# rl_raylib

1.1.6

Generated by Doxygen 1.5.6

Mon Nov 3 18:15:05 2008

# Contents

# 1 rl_RayLib User's Guide

## 1.1 Copyright and License



**Author:**

Terry Gaetz

## 1.2 Purpose

The rl_RayLib library consists of a set of C++ classes for manipulating rays, including the effects of multilayer reflectivity.

### 1.2.1 Ray class

The rl_Ray class generalizes the rl_BasicRay class (found in the rl_basicray package) to include polarization information to the rl_BasicRay's position, direction, energy, and id number components. See the rl_basicray package documentation for further information on the rl_BasicRay and rl_RayMath classes.

The rays can be translated/rotated from a standard coordinate system (STD) to a "body center system" (BCS), and de-rotated/de-translated from the BCS system back to the STD system. Given a surface normal, the ray direction can be reflected to a new direction. This yields much of the transformation functionality needed for basic raytracing.

### 1.2.2 Reflectivity classes

The reflectivity classes include a number of components:

- rl_DielectricData encapsulates an array of energy bins providing the dielectric decrement information and methods to evaluate (interpolate) the decrements at a requested energy. The array is built on a helper "Plain Ol' Data" (POD) struct rl_DielectricPOD which provides the dielectric decrement at a specific energy.
- rl_DielectricLayer encapsulates the information about the interaction of a photon with a single dielectric layer, including the layer thickness, dielectric decrements given the photon energy, the component of the photon wave vector perpendicular to the layer, and various reflection and transmission coefficients, and surface "roughness" information. The layer can be a "substrate" (in which case the layer is considered as semi-infinite), vacuum, or a dielectric layer. These are mediated by helper "Plain Ol' Data" (POD) structs and classes: rl_ReflectionCoefPOD, rl_TransmissionCoefPOD, rl_ReflectionCoefPOD.
- rl_Multilayer encapsulates a stack of rl_DielectricLayer's. Given the energy, sine of the graze angle, the multilayer reflectivity can be evaluated.
- rl_MultilayerSurface adds surface normal information to an rl_Multilayer. Given an rl_Ray, rl_MultilayerSurface can evaluate the reflectivity for the surface. This is also where hooks for surface interception and scattering could be placed.

# 2 Directory Hierarchy

## 2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

# 3 Class Index

## 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:



# 4 Directory Documentation

## 4.1 rl_raylib/ Directory Reference

### Files

- file **rl_DielectricData.cc**
- file **rl_DielectricData.h**
- file **rl_DielectricLayer.cc**
- file **rl_DielectricLayer.h**
- file **rl_DielectricPODArray.cc**

- file **rl_DielectricPODArray.h**
- file **rl_Exception.cc**
- file **rl_Exception.h**
- file **rl_Multilayer.cc**
- file **rl_Multilayer.h**
- file **rl_MultilayerSurface.cc**
- file **rl_MultilayerSurface.h**
- file **rl_Polarization.cc**
- file **rl_Polarization.h**
- file **rl_Ray.cc**
- file **rl_Ray.h**
- file **rl_RayLib.h**
- file **rl_ReflectionCoefPOD.h**
- file **rl_Traits.h**
- file **rl_TransmissionCoefPOD.h**

# 5 Class Documentation

## 5.1 rl_DielectricData Class Reference

`#include <rl_DielectricData.h>`

Collaboration diagram for rl_DielectricData:

### Public Member Functions

- ~rl_DielectricData ()
- rl_DielectricData ()
- rl_DielectricData (rl_DielectricData const &other) throw ( rl_Exception )
- rl_DielectricData (rl_Traits::rl_DielectricPOD const *diel, size_t num_pts, rl_Traits::EInterpMode interp_mode, double bulk_density=1.0) throw ( rl_Exception )
- void init (rl_Traits::rl_DielectricPOD const *diel, size_t num_pts, rl_Traits::EInterpMode interp_mode, double bulk_density=1.0) throw ( rl_Exception )
- int alpha_gamma (double energy, double &alpha, double &gamma)
- double energy_min () const
- double energy_max () const
- double bulk_density_factor () const
- rl_Traits::Bool is_vacuum () const

### Classes

- struct rl_DielectricDataBinPOD

### 5.1.1 Detailed Description

A class encapsulating the dielectric data (alpha, gamma) as a function of energy.

Definition at line 64 of file rl_DielectricData.h.

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 rl_DielectricData::~rl_DielectricData ()

Non-virtual detructor.

Definition at line 53 of file rl_DielectricData.cc.

#### 5.1.2.2 rl_DielectricData::rl_DielectricData () `[inline]`

Default constructor. Constructs a vacuum rl_DielectricData. Use init to initialize the object to a state other than vacuum.

Definition at line 215 of file rl_DielectricData.h.

#### 5.1.2.3 rl_DielectricData::rl_DielectricData (rl_DielectricData const & *other*) throw ( rl_Exception )

Copy constructor

**Parameters:**

***other*** rl_DielectricData to be copied.

Definition at line 60 of file rl_DielectricData.cc.

#### 5.1.2.4 rl_DielectricData::rl_DielectricData (rl_Traits::rl_DielectricPOD const * *diel*, size_t *num_pts*, rl_Traits::EInterpMode *interp_mode*, double *bulk_density* = `1.0`) throw ( rl_Exception )

Constructor. The data for the dielectric constants are obtained from a c-style array of const rl_DielectricPOD; the array contains num_pts points. The interpolation mode will be set to interp_mode. The optical constants will be scaled by bulk_density.

**Parameters:**

***diel*** an array of `num_pts` POD's containing the complex dielectric decrements.

***num_pts*** number of elements in diel.

***interp_mode*** type of interpolation to be used in interpolating the dielectric constants.

***bulk_density*** relative bulk density factor to be used; 1.0 for full bulk density. The dielectric decrements will be scaled by bulk_density.

Definition at line 75 of file rl_DielectricData.cc.

References init().

### 5.1.3 Member Function Documentation

#### 5.1.3.1 void rl_DielectricData::init (rl_Traits::rl_DielectricPOD const * *diel*, size_t *num_pts*, rl_Traits::EInterpMode *interp_mode*, double *bulk_density* = 1.0) throw ( rl_Exception )

Initialization function. The data are obtained from a c-style array of const rl_DielectricPOD; the array contains num_pts points. The interpolation mode will be set to interp_mode. The optical constants will be scaled by bulk_density.

**Parameters:**

***diel*** an array of `num_pts` POD's containing the complex dielectric decrements.

***num_pts*** number of elements in diel.

***interp_mode*** type of interpolation to be used in interpolating the dielectric constants.

***bulk_density*** relative bulk density factor to be used; 1.0 for full bulk density. The dielectric decrements will be scaled by bulk_density.

Definition at line 86 of file rl_DielectricData.cc.

References rl_Traits::rl_DielectricPOD::alpha_, rl_Traits::ELinLin, rl_Traits::ELinLog, rl_Traits::ELogLin, rl_Traits::ELogLog, rl_Traits::rl_DielectricPOD::energy_, rl_Traits::rl_DielectricPOD::gamma_, rl_DielectricData::rl_DielectricDataBinPOD::hi_, and rl_DielectricData::rl_DielectricDataBinPOD::lo_.

Referenced by rl_DielectricLayer::init(), and rl_DielectricData().

#### 5.1.3.2 int rl_DielectricData::alpha_gamma (double *energy*, double & *alpha*, double & *gamma*)

Evaluate the dielectric decrements, alpha and gamma, at the given energy. The energy must be between the minimum and the maximum energies in the `rl_DielectricPOD` array used to initialize the `rl_DielectricData` Returns: 0 on success; -1 if energy is below lower limit, +1 if energy is above upper limit.

**Parameters:**

***energy*** photon energy (in keV).

***alpha*** real part of the complex dielectric decrement.

***gamma*** imaginary part of the complex dielectric decrement.

Definition at line 176 of file rl_DielectricData.cc.

References rl_Traits::rl_DielectricPOD::alpha_, rl_Traits::ELinLin, rl_Traits::ELinLog, rl_Traits::ELogLin, rl_Traits::ELogLog, rl_Traits::rl_DielectricPOD::energy_, rl_Traits::rl_DielectricPOD::gamma_, rl_DielectricData::rl_DielectricDataBinPOD::hi_, and rl_DielectricData::rl_DielectricDataBinPOD::lo_.

Referenced by rl_DielectricLayer::setup_for().

#### 5.1.3.3 double rl_DielectricData::energy_min () const [inline]

Return the minimum energy covered by this dataset.

Definition at line 229 of file rl_DielectricData.h.

Referenced by rl_DielectricLayer::cdump_on(), rl_DielectricLayer::cprint_constraints_on(), rl_DielectricLayer::dump_on(), and rl_DielectricLayer::energy_min().

#### 5.1.3.4 double rl_DielectricData::energy_max () const [inline]

Return the maximum energy covered by this dataset.

Definition at line 234 of file rl_DielectricData.h.

Referenced by rl_DielectricLayer::cdump_on(), rl_DielectricLayer::cprint_constraints_on(), rl_DielectricLayer::dump_on(), and rl_DielectricLayer::energy_max().

#### 5.1.3.5 double rl_DielectricData::bulk_density_factor () const [inline]

Return the maximum energy covered by this dataset. Return the relative bulk density for this set of optical constants. 1.0 is nominal full bulk density

Definition at line 239 of file rl_DielectricData.h.

Referenced by rl_DielectricLayer::bulk_density_factor(), rl_DielectricLayer::cdump_on(), rl_DielectricLayer::cprint_constraints_on(), and rl_DielectricLayer::dump_on().

#### 5.1.3.6 rl_Traits::Bool rl_DielectricData::is_vacuum () const [inline]

Returns True if the object is a vacuum state; False, otherwise.

Definition at line 244 of file rl_DielectricData.h.

Referenced by rl_DielectricLayer::is_vacuum().

