Vertex Analysis Package

— Version Last update (`../../VERSION`) —

Module vertex_analyze

S. Choi, S. Grinstein, J.G. Lima, M. Narain, A. Schwartzman, F. Stichelbaut

Contents

1 Introduction

In Run II, the SMT detector will allow to reconstruct with high precision the position of the primary vertices as well as secondary vertices coming from B or D meson decays.

The vertex reconstruction task is performed under the steering of the framework package VertexReco (*vertexreco* module). Amongst all reconstructed vertices, primary and secondary vertices are then selected to be written out in primary and secondary vertex chunks, for analysis purposes. This selection is handled by the framework package VertexSelect (*vertex_select* module). For more information about the vertex reconstruction system, see vertexreco documentation (`http://www-d0.fnal.gov/d0dist/dist/releases/test/vertexreco/doc/index.html`).

This package is used to analyze the results of the vertex reconstruction and selection. It produces ntuple blocks containing the necessary information about MC generated tracks and vertices, as well as reconstructed tracks and vertices. The corresponding library is:

- libvertex_analyze.a

The *vertex_analyze* module depends on the following modules:

- *vertexutil*
- *vertex_evt*
- *rcp*

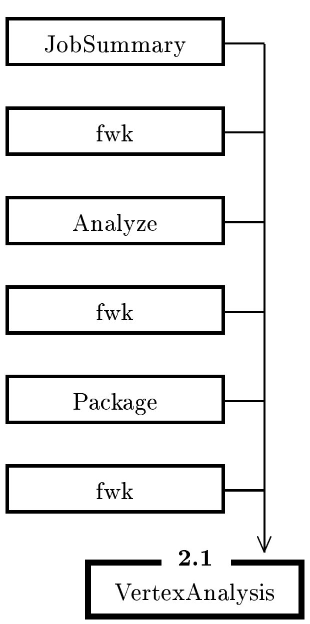
2 Class documentation

Names

2.1 class **VertexAnalysis** : public fwk Package, public fwk Analyze, public fwk JobSummary

Framework package for vertex analysis

Inheritance



Public Members

Private Members

Framework package for vertex analysis.

Based on user input via RCP file, *VertexAnalysis* instantiates one or more of the classes described below. For most classes, there is a one to one correspondence between classes and ntuple blocks. For documentation on the variables stored, please see the documentation for the corresponding class, at the Class documentation (→2, *page 5*) section.

2.1.1 Methods

Names

VertexAnalysis (fwk::Context* context)
Constructor

~VertexAnalysis ()
Destructor

fwk::Result
analyzeEvent (const edm::Event &event)
Process each event

fwk::Result
jobSummary () *Produce summary of job*

std::string **packageName** () *Give package name*

2.1.2 Attributes

Names

int **_debug** *Debug flag*

HepNtuple*
_ptup *Ntuple file*

static const int
_numBlocks *Number of Ntuple blocks*

static const int
_numVRBlocks *Number of vertexreco ntuple blocks*

static const int
_numVtxMatch *Number of vertex matching blocks*

bool **_Need_Block** [_numBlocks]
Ntuple flags

std::string **_block** [_numBlocks]
Block names

vertex::BlockVertex*
_blVertex [_numVRBlocks]
Pointer to vertex block

vertex::BlockVertexMatch*
_blVertexMatch [_numVtxMatch]
Pointer to Vertex Match block

