Pattern

A library for Pattern based Analysis

T. Glebe

Contents

1 Tools

General tools for the pattern code

Names

extern const PkgIni **PatternSteer** *Global object which holds XCVS information.*

enum **CC_Flag** *enum type for handling of charge conjugated pattern*

template <class Pointer> static void **delete_it** (Pointer obj)
delete a single pointer

template <class Container> void **dynamic_delete** (Container& rhs)
call delete for each element of container

template <class Source, class Target, class C, class T> bool **gselect** (const Source& src, Target& trg, const GPattern<C, T>& pat)
A simple global track selection routine.

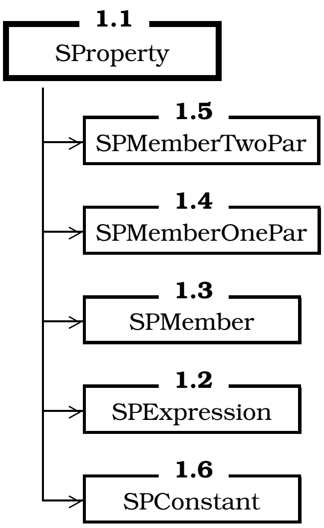
template <class Source, class Target> bool **select** (const Source& src, Target& trg, const Pattern& pat)
gselect adoption for the PatternMatch function

1.1

template <class C, class T> class **SProperty**

class SProperty

Inheritance



Public Members

— Constructors —

SProperty ()

virtual ~**SProperty** ()

— Operators —

virtual T **evaluate** (const C&) const
evaluate pattern

SPExpression <C, T, bool>
lt (const SProperty<C, T>& rhs) const
less than - operator< has problems in CINT

SPExpression <C, T, bool>
lt (const T& rhs) const
less than - operator< has problems in CINT

SPExpression <C, T, bool>
operator< (const SProperty<C, T>& rhs) const

SPExpression <C, T, bool>
operator< (const T& rhs) const

SPExpression <C, T, bool>
operator> (const SProperty<C, T>& rhs) const

SPExpression <C, T, bool>
operator> (const T& rhs) const

SPExpression <C, T, bool>
operator== (const SProperty<C, T>& rhs) const

SPExpression <C, T, bool>
operator== (const T& rhs) const

SPExpression <C, T, bool>
operator!= (const SProperty<C, T>& rhs) const

SPExpression <C, T, bool>
operator!= (const T& rhs) const

SPExpression <C, T, bool>
operator& (const SProperty<C, T>& rhs) const

SPExpression <C, T, bool>

operator— (const SProperty<C, T>& rhs) const

SPExpression <C, T, T>
operator+ (const SProperty<C, T>& rhs) const

SPExpression <C, T, T>
operator+ (const T& rhs) const

SPExpression <C, T, T>
operator- (const SProperty<C, T>& rhs) const

SPExpression <C, T, T>
operator- (const T& rhs) const

SPExpression <C, T, T>
operator* (const SProperty<C, T>& rhs) const

SPExpression <C, T, T>
operator* (const T& rhs) const

SPExpression <C, T, T>
operator/ (const SProperty<C, T>& rhs) const

SPExpression <C, T, T>
operator/ (const T& rhs) const

— Expert functions —

virtual SProperty <C, T> *
propertyClone () const

virtual std::ostream&
print (std::ostream& os) const
used by operator<<()

virtual string
getString () const

Protected Members

SProperty (const SProperty<C, T>& rhs)

const SProperty <C, T> &
operator= (const SProperty<C, T>& rhs)

Expression template base class.

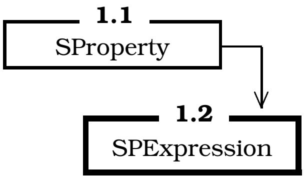
Author: T. Glebe

1.2

template <class C, class T, class X> class **SPExpression** : public SProperty<C, X>

class SPExpression

Inheritance



Public Members

— Constructors —

SPExpression ()

SPExpression (const SProperty<C, T>& lhs, const SProperty<C, T>& rhs, X (*func)(const T&, const T&), const char* nam)

SPExpression (const SPExpression<C, T, X>& rhs)

˜**SPExpression** ()

const SPExpression <C, T, X> & **operator=** (const SPExpression<C, T, X>&)

