My Project

Generated by Doxygen 1.8.11

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Module Documentation

5.1 Abstract Classes

Classes

- class c2_function< float_type >

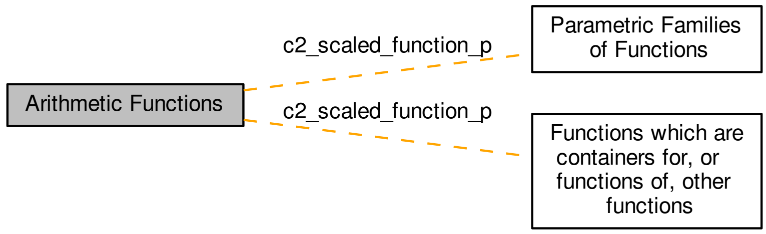
  the parent class for all c2_functions.
  c2_functions know their value, first, and second derivative at almost every point. They can be efficiently combined with binary operators, via c2_binary_function, composed via c2_composed_function_, have their roots found via find_root(), and be adaptively integrated via partial_integrals() or integral(). They also can carry information with them about how to find 'interesting' points on the function. This information is set with set_sampling_grid() and extracted with get_sampling_grid().
- class c2_binary_function< float_type >

  Provides support for c2_function objects which are constructed from two other c2_function objects.

5.1.1 Detailed Description

5.2 Arithmetic Functions

Collaboration diagram for Arithmetic Functions:



Classes

- class c2_scaled_function_p< float_type >

  *Create a very lightweight method to return a scalar multiple of another function. \ *
  The factory function c2_factory::scaled_function() creates ∗new c2_scaled_function_p.
- class c2_composed_function_p< float_type >

  Provides function composition (nesting)
  This allows evaluation of f(g(x)) where f and g are c2_function objects.
- class c2_sum_p< float_type >

  create a c2_function which is the sum of two other c2_function objects.
  This should always be constructed using c2_function::operator+()
- class c2_diff_p< float_type >

  create a c2_function which is the difference of two other c2_functions.
  This should always be constructed using c2_function::operator-()
- class c2_product_p< float_type >

  create a c2_function which is the product of two other c2_functions.
  This should always be constructed using c2_function::operator∗()
- class c2_ratio_p< float_type >

  create a c2_function which is the ratio of two other c2_functions.
  This should always be constructed using c2_function::operator/()

5.2.1 Detailed Description

5.3 Mathemetical Functions

Classes

- class c2_sin_p< float_type >

  compute sin(x) with its derivatives.
  The factory function c2_factory::sin() creates ∗new c2_sin_p
- class c2_cos_p< float_type >

  compute cos(x) with its derivatives.
  The factory function c2_factory::cos() creates ∗new c2_cos_p
- class c2_tan_p< float_type >

  compute tan(x) with its derivatives.
  The factory function c2_factory::tan() creates ∗new c2_tan_p
- class c2_log_p< float_type >

  compute log(x) with its derivatives.
  The factory function c2_factory::log() creates ∗new c2_log_p
- class c2_exp_p< float_type >

  compute exp(x) with its derivatives.
  The factory function c2_factory::exp() creates ∗new c2_exp_p
- class c2_sqrt_p< float_type >

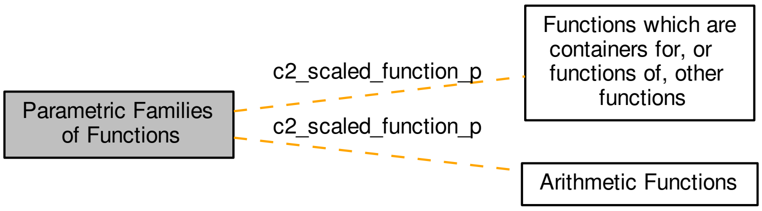
  compute sqrt(x) with its derivatives.
  The factory function c2_factory::sqrt() creates ∗new c2_sqrt_p()
- class c2_identity_p< float_type >

  compute x with its derivatives.
  The factory function c2_factory::identity() creates ∗new c2_identity_p

5.3.1 Detailed Description

5.4 Parametric Families of Functions

Collaboration diagram for Parametric Families of Functions:



Classes

- class c2_scaled_function_p< float_type >

  *Create a very lightweight method to return a scalar multiple of another function. \ \
  The factory function c2_factory::scaled_function() creates ∗new c2_scaled_function_p.*
- class c2_constant_p< float_type >

  *a c2_function which is constant
  The factory function c2_factory::constant() creates ∗new c2_constant_p()*
- class c2_recip_p< float_type >

  *compute scale/x with its derivatives.
  The factory function c2_factory::recip() creates ∗new c2_recip_p*
- class c2_linear_p< float_type >

  *create a linear mapping of another function
  for example, given a c2_function f*
- class c2_quadratic_p< float_type >

  *create a quadratic mapping of another function
  for example, given a c2_function f*
- class c2_power_law_p< float_type >

  *create a power law mapping of another function
  for example, given a c2_function f*
- class c2_connector_function_p< float_type >

  *create a c2_function which smoothly connects two other c2_functions.
  This takes two points and generates a polynomial which matches two c2_function arguments at those two points, with two derivatives at each point, and an arbitrary value at the center of the region. It is useful for splicing together functions over rough spots (0/0, for example).*

5.4.1 Detailed Description

5.5 Interpolating Functions

Classes

- class interpolating_function_p< float_type >

  create a cubic spline interpolation of a set of (x,y) pairs
  This is one of the main reasons for c2_function objects to exist.
- class log_lin_interpolating_function_p< float_type >

  A spline with X transformed into log space.
  Most useful for functions looking like y=log(x) or any other function with a huge X dynamic range, and a slowly varying Y.
- class lin_log_interpolating_function_p< float_type >

  A spline with Y transformed into log space.
  Most useful for functions looking like y=exp(x)
- class log_log_interpolating_function_p< float_type >

  A spline with X and Y transformed into log space.
  Most useful for functions looking like y=x^n or any other function with a huge X and Y dynamic range.
- class arrhenius_interpolating_function_p< float_type >

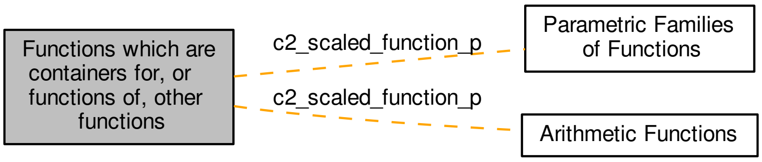
  A spline with X in reciprocal space and Y transformed in log space.
  *Most useful for thermodynamic types of data where Y is roughly A*exp(-B/x). Typical examples are reaction rate data, and thermistor calibration data.*
- class accumulated_histogram< float_type >

  An interpolating_function_p which is the cumulative integral of a histogram.
  Note than binedges should be one element longer than binheights, since the lower & upper edges are specified. Note that this is a malformed spline, since the second derivatives are all zero, so it has less continuity. Also, note that the bin edges can be given in backwards order to generate the reversed accumulation (starting at the high end)

5.5.1 Detailed Description

5.6 Functions which are containers for, or functions of, other functions

Collaboration diagram for Functions which are containers for, or functions of, other functions:



Classes

- class c2_classic_function_p< float_type >

  a container into which any conventional c-style function can be dropped, to create a degenerate c2_function without derivatives. Mostly useful for sampling into interpolating functions. construct a reference to this with c2_classic_↩ function()
  The factory function c2_factory::classic_function() creates ∗new c2_classic_function_p()
- class c2_const_ptr< float_type >

  create a container for a c2_function which handles the reference counting.
  It is useful as a smart container to hold a c2_function and keep the reference count correct. The recommended way for a class to store a c2_function which is handed in from the outside is for it to have a c2_ptr member into which the passed-in function is stored. This way, when the class instance is deleted, it will automatically dereference any function which it was handed.
- class c2_ptr< float_type >

  create a container for a c2_function which handles the reference counting.
- class c2_typed_ptr< float_type, c2_class >

  create a non-generic container for a c2_function which handles the reference counting.
- class c2_plugin_function_p< float_type >

  a container into which any other c2_function can be dropped, to allow expressions with replacable components.
  It is useful for plugging different InterpolatingFunctions into a c2_function expression. It saves a lot of effort in other places with casting away const declarations.
- class c2_const_plugin_function_p< float_type >

  a c2_plugin_function_p which promises not to fiddle with the plugged function.
  The factory function c2_factory::const_plugin_function() creates ∗new c2_const_plugin_function_p()
- class c2_scaled_function_p< float_type >

  *Create a very lightweight method to return a scalar multiple of another function. \ *
  The factory function c2_factory::scaled_function() creates ∗new c2_scaled_function_p.
- class c2_cached_function_p< float_type >

  A container into which any other c2_function can be dropped.
  It allows a function to be pre-evaluated at a point, and used at multiple places in an expression efficiently. If it is re-evaluated at the previous point, it returns the remembered values; otherwise, it re-evauates the function at the new point.
- class c2_inverse_function_p< float_type >

  create the formal inverse function of another function
  for example, given a c2_function f
- class c2_piecewise_function_p< float_type >

  create a c2_function which is a piecewise assembly of other c2_functions.
  The functions must have increasing, non-overlapping domains. Any empty space between functions will be filled with a linear interpolation.

5.6.1 Detailed Description

5.7 Factory classes which reduce silly template typing

Classes

- class c2_factory< float_type >

    a factory of pre-templated c2_function generators

5.7.1 Detailed Description

5.8 Classes which provide coordinate system transformations, wih derivatives

Classes

- class c2_transformation< float_type >

  a transformation of a coordinate, including an inverse
- class c2_transformation_linear< float_type >

  the identity transform
- class c2_transformation_log< float_type >

  log axis transform
- class c2_transformation_recip< float_type >

  reciprocal axis transform
- class c2_function_transformation< float_type >

  a transformation of a function in and out of a coordinate space, using 2 c2_transformations
- class c2_lin_lin_function_transformation< float_type >

  a transformation of a function in and out of lin-lin space
- class c2_log_log_function_transformation< float_type >

  a transformation of a function in and out of log-log space
- class c2_lin_log_function_transformation< float_type >

  a transformation of a function in and out of lin-log space
- class c2_log_lin_function_transformation< float_type >

  a transformation of a function in and out of log-lin space
- class c2_arrhenius_function_transformation< float_type >

  a transformation of a function in and out of Arrhenius (1/x vs. log(y)) space

5.8.1 Detailed Description

Chapter 6

Class Documentation

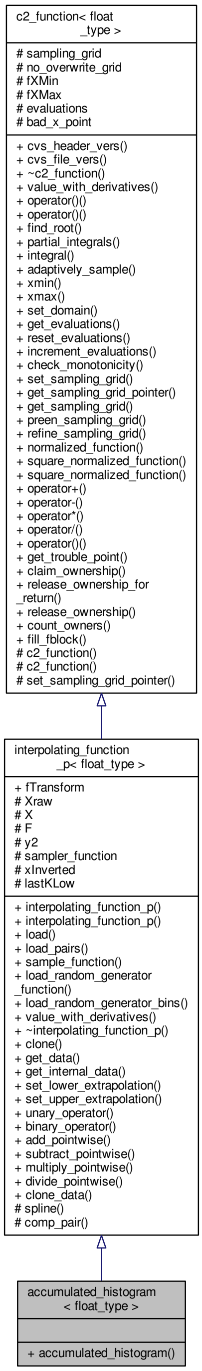
6.1 accumulated_histogram$<$ float_type $>$ Class Template Reference

An interpolating_function_p which is the cumulative integral of a histogram.

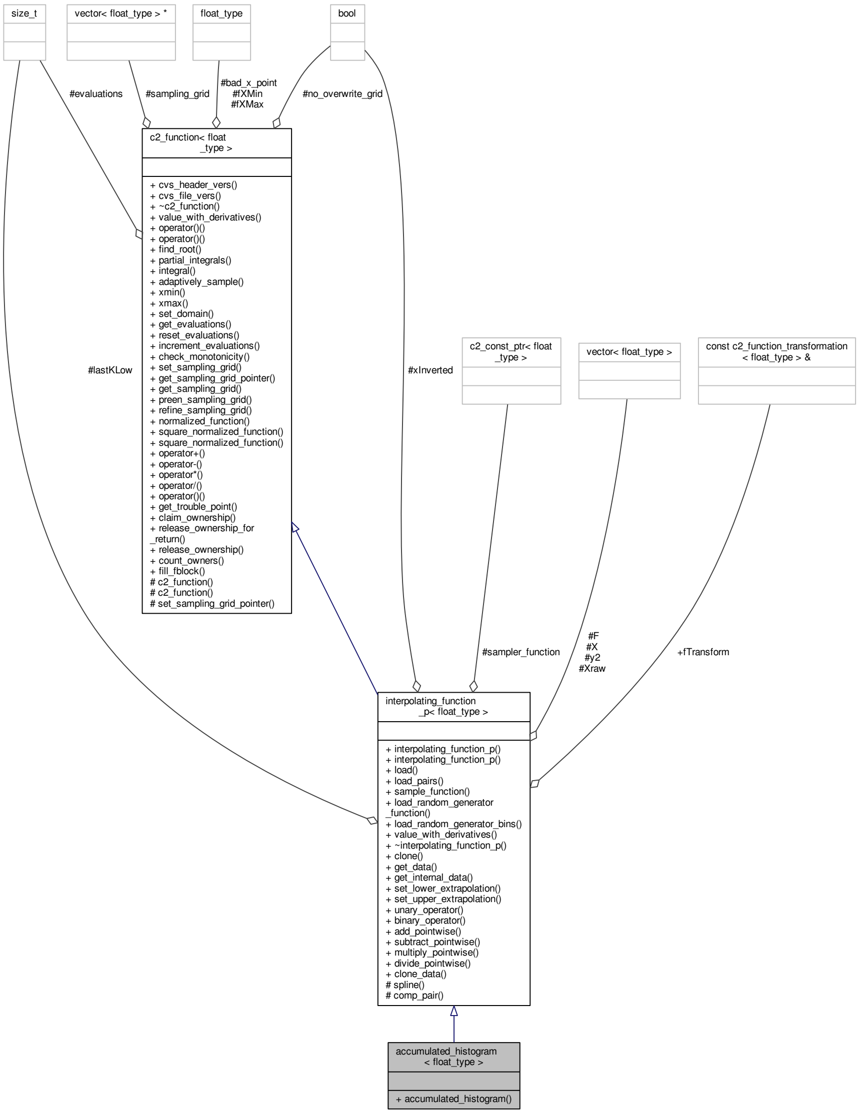
Note than binedges should be one element longer than binheights, since the lower & upper edges are specified. Note that this is a malformed spline, since the second derivatives are all zero, so it has less continuity. Also, note that the bin edges can be given in backwards order to generate the reversed accumulation (starting at the high end)

```
#include "c2_function.hh"
```

Inheritance diagram for accumulated_histogram< float_type >:



Collaboration diagram for accumulated_histogram< float_type >:



Public Member Functions

- accumulated_histogram (const std::vector< float_type >binedges, const std::vector< float_type > binheights, bool normalize=false, bool inverse_function=false, bool drop_zeros=true)

  Construct the integrated histogram.

