Vt++

A C++ library for vertex fitting and matrix calculation in HERA-B.
For further documentation see file `vt++.ps`.

T. Glebe

Contents

1 VERTEX

The vertexing classes

Names

double **distance** (const ArtePointer<RVER>& rver, const ArtePointer<MVER>& mver)
spatial distance betw. RVER & MVER

double **distance** (const ArtePointer<RVER>& rver, const ArtePointer<GTAR>& gtar)
spatial distance between RVER and GTAR

double **distance** (const Vertex& rver, const ArtePointer<GTAR>& gtar)
spatial distance between Vertex and GTAR

double **distance** (const Track& t, const ArtePointer<GTAR>& gtar)
spatial distance of track to wire

double **distanceChi2** (Track& t1, Track& t2)
χ^2 *distance track - track,* $ndf = 1$

double **distance** (const Track& t, const Vertex& v)
spatial distance track - vertex

double **SGNdistance** (const Track& t, const Vertex& v)
signed spatial distance track - vertex

double **SGNdistance** (const Track& t, double xv, double yv, double zv)
signed spatial distance track - space point

double **distanceChi2** (const Vertex& v1, const Vertex& v2)
χ^2 *distance between two vertices,* $ndf = 3$

double **distanceChi2** (const Vertex& v, double x, double y, double z)
$\chi 2$ *distance vertex - space point,* $ndf = 3$

double **SGNdistance** (const Track& t, const ArtePointer<RecoVertex>& ap) *signed spatial distance track - RecoVertex*

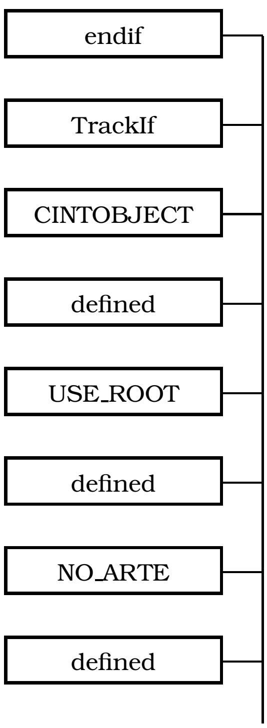
double **SGNdistance** (const Track& t, const VertexIf& ap)
signed spatial distance track - VertexIf

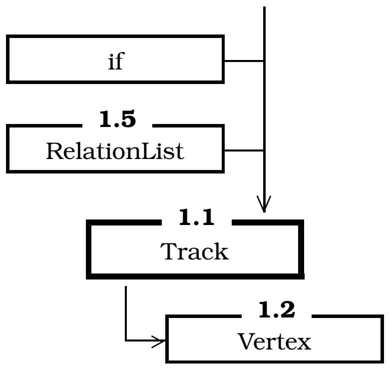
1.1

class **Track** : public RelationList if defined NO_ARTE defined USE_ROOT defined CINTOBJECT, public TrackIf endif

Track representation class

Inheritance





Public Members

	— Constructors —	
	Track ()	
	Track (const Track* const track)	
	Track (const Track& rhs)	
virtual	˜**Track** ()	
	Track (const RecoSegment& r, double ms = 0., double msCC = 0.)	
	Track (const RecoTrack& r, double ms = 0., double msCC = 0.)	
	Track (const ArtePointer<RecoTrack>& r, double ms = 0., double msCC = 0.)	
	Track (const TrackIf& rhs, double ms = 0., double msCC = 0.)	
	Track (const ArtePointer<RTRA>& rtra)	
	Track (const ArtePointer<RSEG>& rseg)	
	Track (const ArtePointer<MTRA>& mtra)	
	— TrackIf access methods —	
float	**x** () const	*x at z=0 (in Vt: p(3, ..))*
float	**y** () const	*y at z=0 (in Vt: p(4, ..))*
float	**x** (float z) const	*Track x position at z*
float	**y** (float z) const	*Track y position at z*
float	**z** () const	*z = 0 in Vt*

float	**tx** () const	*slope (in Vt: p(1, ..))*
float	**ty** () const	*slope (in Vt: p(2, ..))*
float	**p** () const	*momentum (in Vt: p(5, ..))*
float	**pt** () const	*transv. momentum* $\sqrt{p_x^2 + p_y^2}$
float	**pz** () const	$p \cdot e_z$
float	**chi2** () const	*dummy function: always return 0*
unsigned short int	**ndf** () const	*dummy function: always return 0*
float	**phi** () const	*azimuthal angle* ϕ
float	**theta** () const	*polar angle* $\theta = \cos^{-1}(e_z)$
float	**eta** () const	*rapidity* $-\log\tan(\theta/2.)$
int	**charge** () const	*charge*
float	**energy** (double rm = 0.) const	*Energy with given rest-mass (in GeV)* $E = \sqrt{p^2 + m^2}$
float	**xf** (double rm = 0.) const	*Feynman Variable* $x_F = \frac{E+p_z}{(E+p_z)_{max}}$
float	**rap** (double rm = 0.) const	*Rapidity* $y = \frac{1}{2}\ln(\frac{E+p_z}{E-p_z})$
SVector <double, 3>	**xvec** () const	$\vec{v} = (x,\ y,\ z)$
SVector <double, 3>	**tvec** () const	$\vec{v} = (t_x,\ t_y,\ 1.)$ *refitted slope vector*
SVector <double, 3>	**evec** () const	$\vec{v} = (e_x,\ e_y,\ e_z)$ *unit vector along refitted track*
SVector <double, 3>	**pvec** () const	$\vec{v} = (p_x,\ p_y,\ p_z)$ *refitted mom. vector*
const SMatrix <double, 5> &	**COV** () const	*dummy function: return 0 matrix*
const SMatrix <double, 5> &		