— Expert functions —

X **evaluate** (const C& obj) const
evaluate pattern

virtual SProperty <C, X> *

propertyClone () const

virtual std::ostream&
print (std::ostream& os) const
used by operator<<()

virtual string
getString () const

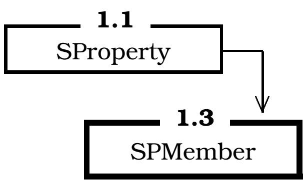
Author: T. Glebe

1.3

template <class C, class T> class **SPMember** : public SProperty<C, T>

class SPMember

Inheritance



Public Members

— Constructors —

SPMember ()

SPMember (T (C::*f)() const, const char* nam)

~**SPMember** ()

— Expert functions —

T **evaluate** (const C& part) const
evaluate pattern

virtual SProperty <C, T> *

propertyClone () const

virtual std::ostream&
print (std::ostream&) const
used by operator<<()

virtual string
getString () const

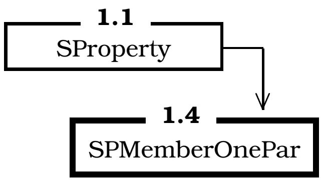
Author: T. Glebe

1.4

template <class C, class T, class P> class **SPMemberOnePar** : public SProperty<C, T>

class SPMemberOnePar

Inheritance



Public Members

— Constructors —

SPMemberOnePar ()

SPMemberOnePar (T (C::*f)(P) const, P par, const char* nam)

~**SPMemberOnePar** ()

— Expert functions —

virtual T **evaluate** (const C& part) const
evaluate pattern

const SPMemberOnePar <C, T, P> &

operator() (const P& val) const
set a new parameter

virtual SProperty <C, T> *
propertyClone () const

virtual std::ostream&
print (std::ostream&) const
used by operator<<()

virtual string
getString () const

Unary expression template.

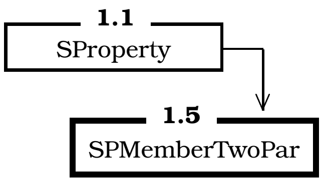
Author: T. Glebe

1.5

template <class C, class T, class P1, class P2> class **SPMemberTwoPar** : public SProperty<C, T>

SPMemberTwoPar

Inheritance



Public Members

— Constructors —

SPMemberTwoPar ()

SPMemberTwoPar (T (C::*f)(P1, P2) const,
P1 par1, P2 par2,
const char* nam)

~**SPMemberTwoPar** ()

— Expert functions —

T **evaluate** (const C& part) const
evaluate pattern

const SPMemberTwoPar <C, T, P1, P2> &
operator() (const P1& val1,
const P2& val2) const
set new parameters

virtual SProperty <C, T> *
propertyClone () const

virtual std::ostream&
print (std::ostream&) const
used by operator<<()

virtual string
getString () const

Binary expression template.

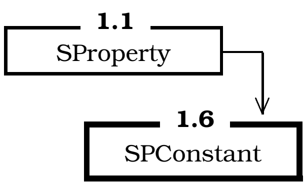
Author: T. Glebe

1.6

template <class C, class T> class **SPConstant** : public SProperty<C, T>

SPConstant

Inheritance



Public Members

— Constructors —

SPConstant ()

SPConstant (T constant)

~**SPConstant** ()

— Expert functions —

T **evaluate** (const C& part) const
evaluate pattern

virtual SProperty <C, T> *
propertyClone () const

virtual std::ostream&
print (std::ostream&) const
used by operator<<()

virtual string
getString () const

Constant expression template.

Author: T. Glebe

2 SPattern

High level tools for data analysis These pattern are designed to work with the SVertex vertexing package.

Names

2.1

template <unsigned int NTR> struct **DP_trait**

type configuration

Members

typedef list <SDecayPatternMatch<NTR> >
SDPM_l

typedef SDPM_l::iterator
SDPM_it

typedef SDPM_l::const_iterator
SDPM_cit

typedef list <SDecayPatternMatch<NTR>*>
SDPMP_l

typedef SDPMP_l::iterator
SDPMP_it

typedef SDPMP_l::const_iterator

SDPMP_cit

typedef SmartPointer <SDecayPattern<NTR> >
SDP_sp

typedef SmartPointer <SDecayPatternMatchContainer<NTR> >
DPMC_sp

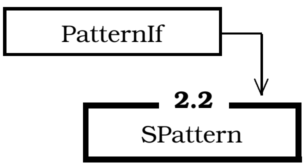
Configuration of types used in the DecayPatternMatch classes

Author: T. Glebe

2.2 class **SPattern** : public PatternIf

Generic class

Inheritance



Public Members

— Constructors —

SPattern ()

SPattern (const SPattern& rhs)

SPattern (const ToppIf& tp,
const SPattern& rhs)
copy rhs and replace ToppIf of rhs by tp