Additional Inherited Members

6.1.1 Detailed Description

template<typename float_type = double>
class accumulated_histogram< float_type >

An interpolating_function_p which is the cumulative integral of a histogram.

Note than binedges should be one element longer than binheights, since the lower & upper edges are specified. Note that this is a malformed spline, since the second derivatives are all zero, so it has less continuity. Also, note that the bin edges can be given in backwards order to generate the reversed accumulation (starting at the high end)

6.1.2 Constructor & Destructor Documentation

6.1.2.1 template<typename float_type = double> accumulated_histogram< float_type >::accumulated_histogram (const std::vector< float_type > *binedges*, const std::vector< float_type > *binheights*, bool *normalize* = `false`, bool *inverse_function* = `false`, bool *drop_zeros* = `true`)

Construct the integrated histogram.

Parameters

binedges	the edges of the bins in *binheights*. It should have one more element than *binheights*
binheights	the number of counts in each bin.
normalize	if true, normalize integral to 1
inverse_function	if true, drop zero channels, and return inverse function for random generation
drop_zeros	eliminate null bins before integrating, so integral is strictly monotonic.

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

- c2_function.hh

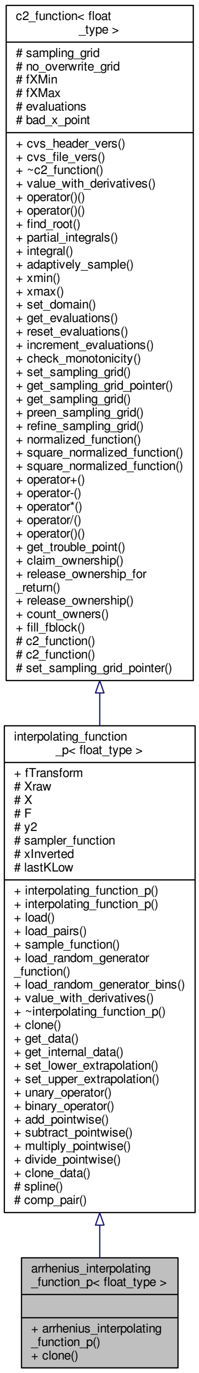
6.2 arrhenius_interpolating_function_p< float_type > Class Template Reference

A spline with X in reciprocal space and Y transformed in log space.

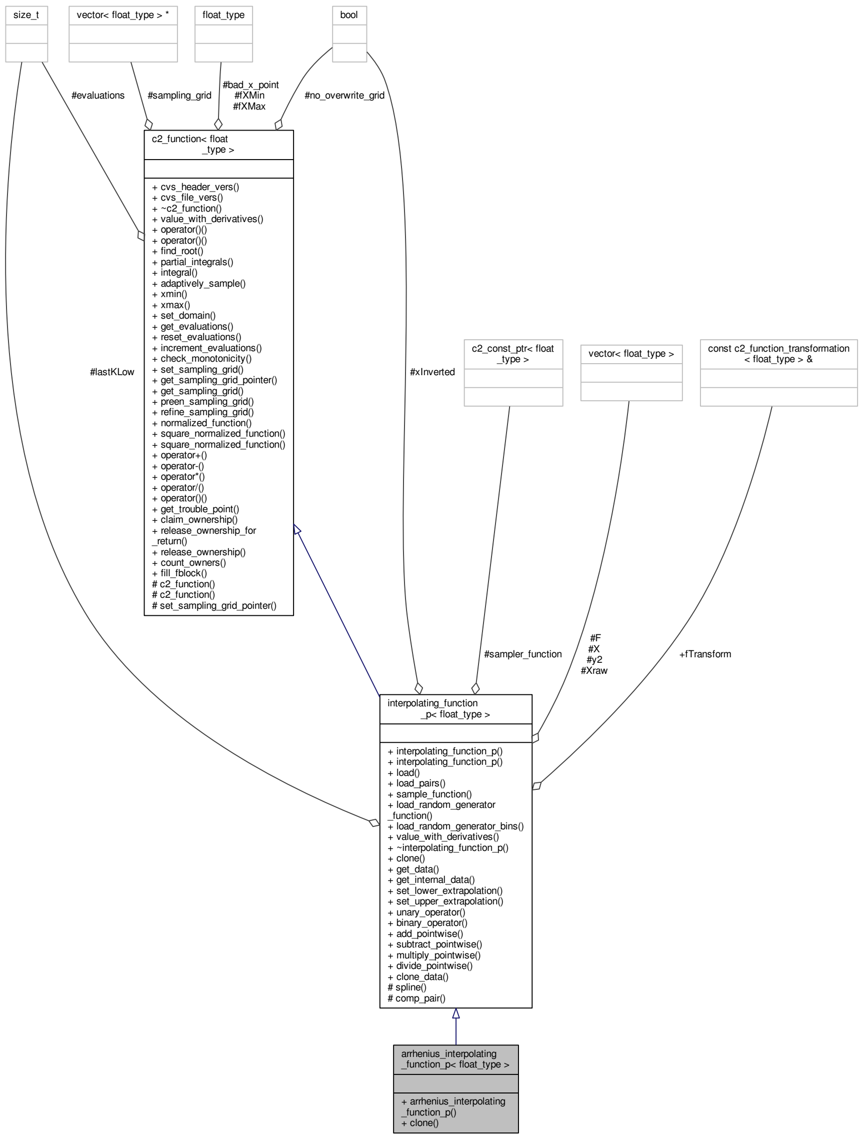
Most useful for thermodynamic types of data where Y is roughly A∗exp(-B/x). Typical examples are reaction rate data, and thermistor calibration data.

```
#include "c2_function.hh"
```

Inheritance diagram for arrhenius_interpolating_function_p< float_type >:



Collaboration diagram for arrhenius_interpolating_function_p< float_type >:



Public Member Functions

- arrhenius_interpolating_function_p ()

  an empty arrhenius cubic-spline interpolating_function_p
- virtual interpolating_function_p< float_type > & clone () const throw (c2_exception)

  create a new, empty interpolating function of this type (virtual constructor)

Additional Inherited Members

6.2.1 Detailed Description

template<typename float_type = double>
class arrhenius_interpolating_function_p< float_type >

A spline with X in reciprocal space and Y transformed in log space.

Most useful for thermodynamic types of data where Y is roughly A∗exp(-B/x). Typical examples are reaction rate data, and thermistor calibration data.

The factory function c2_factory::arrhenius_interpolating_function() creates ∗new arrhenius_interpolating_function↩_p()

6.2.2 Constructor & Destructor Documentation

6.2.2.1 template<typename float_type = double> arrhenius_interpolating_function_p< float_type >::arrhenius_interpolating_function_p ( ) [inline]

an empty arrhenius cubic-spline interpolating_function_p

6.2.3 Member Function Documentation

6.2.3.1 template<typename float_type = double> virtual interpolating_function_p<float_type>& arrhenius_interpolating_function_p< float_type >::clone ( ) const throw c2_exception) [inline], [virtual]

create a new, empty interpolating function of this type (virtual constructor)

Reimplemented from interpolating_function_p< float_type >.

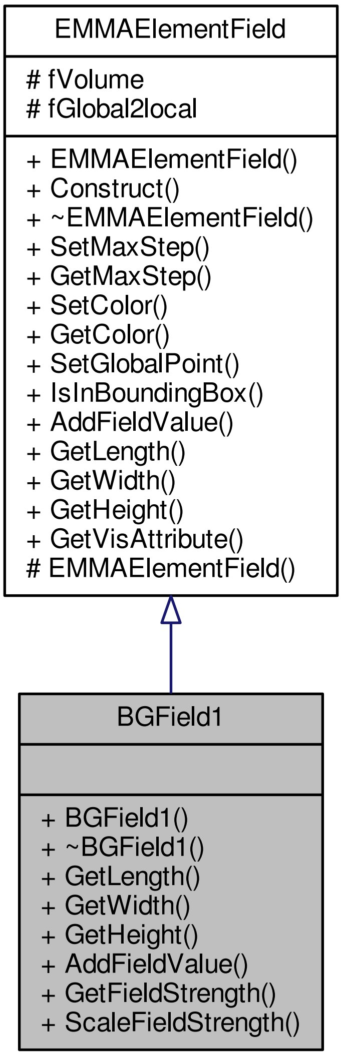
The documentation for this class was generated from the following file:

- c2_function.hh

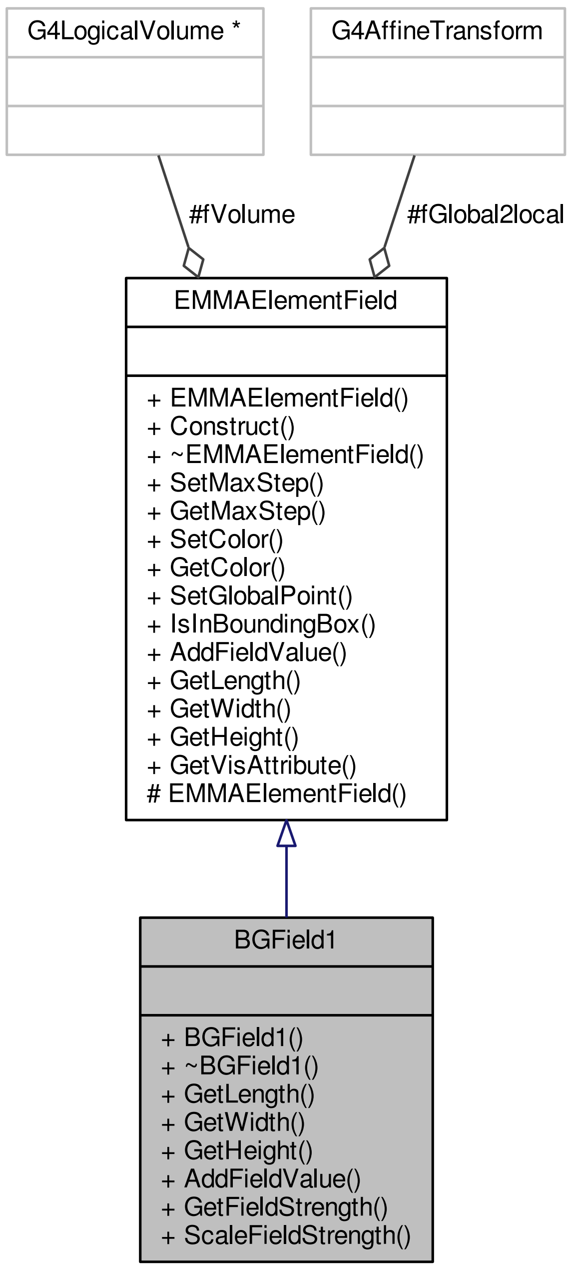
6.3 BGField1 Class Reference

```
#include "BGField1.hh"
```

Inheritance diagram for BGField1:



Collaboration diagram for BGField1:



Public Member Functions

- BGField1 (G4double xoffset, G4double zoffset, G4double zbefore, G4double zafter, G4LogicalVolume $*$, G4ThreeVector)
- ∼BGField1 ()
- virtual G4double GetLength ()
- virtual G4double GetWidth ()

- virtual G4double GetHeight ()
- virtual void AddFieldValue (const G4double Point[3], G4double field[6]) const
- G4double GetFieldStrength ()
- void ScaleFieldStrength (G4double msf)

Additional Inherited Members

6.3.1 Constructor & Destructor Documentation

6.3.1.1 BGField1::BGField1 (G4double *xoffset*, G4double *zoffset*, G4double *zbefore*, G4double *zafter*, G4LogicalVolume ∗ , G4ThreeVector)

6.3.1.2 BGField1::∼BGField1 ( )

6.3.2 Member Function Documentation

6.3.2.1 virtual void BGField1::AddFieldValue (const G4double *Point[3]*, G4double *field[6]*) const [virtual]

6.3.2.2 G4double BGField1::GetFieldStrength ( ) [inline]

6.3.2.3 virtual G4double BGField1::GetHeight ( ) [inline], [virtual]

Implements EMMAElementField.

6.3.2.4 virtual G4double BGField1::GetLength ( ) [inline], [virtual]

Implements EMMAElementField.

6.3.2.5 virtual G4double BGField1::GetWidth ( ) [inline], [virtual]

Implements EMMAElementField.