vertex::BlockMcVtx*
_blMcVtx *Pointer to MC vertices block*

vertex::BlockMcTrk*
_blMcTrk *Pointer to MC tracks block*

vertex::BlockGTrack*
_blGTrack *Pointer to GTracks block*

vertex::BlockKalmanFit*
_blKalmanFit *Pointer to KalmanFit block*

vertex::BlockKTrack*
_blKTrack *Pointer to KTrack block*

vertex::BlockMcInfo*
_blMcInfo *Pointer to MCinfo block*

vertex::BlockMCpGTrackMatch*

_blMCpGTrackMatch *Pointer to MCparticle-Gtrack Match block*

vertex::BlockQuark*
_blQuark *Pointer to Quark block*

vertex::BlockSeed*
_blSeed *Pointer to Seed block*

vertex::BlockVtrackConeAlgo*
_blVtrackCone *Pointer to Track Clustering block*

vertex::BlockSeedCone*
_blSeedCone *Pointer to SeedCone block*

vertex::VertexHistos*
_pHistos *Pointer to class for vertex histograms*

vertex::BlockEvent*
_blEvent *Pointer to class for event information*

2.2 class **VertexHistos**

Class to make histograms for vertexing studies

Public Members

Private Members

Class to make histograms for vertexing studies. It takes the list of reconstructed and selected vertices from the various **VertexCollChunk**'s and fills some histograms.

2.2.1 Methods

Names

	VertexHistos (HepFileManager* man, const int& debug =0)	*Constructor*
	~VertexHistos ()	*Destructor*
void	**fill** (const edm::Event& event)	*Fills histograms using data from event*

2.2.2 Methods

Names

void	**book** ()	*histogram booking method*
void	**fill** (const std::vector<Vertex>* pvcoll)	*histogram filling method*

2.2.3 Attributes

Names

int	**_debug**	*Debug flag*
HepFileManager*	**_man**	*HepTuple file manager*
HepHist1D*	**_hx**	*x-distribution*
HepHist1D*		

	_hy	*y-distribution*
HepHist1D*	**_hz**	*z-distribution*

2.3 namespace vertex class **BlockMcTrk**

Ntuple block for generated tracks

Public Members

Private Members

Ntuple block for generated tracks. This class fetches the generated tracks in the event from the last MCKineChunk, and stores the essential information into an ntuple block called MCTRK.

Type	Name	Description
int	nmctrk	Number of generated charged tracks
float	gpx	x-component of momentum
float	gpy	y-component of momentum
float	gpz	z-component of momentum
int	gvtx	MC vertex where this track was produced
int	index	**mcpp** particle index (see mcpp documentation)
int	id	**Isajet** particle ID code (see Isajet documentation (`http://www-d0.fnal.gov/d0dist/dist/releases/test/isajet/doc/isajet.ps`))

Notes:

- The purpose of this block is to help debugging vertex reconstruction. Therefore, only charged stable particles are stored in this block.
- Variable *gvtx* is an integer, which is the number of the vertex in MCVTX block. See BlockMcVtx (→2.4, *page 13*).

2.3.1

Member functions

Names

2.3.1.1

BlockMcTrk (HepNtuple* ptup, const int& debug = 0)

Constructor

Constructor. Note that the last argument is optional, and the default value is zero.

2.3.2

Attributes

Names

int	**_debug**	*Debug flag*
HepNtuple*	**_ptup**	*Pointer to ntuple file*
int	**_nmctrk**	*Index for arrays of generated vertices*
edm::TKey <MCKineChunk>		

_mcKey *Key to MCKineChunk*

2.4

namespace vertex class **BlockMcVtx**

Ntuple block for generated vertices

Public Members

Private Members

Ntuple block for generated vertices.

It gets the generated vertices from **all** the MCKineChunks in the event, and stores the essential information into an ntuple block, called MCVTX.

Type	**Name**	**Description**
int	nmcvtx	Number of generated vertices
int	gntrk	Number of *charged stable* tracks produced at this vertex
int	gvindex	particle index which points to this vertex
int	gvtype	"Flavour type", see note below
float	gx	Vertex x-coordinate
float	gy	Vertex y-coordinate
float	gz	Vertex z-coordinate

Notes:

- The purpose of this block is to help debugging vertex reconstruction. Therefore, only spatially distinct vertices are stored in this block.
- The following flavour types are used:

  1 primary vertex (hard scattering vertex)

  2 Geant scattering

  3 light hadron decay

4 charm hadron decay

5 bottom hadron decay

11 min-bias

14 charm hadron decay which originally came from a b-decay

- Variable *gvtype* is an integer, which is used in MCTRK blocks so that each track can tell which vertex it comes from. See BlockMcTrk (→2.3, *page 11*).