	CINV () const	*dummy function: return 0 matrix*
inline MATRIX::VtVector	**evec** () const	$\vec{v} = (e_x,\ e_y,\ e_z)$ *unit vector along refitted track*
inline MATRIX::VtVector	**tvec** () const	$\vec{v} = (t_x,\ t_y,\ 1.)$ *refitted slope vector*
inline MATRIX::VtVector	**pvec** () const	$\vec{v} = (p_x,\ p_y,\ p_z)$ *refitted mom. vector*
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
bool	**isValid** () const	*dummy function: always return true*
void	**valid** ()	*dummy function: do nothing*
void	**invalid** ()	*dummy function: do nothing*
bool	**operator==** (const TrackIf& rhs) const	*comparison between Track objects*
void	**collect** (vector<TrackIf*>& c) const	*collect TrackIf pointers*
void	**propagate** (const double zz)	*propagate track to* z
	— Other access methods —	
double	**px** () const	$p \cdot e_x$
double	**py** () const	$p \cdot e_y$
double	**ex** () const	$t_x \cdot e_z$
double	**ey** () const	$t_y \cdot e_z$
double	**ez** () const	$1/\sqrt{1 + t_x^2 + t_y^2}$
bool	**IsRecoTrack** () const	

		is Track constructed from RecoTrack object?
bool	**IsRTRA** () const	*is Track constructed from RTRA object? (n.a. in ROOT)*
double	**xerr** (double dz = 0) const	$\sqrt{\sigma_x^2(z+dz)}$
double	**yerr** (double dz = 0) const	$\sqrt{\sigma_y^2(z+dz)}$
double	**txerr** () const	$\sqrt{\sigma_{tx}^2}$
double	**tyerr** () const	$\sqrt{\sigma_{ty}^2}$
double	**perr** () const	$\sqrt{\sigma_p^2}$
ArtePointer <RSEG>	**rseg** () const	*return RSEG pointer*
ArtePointer <RTRA>	**rtra** () const	*return RTRA pointer*
ArtePointer <MTRA>	**mtra** () const	*return MTRA pointer*
ArtePointer <RecoTrack>	**recotrack** () const	*return RecoTrack pointer*
const MATRIX::CMatrix&	**V** () const	*covariance matrix*
const MATRIX::CMatrix&	**G** () const	*inverse* 5×5 *covariance matrix*
const MATRIX::CMatrix&	**GM** () const	*inverse* 4×4 *covariance matrix, without momentum*

— Expert functions —

std::ostream&	**print** (std::ostream& os) const	*called by cout*
double	**rm** () const	*rest-mass*
void	**rm** (const double mass)	*set rest-mass (needed for mass constrained fits)*
double	**rmCC** () const	*conjugated rest-mass*

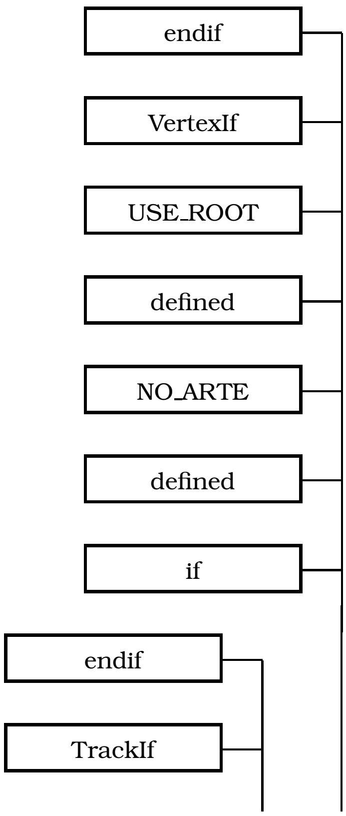
void	**rmCC** (const double mass) *set conjugated rest-mass*
	— Operators —
bool	**operator<** (const Track& rhs)
int	**operator==** (const Track& rhs)