6.3.2.6 void BGField1::ScaleFieldStrength (G4double *msf*) [inline]

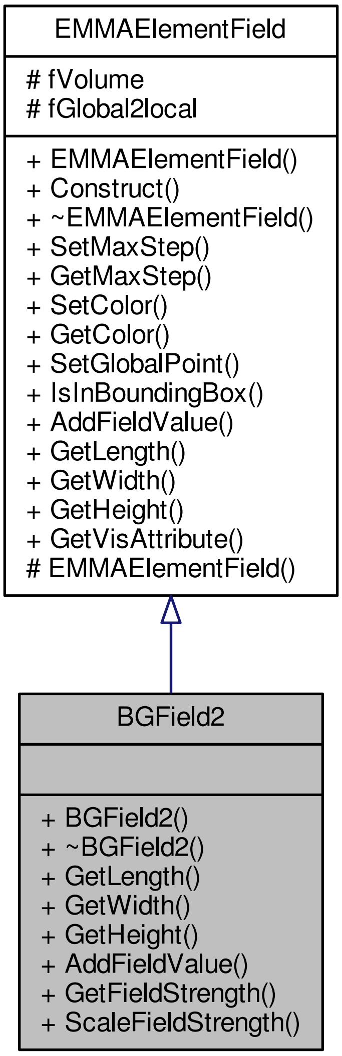
The documentation for this class was generated from the following file:

- BGField1.hh

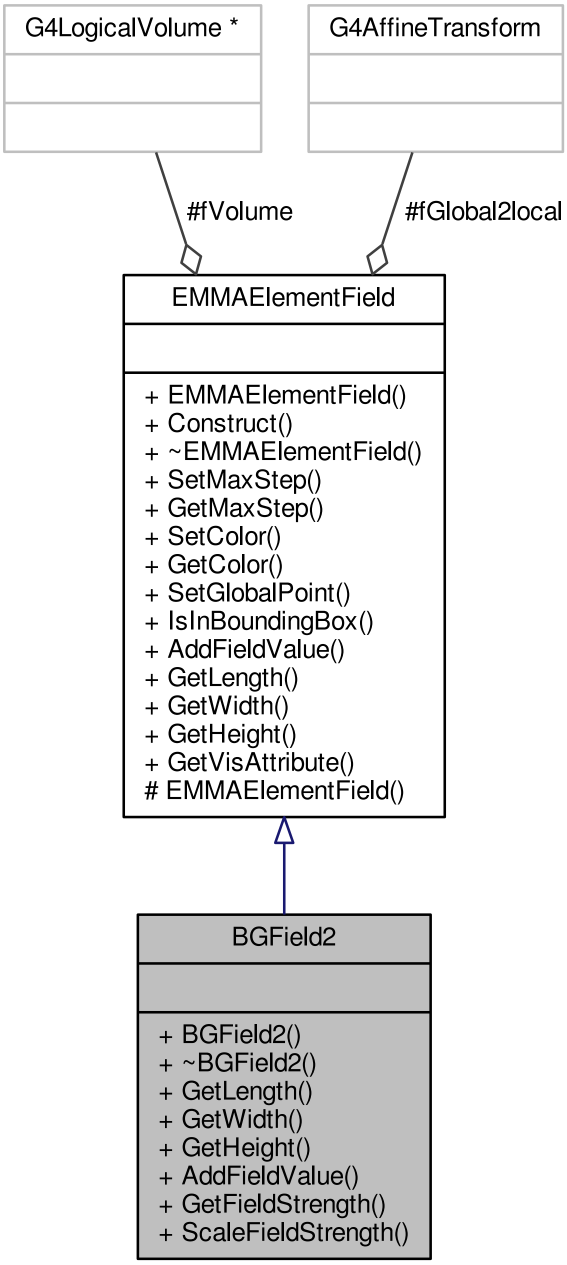
6.4 BGField2 Class Reference

```
#include "BGField2.hh"
```

Inheritance diagram for BGField2:



Collaboration diagram for BGField2:



Public Member Functions

- BGField2 (G4double xoffset, G4double zoffset, G4double zbefore, G4double zafter, G4LogicalVolume $*$, G4ThreeVector)
- $\sim$BGField2 ()
- virtual G4double GetLength ()
- virtual G4double GetWidth ()

- virtual G4double GetHeight ()
- virtual void AddFieldValue (const G4double Point[3], G4double field[6]) const
- G4double GetFieldStrength ()
- void ScaleFieldStrength (G4double msf)

Additional Inherited Members

6.4.1 Constructor & Destructor Documentation

6.4.1.1 BGField2::BGField2 (G4double *xoffset,* G4double *zoffset,* G4double *zbefore,* G4double *zafter,* G4LogicalVolume ∗ , G4ThreeVector)

6.4.1.2 BGField2::∼BGField2 ( )

6.4.2 Member Function Documentation

6.4.2.1 virtual void BGField2::AddFieldValue (const G4double *Point[3],* G4double *field[6]*) const [virtual]

6.4.2.2 G4double BGField2::GetFieldStrength ( ) [inline]

6.4.2.3 virtual G4double BGField2::GetHeight ( ) [inline], [virtual]

Implements EMMAElementField.

6.4.2.4 virtual G4double BGField2::GetLength ( ) [inline], [virtual]

Implements EMMAElementField.

6.4.2.5 virtual G4double BGField2::GetWidth ( ) [inline], [virtual]

Implements EMMAElementField.

6.4.2.6 void BGField2::ScaleFieldStrength (G4double *msf*) [inline]

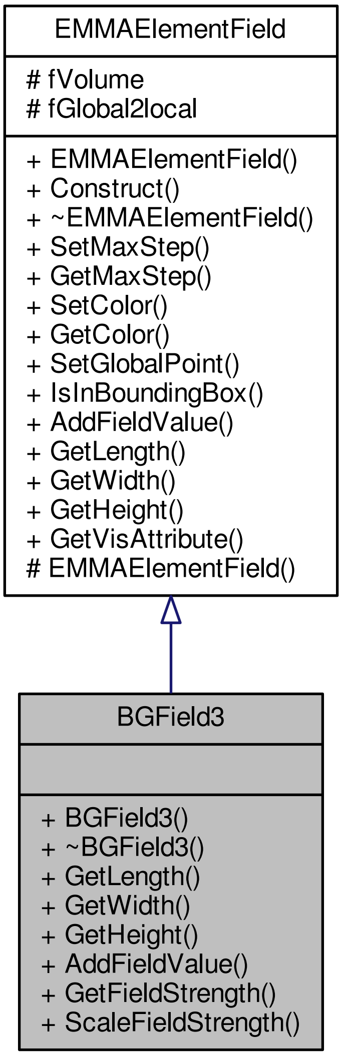
The documentation for this class was generated from the following file:

- BGField2.hh

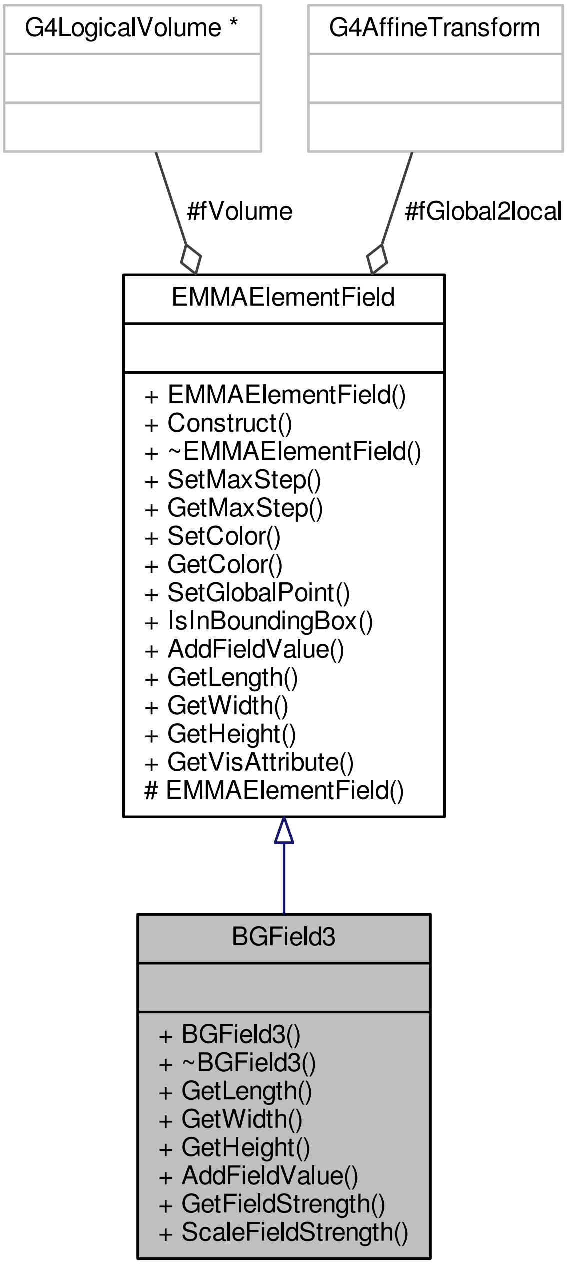
6.5 BGField3 Class Reference

```
#include "BGField3.hh"
```

Inheritance diagram for BGField3:



Collaboration diagram for BGField3:



Public Member Functions

- BGField3 (G4double xoffset, G4double zoffset, G4double zbefore, G4double zafter, G4LogicalVolume ∗, G4ThreeVector)
- ∼BGField3 ()
- virtual G4double GetLength ()
- virtual G4double GetWidth ()

- virtual G4double GetHeight ()
- virtual void AddFieldValue (const G4double Point[3], G4double field[6]) const
- G4double GetFieldStrength ()
- void ScaleFieldStrength (G4double esf)

Additional Inherited Members

6.5.1 Constructor & Destructor Documentation

6.5.1.1 BGField3::BGField3 (G4double *xoffset,* G4double *zoffset,* G4double *zbefore,* G4double *zafter,* G4LogicalVolume ∗ , G4ThreeVector)

6.5.1.2 BGField3::∼BGField3 ( )

6.5.2 Member Function Documentation

6.5.2.1 virtual void BGField3::AddFieldValue (const G4double *Point[3],* G4double *field[6]*) const [virtual]

6.5.2.2 G4double BGField3::GetFieldStrength ( ) [inline]

6.5.2.3 virtual G4double BGField3::GetHeight ( ) [inline], [virtual]

Implements EMMAElementField.

6.5.2.4 virtual G4double BGField3::GetLength ( ) [inline], [virtual]

Implements EMMAElementField.

6.5.2.5 virtual G4double BGField3::GetWidth ( ) [inline], [virtual]

Implements EMMAElementField.

6.5.2.6 void BGField3::ScaleFieldStrength (G4double *esf*) [inline]

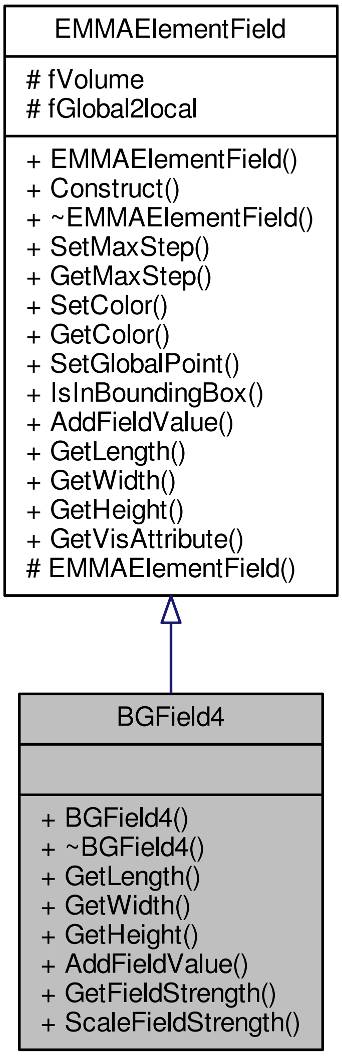
The documentation for this class was generated from the following file:

- BGField3.hh

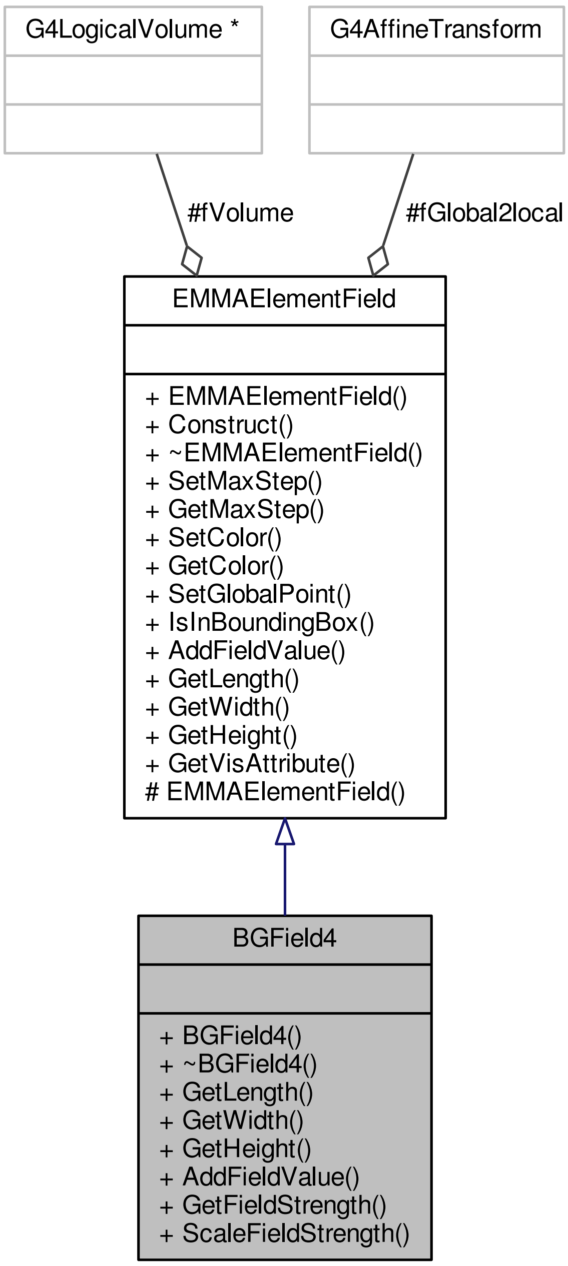
6.6 BGField4 Class Reference

```
#include "BGField4.hh"
```

Inheritance diagram for BGField4:



Collaboration diagram for BGField4:



Public Member Functions

- BGField4 (G4double xoffset, G4double zoffset, G4double zbefore, G4double zafter, G4LogicalVolume ∗, G4ThreeVector)
- ∼BGField4 ()
- virtual G4double GetLength ()
- virtual G4double GetWidth ()

- virtual G4double GetHeight ()
- virtual void AddFieldValue (const G4double Point[3], G4double field[6]) const
- G4double GetFieldStrength ()
- void ScaleFieldStrength (G4double msf)

Additional Inherited Members

6.6.1 Constructor & Destructor Documentation

6.6.1.1 BGField4::BGField4 (G4double *xoffset,* G4double *zoffset,* G4double *zbefore,* G4double *zafter,* G4LogicalVolume ∗ , G4ThreeVector)

6.6.1.2 BGField4::∼BGField4 ( )

6.6.2 Member Function Documentation

6.6.2.1 virtual void BGField4::AddFieldValue (const G4double *Point[3],* G4double *field[6]*) const [virtual]

6.6.2.2 G4double BGField4::GetFieldStrength ( ) [inline]

6.6.2.3 virtual G4double BGField4::GetHeight ( ) [inline], [virtual]

Implements EMMAElementField.

