Table of Contents

Primary vertex re-fitting

Introduction

This wiki is about the process of obtaining and safely using primary vertices that have been re-fitted excluding tracks from the decay of a particle. It does not, at the moment, deal with the tools that perform the re-fitting itself. These are explained elsewhere, for example the primary vertex reconstruction wiki. Here, we present two ways of performing the re-fitting in DaVinci, and give examples on how to use the results in a DaVinci job. This is independent of the method used to re-fit the vertices.

Re-fitting tools

At the moment there are two PV re-fitting tools that can be used in our physics analysis software framework: AdaptivePVReFitter by Yuehong Xie and PVOfflineTool, by Mariusz Witek. For more information about PVOfflineTool and primary vertex reconstruction see here

PV re-fitting in analysis

Primary vertex re-fitting can be achieved either automatically from a DVAlgorithm, or with PVRefitterAlg, an algorithm designed specifically for that purpose. In both cases, the re-fitted primary vertices and relations tables linking the particles to re-fitted vertices are stored on the TES. **It is very important that re-fitted vertices are accessed via the relations tables**. Re-fitted vertices have no meaning without the particles that were used to make them, and a container of re-fitted vertices will have as many versions of the same vertex as there are candidates in the event. This is why the TES location of the vertices is not advertised, but that of the relations tables is. The relations tables can be passed on to any DVAlgorithm via it's PhysDesktop's `P2PVInputLocations` property (see examples below). For more details, see PVRefitterAlg's doxygen documentation

Automatic re-fitting in DVAlgorithm

The DVAlgorithm base class has a property `ReFitPVs` which can be used to automatically re-fit primary vertices each time the `relatedVertex` method is called (see Particle2PV). In practice, this method is called each time a PV-related cut is applied in the creation or selection of an LHCb::Particle. It is set to `false` by default. When set to true, primary vertices for a particle are re-fitted each time the best vertex for a given Particle is asked for. The results are cached, such that the re-fitting isn't performed for a particle that already has a "best" re-fitted primary vertex. At the end of each event, the DVAlgorithm writes the particles out to TES location `"Phys/AlgoName/Particles"` and the relations tables to ="Phys/AlgoName/Particle2VertexRelations". When re-fitting has been performed, these relations link particles to re-fitted vertices.

```
myPMaker = CombineParticles("PMaker")
myPMaker.Code = ....
myPMaker.ReFitPVs = True
MySequence.Members +=[myPMaker]
```

The relations tables can be used by following DVAlgorithms such that the same vertices are used in the selection and in following analysis stages:

```
from Configurables import FilterDesktop # FilterDesktop is a DVAlgorithm
myFilter = FilterDesktop("ParticleFilter")
myFilter.Code = .....
myFilter.addTool(PhysDesktop)
myFilter.PhysDesktop.P2PVInputLocations = ["Phys/" + myPMaker.name() +"/Particle2VertexRelations"
MySequence.Members +=[myFilter]
```

Please note that the re-fitting operation can be time consuming, so bear in mind that selections that perform PV related cuts to reduce a large sample of candidates, and thus perform many re-fits, may incur a significant CPU penalty. If time is an issue, the selection could be re-ordered such that a smaller number of re-fits is performed (rejecting more candidates before the PV-related cuts are applied). It is also important to note that each algorithm for which `ReFitPVs` is set to **True** will perform the re-fits when necessary. If you are dealing with the same particles, there is a lot to be gained (rather, a lot not to be lost) by using the relations tables as described above.

DecayTreeTuple

DecayTreeTuple is just a DVAlgorithm that is used to write out quantities to a ROOT ntuple at some stage of your analysis. Therefore, everything that is said above about using the refitted vertices in a later stage of the analysis must also be applied to your DecayTreeTuple instance. If you do not use the refitted PVs with DecayTreeTuple, the BPV* quantities in your ntuple will be incorrect (i.e., you will see distributions extending past the value of the cut that you applied in your selection).

PVReFitterAlg: re-fit independently from particle selection and/or creation

The PVReFitterAlg is a GaudiAlgorithm which holds a pointer to an IPVReFitter. It produces re-fitted primary vertices and a relations table associating each particle to the vertices that were re-fitted using it. The table has artificial weights to maintain the order of the original primary vertices. The PVReFitterAlg's configuration properties allow the user to set the TES location of primary vertices and LHCb::Particles. In the example below, we re-fit primary vertices and create a `Particle2Vertex::Table` relating `LHCb::Particles` from "Phys/PMaker/Particles" to the re-fitted primary vertices and put it in "Phys/PMaker/myPVReFitterAlg_P2PV".

```
myPMaker = CombineParticles("PMaker")
myPMaker.Code = ....
mainLocation = "Phys/" + myPMaker.name()
MySequence.Members +=[myPMaker]
from Configurables import PVReFitterAlg
PVReFitter = PVReFitterAlg("myPVReFitterAlg")
PVReFitter.ParticleInputLocation = mainLocation+"/Particles"
PVReFitter.OutputLevel=1
MySequence.Members += [PVReFitter]
```

The relations table could be used as input to additional DVAlgorithms. However, the relations have to be sorted according to some "bestness" criteria, since the DVAlgorithm assumes sorted relations (maybe this should be done automatically??) Here, we start by creating a new relations table, sorted according to some "bestness" criteria (for more details of the sorting, see Particle2PV).

```
from Configurables import PVRelatorAlg
p2RefitPV = PVRelatorAlg("P2ReFitVertices")
p2RefitPV.P2PVRelationsInputLocation = mainLocation + "/Particle2ReFittedVertexRelations"
p2RefitPV.P2PVRelationsOutputLocation = mainLocation + "/P2ReFitPVSortedRelations"
```

Next, we pass the sorted relations table to the DVAlgorithm:

```
from Configurables import FilterDesktop # FilterDesktop is a DVAlgorithm
myFilter = FilterDesktop("ParticleFilter")
myFilter.Code = .....
myFilter.addTool(PhysDesktop)
myFilter.PhysDesktop.P2PVInputLocations = mainLocation+"/P2ReFitPVSortedRelations"
MySequence.Members +=[p2RefitPV, myFilter]
```

-- JuanPalacios - 20 Jul 2009

This topic: LHCb > RefitPV
Topic revision: r6 - 2010-07-26 - unknown