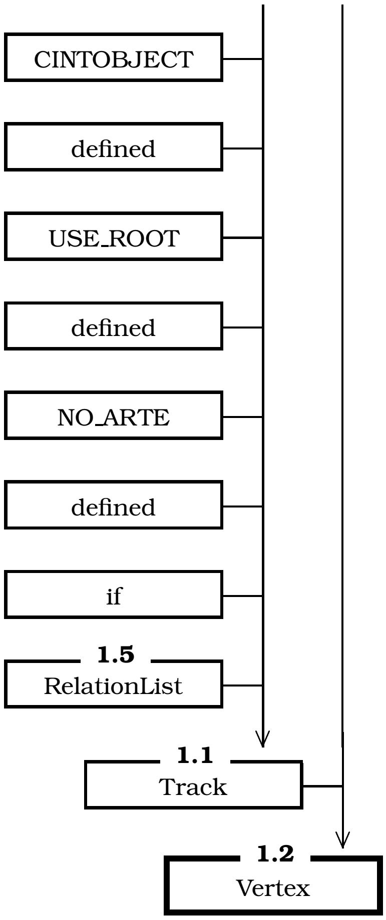
1.2

class **Vertex** : public Track if defined NO_ARTE defined USE_ROOT, public VertexIf endif

Vertex class

Inheritance





Public Members

— Constructors —

Vertex ()

Vertex (const Vertex& rhs)

Vertex (const Track_v& v)

Vertex (Track& t1, Track& t2)

~**Vertex** ()

Vertex (const ArtePointer<RVER>& v)

Vertex (const ArtePointer<RTRA>& r1, const ArtePointer<RTRA>& r2)

Vertex (const ArtePointer<RSEG>& r1,
const ArtePointer<RSEG>& r2)

Vertex (const ArtePointer<MTRA>& m1,
const ArtePointer<MTRA>& m2)

Vertex (const RecoSegment& r1,
const RecoSegment& r2)

Vertex (const RecoTrack& r1,
const RecoTrack& r2)

Vertex (const ArtePointer<RecoTrack>& r1,
const ArtePointer<RecoTrack>& r2)

— VertexIf access functions —

float	**vx** () const	x *of vertex*
float	**vy** () const	y *of vertex*
float	**vz** () const	z *of vertex*
float	**chi2** () const	χ^2 *of vertex fit*
float	**prob** () const	*upper tail* χ^2 *probability*
unsigned short int	**ndf** () const	*degrees of freedom of vertex fit*
unsigned short int	**ntracks** () const	*no of tracks in vertex* n
const TrackIf*	**track** (unsigned int i) const	*dummy function: always return 0*
const TrackIf*&	**track** (unsigned int i)	*dummy function*
SVector <double, 3>	**vpos** () const	$\vec{x} = (x,\ y,\ z)$
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*
	– Other access methods —	
double	**angle** () const	$\sqrt{n \cdot \sum_i t_{x,\,i}^2 - (\sum_i t_{x,\,i})^2 + n \cdot \sum_i t_{y,\,i}^2 - (\sum_i t_{y,\,i})^2}/n$
double	**dist** () const	*available after call to rmsDistAngle()*
double	**vxerr** () const	$\sqrt{\sigma_{vx}^2}$ *vertex* x*-error*
double	**vyerr** () const	$\sqrt{\sigma_{vy}^2}$ *vertex* y*-error*
double	**vzerr** () const	$\sqrt{\sigma_{vz}^2}$ *vertex* z*-error*
double	**pmaxfrac** () const	$\max\lvert p_i/p_j\rvert$ *max. momentum fraction*
const MATRIX::VtSymMatrix&	**CS** () const	3×3 *Vertex covariance matrix*
const MATRIX::VtSymMatrix&	**covn** () const	$(3 \cdot n + 3) \times (3 \cdot n + 3)$ *general covariance matrix*
bool	**valid** () const	*is vertex valid?*
void	**set_invalid** ()	*mark vertex as invalid*
	— Mass calculation routines —	$m = \sqrt{\sum_i p_i^2 + m_i^2 - \sum_i \vec{p}_i^2}$
double	**mass** (const bool use = false) const	*rest-mass 0 or track rest mass*
double	**massCC** (const bool use = false) const	*rest-mass 0 or conjugated track rest mass*
double	**mass** (double m1) const	*single rest-mass calc.,*
double	**mass** (double m1, double m2) const	

2-track mass with particle rest-masses

double **mass** (double m1, double m2, double m3) const
3-track mass with rest-masses

double **mass** (double m1, double m2, double m3, double m4) const
4-track mass with rest-masses

double **mass** (const Vector_d& m02, bool use = false, bool CC = false) const
specify n rest masses

— Vertexing routines —

const bool **findVertex2D** (void)
unweighted 2-D min. dist.

const bool **findVertex3D** (void)
unweighted 3-D min. dist.

const bool **findVertexVt** () *Vt Kalman-filter*

const bool **VtEstimateVertex** ()
weighted 2-D min. dist.

const bool **VtEstimateVertexMath** (double& x, double& y, double& z)
estimate Vertex without changing Vertex object

const bool **VtMass** () *Kalman filter with mass constraints*

— Adding tracks —

void **push_back** (Track& track)

void **push_back** (TrackIf& track)

void **push_back** (const ArtePointer<RTRA>& rtra)

void **push_back** (const ArtePointer<RSEG>& rseg) void **push_back** (const ArtePointer<MTRA>& mtra)

void **push_back** (const RecoSegment& r)

void **push_back** (const RecoTrack& r)

void **push_back** (const RecoTrack& r, double restmass)
add track with rest-mass assigned

void **push_back** (const ArtePointer<RecoTrack>& r)