2.4.1 Methods

Names

2.4.1.1 BlockMcVtx (HepNtuple* ptup, const int& debug = 0)

Constructor

Constructor. Note that the last argument is optional, and the default value is zero.

2.4.2

Attributes

Names

int	**_debug**	*Debug flag*
HepNtuple*	**_ptup**	*Pointer to ntuple file*
int	**_nmcvtx**	*Index for ntuple block*
edm::TKey <MCKineChunk>	**_mcKey**	*Key to MCKineChunk*

2.5

namespace vertex class **BlockQuark**

Ntuple block for generated quarks

Public Members

Private Members

Ntuple block for generated quarks.

Type	Name	Description
int	nquark	Number of generated quarks
float	qeta	eta direction
float	qphi	phi direction
float	qpt	transverse momentum
int	qid	**Isajet** particle ID code (see Isajet documentation (`http://www-d0.fnal.gov/d0dist/dist/releases/test/isajet/doc/isajet.ps`))

2.5.1 Member functions

Names

	~BlockQuark ()	*Destructor*
void	**book** ()	*Book ntuple block*
void	**fill** (const edm::Event& event)	*Fill block with data from MCKineChunk*

2.5.1.1 BlockQuark (HepNtuple* ptup, const int& debug = 0)

Constructor

Constructor. Note that the last argument is optional, and the default value is zero.

2.5.2 Attributes

Names

int	**_debug**	*Debug flag*
HepNtuple*	**_ptup**	*Pointer to ntuple file*
int	**_nquark**	*Index for arrays of generated vertices*
edm::TKey <MCKineChunk>		

_mcKey *Key to MCKineChunk*

2.6

namespace vertex class **BlockGTrack**

Ntuple block for reconstructed global tracks (GTracks)

Public Members

Private Members

Ntuple block for reconstructed global tracks (GTracks). It gets the reconstructed tracks in the event from the GTrackChunks, and stores the essential information into an ntuple block, called GTRACK.

Type	Name	Description
I	ngtrk	Number of GTrack to McTrk associations
I	gtindex	Position of GTrack in GTrackChunk
I	nsmt	Number of SMT hits along reconstructed track
I	ncft	Number of CFT hits along reconstructed track
F	rsig	$r\phi$ impact parameter * sign of α
F	zca	z-coordinate at point of closest approach
F	phidir	ϕ direction of track propagation at dca
F	tanl	tan(λ), λ is angle between track and xy plane
F	qoverpt	q/p_T
F	drsig	Uncertainty in rsig
F	dzca	Uncertainty in zca
F	dphidir	Uncertainty in phidir
F	dtanl	Uncertainty in tanl
F	dqoverpt	Uncertainty in qoverpt
F	drz	covariance of rsig and zca
F	drp	covariance of rsig and phidir
F	drt	covariance of rsig and tanl
F	drq	covariance of rsig and qoverpt
F	dzp	covariance of zca and phidir
F	dzt	covariance of zca and tanl
F	dzq	covariance of zca and qoverpt
F	dpt	covariance of phidir and tanl
F	dpq	covariance of phidir and qoverpt
F	dtq	covariance of tanl and qoverpt
F	rc	ITrack: radius of circle in $r\phi$ view
F	xc	ITrack: x-coordinate of center in $r\phi$ view
F	yc	ITrack: y-coordinate of center in $r\phi$ view
F	zc	ITrack: z-coordinate of center in $r\phi$ view
F	tanth	$tan(\theta)$, θ is the polar angle. the same as 1/tanl
F	drmin	$dr = \sqrt{d\eta^2 + d\phi^2}$ to closest generated particle
I	mcidx	Index to closest generated particle in MCTRK block, see BlockMcTrk (→2.3, *page 11*).
F	dca0	distance of closest approach to the PV selected in transverse plane
F	dca1	z at DCA to the primary vertex
F	dcaerr0	Uncertainty in dca0
F	dcaerr1	Uncertainty in dca1
F	chi_2	χ^2 of the track fit

2.6.1 Methods

Names

	~BlockGTrack ()	*Destructor*
void	**book** ()	*Books the ntuple block*
void	**fill** (const edm::Event& event)	*Fills the ntuple block*

