be where you want it for pions in $K_S$ decays.

```
FilterDesktop.Filter.Code = "PT>1*GeV"
```

Gets particles with $P_T > 1\,\mathrm{GeV}$.

## `TRCHI2`: $\chi^2$ the track fit

Gets the $\chi^2$ of the track fit.

```
FilterDesktop.Filter.Code = "TRCHI2<1000"
```

## `TRCHI2DOF`: $\chi^2$ per degree of freedom of the track fit

Gets the $\chi^2$ per degree of freedom of the track fit.

```
FilterDesktop.Filter.Code = "TRCHI2DOF<20"
```

## `TRPCHI2`: $\chi^2$-probability the track fit

Gets the $\chi^2$-probability of the track fit.

```
FilterDesktop.Filter.Code = "TRPCHI2>0.01"
```

## `VDMIN`: Distance of two vertices

**TODO** : Not yet supported.

## `VFASPF`: Vertex Function as Particle Function.

Allows to apply vertex functors to the particle's `endVertex()`.

```
CombineParticles.MotherCut = "VFASPF(VCHI2/VDOF)<10"
```

Applies a $\chi^2$ cut to the vertex of the particle.

# Particle Array Functors

Applied to a set of particles, typically the daughters

## `AHASCHILD`: Checks if there is a particle in the array that matches the requirements

```
CombineParticles.CombinationCut = "AHASCHILD((ABSID=='pi+') & (PT>1*GeV))"
```

Requires there's at least one pion with a Pt larger than 1 GeV in the combination. This is useful when one requires at least one particle in the decay to meet a requirement.

## `ADAMASS`: The absolute mass difference to the PDG reference value

Calculates the absolute difference between the measured mass and the PDG reference value. It takes the pid of the reference particle as argument.

```
CombineParticles.CombinationCut = "ADAMASS('KS0')<50*MeV"
```

## `AM`: Mass of the combination

Returns $\sqrt{E^2 - p^2}$.

## `AMAXCHILD`, `AMINCHILD`: Maximal, minimal value in array.

```
CombineParticles.CombinationCut = "1*GeV<AMINCHILD(MINTREE('pi-'==ABSID,PT)"
```

In this example one checks that the minimal PT of pions in the decay tree of the particles is in excess of 1 GeV.

## `AMAXDOCA`, `AMINDOCA`: Distance of closest approach cut

```
CombineParticles.CombinationCut = "(AMAXDOCA('') < 0.08*mm)"
```

Checks that the maximum distance of closest approach between all possible pairs of particles is less than 80mum. Equal results can be otained with less CPU time using

```
CombineParticles.CombinationCut = "ACUTDOCA(0.08*mm,'')"
```

## `AMAXDOCACHI2`, `AMINDOCACHI2`: Distance of closest approach significance cut

```
CombineParticles.CombinationCut = "(AMAXDOCACHI2('') < 10)"
```

Checks that the maximum distance of closest approach - chi2 between all possible pairs of particles is less than 10. Equal results can be otained with less CPU time using

```
CombineParticles.CombinationCut = "ACUTDOCACHI2(10,'')"
```

This cut is particularly useful to gain speed on vertex fits. As a rule of thumb, any vertex chi2 cut can be preceded by a DOCACHI2 cut at the same (total, not reduced!) chi2.

## `ANUM`: Number of particles in the array meeting a requirement

```
CombineParticles.CombinationCut = "1<ANUM( ('pi+'==ABSID) & (MIPCHI2DV(PRIMARY)>25))"
```

## `AP`: Sum momentum of the array

Gets the momentum of the array.

```
CombineParticles.CombinationCut = "AP>1*GeV"
```

Gets particles with $P > 1\,\mathrm{GeV}$.

## `APT`: Sum transverse momentum of the array

Gets the transverse momentum of the array (that's the $P_T$ of the sum 4-vector, not the sum of the $P_T$). Note that this is evaluated at the first measurement of the particles, which might not be where you want it for pions in $K_S$ decays.

```
CombineParticles.CombinationCut = "APT>1*GeV"
```

Gets particles with $P_T > 1\,\mathrm{GeV}$.