— Misc functions —

const ArtePointer <RVER> **createRVER** (void) const
write to RVER table

const ArtePointer <RecoVertex> **createRecoVertex** (void) const
write to RecoVertex table

double **rmsDistAngle** () const
calc rms dist and rms angle

bool **calc_mother** (bool use_kalman = true)
calc track params of mother particle

bool **calc_mother_cov** ()
calc cov. matrix of mother particle

const bool **remove_last** ()
remove last track from vertex, do inverse Kalman filter

void **remove_worst** ()
remove track with biggest χ^2 contribution, do inverse Kalman filter

void **use_momentum** (const bool use)
set for all tracks whether momentum should be used or not

void **use_kalman** (const bool use)
use refitted track parameters or not

bool **use_kalman** () const

double **distance** (double x, double y, double z) const

χ^2 *distance to space point,* $ndf = 3$

double **distance** (Track& t) const
χ^2 distance to track, $ndf = 2$

double **distance** (const Vertex& rhs) const
χ^2 distance to vertex, $ndf = 3$

— Expert routines —

void **clear** () *clear all track-vertex relations, makes vertex invalid*

const bool
VtFilter ()

const bool
VtInverseFilter () const

void **VtSmoothX** () *smooth vertex position*

void **VtSmoothQ** () *smooth momenta*

const bool
VtRemoveTrack (Relation& track)
remove track using inverse Kalman filter

const bool
VtFit () *Kalman filter + smoother*

const MATRIX::VtVector&
xv () const

const MATRIX::VtVector&
xvs () const *vector of vertex-positions*

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

double **chi2n** () const *χ^2 after last filter step*

double **chi2l** () const *χ^2 contribution of last filter step*

virtual void
remove (Relation* r)
unbook relation

virtual const iterator
erase (const iterator& pos)
erase a relation without refit

const MATRIX::VtSymMatrix&

bigcov () *compute general* $(3 \cdot n + 3) \times (3 \cdot n + 3)$ *covariance matrix*

— Mass Constraint routines —

void **MassConstr** (double m)
add a mass constraint for all tracks, m *= mass of mother particle*

MassC& **addMassConstr** (double m=0.)
add a mass constraint

void **clearMassConstr** ()
delete the mass constraints

unsigned int **nMassConstr** () const
number of mass constraints

— Operators —

const Vertex& **operator=** (const Vertex& rhs)

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

const double **operator-** (const Vertex& rhs) const

1.3 class **Kalman**

Kalman filter class

Public Members

— Constructors —

Kalman (const Relation* const relation)

~**Kalman** ()

— Access methods —

inline double

	chi2 () const	χ^2 *contribution of track*
inline double	**chi2s** () const	*smoothed* χ^2
inline double	**tx** () const	*refitted* t_x
inline double	**ty** () const	*refitted* t_y
inline double	**p** () const	*refitted* p
inline double	**px** () const	*refitted* p_x
inline double	**py** () const	*refitted* p_y
inline double	**pz** () const	*refitted* p_z
inline double	**ex** () const	*refitted* e_x
inline double	**ey** () const	*refitted* e_y
inline double	**ez** () const	*refitted* e_z
inline double	**E** (double rm = 0.) const	*refitted Energy* $E = \sqrt{p^2 + m^2}$
inline SVector <double, 3>	**tvec** () const	$\vec{v} = (t_x,\ t_y,\ 1.)$ *refitted slope vector*
inline SVector <double, 3>	**evec** () const	$\vec{v} = (e_x,\ e_y,\ e_z)$ *unit vector along refitted track*
inline SVector <double, 3>	**pvec** () const	$\vec{v} = (p_x,\ p_y,\ p_z)$ *refitted mom. vector*
inline MATRIX::VtVector	**evec** () const	$\vec{v} = (e_x,\ e_y,\ e_z)$ *unit vector along refitted track*
inline MATRIX::VtVector		

tvec () const — $\vec{v} = (t_x,\ t_y,\ 1.)$ *refitted slope vector*

inline MATRIX::VtVector
pvec () const — $\vec{v} = (p_x,\ p_y,\ p_z)$ *refitted mom. vector*

— Cov. matrices for internal use —

inline const MATRIX::VtSymMatrix&
W () const

inline const MATRIX::VtSymMatrix&
C () const

inline const MATRIX::VtSymMatrix&
CINV () const

inline const MATRIX::VtSqMatrix&
F () const

inline const MATRIX::VtSqMatrix&
ES () const

inline const MATRIX::VtSymMatrix&
DS () const

inline const MATRIX::VtVector&
xv () const

inline const MATRIX::VtVector&
xnk () const

inline const MATRIX::VtVector&
qvs () const — $\vec{v} = (t_x,\ t_y,\ p)$

— Expert functions —

void **use_momentum** (const bool use)
set if track momentum should be used in fit

inline bool
use_momentum () const

inline double
set_chi2 (const double chi2)