2.6.2 Attributes

Names

int	**_debug**	*Debug flag*	
HepNtuple*	**_ptup**	*Pointer to ntuple file*	
int	**_ngtrk**	*Index for GTRACK ntuple block*	
2.6.2.1 GTrackContentIdSelector	**_selector**	*GTrack Id selector*	19
2.6.2.2 VtrackChunkSelector	**_vtrSelector**	*Vtrack Id selector*	20
VertexType	**_vtxType**	*Vertex type, used to select vertex chunk to get data from*	
edm::TKey <GTrackChunk>	**_gtrackKey**	*Key to GTrackChunk*	
edm::TKey <VtrackCollChunk>	**_vtrKey**	*Key to VtrackCollChunk*	

2.6.2.1 GTrackContentIdSelector **_selector**

GTrack Id selector

GTrack Id selector. This is used to select the GTrackChunk which contains the data to be stored

2.6.2.2

VtrackChunkSelector **_vtrSelector**

Vtrack Id selector

Vtrack Id selector. This is used to select the VtrackChunk which contains the data to be stored

2.7

class **BlockVertex**

Ntuple blocks for reconstructed vertices

Public Members

Private Members

Ntuple blocks for reconstructed vertices.

It takes the list of reconstructed and selected vertices from the various **VertexCollChunk**'s and store the essential information into an ntuple block.

Each instance of the class handles a different chunk, according to a VertexType object passed to its constructor. Possible values are shown on table below (note that VertexType values are under vertex namespace):

VertexType	Block name	Prefix	Description
PRIMARY_RECO	PVRECO	pr	All reconstructed primary vertices
SECONDARY_RECO	SVRECO	sr	All reconstructed secondary vertices
PRIMARY	PVSEL	ps	Vertices selected as primary vertices by VertexSelect
SECONDARY	SVSEL	ss	Vertices selected as secondary vertices by VertexSelect
MCPRIMARY	PVMC	pm	Monte Carlo primary vertices in the same format as reconstructed vertices. The info comes from a VertexCollChunk, like any other reconstructed vertex
MCSECONDARY	SVMC	sm	Monte Carlo secondary vertices in the same format as reconstructed vertices. The info comes from a VertexCollChunk, like any other reconstructed vertex
KALMAN	KALMAN	kl	All vertices reconstructed by Kalman algorithm
KALMAN_PRIMARY	PVKALMAN	pk	Vertices selected as primary vertices by VertexSelectKalman
KALMAN_SECONDARY	SVKALMAN	sk	Vertices selected as secondary vertices by VertexSelectKalman
TOPORECO	TOPO	tp	All vertices reconstructed by topological algorithm
SECPROB	SECPROB	sp	All secondary vertices reconstructed by probabilities algorithm

Since these ntuple blocks are managed by the same class, extra characters had to be added by hand to the names of the ntuple variables, in order to avoid name clashes. Therefore, the first two characters of the variables in each of these blocks vary depending on the block name. The characters used for each block are also shown on the above table, at the column labeled *prefix*. For instance, all variables from PVRECO block start with pr.

In the following table we list all the variables available in the blocks for reconstructed vertices. Note that "XX" in the variable names should be replaced with the two-character prefixes shown in the previous table:

Type	Name	Description
int	XXnvtx (index)	Number of vertices in the chunk
float	XXx	Vertex x-coordinate
float	XXy	Vertex y-coordinate
float	XXz	Vertex z-coordinate
float	XXsigx	Uncertainty in vertex x-coord
float	XXsigy	Uncertainty in vertex y-coord
float	XXsigz	Uncertainty in vertex z-coord
float	XXsigxy	Covariance of x and y position
float	XXsigxz	Covariance of x and z position
float	XXsigyz	Covariance of y and z position
int	XXntrk	Number of tracks attached to this vertex
int	XXattgt$i$	Packed word: See text for more info
float	XXchisq	Vertex chi-square
int	XXndof	Ndof in vertex fit
int	XXid	Vertex index

As mentioned earlier, all variables listed in this table are prefixed by 'xx', where this prefix uniquely identifies the block which holds each variable. Prefixes are listed at the *prefix* column in the first table of this section.