The documentation for this class was generated from the following files:

- rl_DielectricData.h
- rl_DielectricData.cc

## 5.2 rl_DielectricData::rl_DielectricDataBinPOD Struct Reference

`#include <rl_DielectricData.h>`

Collaboration diagram for rl_DielectricData::rl_DielectricDataBinPOD:

### Public Attributes

- rl_Traits::rl_DielectricPOD lo_
  *lower edge of energy bin*
- rl_Traits::rl_DielectricPOD hi_
  *upper edge of energy bin*

### 5.2.1 Detailed Description

A POD describing the lower and upper edge of an energy bin.

Definition at line 73 of file rl_DielectricData.h.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 rl_Traits::rl_DielectricPOD rl_DielectricData::rl_DielectricDataBinPOD::lo_

lower edge of energy bin

Definition at line 76 of file rl_DielectricData.h.

Referenced by rl_DielectricData::alpha_gamma(), and rl_DielectricData::init().

#### 5.2.2.2 rl_Traits::rl_DielectricPOD rl_DielectricData::rl_DielectricDataBinPOD::hi_

upper edge of energy bin

Definition at line 78 of file rl_DielectricData.h.

Referenced by rl_DielectricData::alpha_gamma(), and rl_DielectricData::init().

The documentation for this struct was generated from the following file:

- rl_DielectricData.h

## 5.3 rl_DielectricLayer Class Reference

`#include <rl_DielectricLayer.h>`

Collaboration diagram for rl_DielectricLayer:

### Public Types

- typedef rl_Traits::complex complex
  *complex type*
- typedef rl_Traits::ERoughType ERoughType
  *roughness type*

### Public Member Functions

- ~rl_DielectricLayer ()
- rl_DielectricLayer (char const layer_name[ ]=0) throw ( rl_Exception )
- rl_DielectricLayer (rl_DielectricLayer const &other) throw ( rl_Exception )
- rl_DielectricLayer (rl_Traits::rl_DielectricPOD const *diel, std::size_t ndielpts, double layer_thickness, double roughness, rl_Traits::ERoughType roughness_type, rl_Traits::EInterpMode interp_mode, double bulkdensity, char const *layer_name, rl_Traits::Bool is_substrate=rl_Traits::False) throw ( rl_Exception )
- void init (rl_Traits::rl_DielectricPOD const *diel, std::size_t ndielpts, double layer_thickness, double roughness, rl_Traits::ERoughType roughness_type, rl_Traits::EInterpMode interp_mode, double bulkdensity, char const *layer_name, rl_Traits::Bool is_substrate=rl_Traits::False) throw ( rl_Exception )
- int setup_for (double energy, double sinphi)
- void reflect_nlayer (rl_DielectricLayer layer[ ], int num)
- void reflect_amp (rl_DielectricLayer const &layer, double sinphi)
- complex const & propagator () const
- double alpha () const
- double gamma () const
- double roughness () const

- ERoughType roughness_type () const
- rl_ReflectionCoefPOD const & reflection_coef () const
- double reflectivity (double polarization_factor=0.0) const
- rl_Traits::Bool is_substrate () const
- rl_Traits::Bool is_vacuum () const
- char const * layer_name () const
- double energy_min () const
- double energy_max () const
- double thickness () const
- double zcoat () const
- double bulk_density_factor () const
- std::ostream & dump_on (std::ostream &os, char const pre[ ]="", char const pst[ ]="") const
- void cdump_on (std::FILE *of, char const pre[ ]="", char const pst[ ]="") const
- void cprint_constraints_on (std::FILE *of, char const pre[ ]="", char const pst[ ]="") const

### 5.3.1 Detailed Description

A class encapsulating the multilayer reflection of a ray. Given the energy and the sine of the graze angle, the reflection coefficient is evaluated.

Definition at line 60 of file rl_DielectricLayer.h.

### 5.3.2 Member Typedef Documentation

#### 5.3.2.1 typedef rl_Traits::complex rl_DielectricLayer::complex

complex type

Definition at line 65 of file rl_DielectricLayer.h.

#### 5.3.2.2 typedef rl_Traits::ERoughType rl_DielectricLayer::ERoughType

roughness type

Definition at line 68 of file rl_DielectricLayer.h.

### 5.3.3 Constructor & Destructor Documentation

#### 5.3.3.1 rl_DielectricLayer::~rl_DielectricLayer ()

Destructor

Definition at line 47 of file rl_DielectricLayer.cc.

#### 5.3.3.2 rl_DielectricLayer::rl_DielectricLayer (char const *layer_name*[ ] = 0) throw ( rl_Exception )

Constructor. By default, constructs a VACUUM layer. Use init method to initialize for conditions other than vacuum.

**Parameters:**

***layer_name*** optional string naming the layer.

Definition at line 56 of file rl_DielectricLayer.cc.

References layer_name().

#### 5.3.3.3 rl_DielectricLayer::rl_DielectricLayer (rl_DielectricLayer const & *other*) throw ( rl_Exception )

DEEP Copy constructor

**Parameters:**

***other*** rl_DielectricLayer to be copied.

Definition at line 99 of file rl_DielectricLayer.cc.

#### 5.3.3.4 rl_DielectricLayer::rl_DielectricLayer (rl_Traits::rl_DielectricPOD const * *diel*, std::size_t *ndielpts*, double *layer_thickness*, double *roughness*, rl_Traits::ERoughType *roughness_type*, rl_Traits::EInterpMode *interp_mode*, double *bulkdensity*, char const * *layer_name*, rl_Traits::Bool *is_substrate* = rl_Traits::False) throw ( rl_Exception )

Constructor.

**Parameters:**

***diel*** array of dielectric decrement information.
***ndielpts*** number of points in diel array.
***layer_thickness*** layer thickness (Angstroms).
***roughness*** layer "roughness" (Angstroms).
***roughness_type*** interlayer grading "roughness" type.
***interp_mode*** type of optical constant interpolation to be done.
***bulkdensity*** relative bulk density factor; 1 for full bulk density.
***layer_name*** string describing the layer composition.
***is_substrate*** boolean: True if this is the substrate layer.

Definition at line 142 of file rl_DielectricLayer.cc.

References init(), is_substrate(), layer_name(), roughness(), and roughness_type().

### 5.3.4 Member Function Documentation

#### 5.3.4.1 void rl_DielectricLayer::init (rl_Traits::rl_DielectricPOD const * *diel*, std::size_t *ndielpts*, double *layer_thickness*, double *roughness*, rl_Traits::ERoughType *roughness_type*, rl_Traits::EInterpMode *interp_mode*, double *bulkdensity*, char const * *layer_name*, rl_Traits::Bool *is_substrate* = `rl_Traits::False`) throw ( rl_Exception )

Initializer

**Parameters:**

***diel*** array of dielectric decrement information.

***ndielpts*** number of points in diel array.

***layer_thickness*** layer thickness (Angstroms).

***roughness*** interlayer grading "roughness" (Angstrom).

***roughness_type*** interlayer grading "roughness" type.

***interp_mode*** type of optical constant interpolation to be done.

***bulkdensity*** relative bulk density factor; 1 for full bulk density.

***layer_name*** string describing the layer composition.

***is_substrate*** boolean: True if this is the substrate layer.

Definition at line 160 of file rl_DielectricLayer.cc.

References rl_Traits::ERoughDebyeWaller_CSAO, rl_Traits::ERoughDebyeWaller_RSAO, rl_Traits::ERoughDebyeWaller_Spiller, rl_Traits::ERoughModifiedDebyeWaller, rl_Traits::ERoughNevotCroce, rl_Traits::ERoughNone, rl_DielectricData::init(), is_substrate(), layer_name(), roughness(), and roughness_type().

Referenced by rl_DielectricLayer().

#### 5.3.4.2 int rl_DielectricLayer::setup_for (double *energy*, double *sinphi*)

Set up layer state for given energy and graze angle.

**Parameters:**

***energy*** energy (keV).

***sinphi*** sine of the graze angle between the ray (in vacuum) and the top surface of the multilayer.

Definition at line 224 of file rl_DielectricLayer.cc.

References rl_DielectricData::alpha_gamma(), rl_Traits::ERoughNone, is_substrate(), and is_vacuum().

Referenced by rl_Multilayer::multilayer_reflect_coef().

#### 5.3.4.3 void rl_DielectricLayer::reflect_nlayer (rl_DielectricLayer *layer*[ ], int *num*)

Compute reflectivity for a stack of num layers.

PRECONDITION: The layers must be pre-initialized with setup_for and the reflection coefficients evaluated with reflect_amp.

**Parameters:**

***layer*** array of layers.

***num*** number of layers in the array.

Definition at line 520 of file rl_DielectricLayer.cc.

References cprint_constraints_on(), rl_ReflectionCoefPOD::para(), rl_ReflectionCoefPOD::perp(), propagator(), r2_tij_tji_para_, r2_tij_tji_perp_, and rflcoef_.

Referenced by rl_Multilayer::multilayer_reflect_coef().

#### 5.3.4.4 void rl_DielectricLayer::reflect_amp (rl_DielectricLayer const & *layer*, double *sinphi*)

Compute reflection amplitude for the interface between this layer and the layer immediately above it. sinphi is the graze angle at the topmost layer (i.e., in vacuum). Layer "layer" is assumed to be adjacent to this layer, with "layer" nearer the vacuum and this layer nearer the substrate. The reflection coefficient for this layer is updated.

PRECONDITION: The layers must be pre-initialized with setup_for

**Parameters:**

***layer*** adjacent layer above this layer, where above means closer to the vacuum.

***sinphi*** sine of the graze angle between the ray (in vacuum) and the top surface of the multilayer.

Definition at line 324 of file rl_DielectricLayer.cc.