void **init** () — *initialize the kalman structure*

void **print** (std::ostream& os) const
called by operator<<()

bool **filter** (double z,
const MATRIX::VtSymMatrix& prCINV,
const MATRIX::VtVector& prkal_xv)

double **filter_chi2** (double z, double prChi2, const MATRIX::VtSymMatrix& prCINV, const MATRIX::VtVector& prkal_xv)

bool **inverse_filter** (double z, const MATRIX::VtSymMatrix& CINVn, const MATRIX::VtSymMatrix& prCINV, const MATRIX::VtVector& kal_xvn)

void **smooth** (double z, const MATRIX::VtVector& xvs, const MATRIX::VtSymMatrix& Cn)

const MATRIX::VtVector **calc_dp** (double z, const MATRIX::VtVector& xk, const MATRIX::VtVector& qk) const

const MATRIX::VtVector **calc_pcAx** (const MATRIX::VtVector& xk) const

const MATRIX::VtVector **calc_AGpc** (void) const

const MATRIX::VtVector **calc_qk** (const MATRIX::VtVector& xk) const

void **calc_qvs** (const MATRIX::VtVector& xvs)

double **calc_dchi2** (double z, const MATRIX::VtSymMatrix& prCINV, const MATRIX::VtVector& xk, const MATRIX::VtVector& prxk, const MATRIX::VtVector& qk) const

void **calc_pc** (double z)

— Mass constraint fit functions —

inline void **alpc_init** (void) *fill* $\vec{\alpha}_c^{(0)}$

inline void **alp_init** (void) *fill* $\vec{\alpha}^{(0)}$

void **calc_ealpc** (void) *calculate unit vector* $\vec{\alpha}_c/|\vec{\alpha}_c|$*, energy from alpc*

inline const MATRIX::VtVector&
alpc () const *state vector* $\vec{\alpha_c} = (t_x,\ t_y,\ p)$

inline const MATRIX::VtVector&
alp () const *state vector* $\vec{\alpha} = (t_x,\ t_y,\ p)$

inline double
xn () const $n_x = t_x * e_z$

inline double
yn () const $n_y = t_y * e_z$

inline double
zn () const $n_z = 1/\sqrt{1 + t_x^2 + t_y^2}$

inline double
erg () const $E_i = \sqrt{m^2 + p^2},\ m$ *= track rest-mass*

inline const MATRIX::VtVector&
nalpc () const $\vec{n} = (n_x,\ n_y,\ n_z)$

inline MATRIX::VtVector&
qvs_nc () *return non-const reference to qvs*

This class contains the Vt Kalman filter matrices and vectors. Most of the member functions are for internal use only.

1.4 class **Relation**

for internal use only

Public Members

— Constructors —

Relation (Track& ttrack, Vertex& vvertex)

~**Relation** ()

— Expert functions —

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

inline Track&

getTrack ()

inline Vertex&
getVertex ()

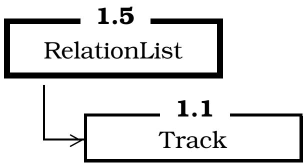
inline Kalman&
getKalman ()

Class for Track-Vertex-Kalman object relations.

1.5 class **RelationList**

Relation container class

Inheritance



Public Members

— Constructors —

RelationList ()

virtual ~**RelationList** ()

— Access functions —

virtual void
clear () *clear all relations*

inline void
push_back (Relation* rel)
add a relation

Relation* **back** () const *return last relation*

inline unsigned int

size () const *return no of relations*

virtual const iterator
erase (const iterator& pos)
Erase an element

— Iterator methods —

inline iterator
begin ()

inline iterator
end ()

inline const_iterator
begin () const

inline const_iterator
end () const

inline reverse_iterator
rbegin ()

inline reverse_iterator
rend ()

inline const_reverse_iterator
rbegin () const

inline const_reverse_iterator
rend () const

Relation* const
operator[] (unsigned int i) const
direct access

— Expert functions —

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

inline void
unbook (Relation* const rel)
get rid of relation pointer

virtual void
remove (Relation* const rel)
remove a relation

1.6

class **MassC**

Mass Constraint class

Public Members

— Constructors —

MassC ()

MassC (double m)
construct from mass of mother particle

~**MassC** ()

— Access functions —

inline void
push_back (Relation& r)
add Track to mass constraint

void **push_back** (Relation* r)
add Track to mass contraint

inline double
mass () const *get mass constraint*

inline void
mass (double m) *set mass constraint*

inline double
galp () const *constraint $\vec{g}(\vec{\alpha}^{(0)})$*

— Expert functions —

void **print** (std::ostream& os) const
called by operator<<()

void **calc_Sl** () *compute mass constraint $\vec{g}(\vec{\alpha}^{(0)})$*

const bool
check_kinematic () const
check kinematic boundary

const MATRIX::VtVector
gam (const Relation&) const
return Γ_{li}

2 MATRIX

A general purpose matrix computation package

Names

2.1 class **VtVector**

Vector class

Public Members

— Constructors —

VtVector (const unsigned int size)