6.6.2.4 virtual G4double BGField4::GetLength ( ) [inline], [virtual]

Implements EMMAElementField.

6.6.2.5 virtual G4double BGField4::GetWidth ( ) [inline], [virtual]

Implements EMMAElementField.

6.6.2.6 void BGField4::ScaleFieldStrength (G4double *msf*) [inline]

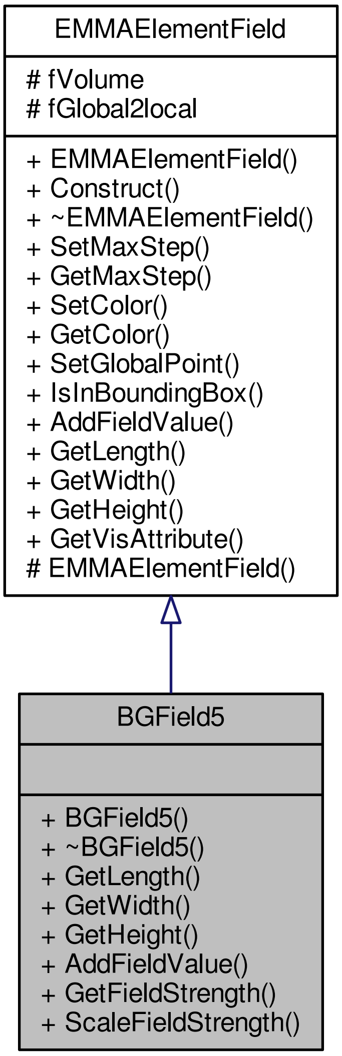
The documentation for this class was generated from the following file:

- BGField4.hh

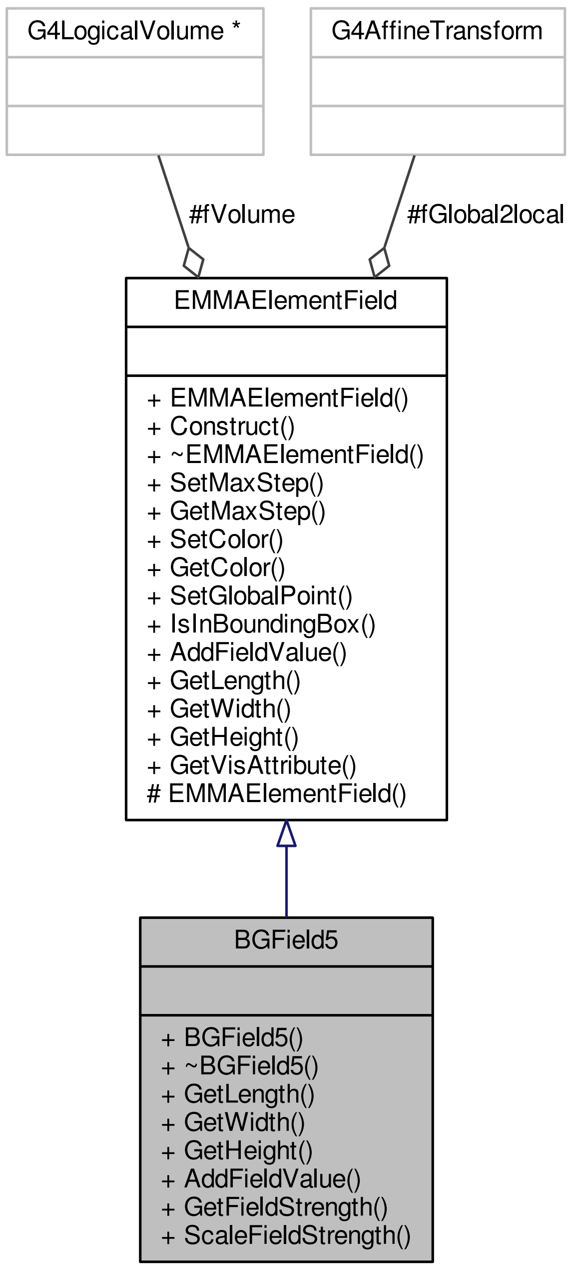
6.7 BGField5 Class Reference

```
#include "BGField5.hh"
```

Inheritance diagram for BGField5:



Collaboration diagram for BGField5:



Public Member Functions

- BGField5 (G4double xoffset, G4double zoffset, G4double zbefore, G4double zafter, G4LogicalVolume ∗, G4ThreeVector)
- ∼BGField5 ()
- virtual G4double GetLength ()
- virtual G4double GetWidth ()

- virtual G4double GetHeight ()
- virtual void AddFieldValue (const G4double Point[3], G4double field[6]) const
- G4double GetFieldStrength ()
- void ScaleFieldStrength (G4double esf)

Additional Inherited Members

6.7.1 Constructor & Destructor Documentation

6.7.1.1 BGField5::BGField5 (G4double *xoffset*, G4double *zoffset*, G4double *zbefore*, G4double *zafter*, G4LogicalVolume ∗ , G4ThreeVector)

6.7.1.2 BGField5::∼BGField5 ( )

6.7.2 Member Function Documentation

6.7.2.1 virtual void BGField5::AddFieldValue (const G4double *Point[3]*, G4double *field[6]*) const [virtual]

6.7.2.2 G4double BGField5::GetFieldStrength ( ) [inline]

6.7.2.3 virtual G4double BGField5::GetHeight ( ) [inline], [virtual]

Implements EMMAElementField.

6.7.2.4 virtual G4double BGField5::GetLength ( ) [inline], [virtual]

Implements EMMAElementField.

6.7.2.5 virtual G4double BGField5::GetWidth ( ) [inline], [virtual]

Implements EMMAElementField.

6.7.2.6 void BGField5::ScaleFieldStrength (G4double *esf*) [inline]

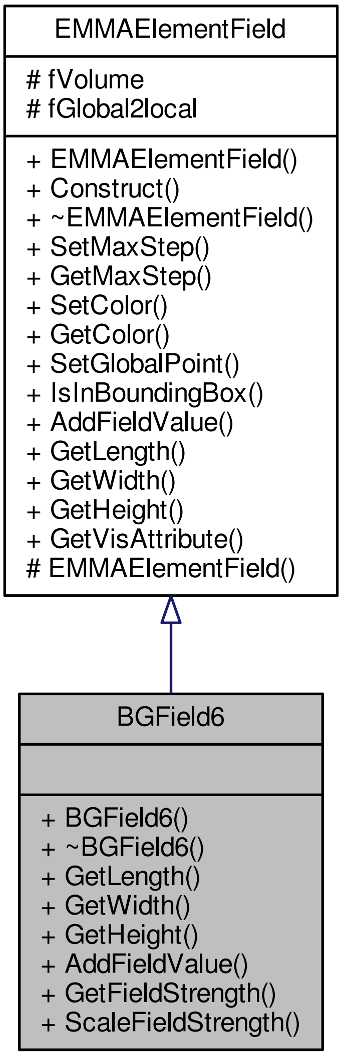
The documentation for this class was generated from the following file:

- BGField5.hh

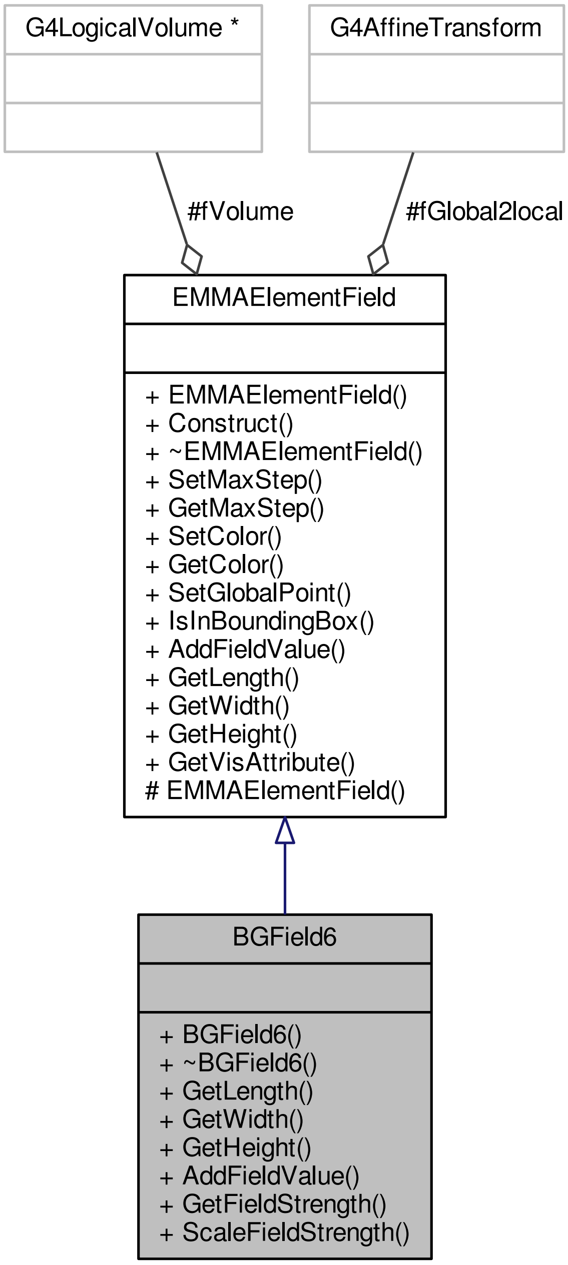
6.8 BGField6 Class Reference

```
#include "BGField6.hh"
```

Inheritance diagram for BGField6:



Collaboration diagram for BGField6:



Public Member Functions

- BGField6 (G4double xoffset, G4double zoffset, G4double zbefore, G4double zafter, G4LogicalVolume $*$, G4ThreeVector)
- ∼BGField6 ()
- virtual G4double GetLength ()
- virtual G4double GetWidth ()

- virtual G4double GetHeight ()
- virtual void AddFieldValue (const G4double Point[3], G4double field[6]) const
- G4double GetFieldStrength ()
- void ScaleFieldStrength (G4double msf)

Additional Inherited Members

6.8.1 Constructor & Destructor Documentation

6.8.1.1 BGField6::BGField6 (G4double *xoffset*, G4double *zoffset*, G4double *zbefore*, G4double *zafter*, G4LogicalVolume ∗ , G4ThreeVector)

6.8.1.2 BGField6::∼BGField6 ( )

6.8.2 Member Function Documentation

6.8.2.1 virtual void BGField6::AddFieldValue (const G4double *Point[3]*, G4double *field[6]*) const [virtual]

6.8.2.2 G4double BGField6::GetFieldStrength ( ) [inline]

6.8.2.3 virtual G4double BGField6::GetHeight ( ) [inline], [virtual]

Implements EMMAElementField.

6.8.2.4 virtual G4double BGField6::GetLength ( ) [inline], [virtual]

Implements EMMAElementField.

6.8.2.5 virtual G4double BGField6::GetWidth ( ) [inline], [virtual]

Implements EMMAElementField.

6.8.2.6 void BGField6::ScaleFieldStrength (G4double *msf*) [inline]

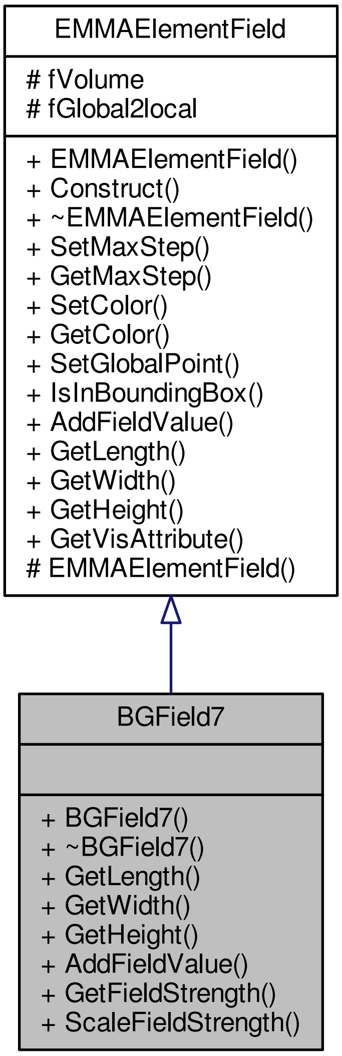
The documentation for this class was generated from the following file:

- BGField6.hh

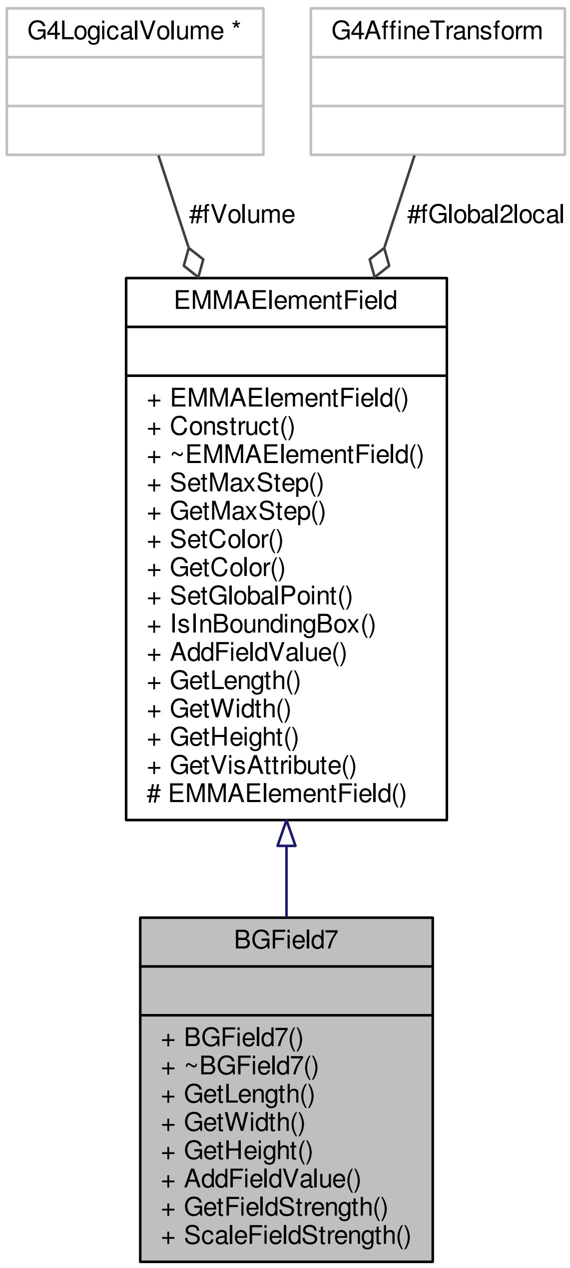
6.9 BGField7 Class Reference

```
#include "BGField7.hh"
```

Inheritance diagram for BGField7:



Collaboration diagram for BGField7:



Public Member Functions

- BGField7 (G4double xoffset, G4double zoffset, G4double zbefore, G4double zafter, G4LogicalVolume $*$, G4ThreeVector)
- $\sim$BGField7 ()
- virtual G4double GetLength ()
- virtual G4double GetWidth ()

- virtual G4double GetHeight ()
- virtual void AddFieldValue (const G4double Point[3], G4double field[6]) const
- G4double GetFieldStrength ()
- void ScaleFieldStrength (G4double msf)

Additional Inherited Members

6.9.1 Constructor & Destructor Documentation

6.9.1.1 BGField7::BGField7 (G4double *xoffset,* G4double *zoffset,* G4double *zbefore,* G4double *zafter,* G4LogicalVolume ∗ , G4ThreeVector)

6.9.1.2 BGField7::∼BGField7 ( )

6.9.2 Member Function Documentation

6.9.2.1 virtual void BGField7::AddFieldValue (const G4double *Point[3],* G4double *field[6]*) const [virtual]

6.9.2.2 G4double BGField7::GetFieldStrength ( ) [inline]

6.9.2.3 virtual G4double BGField7::GetHeight ( ) [inline], [virtual]

Implements EMMAElementField.