# Vertex functors

Vertex functors are accessed using `VFASPF`.

## `VX`, `VY`, `VZ`: Vertex x, y or z-position

```
CombineParticles.MotherCut = "VFASPF(VZ)>-100 & VFASPF(VZ)>100"
```

## `VCHI2` : Vertex $\chi^2$

```
CombineParticles.MotherCut = "VFASPF(VCHI2/VDOF)<10"
```

## `VDOF` : Vertex fit number of degrees of freedom

```
CombineParticles.MotherCut = "VFASPF(VCHI2/VDOF)<10"
```

A two-track vertex has one degree of freedom. A three-track vertex has three degrees of freedom.

## `VMINVDDV`: minimum distance between the vertex and any other vertex

The functor evaluates the minimal 3D-distance between the vertex and the vertices from "list". The list of vertices is extracted from the desktop:

```
CombineParticles.MotherCut = "VFASFP(VMINVDDV(PRIMARY))>1"
```

Requires that the particle's end-vertex has at least one millimeter separation with respect to any primary vertex.

## `VMINVDSOURCE`: minimum distance between the vertex and any other vertex

The functor evaluates the minimal 3D-distance between the vertex and the vertices form "list". The list of vertices is extracted from the "source":

```
CombineParticles.MotherCut = "VFASPF( VMINVDSOURCE( VSOURCEDV ( PRIMARY ) ) ) >1"
```

Requires that the particle's end-vertex has at least one millimeter separation with respect to any primary vertex.

## `VMINVDTES`: minimum distance between the vertex and any other vertex

The functor evaluates the minimal 3D-distance between the vertex and the vertices form "list". The list of vertices is extracted from the TES:

```
CombineParticles.MotherCut = "VFASPF( VMINVDTES ( 'SomeLocationInTES' , PRIMARY ) ) >1"
```

Requires that the particle's end-vertex has at least one millimeter separation with respect to any primary vertex.

## `VPCHI2`: $\chi^2$-probability

The functor evaluates the $\chi^2$-probability of the vertex, taking into account the number of degrees of freedom. The GSL routine `gsl_cdf_chisq_Q` functon is used for evaluation.

```
CombineParticles.MotherCut = "VFASPF(VPCHI2)>1*perCent"
```

The functor is available starting from Phys/LoKiPhys version >= v7r2

# Units

Presently units are not supported easily. This will change in the next release. To avoid confusion between `M` (mass) and `m` (meter) for instance a limited set will be defined. Vanya suggests:

- MeV, GeV
- mm, millimeter, cm , centimeter, meter
- perCent
- ns, fs, nanosecond, femtosecond, ps, picoseconds

These units are available starting from Phys/LoKiPhys version >= v7r2

## Syntax

Cut are combined using the bit-wide `&` and `|` operators, not boolean operators. This also requires that cuts are well protected using parentheses. Example:

```
FilterDesktop.Filter.Code = "( ((ABSID=='pi+') & (PT>400*MeV)) |
                              ((ABSID=='KS0') & (PT>1*GeV)))"
```

Cuts can be split over several lines.

## Types

Some allowed symbols are instances, therefore they do not require any parentheses, e.g. P, PT etc... Some symbols are **types**, e.g. `MIPDV`, and they require to be instantiated. The parentheses means invocation of constructor. E.g. `MIPDV(PRIMARY)` is just invocation of the constructor of **type** `MIPDV` with parameter `PRIMARY` (the predicate to select the vertices). the simplest way to check if the symbol is a type or instance is through inspection using the interactive python:

```
[lxplus097] ~/cmtuser/DaVinci_v19r12/Phys/DaVinci/v19r12/cmt % python
Python 2.5 (r25:51908, Oct 18 2007, 16:04:48)
[GCC 3.4.6 20060404 (Red Hat 3.4.6-8)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>from LoKiPhys.decorators import *
>>>from LoKiArrayFunctors.decorators import *
>>>P
P
>>>MIPDV
<class 'LoKiPhys.functions.LoKi::Particles::MinImpParDV'>
```