VtVector (double v1, double v2)

VtVector (double v1, double v2, double v3)

VtVector (double v1, double v2, double v3, double v4)

VtVector (double v1, double v2, double v3, double v4, double v5)

VtVector (double v1, double v2, double v3, double v4, double v5, double v6)

VtVector (const VtVector& rhs)

~**VtVector** ()

— Operators —

inline double
operator[] (unsigned int row) const

inline double&
operator[] (const unsigned int row)

const VtVector&
operator= (const VtVector& rhs)
$\vec{v} = \vec{w}$

const VtVector&
operator+= (const double rhs)
$\vec{v} = (v_\mu + \alpha)$

const VtVector&
operator-= (const double rhs)
$\vec{v} = (v_\mu - \alpha)$

const VtVector&
operator*= (const double rhs)
$\vec{v} = (v_\mu \cdot \alpha)$

const VtVector&
operator/= (const double rhs)
$\vec{v} = (v_\mu / \alpha)$

const VtVector&
operator+= (const VtVector& rhs)
$\vec{v} = (v_\mu + w_\mu)$

const VtVector&
operator-= (const VtVector& rhs)
$\vec{v} = (v_\mu - w_\mu)$

const VtVector
operator+ (const VtVector& rhs) const
$\vec{v} + \vec{w}$

const VtVector

operator- (const VtVector& rhs) const
$\vec{v} - \vec{w}$

const double
operator* (const VtVector& rhs) const
$\vec{v} * \vec{w} = \sum_\mu v_\mu \cdot w_\mu$

VtVector **operator*** (const double rhs) const
$\vec{v} = (v_\mu \cdot \alpha)$

— Access methods —

inline unsigned int
size () const *vector dimension*

inline unsigned int
nrow () const *vector dimension*

double **mag2** () const $\sum_i v_i^2$

inline double
mag () const $\sqrt{\sum_i v_i^2}$ *vector length*

VtVector **unit** () const *return a unit vector*

— Expert functions —

void **print** (std::ostream& os) const
called by cout

void **place_at** (const VtVector& rhs,
const unsigned int row)
copy a smaller vector at a certain place

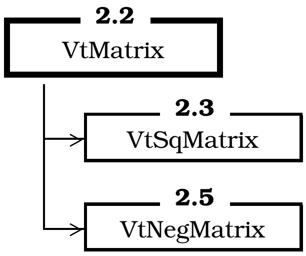
void **copy** (const VtVector& rhs,
const unsigned int offset)
to be used if vector dimensions differ

void **clear** (void) *set vector elements to 0*

2.2 class **VtMatrix**

Simple Matrix computation base class

Inheritance



Public Members

— Constructors —

VtMatrix (const unsigned int row,
const unsigned int col)

VtMatrix (const VtMatrix& rhs)

virtual ~**VtMatrix** ()

— Access functions —

inline unsigned int
nrow () const *no of rows* n

inline unsigned int
ncol () const *no of columns* m

inline int **size** () const $m \times n$

— Matrix operations —

void **VtT** (void) *transform into transpose matrix*

const VtMatrix
T (void) const *return transpose*

virtual void
place_at (const VtMatrix& rhs,
const unsigned int row,
const unsigned int col)
copy a smaller matrix at a certain place

virtual void

place_at (const VtVector& rhs,
const unsigned int row,
const unsigned int col)
copy a vector at a certain place

void **copy** (const VtMatrix& rhs)
to be used if matrix dimensions are not equal

void **clear** (void) *set matrix elements to 0*

const VtMatrix&
operator= (const VtMatrix& rhs)
$\boldsymbol{A} = \boldsymbol{B}$

const VtMatrix&
operator= (const VtNegMatrix& rhs)

virtual const VtMatrix&
operator+= (const double rhs)
$\boldsymbol{A} = (a_{\mu\nu} + \alpha)$

virtual const VtMatrix&
operator-= (const double rhs)
$\boldsymbol{A} = (a_{\mu\nu} - \alpha)$

virtual const VtMatrix&
operator*= (const double rhs)
$\boldsymbol{A} = (a_{\mu\nu} \cdot \alpha)$

virtual const VtMatrix&
operator/= (const double rhs)
$\boldsymbol{A} = (a_{\mu\nu} / \alpha)$

const VtMatrix&
operator+= (const VtMatrix& rhs)
$\boldsymbol{A} = (a_{\mu\nu} + b_{\mu\nu})$

const VtMatrix&
operator-= (const VtMatrix& rhs)
$\boldsymbol{A} = (a_{\mu\nu} - b_{\mu\nu})$

const VtMatrix
operator+ (const VtMatrix& rhs) const
$\boldsymbol{A} + \boldsymbol{B}$

const VtMatrix
operator+ (const VtNegMatrix& rhs) const

const VtMatrix

operator- (const VtMatrix& rhs) const
$\boldsymbol{A} - \boldsymbol{B}$

const VtMatrix
operator- (const VtNegMatrix& rhs) const

const VtNegMatrix
operator- (void) const
$-\boldsymbol{A}$

const VtMatrix
operator* (const VtMatrix& rhs) const
$\boldsymbol{A} \cdot \boldsymbol{B} = \sum_{\nu=1}^{n} a_{\mu\nu} b_{\nu\lambda}$

const VtVector
operator* (const VtVector& rhs) const
$\boldsymbol{A} \cdot \vec{v} = (\sum_{\nu=1}^{n} a_{\mu\nu} v_{\nu})$