2.7.1 Public methods

Names

	BlockVertex (HepNtuple* ptup, const VertexType VtxType, const int& debug =0)	*Constructor*
	~BlockVertex ()	*Destructor*
void	**fill** (const edm::Event& event)	*Filling method*

2.7.2 Private methods

Names

void	**book** ()	*book Ntuple block for reconstructed vertices*
void	**fill** (const std::vector<Vertex>* pvcoll)	*fill Ntuple block for reconstructed vertices*

2.7.3 Attributes

Names

		Arrays size
int	**_debug**	*Debug flag*
HepNtuple*	**_ptup**	*Pointer to ntuple file*
int	**_nvertex**	*Index for ntuple block*

VertexType	**_vtxType**	*Vertex type, used to select vertex chunk to get data from*
std::string	**_block**	*Ntuple block names*

2.8

class **BlockKalmanFit**

Ntuple block for Kalman Fitting analysis

Public Members

Private Members

Ntuple block for Kalman Fitting analysis.

Kalman tracks are built by the Kalman algorithm for vertex reconstruction. This algorithm modifies the track parameters, so that the track goes through the fitted vertex.

The purpose of this block is to study the Kalman filter as a vertex fitting algorithm (without pattern recognition). The Kalman Filter is applied to all MC vertices and the fitted track parameters are stored to compare them with the original Gtracks and MCparticles.

The variables stored in this block are described in the following table:

Type	Name	Description
int	ntrack	Number of KTracks (ntuple index)
float	dca_k	Distance of closest approach to beam axis (KTrack)
float	dca_mc	Distance of closest approach to beam axis (MC track)
float	phi_k	phi direction at the dca (KTrack)
float	phi_mc	phi direction at the dca (MC particle)
float	pxK	x momentum at the vertex (KTrack)
float	pyK	y momentum at the vertex (KTrack)
float	pzK	z momentum at the vertex (KTrack)
float	pzMC	z momentum at the vertex (MC particle)
float	qptK	charge over transverse momentum (Ktrack)
float	qpt_k	charge over transverse momentum (Ktrack)
float	qpt_mc	charge over transverse momentum (MC particle)
float	sdca_k	error in dca (KTrack)
float	sphi_k	error in phi at the dca (KTrack)
float	spxK	error in x momentum at the vertex (KTrack)
float	spyK	error in y momentum at the vertex (KTrack)
float	spzK	error in z momentum at the vertex (KTrack)
float	sqpt_k	error in charge over transverse momentum (KTrack)
float	stanlK	error in tan(lambda) at the vertex (KTrack)
float	stanl_k	error in tan(lambda) at the dca (KTrack)
float	sz_k	error in z at the dca (KTrack)
float	tanl_k	tan(lambda) at the dca (KTrack)
float	tanl_mc	tan(lambda) at the dca (MC particle)
float	z_k	z at the dca (KTrack)
float	zdca_mc	z at the dca (MC particle)
float	x_mc	x fitted vertex position
float	y_mc	y fitted vertex position
float	z_mc	z fitted vertex position
float	sx_mc	σ_x^2 of fitted vertex position
float	sy_mc	σ_y^2 of fitted vertex position
float	sz_mc	σ_z^2 of fitted vertex position
float	mass	vertex mass
float	chisq	vertex χ^2
int	iter	number of iterations taken by the algorithm
int	nt	number of reconstructed tracks in the vertex (matched to MC)

Note: For more information about the Kalman algorithm, please refer to the Kalman algorithm documentation available at `http://www-d0.fnal.gov/d0dist/dist/releases/test/valgkalman/doc/`.

2.8.1 Methods

Names

2.8.1.1 BlockKalmanFit (edm::RCP& rcp, int& selector, HepNtuple* ptup, const int& debug = 0)

Constructor

Constructor. Note that the last argument is optional, and the default value is zero.