6.9.2.4 virtual G4double BGField7::GetLength ( ) [inline], [virtual]

Implements EMMAElementField.

6.9.2.5 virtual G4double BGField7::GetWidth ( ) [inline], [virtual]

Implements EMMAElementField.

6.9.2.6 void BGField7::ScaleFieldStrength (G4double *msf*) [inline]

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

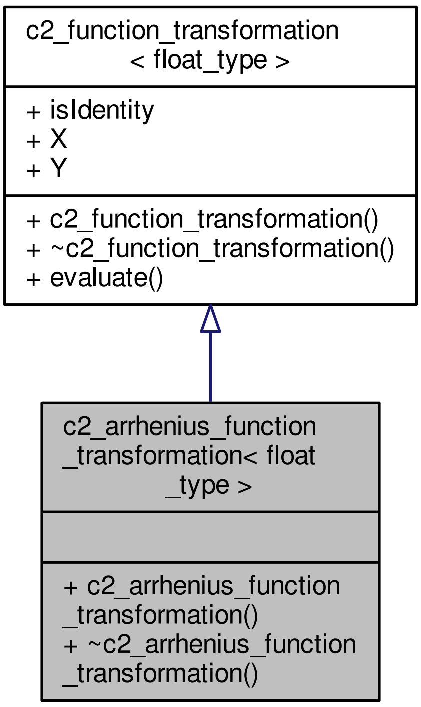
- BGField7.hh

6.10 c2_arrhenius_function_transformation< float_type > Class Template Reference

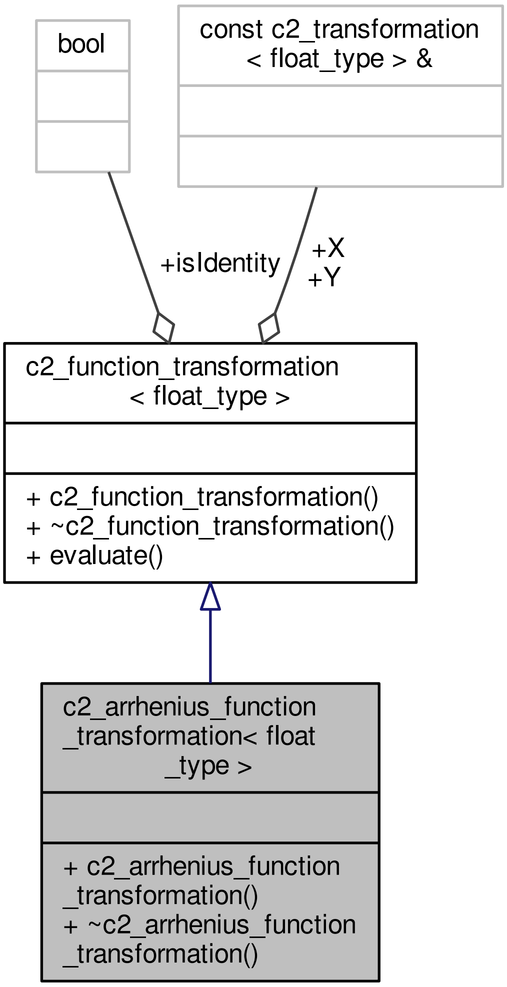
a transformation of a function in and out of Arrhenius (1/x vs. log(y)) space

```
#include "c2_function.hh"
```

Inheritance diagram for c2_arrhenius_function_transformation< float_type >:



Collaboration diagram for c2_arrhenius_function_transformation< float_type >:



Public Member Functions

- c2_arrhenius_function_transformation ()
- virtual ∼c2_arrhenius_function_transformation ()

Additional Inherited Members

6.10.1 Detailed Description

template<typename float_type>
class c2_arrhenius_function_transformation< float_type >

a transformation of a function in and out of Arrhenius (1/x vs. log(y)) space

6.10.2 Constructor & Destructor Documentation

6.10.2.1 template<typename float_type > c2_arrhenius_function_transformation< float_type >::c2_arrhenius_function_transformation ( ) [inline]

6.10.2.2 template<typename float_type > virtual c2_arrhenius_function_transformation< float_type >::~c2_arrhenius_function_transformation ( ) [inline], [virtual]

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

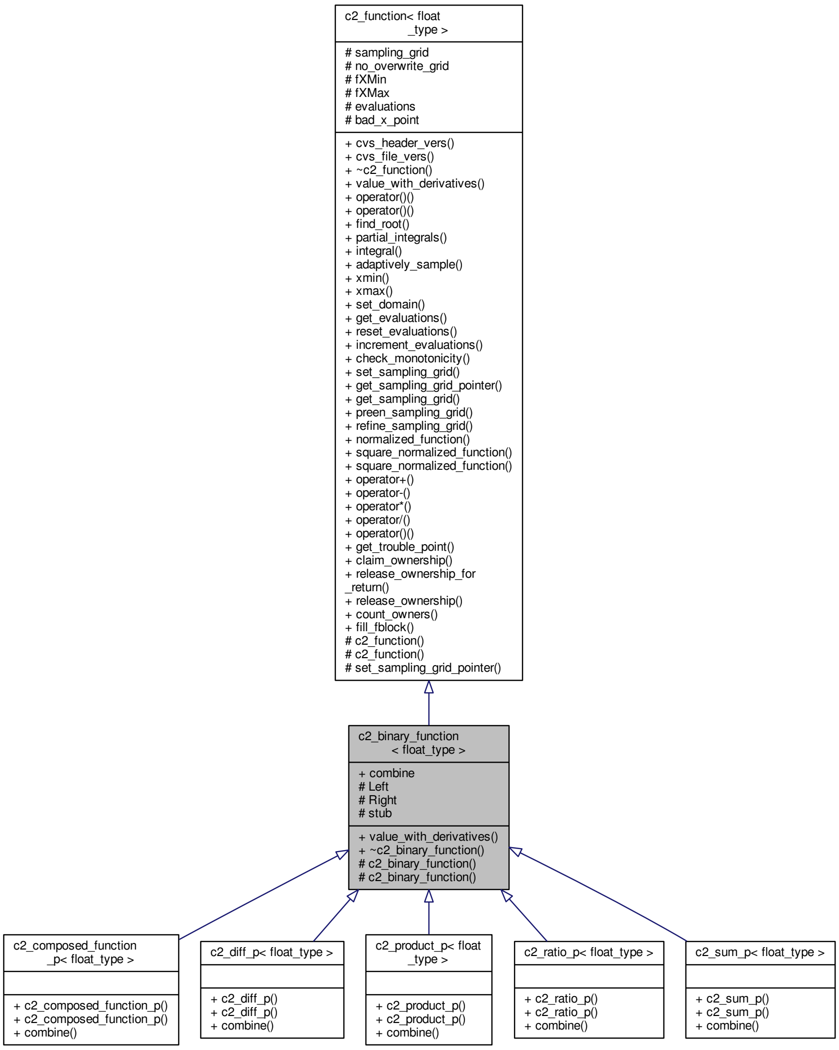
- c2_function.hh

6.11 c2_binary_function< float_type > Class Template Reference

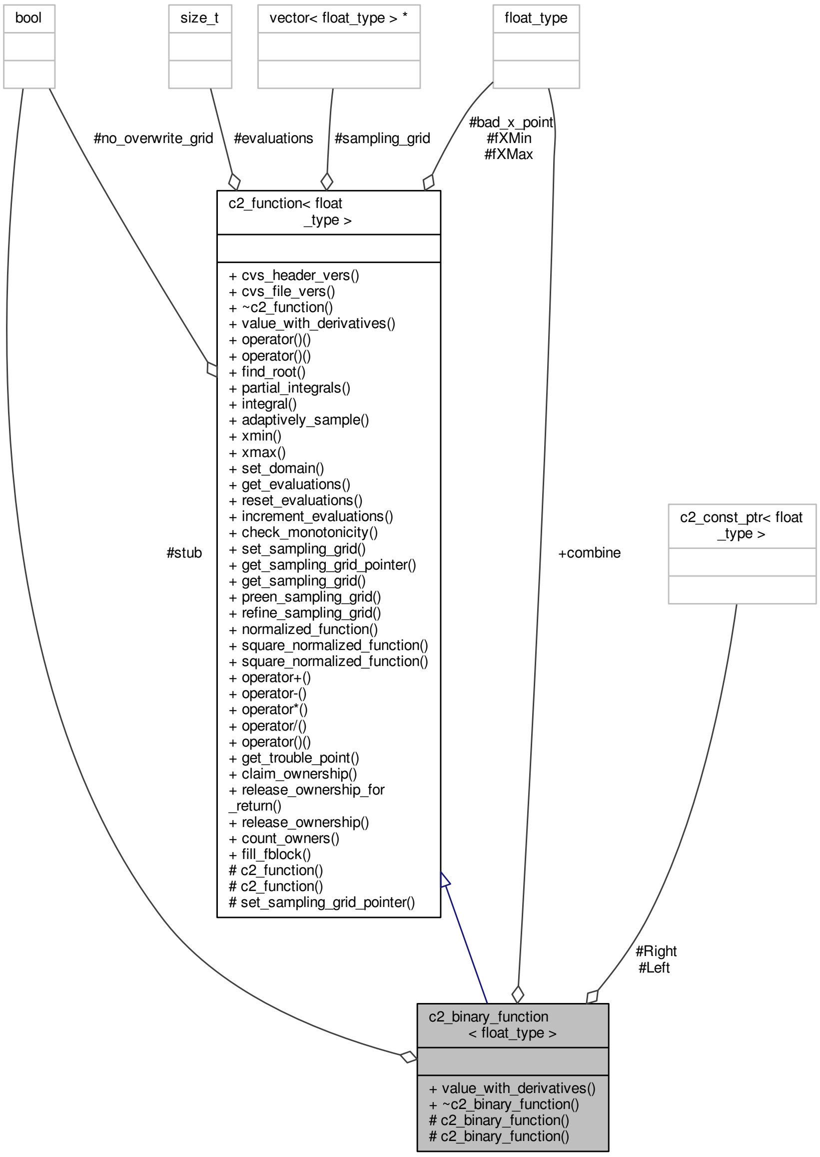
Provides support for c2_function objects which are constructed from two other c2_function objects.

`#include "c2_function.hh"`

Inheritance diagram for c2_binary_function< float_type >:



Collaboration diagram for c2_binary_function< float_type >:



Public Member Functions

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    function to manage the binary operation, used by c2_binary_function::value_with_derivatives()
- virtual ∼c2_binary_function ()

    destructor releases ownership of member functions

Public Attributes

- float_type(∗const combine)(const c2_function< float_type > &left, const c2_function< float_type > &right, float_type x, float_type ∗yprime, float_type ∗yprime2)

Protected Member Functions

- c2_binary_function (float_type(∗combiner)(const c2_function< float_type > &left, const c2_function< float_type > &right, float_type x, float_type ∗yprime, float_type ∗yprime2), const c2_function< float_type > &left, const c2_function< float_type > &right)

  construct the binary function
- c2_binary_function (float_type(∗combiner)(const c2_function< float_type > &left, const c2_function< float_type > &right, float_type x, float_type ∗yprime, float_type ∗yprime2))

  construct a 'stub' c2_binary_function, which provides access to the combine() function

Protected Attributes

- const c2_const_ptr< float_type > Left
- const c2_const_ptr< float_type > Right
- bool stub

  if true, we don't own any functions, we are just a source of a combining function.

6.11.1 Detailed Description

template<typename float_type = double>
class c2_binary_function< float_type >

Provides support for c2_function objects which are constructed from two other c2_function objects.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 template<typename float_type = double> virtual c2_binary_function< float_type >::∼c2_binary_function ( ) [inline], [virtual]

destructor releases ownership of member functions

6.11.2.2 template<typename float_type = double> c2_binary_function< float_type >::c2_binary_function (float_type(∗)(const c2_function< float_type > &left, const c2_function< float_type > &right, float_type x, float_type ∗yprime, float_type ∗yprime2) *combiner*, const c2_function< float_type > & *left*, const c2_function< float_type > & *right*) [inline], [protected]

construct the binary function

Parameters

combiner	pointer to the function which actualy knows how to execute the binary
left	the c2_function to be used in the left side of the binary relation
right	the c2_function to be used in the right side of the binary relation

6.11.2.3 template< typename float_type = double > **c2_binary_function**< float_type >::c2_binary_function (float_type(*)(const **c2_function**< float_type > &left, const **c2_function**< float_type > &right, float_type x, float_type *yprime, float_type *yprime2) *combiner*) [inline], [protected]

construct a 'stub' c2_binary_function, which provides access to the combine() function

Note

Do not evaluate a 'stub' ever. It is only used so that combine() can be called

6.11.3 Member Function Documentation

6.11.3.1 template< typename float_type = double > virtual float_type **c2_binary_function**< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw **c2_exception**) [inline], [virtual]

function to manage the binary operation, used by c2_binary_function::value_with_derivatives()

Implements c2_function< float_type >.