— Expert functions —

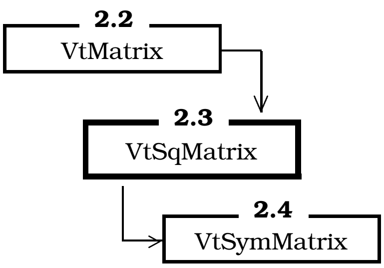
inline double*
array () const *return pointer to internal array*

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

2.3 class **VtSqMatrix** : public VtMatrix

Class for quadratic matrices

Inheritance



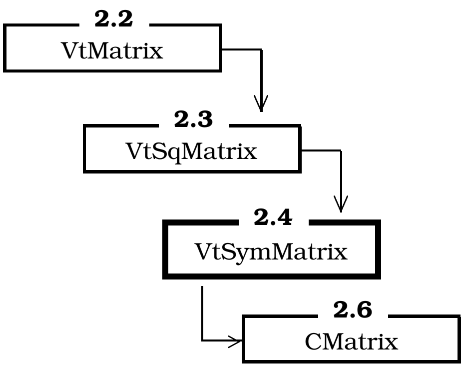
Public Members

	— Constructors —
	VtSqMatrix (const int row)
	VtSqMatrix (const VtSqMatrix& rhs)
	VtSqMatrix (const VtMatrix& rhs)
virtual	~**VtSqMatrix** ()
	— Matrix operations —
bool	**VtDinv** (int dim = 0) *transform to inverse*
const VtSqMatrix	**dinv** (int dim = 0) const *return inverse*
virtual double	**det** () const *return determinant*
const VtVector	**operator*** (const VtVector& rhs) const

2.4 class **VtSymMatrix** : public VtSqMatrix

Class for symmetric matrices

Inheritance



Public Members

— Constructors —

VtSymMatrix (const unsigned int dim)

VtSymMatrix (const unsigned int dim,
double diag)
initalize diagonal elements

VtSymMatrix (const VtSymMatrix& rhs)
copy constructor

VtSymMatrix (const VtSqMatrix& rhs)

VtSymMatrix (const VtMatrix& rhs)

— Matrix operations —

bool **invert** (const bool use_momentum)
calc inverse using momentum or not

bool **VtDsinv** (int dim = 0)
transform to inverse

const VtSymMatrix
dsinv (int dim = 0) const
return inverse

virtual double
det () const *compute determinant via CERNLIB dsfact()*

double **product** (const VtVector& rhs,
unsigned int dim = 0) const
compute $v^t * A * v$

const VtSymMatrix
product (const VtMatrix& rhs) const
compute $B^t * A * B$

virtual void
place_at (const VtMatrix& rhs,
const unsigned int row,
const unsigned int col)
copy a smaller matrix at a certain place

void **copy** (const VtSymMatrix& rhs)
in case matrix dimensions differ

const VtSymMatrix&

operator= (const VtSymMatrix& rhs)
copy

const VtSymMatrix&
operator+= (const double rhs)
$\boldsymbol{A} = (a_{\mu\nu} + \alpha)$

const VtSymMatrix&
operator-= (const double rhs)
$\boldsymbol{A} = (a_{\mu\nu} - \alpha)$

const VtSymMatrix&
operator*= (const double rhs)
$\boldsymbol{A} = (a_{\mu\nu} \cdot \alpha)$

const VtSymMatrix&
operator/= (const double rhs)
$\boldsymbol{A} = (a_{\mu\nu}/\alpha)$

const VtSymMatrix&
operator+= (const VtSymMatrix& rhs)
$\boldsymbol{A} = (a_{\mu\nu} + b_{\mu\nu})$

const VtSymMatrix&
operator-= (const VtSymMatrix& rhs)
$\boldsymbol{A} = (a_{\mu\nu} - b_{\mu\nu})$

const VtSymMatrix
operator+ (const VtSymMatrix& rhs) const

const VtSqMatrix
operator+ (const VtSqMatrix& rhs) const

const VtMatrix
operator+ (const VtMatrix& rhs) const

const VtSymMatrix
operator- (const VtSymMatrix& rhs) const

const VtSymMatrix
operator- (void) const

const VtSqMatrix
operator- (const VtSqMatrix& rhs) const

const VtMatrix
operator- (const VtMatrix& rhs) const

const VtSqMatrix
operator* (const VtSymMatrix& rhs) const

const VtSqMatrix
operator* (const VtSqMatrix& rhs) const

const VtMatrix

operator* (const VtMatrix& rhs) const

const VtVector
operator* (const VtVector& rhs) const

— Expert functions —

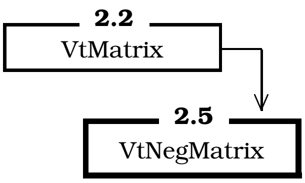
virtual void
print (std::ostream& os) const