2.8.2 Private methods

Names

void **read_vtracks** (const edm::Event& event)
Read Vtracks

2.8.3

Attributes

Names

std::list <Vtrack>		
	_vtracks	*Vtrack collection*
Vertex*	**_pVertex**	*Primary vertex*
MCParticleVtrackMatch*		
	_trackMatch	*Pointer to object which finds MC-particle vs. Vtrack matches*
VertexKalmanFitter*		
	_fitter	*Kalman Fitter*
GTrackContentIdSelector		
	_selector	*GTrack Id selector*
edm::TKey <GTrackChunk>		
	_gtrackKey	*Key to GTrackChunk*
edm::TKey <MCKineChunk>		
	_mcKey	*Key to MCKineChunk*
int	**_debug**	*Debug flag*
HepNtuple*		
	_ptup	*Pointer to ntuple file*
int	**_ntrk**	*Index for vertexreco CWN-tuples*

2.9

class **GTrackMcInfo**

Finds associations between generated and reconstructed tracks

Public Members

Private Members

Finds associations between generated and reconstructed tracks.

The association is based on which MC track has produced the majority of the hits used to reconstruct each global track (GTrack):

- each GTrack keeps a list of its hits
- each hit keeps a record (MCID) of the generated MC track which has produced it.

2.9.1 Methods

Names

GTrackMcInfo () *Default constructor*

GTrackMcInfo (int selDet)
Constructor with bool (1: SMT, 0: CFT, other: both)

~GTrackMcInfo ()
Destructor

TrackMcIdMap
makeMap (const edm::Event& event)
Make map from event

void **dumpMCparticles** (const edm::Event& event)
Dump MCparticles from event

void **dumpMap** () *Dump track map*

int **bestID** (std::vector<int>& mcids)
Get "best" MCID in clusters

std::pair <int, int>
pbestID (std::vector<int>& mcids)
Get "best" MCID in clusters and multiplicity as pair

int **bestID** (std::vector<int>& mcids, int& multiplicity)

Get "best" MCID in clusters and multiplicity

void **dumpCFTcluster** (int gindex)
Dump the CFT cluster of GTrack (index)

void **dumpSMTcluster** (int gindex)
Dump the SMT cluster of GTrack (index)

2.9.2 Function

Names

std::vector <double>
get_xyz_cft (CftCluster& cluster)
Get xyz of cft clusters

2.9.3 Attributes

Names

MCevt* **_ptMCevt** *Pointer to MC event*

std::list <int>
_mcpList *Vector of MCid from MCparticles*

std::vector <int>
_cluMcIds *Vector of MCid from Clusters*

TrackMcIdMap
_trackmap *Map of tracks to ID*

CFT_cluster_map
_cft_cluster_map *CFT clusters*

SMT_cluster_map

	_smt_cluster_map	*SMT clusters*
bool	**_doCFT**	*Use CFT*
bool	**_doSMT**	*Use SMT*
int	**_debug**	*Debug control*

2.10 namespace vertex class **BlockMcInfo**

Fills the ntuple block MCINFO

Public Members

Private Members

Fills the ntuple block MCINFO.

Block MCINFO provides a mapping between generated and reconstructed tracks. It uses GTrackMcInfo, which builds a map relating reconstructed G-Tracks to generated MC tracks.

The MCID variable is an integer, corresponding to the position of the MC track in the ntuple block MCTRK (see BlockMcTrk (→2.3, *page 11*)).

Type	**Name**	**Description**
int	ngtrks	Number of GTrack to McTrk associations
int	mcid	Particle index to McTrk (see BlockMcTrk (→2.3, *page 11*))
int	mult	Multiplicity of CFT/SMT hits from McTrk

2.10.1 public Methods

Names

~BlockMcInfo () *Destructor*

void **book** () *Book ntuple block*

void **fill** (const edm::Event& event)
Fill ntuple block

2.10.1.1 **BlockMcInfo** (HepNtuple* ptup, const int& debug = 0)

Constructor

Constructor. Note that the last argument is optional, and the default value is zero.

2.10.2 Attributes

Names

int **_debug** *Debug flag*

HepNtuple* **_ptup** *Pointer to ntuple file*

int **_ngtrks** *Index for ntuple block*

edm::TKey <MCKineChunk> **_mcKey** *Key to MCKineChunk*

2.11

class **BlockSeed**

Ntuple block for vertex seed analysis

Public Members

Private Members

Ntuple block for vertex seed analysis.