SPattern (const ToppIf& gt)

SPattern (const ToppIf& gt,
const SProperty<TrackIf,
bool>& rhs)

virtual ~**SPattern** ()

SPattern&
operator= (const SPattern& rhs)

— PatternIf access functions —

bool **evalTrack** (const TrackIf& rhs) const
check if single Track passes property criterias

bool **evalTrackProp** (const TrackIf& rhs) const
check track property without charge

const SProperty <TrackIf, bool> &
property () const
get property object

const ToppIf&
topp () const *get topp object*

const double*
rm () const *return pointer to topp mass*

unsigned int
len () const *always return 1*

string **getDecayString** (unsigned int lvl=0) const
get decay string

void **printProperty** (std::ostream& os, unsigned int lvl=0) const
print property

const SPMP_l*
getContainer () const
get pointer to list of pattern matches

const PatternMatchIf*
fillMC (const MC_Track& mt) const
fill container from MC event

bool **isReconstructed** (const MC_Track& rhs) const
same as operator==(MC_Track), but also check whether MC_Track was reconstructed

bool **isNotReconstructed** (const MC_Track& rhs, bool& no_break) const *same as operator==(MC_Track), but also check if MC_Track was not reconstructed*

PatternMatchIf*
fillRECOfromMC (const MC_Track& mt) const
fill container with reconstructed tracks from MC event

PatternMatchIf*
fillNORECOfromMC (const MC_Track& mt) const
fill container with reconstructed tracks or (if not reconstr.) MC tracks from MC event

— Operators —

SPattern **operator—** (const SPattern& rhs) const
OR of two patterns

SDecayPattern <2>
operator& (const SPattern& rhs) const
AND of patterns defines a decay

bool **operator==** (const SPattern& rhs) const

bool **operator==** (const ToppIf& rhs) const

bool **operator!=** (const ToppIf& rhs) const

virtual bool
operator== (const MC_Track& rhs) const

— User functions —

template <class InputContainer, class OutputContainer> bool
select (const InputContainer& src, OutputContainer& trg) const
select all tracks from InputContainer which passes property criteria and put them as PatternMatches in OutputContainer

template <class InputContainer> bool

	exist (const InputContainer& src) const	*return true if a track which matches the pattern exists in InputContainer*
void	**set_cprop** (const SProperty<TrackIf, bool>& rhs)	*set cprop pointer*
int	**chrg** () const	*Topp charge divided by 3*
bool	**hasProperty** () const	*does property exist?*
string	**getName** () const	*get pattern Name*
string	**PropertyString** () const	*get property string*
virtual string	**getString** () const	*get the pattern string*
bool	**check_property** (const SProperty<TrackIf, bool>& rhs) const	*true if rhs is equal to pattern property*
void	**set_cut** (const CutIf& c)	*set a cut object*
const CutIf*	**get_cut** () const	*get the cut object*
	— Expert functions —	
void	**setProperty** (const SProperty<TrackIf, bool>& rhs)	*set property object*
void	**set_container** (SPatternMatchContainer& rhs)	*set pointer to container object*
virtual std::ostream&	**print** (std::ostream& os) const	*used by operator<<*
void	**merge_property** (const SPattern& rhs)	*merge two pattern, only merge property*

A generic template class for defining track selection pattern, based on

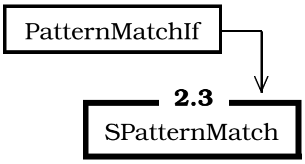
SProperty expressions.

Author: T. Glebe

2.3 class **SPatternMatch** : public PatternMatchIf

Generic class

Inheritance



Public Members

— Constructors —

SPatternMatch ()

SPatternMatch (RecoSegment& rs,
const SPattern& pat)

SPatternMatch (RecoSegment& rs, const
SmartPointer<SPattern>& pat)

SPatternMatch (MC_Track& mt, const
SmartPointer<SPattern>& pat)

SPatternMatch (TrackIf& tr)

SPatternMatch (TrackIf& tr,
const SPattern& pat)

SPatternMatch (TrackIf& tr, const
SmartPointer<SPattern>& pat)

SPatternMatch (const SPatternMatch& rhs)

virtual ~**SPatternMatch** ()