References alpha_, rl_Traits::ERoughDebyeWaller_CSAO, rl_Traits::ERoughDebyeWaller_RSAO, rl_Traits::ERoughDebyeWaller_Spiller, rl_Traits::ERoughModifiedDebyeWaller, rl_Traits::ERoughNevotCroce, rl_Traits::ERoughNone, gamma_, is_substrate(), kt_perp_, name_, rl_TransmissionCoefPOD::para(), rl_ReflectionCoefPOD::para(), rl_TransmissionCoefPOD::perp(), and rl_ReflectionCoefPOD::perp().

Referenced by rl_Multilayer::multilayer_reflect_coef().

#### 5.3.4.5 rl_Traits::complex const & rl_DielectricLayer::propagator () const [inline]

The propagator for this layer.

**Returns:**

propagator for this layer.

Definition at line 524 of file rl_DielectricLayer.h.

Referenced by reflect_nlayer().

#### 5.3.4.6 double rl_DielectricLayer::alpha () const [inline]

Returns this layer's dielectric decrement (real part).

**Returns:**

this layer's dielectric decrement (real part).

Definition at line 528 of file rl_DielectricLayer.h.

#### 5.3.4.7 double rl_DielectricLayer::gamma () const [inline]

Returns this layer's dielectric decrement (imag part).

**Returns:**

this layer's dielectric decrement (imag part).

Definition at line 532 of file rl_DielectricLayer.h.

#### 5.3.4.8 double rl_DielectricLayer::roughness () const [inline]

Returns the roughness parameter of the upper surface of this layer.

**Returns:**

the roughness parameter of the upper surface of this layer.

Definition at line 548 of file rl_DielectricLayer.h.

Referenced by init(), and rl_DielectricLayer().

#### 5.3.4.9 rl_Traits::ERoughType rl_DielectricLayer::roughness_type () const [inline]

Returns the roughness type of the upper surface of this layer.

**Returns:**

the roughness type of the upper surface of this layer.

Definition at line 544 of file rl_DielectricLayer.h.

Referenced by init(), and rl_DielectricLayer().

#### 5.3.4.10 rl_ReflectionCoefPOD const & rl_DielectricLayer::reflection_coef () const [inline]

Returns this layer's complex reflection coefficient.

**Returns:**

this layer's complex reflection coefficient.

Definition at line 552 of file rl_DielectricLayer.h.

Referenced by rl_Multilayer::multilayer_reflect_coef().

#### 5.3.4.11 double rl_DielectricLayer::reflectivity (double *polarization_factor* = 0.0) const [inline]

Returns: this layer's reflectivity.

**Parameters:**

***polarization_factor*** - polarization factor; it must be a value between -1 and 1.

The polarization factor is related to parallel (p) and perpendicular (s) polarization by: polarization_factor $= (I_{perp} - I_{para})/(I_{perp} + I_{para})$ or polarization_factor $= (I_s - I_p)/(I_s + I_p)$ where $I_{perp} and I_{para}$ are the perpendicular and parallel E-field *intensities*, respectively. Thus,

- -1: pure parallel (p) polarization.
- 0: completely unpolarized.
- +1: pure perpendicular (s) polarization.

**Returns:**

this layer's reflectivity

Definition at line 556 of file rl_DielectricLayer.h.

References rl_ReflectionCoefPOD::reflectivity().

#### 5.3.4.12 rl_Traits::Bool rl_DielectricLayer::is_substrate () const [inline]

**Returns:**

rl_Traits::True if this layer is the substrate, rl_Traits::False otherwise.

Definition at line 560 of file rl_DielectricLayer.h.

Referenced by init(), reflect_amp(), rl_DielectricLayer(), and setup_for().

#### 5.3.4.13 rl_Traits::Bool rl_DielectricLayer::is_vacuum () const [inline]

**Returns:**

rl_Traits::True if this layer is the vacuum, rl_Traits::False otherwise.

Definition at line 564 of file rl_DielectricLayer.h.

References rl_DielectricData::is_vacuum().

Referenced by cdump_on(), cprint_constraints_on(), dump_on(), and setup_for().

#### 5.3.4.14 char const * rl_DielectricLayer::layer_name () const [inline]

**Returns:**

the name of this layer as a character string.

Definition at line 568 of file rl_DielectricLayer.h.

Referenced by init(), and rl_DielectricLayer().

#### 5.3.4.15 double rl_DielectricLayer::energy_min () const [inline]

**Returns:**

the minimum energy allowed for this layer.

Definition at line 572 of file rl_DielectricLayer.h.

References rl_DielectricData::energy_min().

#### 5.3.4.16 double rl_DielectricLayer::energy_max () const [inline]

**Returns:**

the maximum energy allowed for this layer.

Definition at line 576 of file rl_DielectricLayer.h.

References rl_DielectricData::energy_max().

#### 5.3.4.17 double rl_DielectricLayer::thickness () const [inline]

**Returns:**

the layer thickness.

Definition at line 536 of file rl_DielectricLayer.h.

#### 5.3.4.18 double rl_DielectricLayer::zcoat () const [inline]

**Returns:**

the layer dimensionless thickness.

Definition at line 540 of file rl_DielectricLayer.h.

#### 5.3.4.19 double rl_DielectricLayer::bulk_density_factor () const [inline]

**Returns:**

the relative bulk density for this layer. 1.0 is nominal full bulk density

Definition at line 580 of file rl_DielectricLayer.h.

References rl_DielectricData::bulk_density_factor().

#### 5.3.4.20 std::ostream & rl_DielectricLayer::dump_on (std::ostream & *os*, char const *pre*[ ] = "", char const *pst*[ ] = "") const

Dumps layer information to a stream.

**Parameters:**

***os*** stream.

***pre*** optional prefix string.

***pst*** optional postfix string.

Definition at line 791 of file rl_DielectricLayer.cc.

References rl_DielectricData::bulk_density_factor(), rl_DielectricData::energy_max(), rl_DielectricData::energy_min(), and is_vacuum().

Referenced by rl_Multilayer::dump_on().

#### 5.3.4.21 void rl_DielectricLayer::cdump_on (std::FILE * *of*, char const *pre*[ ] = "", char const *pst*[ ] = "") const

Dumps layer information to a C-style FILE* stream.

**Parameters:**

***of*** output file.

***pre*** optional prefix string.

***pst*** optional postfix string.

Definition at line 836 of file rl_DielectricLayer.cc.

References rl_DielectricData::bulk_density_factor(), rl_ReflectionCoefPOD::cprint_on(), rl_DielectricData::energy_max(), rl_DielectricData::energy_min(), and is_vacuum().

Referenced by rl_Multilayer::cdump_on().

#### 5.3.4.22 void rl_DielectricLayer::cprint_constraints_on (std::FILE * *of*, char const *pre*[ ] = "", char const *pst*[ ] = "") const

Dumps layer information and constraints to a C-style FILE* stream.

**Parameters:**

***of*** output file.

***pre*** optional prefix string.

***pst*** optional postfix string.

Definition at line 880 of file rl_DielectricLayer.cc.

References rl_DielectricData::bulk_density_factor(), rl_DielectricData::energy_max(), rl_DielectricData::energy_min(), is_vacuum(), rl_ReflectionCoefPOD::para(), rl_TransmissionCoefPOD::para(), rl_ReflectionCoefPOD::perp(), and rl_TransmissionCoefPOD::perp().

Referenced by reflect_nlayer().

The documentation for this class was generated from the following files:

- rl_DielectricLayer.h
- rl_DielectricLayer.cc

## 5.4 rl_DielectricPODArray Class Reference

`#include <rl_DielectricPODArray.h>`

Collaboration diagram for rl_DielectricPODArray:

### Public Member Functions

- ~rl_DielectricPODArray ()
- rl_DielectricPODArray ()
- rl_DielectricPODArray (size_t nelts, double const *energy, double const *alpha, double const *gamma) throw ( rl_Exception )
- rl_DielectricPODArray (size_t nelts, rl_Traits::rl_DielectricPOD *diel) throw ( rl_Exception )
- void init (size_t nelts, double const *energy, double const *alpha, double const *gamma)
- void init (size_t nelts, rl_Traits::rl_DielectricPOD *diel)
- size_t num_elts () const
- rl_Traits::rl_DielectricPOD const * const_data_ptr () const
- void cprint_on (std::FILE *of, char const pre[ ]="", char const pst[ ]="") const

### Protected Attributes

- size_t nelts_

  *number of dielectric decrements read in*

- rl_Traits::rl_DielectricPOD * data_

  *pointer to the data*

### 5.4.1 Detailed Description

A class encapsulating an array of rl_DielectricPODs storing the dielectric contants as a function of energy:

- energy (keV)

- alpha (real part of dielectric decrement)
- gamma (imaginary part of the dielectric decrement)

The complex dielectric constant has real part (1-alpha) and imaginary part (-gamma).

Definition at line 57 of file rl_DielectricPODArray.h.

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 rl_DielectricPODArray::~rl_DielectricPODArray ()

Destructor

Definition at line 65 of file rl_DielectricPODArray.cc.

References data_.

#### 5.4.2.2 rl_DielectricPODArray::rl_DielectricPODArray ()

Default constructor.

An empty uninitialized rl_DielectricPODArray is created and the init method must be called to initialize the object.

Definition at line 69 of file rl_DielectricPODArray.cc.

#### 5.4.2.3 rl_DielectricPODArray::rl_DielectricPODArray (size_t *nelts*, double const * *energy*, double const * *alpha*, double const * *gamma*) throw ( rl_Exception )

Constructor.

**Parameters:**

***nelts*** number of elements in the array

***energy*** array of energies

***alpha*** array of dielectric decrement real part (alpha)

***gamma*** array of dielectric decrement imag part (gamma)

Definition at line 74 of file rl_DielectricPODArray.cc.

References data_, init(), and nelts_.

#### 5.4.2.4 rl_DielectricPODArray::rl_DielectricPODArray (size_t *nelts*, rl_Traits::rl_DielectricPOD * *diel*) throw ( rl_Exception )

Constructor.