The variables stored in this block are described in the following table:

Type	**Name**	**Description**
int	nseed	Number of seeds (ntuple index)

Note: For more information about the Kalman algorithm, please refer to the Kalman algorithm documentation available at `http://www-d0.fnal.gov/d0dist/dist/releases/test/valgkalman/doc/`.

2.11.1

Methods

Names

~BlockSeed () *Destructor*

void **book** () *Book ntuple block*

void **fill** (const edm::Event& event)

Fill ntuple block

2.11.1.1

BlockSeed (edm::RCP& rcp, HepNtuple* ptup, int& selector, const int& debug = 0)

Constructor

Constructor. Note that the last argument is optional, and the default value is zero.

2.11.2

Private methods

Names

void **initialize** (const edm::Event& event)
Read Vtracks

2.11.3

Attributes

Names

std::list <Vtrack>
_vtracks *Vtrack collection*

VertexType
_vertexType *Primary vertex*

VertexKalmanFitter*
_fitter *Kalman Fitter*

GTrackContentIdSelector

	_selector	*GTrack Id selector*
edm::TKey <GTrackChunk>	**_gtrackKey**	*Key to GTrackChunk*
int	**_debug**	*Debug flag*
HepNtuple*	**_ptup**	*Pointer to ntuple file*
int	**_nseed**	*Index for vertexreco CWN-tuples*

2.12 class **BlockSeedCone**

Ntuple block for vertex seed analysis

Public Members

Private Members

Ntuple block for vertex seed analysis.

The variables stored in this block are described in the following table:

Type	**Name**	**Description**
int	nseed	Number of seeds (ntuple index)

Note: For more information about the Kalman algorithm, please refer to the Kalman algorithm documentation available at `http://www-d0.fnal.gov/d0dist/dist/releases/test/valgkalman/doc/`.

2.12.1 Methods

Names

2.12.1.1 BlockSeedCone (edm::RCP& rcp, HepNtuple* ptup, int& selector, const int& debug = 0)

Constructor

Constructor. Note that the last argument is optional, and the default value is zero.

2.12.2 Private methods

Names

void **initialize** (const edm::Event& event)
Read Vtracks

2.12.3

Attributes

Names

std::vector <Vtrack>
_vtracks *Vtrack collection*

VertexType
_vertexType *Primary vertex*

VertexInitAlg*
_coneAlgo *cone algorithm*

VertexKalmanFitter*
_fitter *Kalman Fitter*

GTrackContentIdSelector
_selector *GTrack Id selector*

edm::TKey <GTrackChunk>
_gtrackKey *Key to GTrackChunk*

int **_debug** *Debug flag*

HepNtuple*
_ptup *Pointer to ntuple file*

int **_nseedc** *Index for vertexreco CWN-tuples*

int **_eventNumber** *Event mumber*

2.13

class **BlockMCpGTrackMatch**

Ntuple block for GTrack-MCparticle matching analysis

Public Members

Private Members

Ntuple block for GTrack-MCparticle matching analysis.

Type	Name	Description
float		
int		

2.13.1 Methods

Names

2.13.1.1 **BlockMCpGTrackMatch** (edm::RCP& rcp,
HepNtuple* ptup,
int& selector,
const int& debug = 0)
Constructor 36

˜BlockMCpGTrackMatch ()
Destructor

void **book** () *Book ntuple blocks*

void **fill** (const edm::Event& event)
Fill ntuple blocks

2.13.1.1 BlockMCpGTrackMatch (edm::RCP& rcp, HepNtuple* ptup, int& selector, const int& debug = 0)

Constructor

Constructor. Note that the last argument is optional, and the default value is zero.

2.13.2 Attributes

Names

int	**_debug**	*Debug flag*
HepNtuple*	**_ptup**	*Pointer to ntuple file*
GTrackContentIdSelector	**_selector**	*GTrack Id selector*
edm::TKey <GTrackChunk>	**_gtrackKey**	*Key to GTrackChunk*
edm::TKey <MCKineChunk>	**_mcKey**	*Key to MCKineChunk*

2.14 namespace vertex class **BlockVertexMatch**

Ntuple block for MC-Reco vertex matching analysis

Public Members

Private Members

Ntuple block for MC-Reco vertex matching analysis.