— TrackIf access methods —

float	**x** () const	*Track x position*
float	**y** () const	*Track y position*
float	**x** (float z) const	*Track x position at z*
float	**y** (float z) const	*Track y position at z*
float	**z** () const	*Track z position*
float	**tx** () const	*Track slope t_x*
float	**ty** () const	*Track slope t_y*
float	**p** () const	*Track momentum p*
float	**pt** () const	*transverse Track momentum p_t*
float	**pz** () const	*Track momentum along z*
float	**chi2** () const	*Track χ^2*
unsigned short int	**ndf** () const	*degrees of freedom of track fit*
float	**phi** () const	*azimuthal angle ϕ [deg]*
float	**theta** () const	*polar angle $\theta = \cos^{-1}(e_z)$ [deg]*
float	**eta** () const	*rapidity $\eta = -\log\tan(\theta/2.)$*
int	**charge** () const	*track charge*
float	**energy** (double rm = 0.) const	*track energy $E = \sqrt{m^2 + p^2}$*
float	**xf** (double rm = 0.) const	*Feynman-x: $x_F = \frac{E+p_z}{(E+p_z)_{max}}$*
float	**rap** (double rm = 0.) const	*Rapidity $y = \frac{1}{2}\ln(\frac{E+p_z}{E-p_z})$*
bool	**isValid** () const	*is PatternMatch valid?*
void	**valid** ()	*mark PatternMatch as valid*
void	**invalid** ()	*mark PatternMatch as invalid*
SVector <double, 3>	**xvec** () const	$\vec{x} = (x,\ y,\ z)$
SVector <double, 3>	**tvec** () const	$\vec{t} = (t_x,\ t_y,\ 1)$
SVector <double, 3>		

evec () const $\vec{e} = (e_x,\ e_y,\ ez)$ *unit vector along* $\vec{p}$

SVector <double, 3>
pvec () const $\vec{p} = (p_x,\ p_y,\ p_z)$

const SMatrix <double, 5> &
COV () const *cov. matrix*

const SMatrix <double, 5> &
CINV () const *inverse cov. matrix*

float **cov_x** (double dz=0.) const
get $\sigma_x^2|_{z+dz}$

float **cov_y** (double dz=0.) const
get $\sigma_y^2|_{z+dz}$

float **cov_tx** () const *get* σ_{t_x}

float **cov_ty** () const *get* σ_{t_y}

float **cov_p** () const *get* σ_p^2

void **collect** (vector<TrackIf*>& c) const
collect pointers

void **propagate** (const double z)
propagate track parameters & covariance matrix to z

— PatternMatchIf access methods —

virtual const TrackIf&
getTrackIf () const
read only track access on interface-level

virtual const RecoTrack&
getRecoTrack () const
get the RecoTrack object

virtual const MC_Track&
getMC_Track () const
get the MC_Track object

const RecoSegment&
getRecoSegment () const
read only track access to full track

const SPattern&

	getPattern () const	*read only pattern access*
double	**mass** () const	*assumed rest-mass of track (defined by pattern)*
double	**massError** () const	*dummy function: always returns 0*
double	**massProb** (double scale) const	*dummy function: same as track prob() (as massError()==0)*
bool	**isMCmatch** () const	*is this a match with a MC track?*
const PatternMatchIf*	**PatternMatch** (unsigned int i) const	*dummy function: always return 0*
bool	**calc_mother** ()	*dummy function: always return true*
bool	**calc_mother_tr** ()	*dummy function: always return true*
bool	**calc_mother_cov** ()	*dummy function: always return true*
	— Operators —	
bool	**operator==** (const TrackIf& rhs) const	*compare Track member with argument pointers*
bool	**operator==** (const SPatternMatch& rhs) const	*compare Track members pointers*
bool	**operator==** (const ToppIf& rhs) const	*compare topp pointer with pattern*
bool	**operator==** (const GTopp& rhs) const	*compare with GTopp*
	— Expert functions —	
float*	**set_user** (unsigned int size)	

		set a user field of given size, return pointer to it
float*	**user** () const	*get pointer to user field*
unsigned int	**user_size** () const	*get size of user field*
std::ostream&	**print** (std::ostream& os) const	*used by operator<<*
	— Relations —	
GRelationList <DecayPatternMatchIf>	**vtx**	*PatternMatch → DecayPatternMatch: relation of outgoing particles to decay objects*
const GRelationList <DecayPatternMatchIf> &	**Vtx** () const	
GRelationList <DecayPatternMatchIf> &	**Vtx** ()	
GRelationList <DecayPatternMatchIf> &	**container** (const DecayPatternMatchIf& p)	

Class which represents a selected track object together with the cuts which were used to select it. The cuts are represented by a ClProperty expression object.