**Parameters:**

***nelts*** number of elements in the array

***diel*** array of dielectric decrement PODs

Definition at line 101 of file rl_DielectricPODArray.cc.

References data_, init(), and nelts_.

### 5.4.3 Member Function Documentation

#### 5.4.3.1 void rl_DielectricPODArray::init (size_t *nelts*, double const * *energy*, double const * *alpha*, double const * *gamma*)

Initializer.

**Parameters:**

***nelts*** number of elements in the array

***energy*** array of energies

***alpha*** array of dielectric decrement real part (alpha)

***gamma*** array of dielectric decrement imag part (gamma)

initialize this rl_DielectricPODArray from the energy, alpha, and gamma arrays. The rl_DielectricPODArray is sorted on the energy field.

Definition at line 122 of file rl_DielectricPODArray.cc.

References rl_Traits::rl_DielectricPOD::alpha_, data_, rl_Traits::rl_DielectricPOD::energy_, rl_Traits::rl_DielectricPOD::gamma_, and nelts_.

Referenced by rl_DielectricPODArray().

#### 5.4.3.2 void rl_DielectricPODArray::init (size_t *nelts*, rl_Traits::rl_DielectricPOD * *diel*)

Initializer.

**Parameters:**

***nelts*** number of elements in the array

***diel*** array of dielectric decrement PODs

initialize this rl_DielectricPODArray from the input array of rl_DielectricPODs. The rl_DielectricPODArray is sorted on the energy field.

Definition at line 140 of file rl_DielectricPODArray.cc.

References rl_Traits::rl_DielectricPOD::alpha_, data_, rl_Traits::rl_DielectricPOD::energy_, rl_Traits::rl_DielectricPOD::gamma_, and nelts_.

#### 5.4.3.3 size_t rl_DielectricPODArray::num_elts () const [inline]

Accessor.

**Returns:**

number of elements in the rl_DielectricPOD array.

Definition at line 170 of file rl_DielectricPODArray.h.

References nelts_.

#### 5.4.3.4 rl_Traits::rl_DielectricPOD const * rl_DielectricPODArray::const_data_ptr () const [inline]

Accessor.

**Returns:**

pointer-to-const to rl_DielectricPOD array.

Definition at line 174 of file rl_DielectricPODArray.h.

References data_.

#### 5.4.3.5 void rl_DielectricPODArray::cprint_on (std::FILE * *of*, char const *pre*[ ] = "", char const *pst*[ ] = "") const

Accessor.

**Returns:**

pointer-to-const to rl_DielectricPOD array. Print reflectivity information to output FILE* stream.

**Parameters:**

***of*** output FILE* stream.

***pre*** optional prefix (char*) string.

***pst*** optional postfix (char*) string.

Definition at line 156 of file rl_DielectricPODArray.cc.

References data_, and nelts_.

### 5.4.4 Member Data Documentation

#### 5.4.4.1 size_t rl_DielectricPODArray::nelts_ [protected]

number of dielectric decrements read in

Definition at line 62 of file rl_DielectricPODArray.h.

Referenced by cprint_on(), init(), num_elts(), and rl_DielectricPODArray().

#### 5.4.4.2 rl_Traits::rl_DielectricPOD* rl_DielectricPODArray::data_ [protected]

pointer to the data

Definition at line 64 of file rl_DielectricPODArray.h.

Referenced by const_data_ptr(), cprint_on(), init(), rl_DielectricPODArray(), and ~rl_DielectricPODArray().

The documentation for this class was generated from the following files:

- rl_DielectricPODArray.h
- rl_DielectricPODArray.cc

## 5.5 rl_Exception Class Reference

`#include <rl_Exception.h>`

### Public Member Functions

- ~rl_Exception () throw ( )
  *Destructor.*
- rl_Exception (const string &msg)
  *Include a string describing the exception.*
- rl_Exception (const char *format,...)
  *Format a string describing the exception.*

### 5.5.1 Detailed Description

The exception thrown by the rl_RayLib and rl_RaySupLib libraries.

Definition at line 35 of file rl_Exception.h.

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 rl_Exception::~rl_Exception () throw ( )

Destructor.

Definition at line 27 of file rl_Exception.cc.

#### 5.5.2.2 rl_Exception::rl_Exception (const string & *msg*)

Include a string describing the exception.

Definition at line 29 of file rl_Exception.cc.

#### 5.5.2.3 rl_Exception::rl_Exception (const char * *format*, ...)

Format a string describing the exception.

Definition at line 32 of file rl_Exception.cc.

The documentation for this class was generated from the following files:

- rl_Exception.h
- rl_Exception.cc

## 5.6 rl_Multilayer Class Reference

`#include <rl_Multilayer.h>`

Collaboration diagram for rl_Multilayer:

### Public Types

- typedef rl_Traits::EInterpMode EInterpMode
  *interpolation mode for dielectric data*

### Public Member Functions

- virtual ~rl_Multilayer ()
- rl_Multilayer (int num_layers, rl_DielectricLayer *layers, rl_Traits::Bool adopt_data=rl_Traits::True)
- void init (int num_layers, rl_DielectricLayer *layers, rl_Traits::Bool adopt_data=rl_Traits::True)
- int multilayer_reflect_coef (rl_ReflectionCoefPOD &rfl, double energy, double sg)

- int multilayer_reflectivity (double &rfl, double energy, double sg, double polarization_factor=0.0)
- rl_DielectricLayer const & layer (int layer_no) const
- int num_layers () const
- std::ostream & dump_on (std::ostream &os, int layer_no, char const pre[ ]="", char const pst[ ]="") const
- void cdump_on (std::FILE *of, int layer_no, char const pre[ ]="", char const pst[ ]="") const

**Protected Attributes**

- int num_layers_

  *number of multilayers*

- rl_DielectricLayer vacuum_

  *the top (vacuum) layer*

- rl_DielectricLayer * layer_

  *array of dielectric layers*

- rl_Traits::Bool own_data_

  *do we own the rl_DielectricLayer array?*

### 5.6.1 Detailed Description

A class encapsulating reflection of a ray from a multilayer surface. Given the energy and the sine of the graze angle, reflection coefficient can be evaluated.

Definition at line 60 of file rl_Multilayer.h.

### 5.6.2 Member Typedef Documentation

#### 5.6.2.1 typedef rl_Traits::EInterpMode rl_Multilayer::EInterpMode

interpolation mode for dielectric data

Definition at line 85 of file rl_Multilayer.h.

### 5.6.3 Constructor & Destructor Documentation

#### 5.6.3.1 rl_Multilayer::~rl_Multilayer () [virtual]

Destructor.

Frees up rl_DielectricLayer array if own_data_ is rl_Traits::True.

Definition at line 46 of file rl_Multilayer.cc.

References layer_, and own_data_.

#### 5.6.3.2 rl_Multilayer::rl_Multilayer (int *num_layers*, rl_DielectricLayer * *layers*, rl_Traits::Bool *adopt_data* = `rl_Traits::True`)

Construct multilayer from num_layers individual layers.

**Parameters:**

***num_layers*** number of layers to construct.

***layers*** pointer to array of layer data

***adopt_data*** boolean flag

- if rl_Traits::True, rl_Multilayer assumes responsibility for deleting the rl_DielectricLayer array;
- if rl_Traits::False, it is the caller's responsibility.

Definition at line 53 of file rl_Multilayer.cc.

### 5.6.4 Member Function Documentation

#### 5.6.4.1 void rl_Multilayer::init (int *num_layers*, rl_DielectricLayer * *layers*, rl_Traits::Bool *adopt_data* = `rl_Traits::True`)

Initialize multilayer from num_layers individual layers.

**Parameters:**

***num_layers*** number of layers to construct.

***layers*** pointer to array of layer data

***adopt_data*** boolean flag

- if rl_Traits::True, rl_Multilayer assumes responsibility for deleting the rl_DielectricLayer array;
- if rl_Traits::False, it is the caller's responsibility.

Definition at line 64 of file rl_Multilayer.cc.

References layer_, num_layers_, and own_data_.

#### 5.6.4.2 int rl_Multilayer::multilayer_reflect_coef (rl_ReflectionCoefPOD & *rfl*, double *energy*, double *sg*)

Evaluate the multilayer reflection coefficients

**Returns:**

0 if successful, the number of the invalid layer if not.

**Parameters:**

***rfl*** the computed reflection coefficient.

***energy*** ray energy.

***sg*** sine of the graze angle between the ray and the surface.

Definition at line 90 of file rl_Multilayer.cc.

References layer_, num_layers_, rl_DielectricLayer::reflect_amp(), rl_DielectricLayer::reflect_nlayer(), rl_DielectricLayer::reflection_coef(), rl_DielectricLayer::setup_for(), and vacuum_.

Referenced by multilayer_reflectivity(), and rl_MultilayerSurface::reflect().

#### 5.6.4.3 int rl_Multilayer::multilayer_reflectivity (double & *rfl*, double *energy*, double *sg*, double *polarization_factor* = `0.0`)

Evaluate the multilayer reflectivity (assuming unpolarized rays)

**Returns:**

0 if successful, the number of the invalid layer if not.

**Parameters:**

***rfl*** multilayer reflectivity.

***energy*** ray energy.

***sg*** sine of the graze angle between the ray and the surface.

***polarization_factor*** polarization factor; it must be a value between -1 and 1. The polarization factor is related to parallel (p) and perpendicular (s) polarization by:

$$\texttt{polarization\_factor} = \frac{(I_\perp - I_\parallel)}{(I_\perp + I_\parallel)}$$

or

$$\texttt{polarization\_factor} = \frac{(I_s - I_p)}{(I_s + I_p)}$$

where $I_\perp$ and $I_\parallel$ are the perpendicular and parallel E-field *intensities*, respectively. Thus,

- -1: pure parallel (p) polarization.
- 0: completely unpolarized.
- +1: pure perpendicular (s) polarization.