# Use of `CombineParticles`.

`CombineParticles` is a `LoKI::Algo`, and a `DVAlgorithm` that combines the input particles according to the decay descriptor. There are three cuts applied :

## `DaughtersCuts` :

On the incoming daughter particles. It is a map of `"particle" : "cuts"`

```
HltSelB2DstarMu.DaughtersCuts = {
// PT cut for muons
"mu-" :  "PT>400*MeV",
// IP_chi2 cut for D0 in D*+
"D*(2010)+'" : "MAXTREE('D0'==ABSID,BPVVDCHI2) >6.25 "
}
```

## `CombinationCut`

Once a combination has been made according to the decay descriptor but before the vertex fit. This cut is applied to the set of daughters. Therefore only Particle array functors can be used. They all start with an `A`.

## `MotherCut`

Applied the vertex fit on the mother. All the cuts that require the position of the vertex must be applied in `MotherCut`, while the others can be applied earlier (saving CPU). Note that for long lived particles like Ks it pays off to apply a loose mass cut in `CombinationCut` and a harder in `MotherCut`. The reason is that the vertex fit does a propagation of the momenta through the detector. You thus get the momentum at the Ks vertex, while in `CombinationCut` it's just the sum of the momenta of the daughters at their first measurement.

A simple example:

```
HltShared.Members += { "CombineParticles/HltSharedPhi2KK" }

HltSharedPhi2KK.PhysDesktop.InputLocations = { "Phys/HltNoPIDsKaons" }

HltSharedPhi2KK.DecayDescriptor = "phi(1020) -> K+ K-"

// (IP > 2 sigma^2)
HltSharedPhi2KK.DaughtersCuts = { "K+" : "MIPCHI2DV(PRIMARY)>4" }

// mass window +/- 300 MeV
HltSharedPhi2KK.CombinationCut = "ADAMASS('phi(1020)')<50"

// chi2/ degrees of freedom < 25
HltSharedPhi2KK.MotherCut = "VFASPF(VCHI2/VDOF)<25"
```

The decay descriptor can be overwritten by `DecayDescriptors` which allows to reconstruct several decays in one go.

```
HltShared.Members += { "CombineParticles/HltSharedDstarWithD02KPi" }
HltSharedDstarWithD02KPi.PhysDesktop.InputLocations = { "Phys/HltSharedD02KPi", "Phys/HltSharedSl

// also wrong-sign ones
HltSharedDstarWithD02KPi.DecayDescriptors = {"[D*(2010)+ -> pi+ D0]cc", "[D*(2010)+ -> pi+ D~0]cc

// mass window +/- 50 MeV, PT> 1.25 GeV
HltSharedDstarWithD02KPi.CombinationCut = "(ADAMASS('D*(2010)+')<50) & (APT>1250)"
```

```
// Chi2/nDoF < 25 .  m(D*-D0) = 145.5 MeV, +2
HltSharedDstarWithD02KPi.MotherCut = "(VFASPF(VCHI2/VDOF)<25) & (M-MAXTREE('D0'==ABSID,M)<165.5)"
```

## Mother Cut, Daughter Cut, Combination Cut

Each cut, which is applicable to mother is also applicable to daughter (and vice versa). Moreover the cut, used for daughter particle, being equipped with the proper `CHILD,CHILDFUN,CHILDCUT,MINTREE,MAXTREE,INTREE,NINTREE` meta- function, gives **IDENTICAL** result being applied for the mother particle.

From pure C++ point of view, both `DaughterCuts` and `MotherCuts` are objects which get as argument `const LHCb::Particle*`. But `CombinationCut` accepts as an argument `LoKi::Range_<LHCb::Particle::ConstVector>"`.

Therefore is requires the DIFFERENT objects. `CombinationCut` is applied for the combination of "good" daughter particles, and only for the combinations, which satisfy `CombinationCut`, the effective mother particle is created through teh vertex fit procedure, and `MotherCut` is applied for this effective mother particle.