Author: T. Glebe

2.4

class **SPatternMatchContainer**

Generic class

Public Members

— Constructors —

SPatternMatchContainer ()

SPatternMatchContainer (const SPattern& rhs)

SPatternMatchContainer (const SmartPointer<SPattern>& rhs)

~**SPatternMatchContainer** ()

— Access methods —

SPatternMatch& **push_back** (RecoSegment& rs)
add a RecoSegment

SPatternMatch& **push_back** (MC_Track& mtra)
add a MC track

SPatternMatch& **push_back** (TrackIf& t)
add a TrackIf

void **clear** () *clear list of pattern matches*

const SPM_l& **matches** () const *readonly access to pattern matches*

SPM_l& **matches** () *read/write access to pattern matches*

const SPMP_l& **pmatches** () const
readonly access to pattern match pointers

SPMP_l& **pmatches** () *read/write access to pattern match pointers*

const SP_sp& **pattern** () const *get pattern*

void **with** (const SProperty<TrackIf, bool>& rhs)
set a Track Property

— Expert methods —

bool **evalTrackProp** (const TrackIf& rhs) const
check track property without charge

template <class Input> bool

select (const Input& inp)
find pattern matches from Input container

std::ostream&
print (std::ostream& os) const

Class which represents a container for Pattern matches.

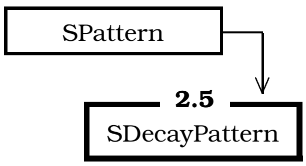
Author: T. Glebe

2.5

template <unsigned int NTR> class **SDecayPattern** : public SPattern

Generic class

Inheritance



Public Members

— Constructors —

SDecayPattern ()

SDecayPattern (const SDecayPattern<NTR>& rhs)

SDecayPattern (const SDecayPattern<NTR-1>& lhs, const SP_sp& rhs)

template <unsigned int NTR2>

SDecayPattern (const Smart-Pointer<SDecayPattern<NTR2> >& lhs, const SP_sp& rhs)

SDecayPattern (const ToppIf& tp, const SDecayPattern<NTR>& rhs)

SDecayPattern (const ToppIf& tp, const SP_sp& lhs, const SP_sp& rhs)

virtual ˜**SDecayPattern** ()

SDecayPattern <NTR> &
operator= (const SDecayPattern<NTR>& rhs)

— Operators —

SDecayPattern <NTR+1>
operator& (const SP_sp& rhs) const

bool **operator==** (const MC_Track& rhs) const

— overloaded PatternIf access functions —

const double*
rm () const *return pointer to vector with rest-masses*

unsigned int
len () const *return length of vector (=NTR)*

string **getDecayString** (unsigned int lvl=0) const
string with decay

void **printProperty** (std::ostream& os, unsigned int lvl=0) const
print propertyString

const SPMP_l*
getContainer () const
get pointer to list of pattern matches

const PatternMatchIf*
fillMC (const MC_Track& rhs) const
fill container from MC structure

bool **isReconstructed** (const MC_Track& rhs) const *same as operator==(MC_Track), but also check whether MC_Track was reconstructed*

bool **isNotReconstructed** (const MC_Track& rhs, bool& no_break) const
same as operator==(MC_Track), but also check if MC_Track was not reconstructed

PatternMatchIf*
fillRECOfromMC (const MC_Track& mt) const
fill container with reconstructed tracks from MC event

PatternMatchIf*
fillNORECOfromMC (const MC_Track& mt) const
fill container with reconstructed tracks or (if not reconstr.) MC tracks from MC event

— Access functions —

static const unsigned int
size

const SP_sp&
pattern (unsigned int i) const
read-only pattern access

SP_sp& **pattern** (unsigned int i)
read/write pattern access

void **massrange** (double mr)
set mass range cut around expected invariant mass

double **massrange** () const
return mass range cut

string **getName** () const
get particle name

string **getString** () const
get the pattern string

string **getPatternString** () const

bool **evalVtxProperty** (const VertexIf& rhs) const
check vertex property

const SVector <double, NTR> &
rmvec () const *return rest-mass vector*

— Expert functions —

void **setVtxProperty** (const SProperty<VertexIf, bool>& rhs)
set Vertex property object

SmartPointer <SProperty<VertexIf, bool> > &
getVtxProperty () const
get Vertex property object

void **set_container** (SDecayPatternMatchContainer<NTR>* rhs)
set pointer to container object

const SP_sp
get (const ToppIf& topp) const
get a pattern by TOPP id

void **fill_SPMP_l** (const SPMP_l** array)
*fill an array with a sorted list of list<PatternMatchIf>**

virtual std::ostream&
print (std::ostream& os) const
used by operator<<()

Protected Members

void **fillmtramap** (const MC_Track& rhs) const
internal utility function

A class which handles selection pattern for decaying particles. This class handles the kind of information which is independent from the Vertex object, like the Topp table entry of the decay and the list of track patterns which are used to select the tracks which will then be combined and vertexed.