Definition at line 136 of file rl_Multilayer.cc.

References rl_ReflectionCoefPOD::init(), multilayer_reflect_coef(), and rl_ReflectionCoefPOD::reflectivity().

#### 5.6.4.4 rl_DielectricLayer const & rl_Multilayer::layer (int *layer_no*) const

**Returns:**

const reference to layer layer_no

Definition at line 156 of file rl_Multilayer.cc.

References layer_.

Referenced by rl_MultilayerSurface::layer().

#### 5.6.4.5 int rl_Multilayer::num_layers () const [inline]

**Returns:**

number of layers

Definition at line 231 of file rl_Multilayer.h.

References num_layers_.

#### 5.6.4.6 std::ostream & rl_Multilayer::dump_on (std::ostream & *os*, int *layer_no*, char const *pre*[ ] = "", char const *pst*[ ] = "") const

Dump information about layer layer_no to stream os.

**Returns:**

reference to stream os.

**Parameters:**

***os*** output stream

***layer_no*** layer number to be dumped

***pre*** optional prefix string

***pst*** optional postfix string

Definition at line 164 of file rl_Multilayer.cc.

References rl_DielectricLayer::dump_on(), and layer_.

Referenced by rl_MultilayerSurface::dump_on().

#### 5.6.4.7 void rl_Multilayer::cdump_on (std::FILE * *of*, int *layer_no*, char const *pre*[ ] = "", char const *pst*[ ] = "") const

Dump information about layer layer_no to output FILE*.

**Parameters:**

***of*** output FILE*

***layer_no*** layer number to be dumped

***pre*** optional prefix string

***pst*** optional postfix string

Definition at line 181 of file rl_Multilayer.cc.

References rl_DielectricLayer::cdump_on(), and layer_.

### 5.6.5 Member Data Documentation

#### 5.6.5.1 int rl_Multilayer::num_layers_ [protected]

number of multilayers

Definition at line 74 of file rl_Multilayer.h.

Referenced by init(), multilayer_reflect_coef(), and num_layers().

#### 5.6.5.2 rl_DielectricLayer rl_Multilayer::vacuum_ [protected]

the top (vacuum) layer

Definition at line 76 of file rl_Multilayer.h.

Referenced by multilayer_reflect_coef().

#### 5.6.5.3 rl_DielectricLayer* rl_Multilayer::layer_ [protected]

array of dielectric layers

Definition at line 78 of file rl_Multilayer.h.

Referenced by cdump_on(), dump_on(), init(), layer(), multilayer_reflect_coef(), and ~rl_Multilayer().

#### 5.6.5.4 rl_Traits::Bool rl_Multilayer::own_data_ [protected]

do we own the rl_DielectricLayer array?

Definition at line 80 of file rl_Multilayer.h.

Referenced by init(), and ~rl_Multilayer().

The documentation for this class was generated from the following files:

- rl_Multilayer.h
- rl_Multilayer.cc

## 5.7 rl_MultilayerSurface Class Reference

`#include <rl_MultilayerSurface.h>`

Collaboration diagram for rl_MultilayerSurface:

### Public Member Functions

- virtual ~rl_MultilayerSurface ()
- rl_MultilayerSurface (rl_Multilayer &ml, dvm3_Vector &norm)
- dvm3_Vector const & normal_vector () const
- void set_normal (dvm3_Vector const &norm)
- int reflect (rl_Ray &ray)
- rl_ReflectionCoefPOD const & reflection_coefs () const
- rl_DielectricLayer const & layer (int layer_no) const
- std::ostream & dump_on (std::ostream &os, int layer_no, char const pre[ ]="", char const pst[ ]="") const

### 5.7.1 Detailed Description

A class encapsulating a multilayer surface, including surface normal. This class contains the multilayer data and surface normal *by reference*. That is, the rl_Multilayer and dvm3_Vector objects must already exist.

Definition at line 51 of file rl_MultilayerSurface.h.

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 rl_MultilayerSurface::~rl_MultilayerSurface () [virtual]

Destructor.

Definition at line 35 of file rl_MultilayerSurface.cc.

#### 5.7.2.2 rl_MultilayerSurface::rl_MultilayerSurface (rl_Multilayer & *ml*, dvm3_Vector & *norm*) [inline]

Constructor. Constructs multilayer surface information from the provided rl_Multilayer object and normal vector.

**Parameters:**

***ml*** the rl_Multilayer object to be used (must already exist)

***norm*** the surface normal dvm3_Vector to be used (must already exist)

Definition at line 181 of file rl_MultilayerSurface.h.

### 5.7.3 Member Function Documentation

#### 5.7.3.1 dvm3_Vector const & rl_MultilayerSurface::normal_vector () const [inline]

Accessor.

**Returns:**

the surface normal vector at the ray intercept

Definition at line 190 of file rl_MultilayerSurface.h.

#### 5.7.3.2 void rl_MultilayerSurface::set_normal (dvm3_Vector const & *norm*) [inline]

Mutator. Set the surface normal at the ray/surface intercept. Until the full surface intercept apparatus is available, applications need to set the surface normal at the intercept.

**Parameters:**

***norm*** the surface normal vector at the ray intercept

Definition at line 203 of file rl_MultilayerSurface.h.

#### 5.7.3.3 int rl_MultilayerSurface::reflect (rl_Ray & *ray*) [inline]

Mutator. Reflect ray direction vector at this surface

**Parameters:**

***ray*** to be reflected. The ray is assumed to be already at a valid surface intercept point. The ray direction vector and polarization state are updated.

**Returns:**

0 if successful.

Definition at line 213 of file rl_MultilayerSurface.h.

References rl_Multilayer::multilayer_reflect_coef(), and rl_Ray::reflect().

#### 5.7.3.4 rl_ReflectionCoefPOD const & rl_MultilayerSurface::reflection_coefs () const [inline]

Accessor.

**Returns:**

rl_ReflectionCoefPOD holding the complex perpendicular and parallel reflection coefficients.

Definition at line 195 of file rl_MultilayerSurface.h.

#### 5.7.3.5 rl_DielectricLayer const & rl_MultilayerSurface::layer (int *layer_no*) const [inline]

Accessor.

**Parameters:**

***layer_no*** layer number for which the information is requested.

**Returns:**

const reference to the rl_DielectricLayer for layer layer_no.

Definition at line 226 of file rl_MultilayerSurface.h.

References rl_Multilayer::layer().

#### 5.7.3.6 std::ostream & rl_MultilayerSurface::dump_on (std::ostream & *os*, int *layer_no*, char const *pre*[ ] = "", char const *pst*[ ] = "") const [inline]

Dump information about a layer.

**Parameters:**

***os*** output stream

***layer_no*** layer number for which the information is requested.

***pre*** optional char* prefix string.

***pst*** optional char* postfix string.

Definition at line 231 of file rl_MultilayerSurface.h.

References rl_Multilayer::dump_on().

The documentation for this class was generated from the following files:

- rl_MultilayerSurface.h
- rl_MultilayerSurface.cc

## 5.8 rl_Polarization Class Reference

`#include <rl_Polarization.h>`

### Public Types

- typedef rl_Traits::complex complex

  *complex type*

### Public Member Functions

- ~rl_Polarization ()
- rl_Polarization ()
- rl_Polarization (rl_Polarization const &rhs)
- rl_Polarization (rl_PolCSPOD const &cs, dvm3_Vector const &dir)
- void init (rl_PolCSPOD const &cs, dvm3_Vector const &dir)
- void get_PolCSPOD (rl_PolCSPOD &cs, dvm3_Vector const &dir) const
- double intensity () const
- void rotate (rl_Polarization &rotated, dvm3_RotMat const &rot_mtx)
- void derotate (rl_Polarization &derotated, dvm3_RotMat const &rot_mtx)
- void attenuate (double factor)
- void reflect (dvm3_Vector const &normal, dvm3_Vector const &dir_in, dvm3_-Vector const &dir_out, rl_ReflectionCoefPOD const &rflcoef)
- dvm3_Vector const & C_r () const

  *return the real "cosine' polarization vector*

- dvm3_Vector const & C_i () const

  *return the imaginary "cosine' polarization vector*

- dvm3_Vector const & S_r () const

  *return the real "sine' polarization vector*

- dvm3_Vector const & S_i () const

  *return the imaginary "sine' polarization vector*

- std::ostream & print_on (std::ostream &os) const
- void cprint_on (std::FILE *of) const

### 5.8.1 Detailed Description

Encapsulates the polarization state of a ray. The class is a simple class to handle the ray polarization state resolved onto a specific coordinate system. The polarization axes are constructed from a direction vector and the coordinate axes. A coordinate triple is constructed in which one axis is the direction vector.

Definition at line 186 of file rl_Polarization.h.

### 5.8.2 Member Typedef Documentation

#### 5.8.2.1 typedef rl_Traits::complex rl_Polarization::complex

complex type

Definition at line 191 of file rl_Polarization.h.

### 5.8.3 Constructor & Destructor Documentation

#### 5.8.3.1 rl_Polarization::~rl_Polarization () [inline]

Destructor (NON-VIRTUAL)

Definition at line 343 of file rl_Polarization.h.

#### 5.8.3.2 rl_Polarization::rl_Polarization () [inline]

Default constructor; constructs rl_Polarization with INVALID fields.

Definition at line 349 of file rl_Polarization.h.

#### 5.8.3.3 rl_Polarization::rl_Polarization (rl_Polarization const & *rhs*) [inline]

Copy constructor; constructs rl_Polarization with INVALID fields.

**Parameters:**

***rhs*** - rl_Polarization object to be copied.

Definition at line 359 of file rl_Polarization.h.

#### 5.8.3.4 rl_Polarization::rl_Polarization (rl_PolCSPOD const & *cs*, dvm3_Vector const & *dir*) [inline]

Conversion constructor. Convert OSAC-style polarization amplitudes into polarization vector set.