6.11.4 Member Data Documentation

6.11.4.1 template< typename float_type = double > float_type(* const **c2_binary_function**< float_type >::combine) (const **c2_function**< float_type > &left, const **c2_function**< float_type > &right, float_type x, float_type *yprime, float_type *yprime2)

6.11.4.2 template< typename float_type = double > const **c2_const_ptr**<float_type> **c2_binary_function**< float_type >::Left [protected]

6.11.4.3 template< typename float_type = double > const **c2_const_ptr**<float_type> **c2_binary_function**< float_type >::Right [protected]

6.11.4.4 template< typename float_type = double > bool **c2_binary_function**< float_type >::stub [protected]

if true, we don't own any functions, we are just a source of a combining function.

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

- c2_function.hh

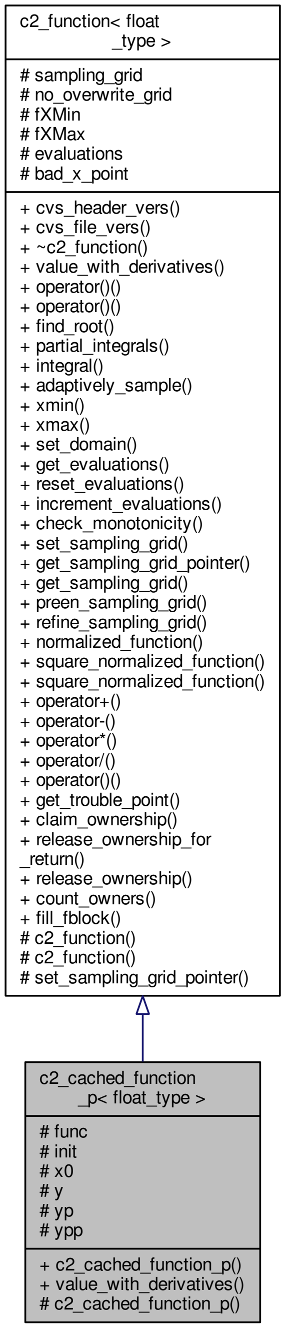
6.12 c2_cached_function_p< float_type > Class Template Reference

A container into which any other c2_function can be dropped.

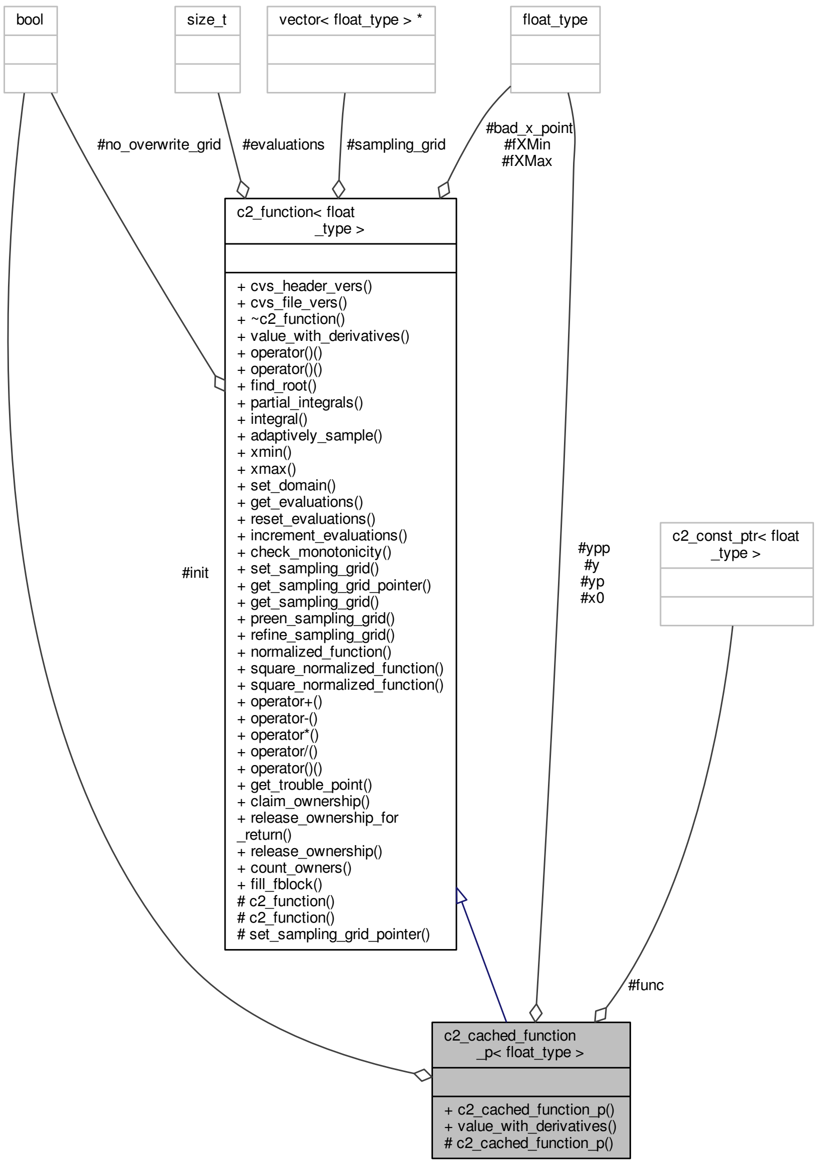
It allows a function to be pre-evaluated at a point, and used at multiple places in an expression efficiently. If it is re-evaluated at the previous point, it returns the remembered values; otherwise, it re-evauates the function at the new point.

```
#include "c2_function.hh"
```

Inheritance diagram for c2_cached_function_p< float_type >:



Collaboration diagram for c2_cached_function_p< float_type >:



Public Member Functions

- c2_cached_function_p (const c2_function< float_type > &f)

  construct the container
- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

  get the value and derivatives.

Protected Member Functions

- c2_cached_function_p ()

Protected Attributes

- const c2_const_ptr< float_type > func
- bool init
- float_type x0
- float_type y
- float_type yp
- float_type ypp

6.12.1 Detailed Description

template<typename float_type = double>
class c2_cached_function_p< float_type >

A container into which any other c2_function can be dropped.

It allows a function to be pre-evaluated at a point, and used at multiple places in an expression efficiently. If it is re-evaluated at the previous point, it returns the remembered values; otherwise, it re-evauates the function at the new point.

The factory function c2_factory::cached_function() creates *new c2_cached_function_p

6.12.2 Constructor & Destructor Documentation

6.12.2.1 template<typename float_type = double> c2_cached_function_p< float_type >::c2_cached_function_p (const c2_function< float_type > & *f*) [inline]

construct the container

Parameters

f	the function to be cached

6.12.2.2 template<typename float_type = double> c2_cached_function_p< float_type >::c2_cached_function_p ( ) [inline], [protected]

6.12.3 Member Function Documentation

6.12.3.1 template<typename float_type = double> virtual float_type c2_cached_function_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Checks to see if the function is being re-evaluated at the previous point, and returns remembered values if so.

Implements c2_function< float_type >.

6.12.4 Member Data Documentation

6.12.4.1 template<typename float_type = double> const c2_const_ptr<float_type> c2_cached_function_p< float_type >::func [protected]

6.12.4.2 template<typename float_type = double> bool c2_cached_function_p< float_type >::init [mutable], [protected]

6.12.4.3 template<typename float_type = double> float_type c2_cached_function_p< float_type >::x0 [mutable], [protected]

6.12.4.4 template<typename float_type = double> float_type c2_cached_function_p< float_type >::y [mutable], [protected]

6.12.4.5 template<typename float_type = double> float_type c2_cached_function_p< float_type >::yp [mutable], [protected]

6.12.4.6 template<typename float_type = double> float_type c2_cached_function_p< float_type >::ypp [mutable], [protected]

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

- c2_function.hh

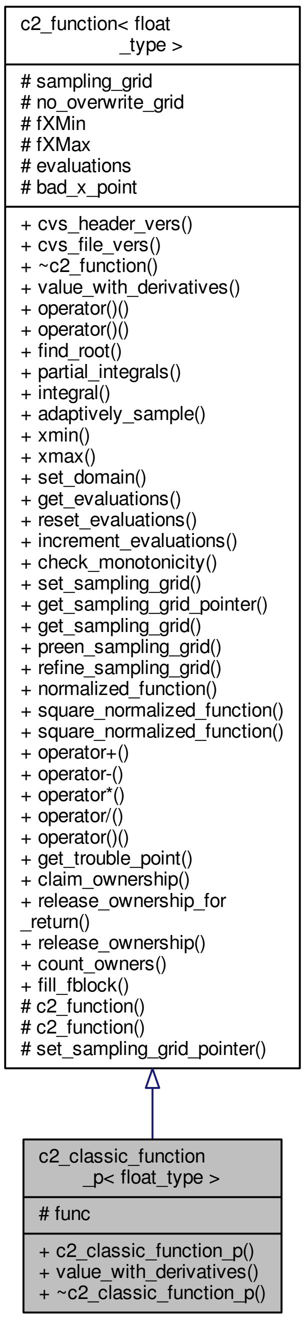
6.13 c2_classic_function_p< float_type > Class Template Reference

a container into which any conventional c-style function can be dropped, to create a degenerate c2_function without derivatives. Mostly useful for sampling into interpolating functions. construct a reference to this with c2_classic_function()

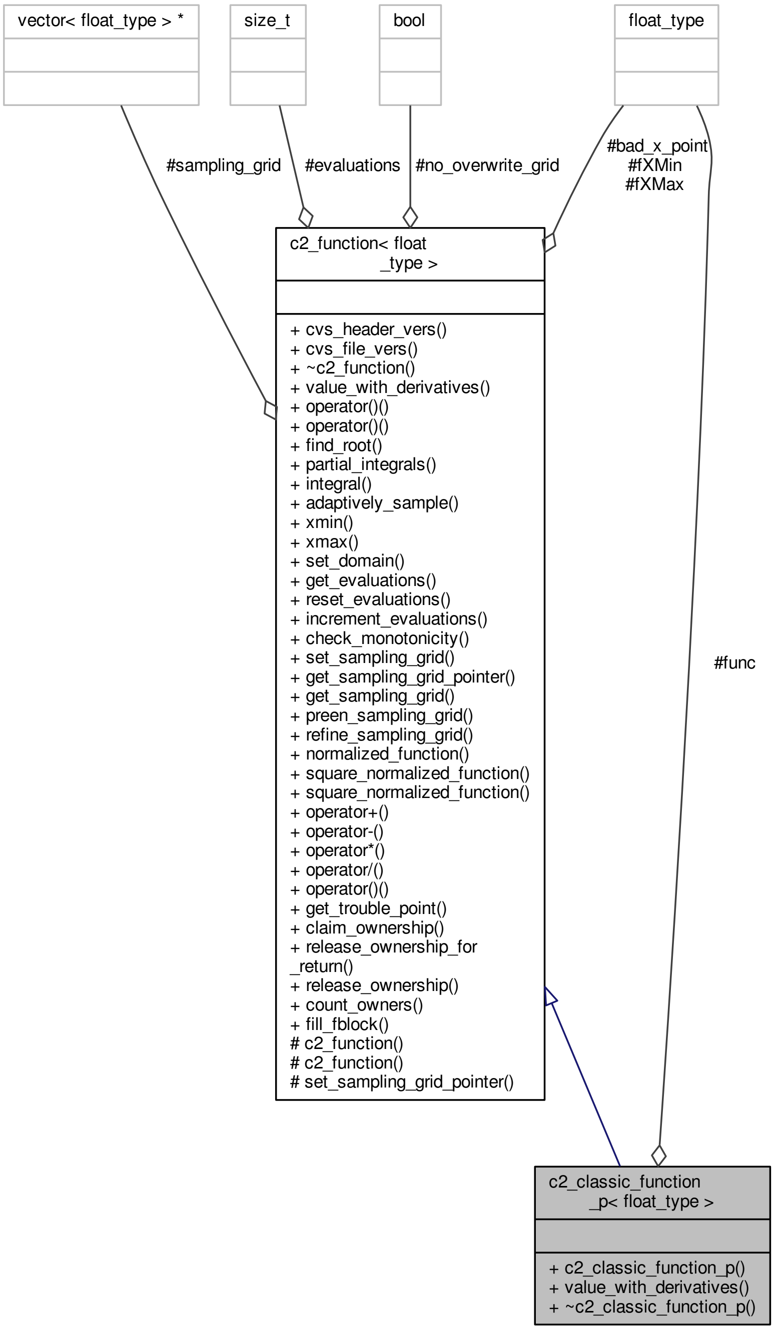
The factory function c2_factory::classic_function() creates *new c2_classic_function_p()

```
#include "c2_function.hh"
```

Inheritance diagram for c2_classic_function_p< float_type >:



Collaboration diagram for c2_classic_function_p< float_type >:



Public Member Functions

- c2_classic_function_p (const float_type(∗c_func)(float_type))

    construct the container
- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.
- virtual ∼c2_classic_function_p ()

Protected Attributes

- const float_type(* func)(float_type)

  pointer to our function

Additional Inherited Members

6.13.1 Detailed Description

template<typename float_type = double>
class c2_classic_function_p< float_type >

a container into which any conventional c-style function can be dropped, to create a degenerate c2_function without derivatives. Mostly useful for sampling into interpolating functions. construct a reference to this with c2_classic_↩
function()

The factory function c2_factory::classic_function() creates *new c2_classic_function_p()

6.13.2 Constructor & Destructor Documentation

6.13.2.1 template<typename float_type = double> c2_classic_function_p< float_type >::c2_classic_function_p (const float_type(*)(float_type) *c_func*) [inline]

construct the container

Parameters

c_func	a pointer to a conventional c-style function

6.13.2.2 template<typename float_type = double> virtual c2_classic_function_p< float_type >::~c2_classic_function_p ( ) [inline], [virtual]

6.13.3 Member Function Documentation

6.13.3.1 template<typename float_type = double> virtual float_type c2_classic_function_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function Uses the internal function pointer set by set_function().