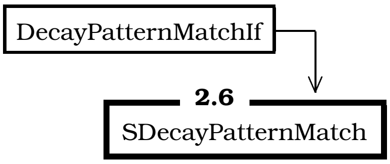
Author: T. Glebe

2.6

template <unsigned int NTR> class **SDecayPatternMatch** : public DecayPatternMatchIf

Generic class

Inheritance



Public Members

— Constructors —

SDecayPatternMatch ()

SDecayPatternMatch (const MC_Track& t, const SmartPointer<SPattern>& vpat)
create from MC

SDecayPatternMatch (const TrackIf& t, const SmartPointer<SPattern>& vpat)
create from reconstructed track

SDecayPatternMatch (SmartPointer<SVertex<NTR> >& v, const SmartPointer<SPattern>& vpat)
create from Vertex

SDecayPatternMatch (SVertex<NTR>* v, const SmartPointer<SPattern>& vpat)

		take over ownership of vertex object
virtual	~**SDecayPatternMatch** ()	
	— TrackIf access methods —	
float	**x** () const	*Track x position*
float	**y** () const	*Track y position*
float	**x** (float z) const	*Track x position at z*
float	**y** (float z) const	*Track y position at z*
float	**z** () const	*Track z position*
float	**tx** () const	*Track slope t_x*
float	**ty** () const	*Track slope t_y*
float	**p** () const	*Track momentum p*
float	**pt** () const	*transverse Track momentum p_t*
float	**pz** () const	*Track momentum along z*
float	**phi** () const	*azimuthal angle ϕ [deg]*
float	**theta** () const	*polar angle $\theta = \cos^{-1}(e_z)$ [deg]*
float	**eta** () const	*rapidity $\eta = -\log\tan(\theta/2.)$*
int	**charge** () const	*track charge*
float	**energy** (double rm = 0.) const	*track energy $E = \sqrt{m^2 + p^2}$*
float	**xf** (double rm = 0.) const	*Feynman-x: $x_F = \frac{E+p_z}{(E+p_z)_{max}}$*
float	**rap** (double rm = 0.) const	*Rapidity $y = \frac{1}{2}\ln(\frac{E+p_z}{E-p_z})$*
virtual bool	**isValid** () const	*is PatternMatch valid?*
virtual void	**valid** ()	*mark PatternMatch as valid*
virtual void	**invalid** ()	*mark PatternMatch as invalid*
SVector <double, 3>	**xvec** () const	$\vec{x} = (x,\ y,\ z)$
SVector <double, 3>		

tvec () const $\vec{t} = (t_x,\ t_y,\ 1)$

SVector <double, 3>
evec () const $\vec{e} = (e_x,\ e_y,\ ez)$ *unit vector along* $\vec{p}$

SVector <double, 3>
pvec () const $\vec{p} = (p_x,\ p_y,\ p_z)$

const SMatrix <double, 5> &
COV () const *cov. matrix*

const SMatrix <double, 5> &
CINV () const *inverse cov. matrix*

float **cov_x** (double dz=0.) const
get $\sigma_x^2|_{z+dz}$

float **cov_y** (double dz=0.) const
get $\sigma_y^2|_{z+dz}$

float **cov_tx** () const *get* σ_{t_x}

float **cov_ty** () const *get* σ_{t_y}

float **cov_p** () const *get* σ_p^2

void **collect** (vector<TrackIf*>& c) const
collect pointers

void **propagate** (const double z)
propagate track parameters & covariance matrix to z

— PatternMatchIf access methods —

const TrackIf&
getTrackIf () const
get TrackIf part of reconstructed particle

const RecoTrack&
getRecoTrack () const
get the mother RecoTrack object (if available)

const MC_Track&
getMC_Track () const
get the mother MC_Track object (if available)