2.5

class **VtNegMatrix** : public VtMatrix

Negative matrix class

Inheritance



Public Members

— Constructors —

VtNegMatrix (const unsigned int row,
const unsigned int col)

VtNegMatrix (const VtNegMatrix& rhs)

VtNegMatrix (const VtMatrix& rhs)

~**VtNegMatrix** ()

— Matrix operations —

const VtNegMatrix
T (void) const *return transposed matrix*

const VtNegMatrix&
operator+= (const double rhs)

const VtNegMatrix&

operator-= (const double rhs)

const VtNegMatrix&
operator*= (const double rhs)

const VtNegMatrix&
operator/= (const double rhs)

const VtNegMatrix&
operator+= (const VtMatrix& rhs)

const VtNegMatrix&
operator+= (const VtNegMatrix& rhs)

const VtNegMatrix&
operator-= (const VtMatrix& rhs)

const VtNegMatrix&
operator-= (const VtNegMatrix& rhs)

const VtMatrix
operator+ (const VtMatrix& rhs) const

const VtMatrix
operator+ (const VtNegMatrix& rhs) const

const VtMatrix
operator- (const VtMatrix& rhs) const

const VtMatrix
operator- (void) const

const VtMatrix
operator* (const VtMatrix& rhs) const

const VtVector
operator* (const VtVector& rhs) const

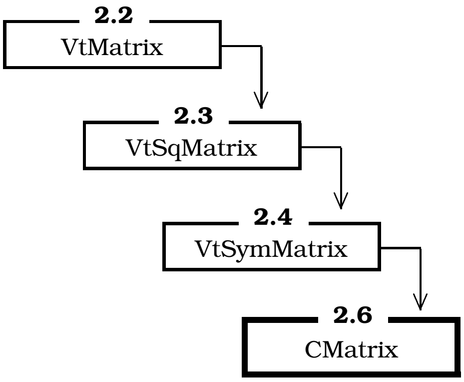
A class for negative matrices to optimize expressions like $C = -A * B$. For internal use only!

2.6

class **CMatrix** : public VtSymMatrix

HERA-B Track covariance matrix class

Inheritance



Public Members

— Constructors —

CMatrix ()

CMatrix (const CMatrix& rhs)

CMatrix (const VtSymMatrix& rhs)

CMatrix (const VtSqMatrix& rhs)

CMatrix (const VtMatrix& rhs)

CMatrix (const SMatrix<double, 5>& rhs)
not available in CINT

CMatrix (const ArtePointer<RTRA>& rtra)

CMatrix (const ArtePointer<RSEG>& rseg)

CMatrix (const ArtePointer<MTRA>& mtra)

CMatrix (const RecoSegment& r)

CMatrix (const ArtePointer<RecoTrack>& r)

— Access methods —

double	**x** () const	*rtra->cvf[0]*
double	**x_y** () const	*rtra->cvf[1]*
double	**x_tx** () const	*rtra->cvf[3]*
double	**x_ty** () const	*rtra->cvf[6]*
double	**x_p** () const	*rtra->cvf[10]* $*Q * -p^2$
double	**y** () const	*rtra->cvf[2]*

double	**y_tx** () const	*rtra->cvf[4]*
double	**y_ty** () const	*rtra->cvf[7]*
double	**y_p** () const	*rtra->cvf[11]* $*Q*-p^2$
double	**tx** () const	*rtra->cvf[5]*
double	**tx_ty** () const	*rtra->cvf[8]*
double	**tx_p** () const	*rtra->cvf[12]* $*Q*-p^2$
double	**ty** () const	*rtra->cvf[9]*
double	**ty_p** () const	*rtra->cvf[13]* $*Q*-p^2$
double	**p** () const	*rtra->cvf[14]* $*p^4$
	— Modification of matrix elements —	
void	**set_x** (const double x)	
void	**set_x_y** (const double xy)	
void	**set_x_tx** (const double xtx)	
void	**set_x_ty** (const double xty)	
void	**set_x_p** (const double xp)	
void	**set_y** (const double y)	
void	**set_y_tx** (const double ytx)	
void	**set_y_ty** (const double yty)	
void	**set_y_p** (const double yp)	
void	**set_tx** (const double tx)	
void	**set_tx_ty** (const double txty)	
void	**set_tx_p** (const double txp)	
void	**set_ty** (const double ty)	

void	**set_ty_p** (const double typ)
void	**set_p** (const double p)
	— Error propagation —
double	**x_prop** (double dz) const
double	**x_y_prop** (double dz) const
double	**x_tx_prop** (double dz) const
double	**x_ty_prop** (double dz) const
double	**x_p_prop** (double dz) const
double	**y_prop** (double dz) const
double	**y_tx_prop** (double dz) const
double	**y_ty_prop** (double dz) const
double	**y_p_prop** (double dz) const
void	**propagate** (const double dz) *propagate all matrix elements*

Class Graph