## Help

Please do not send mails directly to the authors, but send the questions and requests through the following mailing lists:

`lhcb-davinci@cern.ch`, `lhcb-loki@cern.ch`, `lhcb-bender@cern.ch`

In this case more colleagues can profit from solutions to the problems and it will allow a bit more wide distribution of useful information, tricks, recipes, experience and the solutions. Also it will simplify the monitoring of the progress with the implementation of the missing functionality. As alternative to the mailing lists the following actions are also accepted:

1. One can submit the question to LoKi FAQ.
2. One can submit new task or request new feature through Savannah LoKi portal
3. One can submit new task or request new feature through Savannah Bender portal

## Examples

Many working examples can be found in the `Hlt/HltSelections` package from version `v6r0` released with `DaVinci v19r11`.

## Feature requests

Feature requests should be sent to the same mailing lists as above. For every new feature a suggested name should be provided. This drastically simplifies the life of developers and results in a ***MUCH HIGHER PRIORITY*** for the implementation.

## Change of names

Although it is very simple to define aliases for these functors (and many of them are actually available) their use is not encouraged. This is because we would like to define a minimum vocabulary everyone would have to be able to understand. Since this is rather new, suggestions for changes of names are accepted but should be agreed by the community. Please send requests to the mailing lists above. It is suggested to freeze this in the next version.

Suggestions for the names of the filters in `CombineParticles` are also welcome.

# Few related tools/algorithms.

A few simple tools can be used for debugging and/or validation/testing the various `LoKi::Hybrid` construction.

## `LoKi::Hybrid::PrintTool`

The most primitive tool, it implements the abstract interface `IPrintDecay`, and it allows to print some information about the decay tree. It is convinient to use the high-level algorithm `PrintDecayTree` to drive the tool:

```
// append the algorithm to the list of top level algorithms:
ApplicationMgr.TopAlg += { "PrintDecayTree/Printer"} ;

// declare the actual type/name of the IPrintDecay tool:
Printer.PrintDecayTool = "LoKi::Hybrid::PrintTool/PRINT" ;

// declare the variables to be printed by the tool, each "variable" is just LoKi-functor:
Printer.PRINT.Variables = {  "ID"   , "P/GeV"   ,   "PT/GeV"   , "M/GeV"    ,   "KEY"    } ;
```

With this configuration one obtains the following printout:

```
Printer.PRINT              INFO Print the decay tree:  ( B0 -> mu+  mu-  K+  K-  )
 ------------------------------------------------------------------------------------------------------------------
 |#          Decay          |        ID       |     (P/1000)    |    (PT/1000)    |        M        |       K
 ------------------------------------------------------------------------------------------------------------------
 |0 |--> B0                 |       511       |      262.81     |      6.1192     |      5099.1     |
 |1    |--> mu+             |       -13       |      56.877     |      1.6668     |      105.66     |
 |1    |--> mu-             |        13       |      128.8      |      3.7141     |      105.66     |
 |1    |--> K+              |       321       |      56.877     |      1.6668     |      493.68     |
 |1    |--> K-              |      -321       |      20.261     |     0.63129     |      493.68     |
 ------------------------------------------------------------------------------------------------------------------
```

For the default configuration, all columns are printed using `14.5g` "scientific" format and the column title is constructed from the functor itself. However both the format and the table header could be modified, e.g. if one adds into the previous configuration the explicit specification of `Format` and `Header`:

```
// Format: print only 5th (KEY) and 1st (ID) variable from the list of "Variables":
Printer.PRINT.Format = "| %5% %|6t|| %1% %|14t||"

// specify explicitly the table header: column names
Printer.PRINT.Header = "| Key | PdgID |"
```

In this case the output table looks like:

```
Printer.PRINT                        INFO Print the decay tree:  ( B0 ->  mu+  mu-  K+  K-  )
 ---------------------------------------
 |#          Decay          | Key  | PdgID  |
 ---------------------------------------
 |0 |--> B0                 | 5    | 511    |
 |1    |--> mu+             | 0    | -13    |
 |1    |--> mu-             | 1    | 13     |
 |1    |--> K+              | 2    | 321    |
 |1    |--> K-              | 5    | -321   |
 ---------------------------------------
```

For more details see the Doxygen documentation here.