const SPattern&

	getPattern () const	*read only pattern access*
double	**mass** () const	*invariant mass of vertex*
double	**massError** () const	*error on invariant mass*
double	**massProb** (double scale) const	*calculate a probabilty including the mass*
bool	**isMCmatch** () const	*does this match contain MC tracks?*
bool	**calc_mother** ()	*calculate mother track + cov. matrix*
bool	**calc_mother_tr** ()	*calculate mother track*
bool	**calc_mother_cov** ()	*calculate mother covariance matrix*
	— VertexIf Access methods —	
float	**vx** () const	*vertex x position*
float	**vy** () const	*vertex y position*
float	**vz** () const	*vertex z position*
float	**chi2** () const	*vertex χ^2*
unsigned short int	**ndf** () const	*degrees of freedom of vertex fit*
float	**prob** () const	*upper tail χ^2 probability*
unsigned short int	**ntracks** () const	*no of tracks in vertex*
const TrackIf*	**track** (unsigned int i) const	*return a track pointer*
const TrackIf*&	**track** (unsigned int i)	*read/write track access*
SVector <double, 3>	**vpos** () const	$\vec{v} = (x,\ y,\ z)$
const SMatrix <double, 3> &		

	VCOV () const	*vertex cov. matrix*
float	**vtx_cov_x** () const	σ_x^2 *of vertex*
float	**vtx_cov_y** () const	σ_y^2 *of vertex*
float	**vtx_cov_z** () const	σ_z^2 *of vertex*
const SVector <double, 3> &	**vposR** () const	$\vec{x} = (x,\ y,\ z)$ *(fast access)*
	— SVertex specific members —	
const SMatrix <double, 3> &	**VCINV** () const	*inverse vertex cov. matrix*
const SVector <double, 6> &	**mother** () const	*mother track parameters*
	— Operators —	
bool	**operator==** (const TrackIf& rhs) const	*compare Track member with argument pointers*
bool	**operator==** (const ToppIf& rhs) const	*compare Topp pointer with decay pattern*
bool	**operator==** (const GTopp& rhs) const	*compare with GTopp*
	— DecayPatternMatchIf access functions —	
const PatternMatchIf*	**PatternMatch** (unsigned int i) const	*return a PatternMatchIf pointer*
	— Expert functions —	
SmartPointer <SVertex<NTR> > &	**vertex** ()	*r/w access to SVertex object*
void	**set_mother** (const TrackIf& rhs)	*set mother track manually*
void	**set_MC** (bool ismc)	*set MC flag manually*
float*	**set_user** (unsigned int size)	

		set a user field of given size, return pointer to it
float*	**user** () const	*get pointer to user field*
unsigned int	**user_size** () const	*get size of user field*
std::ostream&	**print** (std::ostream& os) const	*used by operator<<*
	— Relations —	
GRelationList <SPatternMatch>	**out**	*DecayPatternMatch → PatternMatch (relation to outgoing tracks)*
const GRelationList <SPatternMatch> &	**Out** () const	
GRelationList <SPatternMatch> &	**Out** ()	
GRelationList <SPatternMatch> &	**container** (const SPatternMatch& p)	

Pattern Match class for Decays. SDecayPatternMatch represents a SVertex vertex object together with the DecayPattern which was used to fit and select the vertex object.

Author: T. Glebe

2.7 template <unsigned int NTR> class **SDecayPatternMatchContainer**

Generic class

Public Members

— Constructors —

SDecayPatternMatchContainer ()

SDecayPatternMatchContainer (const ToppIf& topp, const SPatternExpression<NTR>& rhs)

virtual ˜**SDecayPatternMatchContainer** ()

— Access methods —

const SP_sp&
pattern () const *get pattern pointer*

const SmartPointer <SDecayPattern<NTR> > &
dpattern () const
get decay pattern pointer

void **with** (const SProperty<TrackIf, bool>& rhs)
set a Track Property

void **with** (const SProperty<VertexIf, bool>& rhs)
set a Vertex Property

const typename DP_trait <NTR> ::SDPM_l&
matches () const *read only access to matches*

typename DP_trait <NTR> ::SDPM_l&
matches () *read/write access to matches*

const SPMP_l&
pmatches () const
read only access to matches

SPMP_l& **pmatches** () *read/write access to matches*

void **massrange** (double mr)
set mass range cut around expected invariant mass

double **massrange** () const
return mass range cut

const SmartPointer <SPatternMatchContainer> *
pcList () const

const SPMP_l* const*
pmList () const

SDecayPatternMatch <NTR> &

push_back (const MC_Track& mtra)
add a MC decay pattern match

SDecayPatternMatch <NTR> &
push_back (const TrackIf& track)
add a reconstructed decay pattern match

SDecayPatternMatch <NTR> &
push_back (SVertex<NTR>* vtx)
add a SVertex decay pattern match

SDecayPatternMatch <NTR> &
push_back (const SDecayPatternMatch<NTR>& dpm)
add a decay pattern match