**Parameters:**

***cs*** OSAC-style polarization amplitude

***dir*** ray direction vector

Definition at line 354 of file rl_Polarization.h.

References init().

### 5.8.4 Member Function Documentation

#### 5.8.4.1 void rl_Polarization::init (rl_PolCSPOD const & *cs*, dvm3_Vector const & *dir*)

Initialize the polarization state.

**Parameters:**

***cs*** OSAC-style polarization amplitude

***dir*** ray direction vector

Definition at line 171 of file rl_Polarization.cc.

References rl_PolCSPOD::c2_, and rl_PolCSPOD::s2_.

Referenced by rl_Ray::init_ray(), rl_Polarization(), and rl_Ray::set_polarization().

#### 5.8.4.2 void rl_Polarization::get_PolCSPOD (rl_PolCSPOD & *cs*, dvm3_Vector const & *dir*) const

Evaluate rl_PolCSPOD corresponding to this polarization state.

**Parameters:**

***cs*** OSAC-style polarization amplitude

***dir*** ray direction vector

Definition at line 217 of file rl_Polarization.cc.

References rl_PolCSPOD::c2_, and rl_PolCSPOD::s2_.

Referenced by rl_Ray::get_PolCSPOD().

#### 5.8.4.3 double rl_Polarization::intensity () const

Evaluate intensity.

**Returns:**

intensity (i.e., Stoke's $s_0$).

Definition at line 208 of file rl_Polarization.cc.

Referenced by rl_Ray::intensity().

#### 5.8.4.4 void rl_Polarization::rotate (rl_Polarization & *rotated*, dvm3_RotMat const & *rot_mtx*) [inline]

Rotate polarization state to frame described by rot_mtx

**Parameters:**

***rotated*** output rotated polarization state. The state is now in the frame obtained by applying rot_mtx.

***rot_mtx*** rotation matrix.

Definition at line 368 of file rl_Polarization.h.

References C_i_, C_r_, S_i_, and S_r_.

Referenced by rl_Ray::translate_rotate().

#### 5.8.4.5 void rl_Polarization::derotate (rl_Polarization & *derotated*, dvm3_RotMat const & *rot_mtx*) [inline]

Rotate polarization state back from frame described by rot_mtx

**Parameters:**

***derotated*** output derotated polarization state. The state is now back in the original frame.

***rot_mtx*** rotation matrix.

Definition at line 378 of file rl_Polarization.h.

References C_i_, C_r_, S_i_, and S_r_.

Referenced by rl_Ray::derotate_detranslate().

#### 5.8.4.6 void rl_Polarization::attenuate (double *factor*)

Attenuate reflectivity by a factor.

**Parameters:**

***factor*** attenuation factor. (NOTE: this multiplies the polarization components by sqrt(factor).)

Definition at line 243 of file rl_Polarization.cc.

Referenced by rl_Ray::attenuate().

#### 5.8.4.7 void rl_Polarization::reflect (dvm3_Vector const & *normal*, dvm3_Vector const & *dir_in*, dvm3_Vector const & *dir_out*, rl_ReflectionCoefPOD const & *rflcoef*)

Evaluate the reflected polarization vectors. Given a surface normal, incident and reflected ray direction vectors, and the parallel and perpendicular reflection coefficients, evaluate the reflected polarization vectors.

**Parameters:**

***normal*** surface normal vector.

***dir_in*** direction vector for incoming ray.

***dir_out*** direction vector for reflected ray.

***rflcoef*** complex reflection coefficients for surface.

Definition at line 262 of file rl_Polarization.cc.

References rl_ReflectionCoefPOD::para(), and rl_ReflectionCoefPOD::perp().

Referenced by rl_Ray::reflect().

#### 5.8.4.8 dvm3_Vector const & rl_Polarization::C_r () const [inline]

return the real "cosine' polarization vector

Definition at line 388 of file rl_Polarization.h.

#### 5.8.4.9 dvm3_Vector const & rl_Polarization::C_i () const [inline]

return the imaginary "cosine' polarization vector

Definition at line 395 of file rl_Polarization.h.

#### 5.8.4.10 dvm3_Vector const & rl_Polarization::S_r () const [inline]

return the real “sine’ polarization vector

Definition at line 402 of file rl_Polarization.h.

#### 5.8.4.11 dvm3_Vector const & rl_Polarization::S_i () const [inline]

return the imaginary “sine’ polarization vector

Definition at line 409 of file rl_Polarization.h.

#### 5.8.4.12 std::ostream & rl_Polarization::print_on (std::ostream & *os*) const

Write the polarization vectors to stream os. The state is written as "[x y z][x y z][x y z][x y z]".

**Parameters:**

***os*** output stream.

Definition at line 339 of file rl_Polarization.cc.

Referenced by rl_Ray::print_on().

#### 5.8.4.13 void rl_Polarization::cprint_on (std::FILE * *of*) const

Write the polarization amplitude to FILE* of. The state is written as "[x y z][x y z][x y z][x y z]".

**Parameters:**

***of*** output FILE*.

The documentation for this class was generated from the following files:

- rl_Polarization.h
- rl_Polarization.cc

## 5.9 rl_PolCSPOD Struct Reference

`#include <rl_Polarization.h>`

### Public Member Functions

- void init (double b_over_a=1.0, double psi=0.0)

- void attenuate (double factor)
- double intensity () const
- std::ostream & print_on (std::ostream &os) const
- void cprint_on (FILE *of) const

**Public Attributes**

- rl_Traits::complex c2_ [2]

  *polarization amplitude (cosine component)*

- rl_Traits::complex s2_ [2]

  *polarization amplitude (sine component)*

### 5.9.1 Detailed Description

A Plain Ol' Data struct (POD) encapsulating the OSAC-style complex polarization amplitudes. This is a utility POD to simplify communication with BPipe.

Relation to OSAC polarization amplitudes:

`c2_.q1` $\Longleftrightarrow$ `c2comp(1)`, `c2_.q2` $\Longleftrightarrow$ `c2comp(2)` `s2_.q1` $\Longleftrightarrow$ `s2comp(1)`, `s2_.q2` $\Longleftrightarrow$ `s2comp(2)`

For random polarization:

$$\texttt{wt} = \texttt{c2.q1.r}^2+\texttt{c2.q2.r}^2+\texttt{c2.q1.i}^2+\texttt{c2.q2.i}^2+\texttt{s2.q1.r}^2+\texttt{s2.q2.r}^2+\texttt{s2.q1.i}^2+\texttt{s2.q2.i}^2$$

or

$$\texttt{wt} = |\texttt{c2}[0]|^2 + |\texttt{s2}[0]|^2$$

For discrete polarization:

$$\texttt{wt} = \texttt{c2.q1.r}^2+\texttt{c2.q2.r}^2+\texttt{c2.q1.i}^2+\texttt{c2.q2.i}^2+\texttt{s2.q1.r}^2+\texttt{s2.q2.r}^2+\texttt{s2.q1.i}^2+\texttt{s2.q2.i}^2+2\left(\texttt{c2.q1.is2.q1.r} + \texttt{c2.c}\right.$$

or

$$\texttt{wt} = |\texttt{c2}[0]|^2 + |\texttt{s2}[0]|^2 + 2\left(\texttt{c2}[0]\texttt{s2}^*[0] + \texttt{c2}[1]\texttt{s2}^*[1]\right)$$

where the $^*$ superscript denotes a complex conjugate.

Definition at line 106 of file rl_Polarization.h.

### 5.9.2 Member Function Documentation

#### 5.9.2.1 void rl_PolCSPOD::init (double *b_over_a* = `1.0`, double *psi* = `0.0`)

Initialization method.

**Parameters:**

***b_over_a*** ratio of semiminor axis to semi-major axis.

***psi*** angle of the major axis from the +X axis, with psi increasing towards +Y

Definition at line 76 of file rl_Polarization.cc.

References c2_, and s2_.

#### 5.9.2.2 void rl_PolCSPOD::attenuate (double *factor*)

Attenuate intensity by a factor.

**Parameters:**

***factor*** the attenuation factor; the ray intensity is multiplied by factor.

Definition at line 121 of file rl_Polarization.cc.

References c2_, and s2_.

#### 5.9.2.3 double rl_PolCSPOD::intensity () const

Evaluate the intensity.

**Returns:**

the intensity (i.e., Stoke's $s_0$.

Definition at line 99 of file rl_Polarization.cc.

References c2_, and s2_.

Referenced by cprint_on().

#### 5.9.2.4 std::ostream & rl_PolCSPOD::print_on (std::ostream & *os*) const

Write the polarization amplitude to stream os. The state is written out as "[(r,i)(r,i)][(r,i)(r,i)]".

**Parameters:**

***os*** output stream.

Definition at line 135 of file rl_Polarization.cc.

References c2_, and s2_.

#### 5.9.2.5 void rl_PolCSPOD::cprint_on (FILE * *of*) const

Write the polarization amplitude to FILE* of. The state is written out as "[(r,i)(r,i)]\n[(r,i)(r,i)]".

**Parameters:**

***of*** output FILE*.

Definition at line 149 of file rl_Polarization.cc.

References c2_, intensity(), and s2_.

### 5.9.3 Member Data Documentation

#### 5.9.3.1 rl_Traits::complex rl_PolCSPOD::c2_[2]

polarization amplitude (cosine component)

Definition at line 109 of file rl_Polarization.h.

Referenced by attenuate(), cprint_on(), rl_Polarization::get_PolCSPOD(), rl_Polarization::init(), init(), intensity(), and print_on().

#### 5.9.3.2 rl_Traits::complex rl_PolCSPOD::s2_[2]

polarization amplitude (sine component)

Definition at line 111 of file rl_Polarization.h.

Referenced by attenuate(), cprint_on(), rl_Polarization::get_PolCSPOD(), rl_Polarization::init(), init(), intensity(), and print_on().