### LoKi::Hybrid::PlotTool

The simple tool, which implements the abstract interface `IPlotTool` and allows to fill some histograms for the selected particles.

For more details see the Doxygen documentation here.

### LoKi::Hybrid::TupleTool

The simple tool, which implements the abstract interface here and allows to fill N-tuple with useful information for the selected particles.

The tool is a specialization of general "tuple tools", the concept developed by Jeremie Borel. The usage is failrly trivial:

```
ApplicationMgr.TopAlg += { "DecayTreeTuple/MyDecayTreeTuple" } ;

// specify location of input data in TES
MyDecayTreeTuple.PhysDesktop.InputLocations = {"Phys/HltSelBd2MuMuKstar"} ;

// use various tools, developed by Jeremie
MyDecayTreeTuple.ToolList = {
      "TupleToolTrigger"
      , "TupleToolMCTruth"
      , "TupleToolMCBackgroundInfo"
      , "TupleToolGeometry"
      , "TupleToolKinematic"
      , "TupleToolPropertime"
      , "TupleToolPrimaries"
     , "TupleToolEventInfo"
     // use also LoKi-based Tuple Tool!!
     , "LoKi::Hybrid::TupleTool/LoKiTool" /// < THIS LINE
};

// configure LoKi tuple tool:
MyDecayTreeTuple.LoKiTool.Variables = {
  "mass" : "M/GeV" ,
  "p"       : "P/GeV" ,
  "pt"      : "PT/GeV"
};
```

For this example, the tool adds three columns into N-Tuple, defines the names of the column to be `"head"+"mass"`, `"head"+"p"` and `"head"+"pt"`. The content of the colums will be filled with tresult of evaluation of the following LoKi functors: **`M/GeV`**, **`P/GeV`** and **`PT/GeV`**.

For more details see the Doxygen documentation here.

## Direct manipulatuon with factories

Sometime for debugging purposes the direct manipulation with factories could be useful. Also it could have some sense for validation of functors and comparison with some other calculations.

Since the factory is **`AlgTool`** in Gaudi sense, first one needs to acquire the factory

```
#include "LoKi/IHybridFactory.h"

...
const std::string factoryName = "LoKi::Hybrid::Tool/HybridFactory:PUBLIC" ;
```

```
LoKi::IHybridFactory* factory = tool<LoKi::IHybridFactory> ( factoryName ) ;
```

And now one can use the factory for creation of of functors (functions and predicates) :

```
using namespace LoKi::Types ;
using namespace LoKi::Cuts   ;

// initialize the function with *some* values

Fun fun = -1.E+10 * ONE ;

// get the code
const std::string& code = .... ;

// instantiate the functor:
StatusCode sc = factory->get ( code , fun ) ;

if ( sc.isFailure() ) { .... } // handle the error
```

In a similar way one can create predicate (the functor with returns `bool`):

```
using namespace LoKi::Types ;
using namespace LoKi::Cuts   ;

// initialize the predicate with *some* values

Cut cut = NONE ;

// get the code
const std::string& code = .... ;

// instantiate the functor:
StatusCode sc = factory->get ( code , cut ) ;

if ( sc.isFailure() ) { .... } // handle the error
```

Finally one can operate with **`fun`** and **`cut`** objects as with regular LoKi functors.

The complete example is available here.

-- PatrickKoppenburg - 28 Mar 2008 -- PatrickKoppenburg - 25 Apr 2008 -- Karol Hennessy - 15 May 2008 -- Vanya Belyaev - 12 Jun 2008 -- PatrickKoppenburg - 18 Aug 2008

--- This topic: LHCb > LoKiHybridFilters
Topic revision: r39 - 2019-08-02 - NiklasStefanNolte