— Expert methods —

bool **select** () *get tracks via SPatternMatchContainer::select(ArteTable<C>()) and do combinatorics + vertexing*

bool **selectMCtruth** () *Select decay pattern from MC truth (MC_Track table) and fill tracks into container*

bool **selectRECOfromMCtruth** ()
Select decay pattern from MC truth (MC_Track table) which have been reconstructed and fill (reconstructed) tracks into container

2.7.3 bool **selectNORECOfromMCtruth** ()

		Select decay pattern from MC truth (MC_Track table) which could not be fully reconstructed and fill tracks into container . 39
void	**set_cut** (const CutIf& c)	*set a cut object*
bool	**prob_collision** ()	*if DecayPatternMatches share same track, flag DPM with smaller prob as invalid*
bool	**massProb_collision** (double scale = 0.1)	*if DecayPatternMatches share same track, flag DPM with smaller massProb as invalid*
bool	**massProb_bg_collision** ()	*massProb_collision function for background studies*
bool	**getComplement** (const unsigned int nr, PMC_sp& container)	*fill container with tracks NOT used in decaypatternmatches*
bool	**calc_mother** ()	*calculate mother tracks with covariance matrix, ignore invalid pattern matches*
bool	**calc_mother_tr** ()	*calculate mother tracks, ignore invalid pattern matches*
bool	**calc_mother_cov** ()	*calculate cov. matrix of mother tracks, ignore invalid pattern matches*
	— Expert methods —	
std::ostream&	**print** (std::ostream& os) const	*used in operator<<()*

Protected Members

bool	**check_combination** (const GSC_it& it) const	

For internal use only. Checks whether a track appears more than once in a combination.

virtual bool
do_combine (bool check)
routine which does the work

virtual bool
do_combine_no_vertex (bool check)
routine which does the work

Class which represents a container for DecayPattern matches.

Author: T. Glebe

2.7.1

bool **combine** (bool check = false)

Do combinatorics and fill decay pattern match container

Do combinatorics and fill decay pattern match container. Compute vertex via Kalman filter and check vertex properties. Accept only combinations whose inv. mass lies within mass window (to be set via `massrange()`. If argument is `true`, then it is explicitly checked whether in a combination a track apprears more than once. If yes, the combination is skipped.

2.7.2

bool **combine_no_vertex** (bool check = false)

Do combinatorics and fill decay pattern match container

Do combinatorics and fill decay pattern match container. Do not compute a vertex. Just accept combinations whose inv. mass lies within mass window (to be set via `massrange()`. If argument is `true`, then it is explicitly checked whether in a combination a track apprears more than once. If yes, the combination is skipped.

2.7.3

bool **selectNORECOfromMCtruth** ()

Select decay pattern from MC truth (MC_Track table) which could not be fully reconstructed and fill tracks into container

Select decay pattern from MC truth (MC_Track table) which could not be fully reconstructed and fill tracks into container. In case of reconstructed tracks, fill RecoTrack, in case of not reconstructed tracks, fill MC_Track.

2.8

template <unsigned int NTR> class **SPatternExpression**

Generic class

Public Members

— Constructors —

SPatternExpression ()

SPatternExpression (const SDecayPattern<NTR>& dp, const PMC_sp& lhs, const PMC_sp& rhs)

template <unsigned int NTR2> **SPatternExpression** (const SDecayPattern<NTR>& dp, const PMC_sp& lhs, const SmartPointer<SDecayPatternMatchContainer<NTR2> >& rhs)

SPatternExpression (const SDP_sp& dp, const PMC_sp& lhs, const PMC_sp& rhs)

SPatternExpression (const ToppIf& topp, const SPatternExpression<NTR>& rhs)

SPatternExpression (const SDecayPattern<NTR>& dp, const SPatternExpression<NTR-1>& lhs, const PMC_sp& rhs)

template <unsigned int NTR2>
SPatternExpression (const SDecayPattern<NTR>& dp, const SPatternExpression<NTR-1>& lhs, const SmartPointer<SDecayPatternMatchContainer<NTR2> >& rhs)

template <unsigned int NTR1, unsigned int NTR2>
SPatternExpression (const SDecayPattern<NTR>& dp, const SmartPointer<SDecayPatternMatchContainer<NTR1> >& lhs, const SmartPointer<SDecayPatternMatchContainer<NTR2> >& rhs)

— Access methods —

const SDecayPattern <NTR> &
dpattern () const
get decay pattern

PMC_sp **pcList** (unsigned int i) const

const SPMP_l*
pmList (unsigned int i) const

Class which represents a [Decay]PatternMatchContainer expression.

Author: T. Glebe

Class Graph