The documentation for this struct was generated from the following files:

- rl_Polarization.h
- rl_Polarization.cc

## 5.10 rl_Ray Class Reference

`#include <rl_Ray.h>`

Collaboration diagram for rl_Ray:

### Public Member Functions

- virtual ~rl_Ray ()
- rl_Ray ()

- rl_Ray (rl_BasicRay const &ray, rl_PolCSPOD const &cspol)
- rl_Ray (dvm3_Vector const &pos, dvm3_Vector const &dir, double energy, long int id, rl_PolCSPOD const &cspol)
- void init_ray (dvm3_Vector const &pos, dvm3_Vector const &dir, double energy, long int id, rl_PolCSPOD const &cspol)
- rl_Polarization const & polarization () const
- void set_polarization (rl_PolCSPOD const &cspol)
- void get_PolCSPOD (rl_PolCSPOD &cs) const
- double intensity () const
- void attenuate (double by_how_much)
- void reflect (dvm3_Vector const &normal, rl_ReflectionCoefPOD const &rflcoef)
- void translate_rotate (dvm3_Vector const &trans, dvm3_RotMat const &rotmat)
- void derotate_detranslate (dvm3_Vector const &trans, dvm3_RotMat const &rotmat)
- std::ostream & print_on (std::ostream &os, char const pre[ ]="", char const pst[ ]="") const

### 5.10.1 Detailed Description

An rl_BasicRay with added polarization information.

Definition at line 45 of file rl_Ray.h.

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 rl_Ray::~rl_Ray () [virtual]

Destructor

Definition at line 41 of file rl_Ray.cc.

#### 5.10.2.2 rl_Ray::rl_Ray () [inline]

Default constructor. Constructs a ray in an INVALID state; use init_ray to initialize the fields.

Definition at line 193 of file rl_Ray.h.

#### 5.10.2.3 rl_Ray::rl_Ray (rl_BasicRay const & *ray*, rl_PolCSPOD const & *cspol*) [inline]

Constructor. Construct a ray given an rl_BasicRay and a polarization state.

Definition at line 198 of file rl_Ray.h.

#### 5.10.2.4 rl_Ray::rl_Ray (dvm3_Vector const & *pos*, dvm3_Vector const & *dir*, double *energy*, long int *id*, rl_PolCSPOD const & *cspol*) [inline]

Constructor. Construct a ray.

**Parameters:**

***pos*** ray position vector.

***dir*** ray direction unit vector.

***energy*** ray energy (keV).

***id*** a numeric ray identifier.

***cspol*** an OSAC-style complex polarization amplitude vector.

Definition at line 203 of file rl_Ray.h.

### 5.10.3 Member Function Documentation

#### 5.10.3.1 void rl_Ray::init_ray (dvm3_Vector const & *pos*, dvm3_Vector const & *dir*, double *energy*, long int *id*, rl_PolCSPOD const & *cspol*) [inline]

Initialize a ray.

**Parameters:**

***pos*** ray position vector.

***dir*** ray direction unit vector.

***energy*** ray energy (keV).

***id*** a numeric ray identifier.

***cspol*** an OSAC-style complex polarization amplitude vector.

Definition at line 211 of file rl_Ray.h.

References rl_Polarization::init().

#### 5.10.3.2 rl_Polarization const & rl_Ray::polarization () const [inline]

Return the polarization state.

**Returns:**

polarization state

Definition at line 224 of file rl_Ray.h.

#### 5.10.3.3 void rl_Ray::set_polarization (rl_PolCSPOD const & *cspol*) [inline]

Set the polarization state.

**Parameters:**

***cspol*** an OSAC-style complex polarization amplitude vector.

Definition at line 242 of file rl_Ray.h.

References rl_Polarization::init().

#### 5.10.3.4 void rl_Ray::get_PolCSPOD (rl_PolCSPOD & *cs*) const [inline]

Evaluate rl_PolCSPOD corresponding to this ray's polarization state.

**Parameters:**

***cs*** return an OSAC-style complex polarization amplitude vector.

Definition at line 229 of file rl_Ray.h.

References rl_Polarization::get_PolCSPOD().

#### 5.10.3.5 double rl_Ray::intensity () const [inline]

Returns this ray's normalized intensity (i.e., "weight")

**Returns:**

this ray's normalized intensity (i.e., "weight")

Definition at line 234 of file rl_Ray.h.

References rl_Polarization::intensity().

#### 5.10.3.6 void rl_Ray::attenuate (double *by_how_much*) [inline]

Attenuate this ray by by_how_much

**Parameters:**

***by_how_much*** how much to attenuate this ray.

Definition at line 247 of file rl_Ray.h.

References rl_Polarization::attenuate().

#### 5.10.3.7 void rl_Ray::reflect (dvm3_Vector const & *normal*, rl_ReflectionCoefPOD const & *rflcoef*)

Reflect this ray's direction vector and polarization state.

**Parameters:**

***normal*** surface normal unit vector.

***rflcoef*** complex reflectances for the surface.

Definition at line 51 of file rl_Ray.cc.

References rl_Polarization::reflect().

Referenced by rl_MultilayerSurface::reflect().

#### 5.10.3.8 void rl_Ray::translate_rotate (dvm3_Vector const & *trans*, dvm3_RotMat const & *rotmat*)

Translate to BCS origin and rotate from STD to BCS coordinates.

**Parameters:**

***trans*** translation vector

***rotmat*** rotation matrix to be applied. rotmat specifies a rotation from std to bcs.

Definition at line 63 of file rl_Ray.cc.

References rl_Polarization::rotate().

#### 5.10.3.9 void rl_Ray::derotate_detranslate (dvm3_Vector const & *trans*, dvm3_RotMat const & *rotmat*)

Derotate back to std coordinates; detranslate back to std origin

**Parameters:**

***trans*** translation vector.

***rotmat*** rotation matrix to be applied. rotmat specifies a rotation from std to bcs; the inverse of rotmat is applied to the ray.

Definition at line 75 of file rl_Ray.cc.

References rl_Polarization::derotate().

#### 5.10.3.10 std::ostream & rl_Ray::print_on (std::ostream & *os*, char const *pre*[ ] = "", char const *pst*[ ] = "") const

Write the ray contents with optional pre and post comment strings.

**Parameters:**

***os*** output stream.

***pre*** optional prefix c-string.

***pst*** optional postfix c-string.

Definition at line 86 of file rl_Ray.cc.

References rl_Polarization::print_on().

The documentation for this class was generated from the following files:

- rl_Ray.h
- rl_Ray.cc

## 5.11 rl_ReflectionCoefPOD Class Reference

`#include <rl_ReflectionCoefPOD.h>`

### Public Member Functions

- double reflectivity (double polarization_factor=0.0) const
- void init ()
- void init (rl_Traits::complex &para, rl_Traits::complex &perp)
- rl_Traits::complex para () const
- rl_Traits::complex & para ()
- rl_Traits::complex perp () const
- rl_Traits::complex & perp ()
- std::ostream & print_on (std::ostream &os, char const pre[ ]="", char const pst[ ]="") const
- void cprint_on (std::FILE *of, char const pre[ ]="", char const pst[ ]="") const

### 5.11.1 Detailed Description

A Plain Ol' Data class representing complex reflection coefficients.

Definition at line 76 of file rl_ReflectionCoefPOD.h.

### 5.11.2 Member Function Documentation

#### 5.11.2.1 double rl_ReflectionCoefPOD::reflectivity (double *polarization_factor* = `0.0`) const `[inline]`

evaluate the reflectivity.

**Parameters:**

***polarization_factor*** polarization factor; it must be a value between -1 and 1. The polarization factor is related to parallel (p) and perpendicular (s) polarization by:

$$\texttt{polarization\_factor} = \frac{(I_\perp - I_\parallel)}{(I_\perp + I_\parallel)}$$

or

$$\texttt{polarization\_factor} = \frac{(I_s - I_p)}{(I_s + I_p)}$$

where $I_\perp$ and $I_\parallel$ are the perpendicular and parallel E-field *intensities*, respectively. Thus,

- -1: pure parallel (p) polarization.
- 0: completely unpolarized.
- +1: pure perpendicular (s) polarization.

**Returns:**

const reference to layer layer_no

Definition at line 203 of file rl_ReflectionCoefPOD.h.

Referenced by rl_Multilayer::multilayer_reflectivity(), and rl_DielectricLayer::reflectivity().

#### 5.11.2.2 void rl_ReflectionCoefPOD::init () `[inline]`

initialize perpendicular (s) and parallel (p) reflection coefficients to zero.

Definition at line 172 of file rl_ReflectionCoefPOD.h.

Referenced by rl_Multilayer::multilayer_reflectivity().

#### 5.11.2.3 void rl_ReflectionCoefPOD::init (rl_Traits::complex & *para*, rl_Traits::complex & *perp*) `[inline]`

initialize perpendicular (s) and parallel (p) reflection coefficients. to zero.

**Parameters:**

***para*** parallel reflection coefficient.

***perp*** perpendicular reflection coefficient.

Definition at line 179 of file rl_ReflectionCoefPOD.h.

#### 5.11.2.4 rl_Traits::complex rl_ReflectionCoefPOD::para () const [inline]

**Returns:**

parallel (p) reflection coefficient.

Definition at line 187 of file rl_ReflectionCoefPOD.h.

Referenced by rl_DielectricLayer::cprint_constraints_on(), cprint_on(), rl_Polarization::reflect(), rl_DielectricLayer::reflect_amp(), and rl_-DielectricLayer::reflect_nlayer().

#### 5.11.2.5 rl_Traits::complex & rl_ReflectionCoefPOD::para () [inline]

**Returns:**

parallel (p) reflection coefficient (read/write access).

Definition at line 195 of file rl_ReflectionCoefPOD.h.

#### 5.11.2.6 rl_Traits::complex rl_ReflectionCoefPOD::perp () const [inline]

**Returns:**

perpendicular (p) reflection coefficient.