Implements c2_function< float_type >.

6.13.4 Member Data Documentation

6.13.4.1 template<typename float_type = double> const float_type(* c2_classic_function_p< float_type >::func) (float_type) [protected]

pointer to our function

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

- c2_function.hh

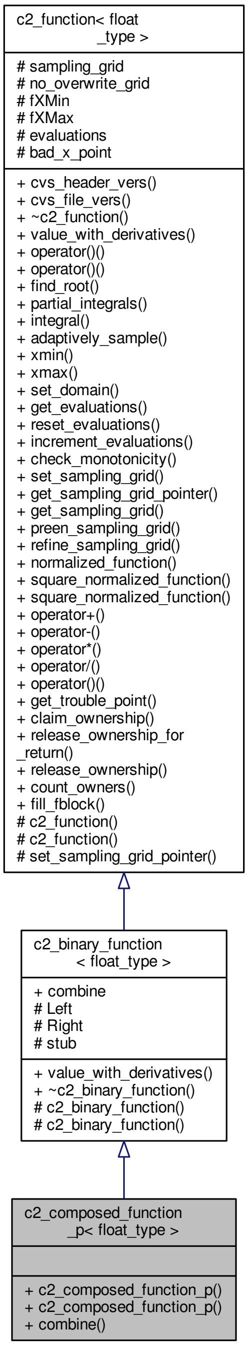
6.14 c2_composed_function_p< float_type > Class Template Reference

Provides function composition (nesting)

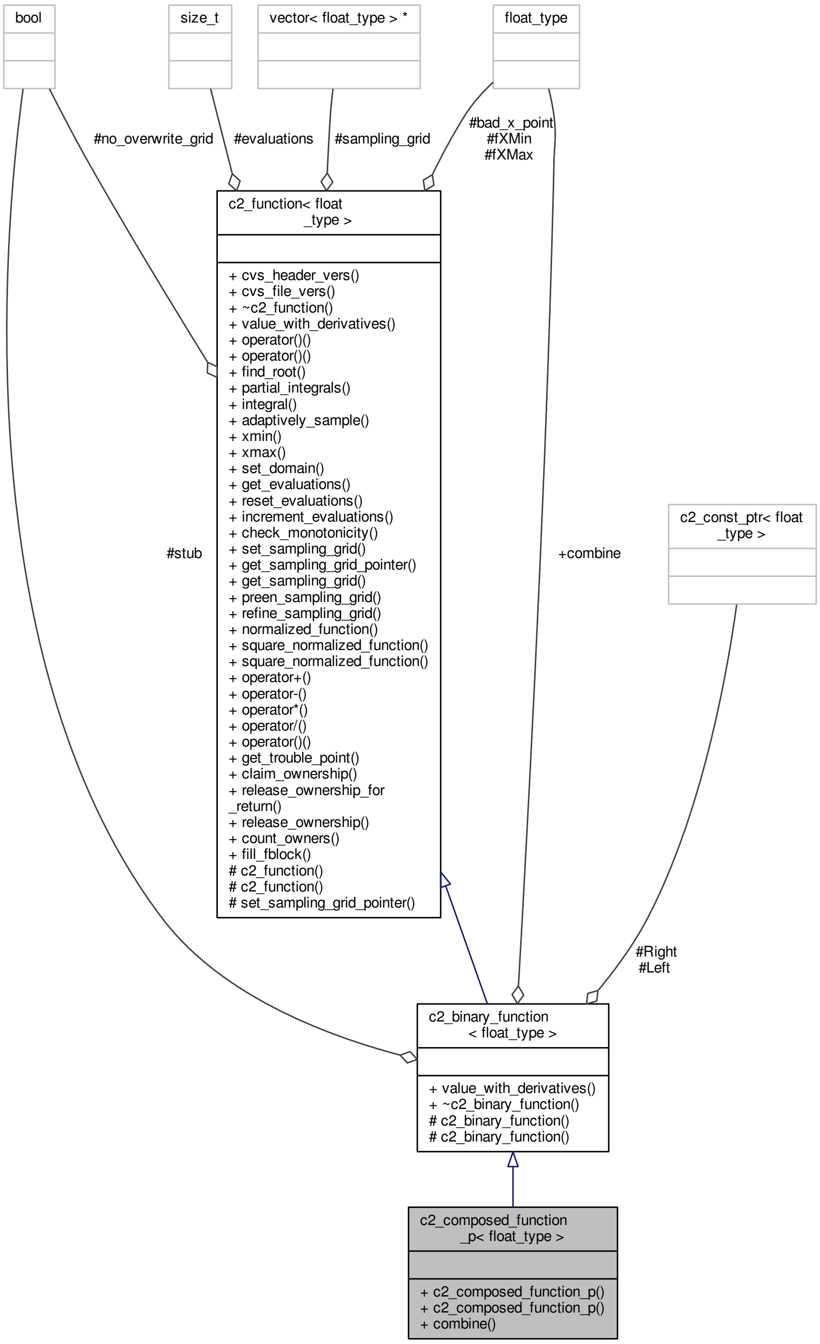
This allows evaluation of *f(g(x))* where *f* and *g* are c2_function objects.

```
#include "c2_function.hh"
```

Inheritance diagram for c2_composed_function_p< float_type >:



Collaboration diagram for c2_composed_function_p< float_type >:



Public Member Functions

- c2_composed_function_p (const c2_function< float_type > &outer, const c2_function< float_type > &inner)

    construct outer(inner (x))

- c2_composed_function_p ()

    Create a stub just for the combiner to avoid statics.

Static Public Member Functions

- static float_type combine (const c2_function< float_type > &left, const c2_function< float_type > &right, float_type x, float_type ∗yprime, float_type ∗yprime2) throw (c2_exception)

    execute math necessary to do composition

Additional Inherited Members

6.14.1 Detailed Description

template<typename float_type = double>
class c2_composed_function_p< float_type >

Provides function composition (nesting)

This allows evaluation of *f(g(x))* where *f* and *g* are c2_function objects.

This should always be constructed using c2_function::operator()

6.14.2 Constructor & Destructor Documentation

6.14.2.1 template<typename float_type = double> c2_composed_function_p< float_type >::c2_composed↩_function_p (const c2_function< float_type > & *outer,* const c2_function< float_type > & *inner*) [inline]

construct *outer*(*inner* (x))

Note

See c2_binary_function for discussion of ownership.

Parameters

outer	the outer function
inner	the inner function

6.14.2.2 template<typename float_type = double> c2_composed_function_p< float_type >::c2_composed_function_p ( ) [inline]

Create a stub just for the combiner to avoid statics.

6.14.3 Member Function Documentation

6.14.3.1 template<typename float_type = double> static float_type c2_composed_function_p< float_type >::combine (const c2_function< float_type > & *left,* const c2_function< float_type > & *right,* float_type *x,* float_type ∗ *yprime,* float_type ∗ *yprime2*) throw c2_exception) [inline], [static]

execute math necessary to do composition

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

- c2_function.hh

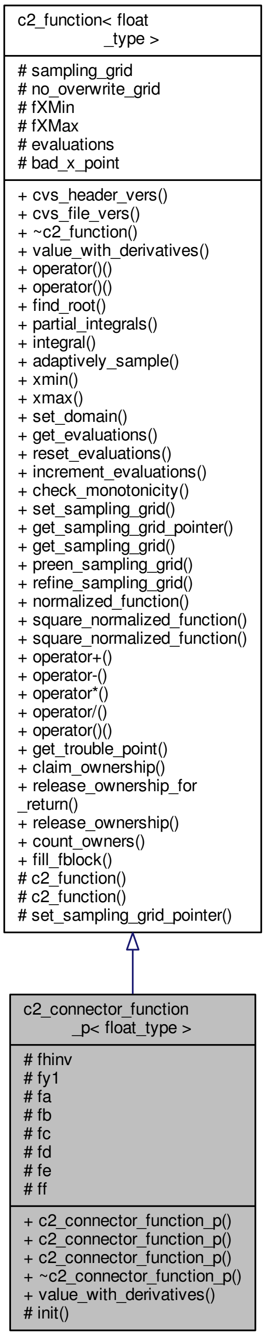
6.15 c2_connector_function_p< float_type > Class Template Reference

create a c2_function which smoothly connects two other c2_functions.

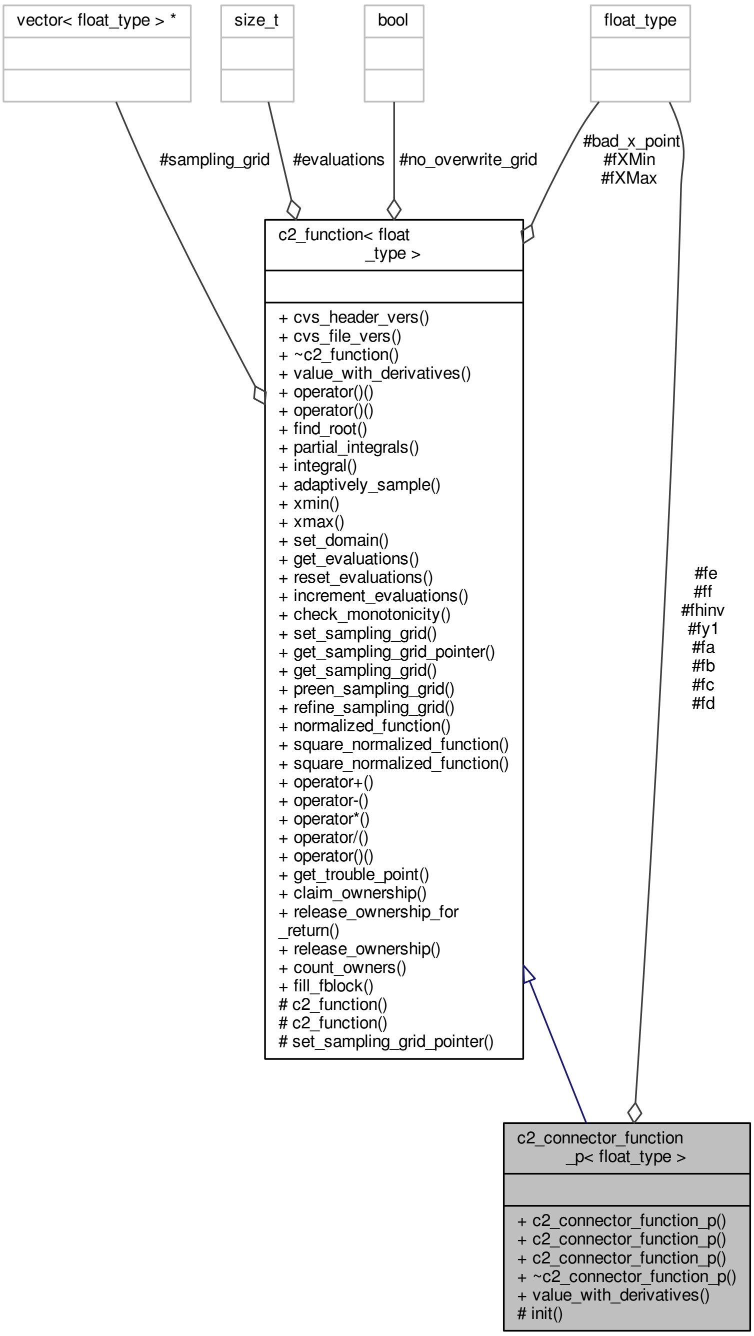
This takes two points and generates a polynomial which matches two c2_function arguments at those two points, with two derivatives at each point, and an arbitrary value at the center of the region. It is useful for splicing together functions over rough spots (0/0, for example).

```
#include "c2_function.hh"
```

Inheritance diagram for c2_connector_function_p< float_type >:



Collaboration diagram for c2_connector_function_p< float_type >:



Public Member Functions

- c2_connector_function_p (float_type x0, const c2_function< float_type > &f0, float_type x2, const c2_function< float_type > &f2, bool auto_center, float_type y1)

  construct the container from two functions

- c2_connector_function_p (float_type x0, float_type y0, float_type yp0, float_type ypp0, float_type x2, float_type y2, float_type yp2, float_type ypp2, bool auto_center, float_type y1) *construct the container from numerical values*

- c2_connector_function_p (const c2_fblock< float_type > &fb0, const c2_fblock< float_type > &fb2, bool auto_center, float_type y1)

  construct the container from c2_fblock<float_type> objects

- virtual ~c2_connector_function_p ()

  destructor

- virtual float_type value_with_derivatives (float_type x, float_type *yprime, float_type *yprime2) const throw (c2_exception)

  get the value and derivatives.

Protected Member Functions

- void init (const c2_fblock< float_type > &fb0, const c2_fblock< float_type > &fb2, bool auto_center, float_type y1)

  fill container numerically

Protected Attributes

- float_type fhinv
- float_type fy1
- float_type fa
- float_type fb
- float_type fc
- float_type fd
- float_type fe
- float_type ff

6.15.1 Detailed Description

template<typename float_type = double>
class c2_connector_function_p< float_type >

create a c2_function which smoothly connects two other c2_functions.

This takes two points and generates a polynomial which matches two c2_function arguments at those two points, with two derivatives at each point, and an arbitrary value at the center of the region. It is useful for splicing together functions over rough spots (0/0, for example).

If *auto_center* is true, the value at the midpoint is computed so that the resulting polynomial is of order 5. If *auto_center* is false, the value *y1* is used at the midpoint, resulting in a polynomial of order 6.

This is usually used in conjunction with c2_piecewise_function_p to assemble an apparently seamless function from a series of segments.