This class takes lists of reconstructed and generated vertices from the event chunks, and stores the essential information into ntuple blocks.

Each instance of the class handles a different pair of chunks: one for MC vertices, the other for reconstructed vertices. Reconstructed vertices are taken from a VertexCollChunk, according to the VertexType object passed to its constructor (see vertexutil module (`http://www-d0.fnal.gov/d0dist/dist/releases/test/vertexutil/doc/`)).

The matching criteria are controlled via RCP file. The default RCP file is available at vertexmatch/rcp/VertexMatch.rcp (`../../../vertexmatch/rcp/VertexMatch.rcp`).

The following table lists all the variables available in the block for vertex matching:

Type	**Name**	**Description**
int		
float		

2.14.1

Methods

Names

2.14.1.1		**BlockVertexMatch** (edm::RCP& rcp, const VertexType type, HepNtuple* ptup, const int& debug = 0) *Constructor*	39
		~BlockVertexMatch () *Destructor*	
	void	**book** () *Book ntuple block*	
	void	**fill** (const edm::Event& event) *Fill ntuple block*	

2.14.1.1

BlockVertexMatch (edm::RCP& rcp, const VertexType type, HepNtuple* ptup, const int& debug = 0)

Constructor

Constructor. Note that the last argument is optional, and the default value is zero.

2.14.2 Attributes

Names

int	**_debug**	*Debug flag*
HepNtuple*	**_ptup**	*Pointer to ntuple file*
int	**_matchedVertex**	*Index for vertexreco CWN-tuples*
VertexType	**_vtxType**	*Vertex type, used to select vertex chunk to get data from*

2.15 namespace vertex class **BlockEvent**

Ntuple block for event information for real data

Public Members

Private Members

Ntuple block for event information for real data. The information is stored in a block called EVENT.

Type	Name	Description
I	evtnum	Event number
I	runnum	Run number
I	lumblk	Luminosity block number
I[256]	L1TFW	Level 1 trigger framework AndOr terms
I[128]	L3bits	Level 3 trigger bit information

2.15.1 Methods

Names

	BlockEvent (HepNtuple* ptup, const int& debug = 0)	*Constructor*
	˜BlockEvent	*Destructor*
void	**book** ()	*Booker function. Books event information for real data.*
void	**fill** (const edm::Event& event)	*Fills the ntuple block*

2.15.2 Attributes

Names

int	**_debug**	*Debug flag*
HepNtuple*	**_ptup**	*Pointer to ntuple file*
int	**_evtno**	*Event number*
int	**_runno**	*Run number*
int	**_lumblk**	*Luminosity block number*

3 Global function in vertex namespace

Names

3.1

namespace vertex void **book_block** (HepNtuple* _ptup, const std::string& block, const std::string& idxName, const int& blsize, int& index, const int& numVar, const int& numInt, const std::string names[])

Books any ntuple blocks containing only int*s and* float*s*

Books any ntuple blocks containing only *int*s and *float*s.

This is a global function in vertex namespace.

Function book_block takes a list of variables, and a name for the block to be created. It currently handles only integer and float ntuple variables, but can be easily extended for other types as well.

4 Release Notes

Please check this link (`../rel-notes.html`) for the most recent release comments, directly extracted from the cvs repository.

We list below only the oldest release notes. These were replaced by the link above, because it was too hard to be kept up-to-date.

- version 1.4: (Jan/2000) Split VertexAnalysis into several smaller blocks, corresponding to each ntuple block. New block mapping generated to reconstructed tracks
- version 1.3: (Oct/1999) Append blocks to the global ntuple, managed by the NtupleMgr package
- version 1.2: (Jul/1999) Small modification in VertexAnalysis
- version 1.1: (Mar/1999) New VertexAnalysis with only one large ntuple
- version 1.0: (Jan/1999) First version

Class Graph