Definition at line 191 of file rl_ReflectionCoefPOD.h.

Referenced by rl_DielectricLayer::cprint_constraints_on(), cprint_on(), rl_Polarization::reflect(), rl_DielectricLayer::reflect_amp(), and rl_-DielectricLayer::reflect_nlayer().

#### 5.11.2.7 rl_Traits::complex & rl_ReflectionCoefPOD::perp () [inline]

**Returns:**

perpendicular (p) reflection coefficient (read/write access).

Definition at line 199 of file rl_ReflectionCoefPOD.h.

#### 5.11.2.8 std::ostream & rl_ReflectionCoefPOD::print_on (std::ostream & *os*, char const *pre*[ ] = "", char const *pst*[ ] = "") const [inline]

Print reflectivity information to output stream.

**Parameters:**

***os*** output stream.

***pre*** optional prefix (char∗) string.

***pst*** optional postfix (char∗) string.

Definition at line 210 of file rl_ReflectionCoefPOD.h.

#### 5.11.2.9 void rl_ReflectionCoefPOD::cprint_on (std::FILE ∗ *of*, char const *pre*[ ] = "", char const *pst*[ ] = "") const [inline]

Print reflectivity information to output FILE∗ stream.

**Parameters:**

***of*** output FILE∗ stream.

***pre*** optional prefix (char∗) string.

***pst*** optional postfix (char∗) string.

Definition at line 219 of file rl_ReflectionCoefPOD.h.

References para(), and perp().

Referenced by rl_DielectricLayer::cdump_on().

The documentation for this class was generated from the following file:

- rl_ReflectionCoefPOD.h

## 5.12 rl_Traits Class Reference

`#include <rl_Traits.h>`

### Public Types

- enum Bool { **False**, **True** }

  *Typedef for the Boolean type.*

- enum EInterpMode { ELinLin, ELinLog, ELogLin, ELogLog }

- enum ERoughType {
  ERoughNone, ERoughDebyeWaller_RSAO, ERoughDebyeWaller_CSAO, ERoughDebyeWaller_Spiller,
  ERoughModifiedDebyeWaller, ERoughNevotCroce }
- typedef std::complex< double > complex
  *Typedef for the complex type.*

**Classes**

- struct rl_DielectricPOD

### 5.12.1 Detailed Description

rl_Traits is a "traits" class for the rl_RayLib library. It defines typedefs (e.g., abstracting out the complex class) and enums used in the library. It also declares a Plain Old Data (POD) struct to encapsulate the dielectric constant data.

Definition at line 55 of file rl_Traits.h.

### 5.12.2 Member Typedef Documentation

#### 5.12.2.1 typedef std::complex<double> rl_Traits::complex

Typedef for the complex type.

Definition at line 61 of file rl_Traits.h.

### 5.12.3 Member Enumeration Documentation

#### 5.12.3.1 enum rl_Traits::Bool

Typedef for the Boolean type.

Definition at line 64 of file rl_Traits.h.

#### 5.12.3.2 enum rl_Traits::EInterpMode

Enumeration specifying the interpolation of the optical constants.

**Enumerator:**

***ELinLin*** linear in energy, linear in optical constants.

***ELinLog*** log in energy, linear in optical constants.

***ELogLin*** linear in energy, log in optical constants.

***ELogLog*** log in energy, log in optical constants.

Definition at line 69 of file rl_Traits.h.

#### 5.12.3.3 enum rl_Traits::ERoughType

Enumeration specifying the type of interlayer diffusion treatment

**Enumerator:**

***ERoughNone*** no interlayer diffusion
***ERoughDebyeWaller_RSAO*** Debye-Waller factor
***ERoughDebyeWaller_CSAO*** Debye-Waller factor
***ERoughDebyeWaller_Spiller*** Debye-Waller factor
***ERoughModifiedDebyeWaller*** Modified Debye-Waller factor
***ERoughNevotCroce*** Nevot-Croce factor

Definition at line 80 of file rl_Traits.h.

The documentation for this class was generated from the following file:

- rl_Traits.h

## 5.13 rl_Traits::rl_DielectricPOD Struct Reference

`#include <rl_Traits.h>`

### Public Attributes

- double energy_
  *energy (keV)*
- double alpha_
  *dielectric decrement, real part*
- double gamma_
  *dielectric decrement, imag part*

### 5.13.1 Detailed Description

Plain Ol' Data: dielectric data (alpha, gamma) as a function of energy.

Definition at line 93 of file rl_Traits.h.

### 5.13.2 Member Data Documentation

#### 5.13.2.1 double rl_Traits::rl_DielectricPOD::energy_

energy (keV)

Definition at line 96 of file rl_Traits.h.

Referenced by rl_DielectricData::alpha_gamma(), rl_DielectricPODArray::init(), and rl_DielectricData::init().

#### 5.13.2.2 double rl_Traits::rl_DielectricPOD::alpha_

dielectric decrement, real part

Definition at line 98 of file rl_Traits.h.

Referenced by rl_DielectricData::alpha_gamma(), rl_DielectricPODArray::init(), and rl_DielectricData::init().

#### 5.13.2.3 double rl_Traits::rl_DielectricPOD::gamma_

dielectric decrement, imag part

Definition at line 100 of file rl_Traits.h.

Referenced by rl_DielectricData::alpha_gamma(), rl_DielectricPODArray::init(), and rl_DielectricData::init().

The documentation for this struct was generated from the following file:

- rl_Traits.h

## 5.14 rl_TransmissionCoefPOD Class Reference

`#include <rl_TransmissionCoefPOD.h>`

### Public Member Functions

- void init ()
- void init (rl_Traits::complex &para, rl_Traits::complex &perp)
- double transmission (double polarization_factor=0.0) const
- rl_Traits::complex para () const
- rl_Traits::complex & para ()
- rl_Traits::complex perp () const
- rl_Traits::complex & perp ()
- std::ostream & print_on (std::ostream &os, char const pre[ ]="", char const pst[ ]="") const

- void cprint_on (std::FILE *of, char const pre[ ]="", char const pst[ ]="") const

### 5.14.1 Detailed Description

A Plain Ol' Data class representing complex reflection coefficients.

Definition at line 75 of file rl_TransmissionCoefPOD.h.

### 5.14.2 Member Function Documentation

#### 5.14.2.1 void rl_TransmissionCoefPOD::init () [inline]

Initialize perpendicular (s) and parallel (p) transmission coefficients to zero.

Definition at line 171 of file rl_TransmissionCoefPOD.h.

#### 5.14.2.2 void rl_TransmissionCoefPOD::init (rl_Traits::complex & *para*, rl_Traits::complex & *perp*) [inline]

Iinitialize perpendicular (s) and parallel (p) transmission coefficients.

**Parameters:**

***para*** parallel transmission coefficient

***perp*** perpendicular transmission coefficient

Definition at line 178 of file rl_TransmissionCoefPOD.h.

#### 5.14.2.3 double rl_TransmissionCoefPOD::transmission (double *polarization_factor* = 0.0) const [inline]

Transmission factor.

**Parameters:**

***polarization_factor*** polarization factor; it must be a value between -1 and 1. The polarization factor is related to parallel (p) and perpendicular (s) polarization by:

$$\texttt{polarization\_factor} = \frac{(I_\perp - I_\parallel)}{(I_\perp + I_\parallel)}$$

or

$$\texttt{polarization\_factor} = \frac{(I_s - I_p)}{(I_s + I_p)}$$

where $I_\perp$ and $I_\parallel$ are the perpendicular and parallel E-field *intensities*, respectively. Thus,

- -1: pure parallel (p) polarization.
- 0: completely unpolarized.
- +1: pure perpendicular (s) polarization.

**Returns:**

transmission factor

Definition at line 202 of file rl_TransmissionCoefPOD.h.

#### 5.14.2.4 rl_Traits::complex rl_TransmissionCoefPOD::para () const [inline]

**Returns:**

parallel (p) transmission coefficient.

Definition at line 186 of file rl_TransmissionCoefPOD.h.

Referenced by rl_DielectricLayer::cprint_constraints_on(), and rl_DielectricLayer::reflect_amp().

#### 5.14.2.5 rl_Traits::complex & rl_TransmissionCoefPOD::para () [inline]

**Returns:**

parallel (p) transmission coefficient (read/write access).

Definition at line 194 of file rl_TransmissionCoefPOD.h.

#### 5.14.2.6 rl_Traits::complex rl_TransmissionCoefPOD::perp () const [inline]

**Returns:**

perpendicular (s) transmission coefficient.

Definition at line 190 of file rl_TransmissionCoefPOD.h.

Referenced by rl_DielectricLayer::cprint_constraints_on(), and rl_DielectricLayer::reflect_amp().

#### 5.14.2.7 rl_Traits::complex & rl_TransmissionCoefPOD::perp () [inline]

**Returns:**

perpendicular (s) transmission coefficient (read/write access).

Definition at line 198 of file rl_TransmissionCoefPOD.h.

#### 5.14.2.8 std::ostream & rl_TransmissionCoefPOD::print_on (std::ostream & *os*, char const *pre*[ ] = "", char const *pst*[ ] = "") const [inline]

Print reflectivity information to output stream.

**Parameters:**

***os*** output stream.

***pre*** optional prefix (char∗) string.

***pst*** optional postfix (char∗) string.

Definition at line 209 of file rl_TransmissionCoefPOD.h.

#### 5.14.2.9 void rl_TransmissionCoefPOD::cprint_on (std::FILE ∗ *of*, char const *pre*[ ] = "", char const *pst*[ ] = "") const [inline]

Print reflectivity information to output FILE∗ stream.

**Parameters:**

***of*** output FILE∗ stream.

***pre*** optional prefix (char∗) string.

***pst*** optional postfix (char∗) string.

Definition at line 218 of file rl_TransmissionCoefPOD.h.

The documentation for this class was generated from the following file:

- rl_TransmissionCoefPOD.h

# Index