See also

Sample Applications and Adaptive sampling

The factory function c2_factory::connector_function() creates *new c2_connector_function_p

6.15.2 Constructor & Destructor Documentation

6.15.2.1 template<typename float_type = double> c2_connector_function_p< float_type >::c2_connector_function_p (float_type *x0*, const c2_function< float_type > & *f0*, float_type *x2*, const c2_function< float_type > & *f2*, bool *auto_center*, float_type *y1*)

construct the container from two functions

Parameters

x0	the point at which to match *f1* and its derivatives
f0	the function on the left side to be connected
x2	the point at which to match *f2* and its derivatives
f2	the function on the right side to be connected
auto_center	if true, no midpoint value is specified. If false, match the value *y1* at the midpoint
y1	the value to match at the midpoint, if *auto_center* is false

Returns

a c2_function with domain (*x0*,*x2*) which smoothly connects *f0(x0)* and *f2(x2)*

6.15.2.2 template<typename float_type = double> c2_connector_function_p< float_type >::c2_connector_function_p (float_type *x0*, float_type *y0*, float_type *yp0*, float_type *ypp0*, float_type *x2*, float_type *y2*, float_type *yp2*, float_type *ypp2*, bool *auto_center*, float_type *y1*)

construct the container from numerical values

Parameters

x0	the position of the left edge
y0	the function derivative on the left boundary
yp0	the function second derivative on the left boundary
ypp0	the function value on the left boundary
x2	the position of the right edge
y2	the function derivative on the right boundary
yp2	the function second derivative on the right boundary
ypp2	the function value on the right boundary
auto_center	if true, no midpoint value is specified. If false, match the value *y1* at the midpoint
y1	the value to match at the midpoint, if *auto_center* is false

Returns

a c2_function with domain (*x0*,*x2*) which smoothly connects the points described

6.15.2.3 template<typename float_type = double> c2_connector_function_p< float_type >::c2_connector_function_p (const c2_fblock< float_type > & *fb0*, const c2_fblock< float_type > & *fb2*, bool *auto_center*, float_type *y1*)

construct the container from c2_fblock<float_type> objects

Parameters

fb0	the left edge
fb2	the right edge
auto_center	if true, no midpoint value is specified. If false, match the value *y1* at the midpoint
y1	the value to match at the midpoint, if *auto_center* is false

Returns

a c2_function with domain (*fb0.x*,*fb2.x*) which smoothly connects *fb0* and *fb2*

6.15.2.4 template<typename float_type = double> virtual c2_connector_function_p< float_type >::~c2_connector_function_p ( ) [virtual]

destructor

6.15.3 Member Function Documentation

6.15.3.1 template<typename float_type = double> void c2_connector_function_p< float_type >::init (const c2_fblock< float_type > & *fb0*, const c2_fblock< float_type > & *fb2*, bool *auto_center*, float_type *y1*) [protected]

fill container numerically

6.15.3.2 template<typename float_type = double> virtual float_type c2_connector_function_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

6.15.4 Member Data Documentation

6.15.4.1 template<typename float_type = double> float_type c2_connector_function_p< float_type >::fa [protected]

6.15.4.2 template<typename float_type = double> float_type c2_connector_function_p< float_type >::fb [protected]

6.15.4.3 template<typename float_type = double> float_type c2_connector_function_p< float_type >::fc [protected]

6.15.4.4 template<typename float_type = double> float_type c2_connector_function_p< float_type >::fd [protected]

6.15.4.5 template<typename float_type = double> float_type c2_connector_function_p< float_type >::fe [protected]

6.15.4.6 template<typename float_type = double> float_type c2_connector_function_p< float_type >::ff [protected]

6.15.4.7 template<typename float_type = double> float_type c2_connector_function_p< float_type >::fhinv [protected]

6.15.4.8 template<typename float_type = double> float_type c2_connector_function_p< float_type >::fy1 [protected]

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

- c2_function.hh

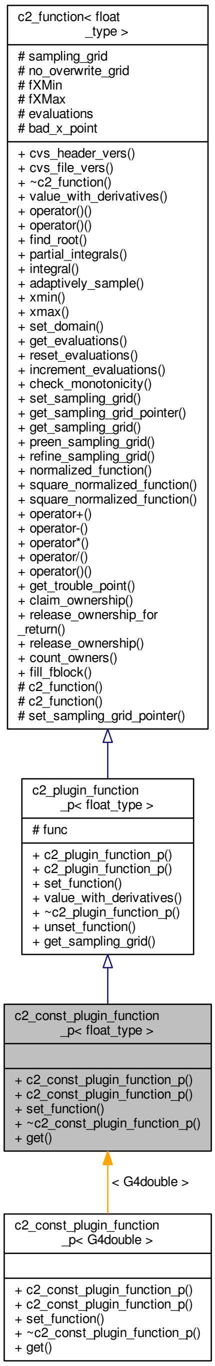
6.16 c2_const_plugin_function_p< float_type > Class Template Reference

a c2_plugin_function_p which promises not to fiddle with the plugged function.

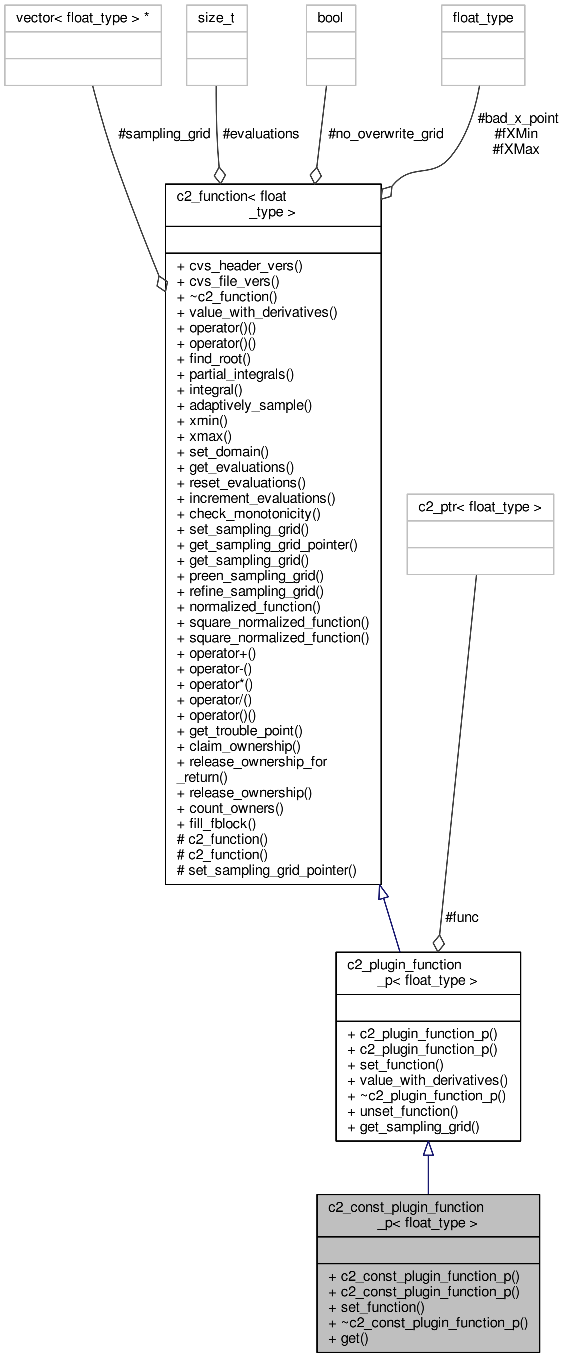
The factory function c2_factory::const_plugin_function() creates *new c2_const_plugin_function_p()

```
#include "c2_function.hh"
```

Inheritance diagram for c2_const_plugin_function_p< float_type >:



Collaboration diagram for c2_const_plugin_function_p< float_type >:



Public Member Functions

- c2_const_plugin_function_p ()

    construct the container with no function
- c2_const_plugin_function_p (const c2_function< float_type > &f)

    construct the container with a pre-defined function
- void set_function (const c2_function< float_type > ∗f)

fill the container with a new function, or clear it with a null pointer

- virtual ~c2_const_plugin_function_p ()

  destructor

- const c2_function< float_type > & get () const throw (c2_exception)

  get a const reference to our owned function, for direct access

Additional Inherited Members

6.16.1 Detailed Description

template<typename float_type = double>
class c2_const_plugin_function_p< float_type >

a c2_plugin_function_p which promises not to fiddle with the plugged function.

The factory function c2_factory::const_plugin_function() creates *new c2_const_plugin_function_p()

6.16.2 Constructor & Destructor Documentation

6.16.2.1 template<typename float_type = double> c2_const_plugin_function_p< float_type >::c2_const_plugin_function_p ( ) [inline]

construct the container with no function

6.16.2.2 template<typename float_type = double> c2_const_plugin_function_p< float_type >::c2_const_plugin_function_p (const c2_function< float_type > & *f*) [inline]

construct the container with a pre-defined function

6.16.2.3 template<typename float_type = double> virtual c2_const_plugin_function_p< float_type >::~c2_const_plugin_function_p ( ) [inline], [virtual]

destructor

6.16.3 Member Function Documentation

6.16.3.1 template<typename float_type = double> const c2_function<float_type>& c2_const_plugin_function_p< float_type >::get ( ) const throw c2_exception) [inline]

get a const reference to our owned function, for direct access

6.16.3.2 template< typename float_type = double > void c2_const_plugin_function_p< float_type >::set_function (const c2_function< float_type > * f) [inline]

fill the container with a new function, or clear it with a null pointer

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

- c2_function.hh

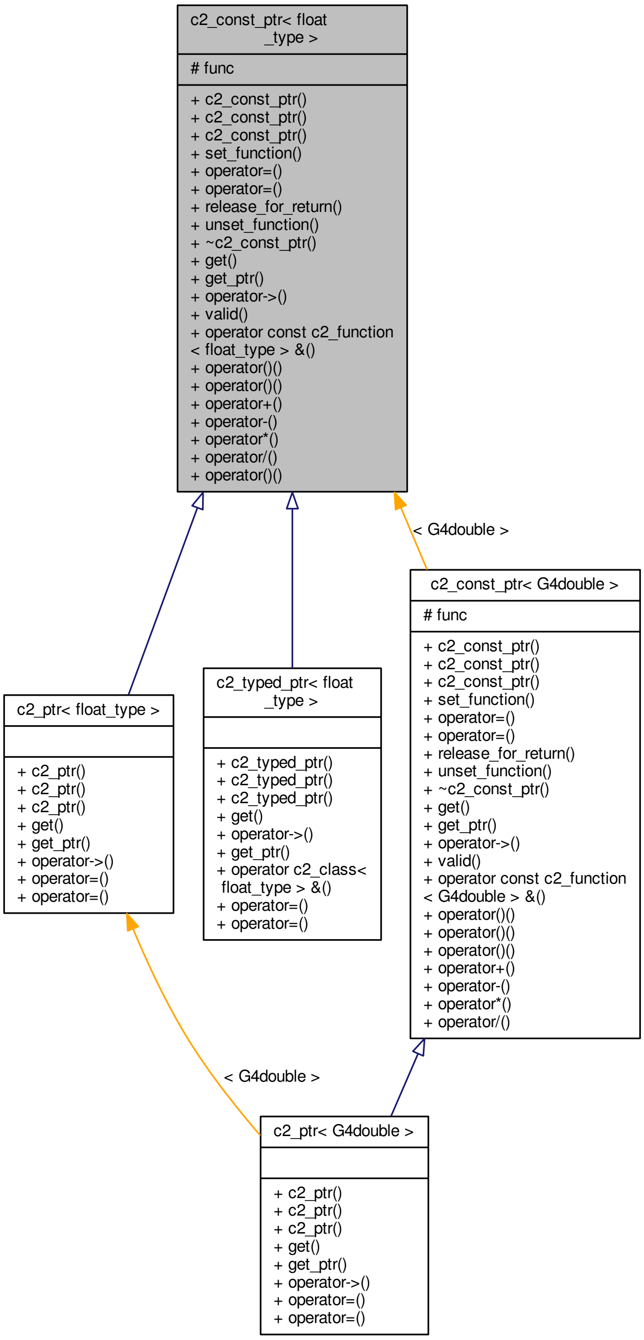
6.17 c2_const_ptr< float_type > Class Template Reference

create a container for a c2_function which handles the reference counting.

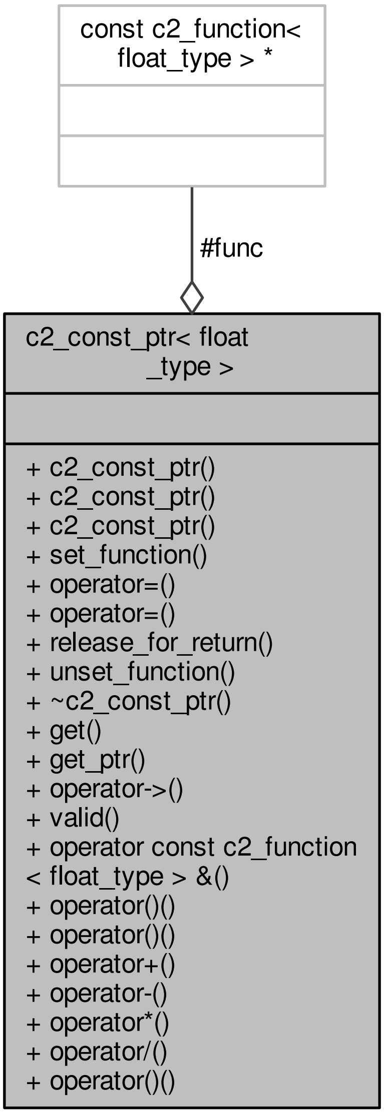
It is useful as a smart container to hold a c2_function and keep the reference count correct. The recommended way for a class to store a c2_function which is handed in from the outside is for it to have a c2_ptr member into which the passed-in function is stored. This way, when the class instance is deleted, it will automatically dereference any function which it was handed.

```
#include "c2_function.hh"
```

Inheritance diagram for c2_const_ptr< float_type >:



Collaboration diagram for c2_const_ptr< float_type >:



Public Member Functions

- c2_const_ptr ()

  construct the container with no function
- c2_const_ptr (const c2_function< float_type > &f)

  construct the container with a pre-defined function
- c2_const_ptr (const c2_const_ptr< float_type > &src)

  copy constructor
- void set_function (const c2_function< float_type > ∗f)

  fill the container with a new function, or clear it with a null pointer

- const c2_const_ptr< float_type > & operator= (const c2_const_ptr< float_type > &f)

    fill the container from another container
- const c2_function< float_type > & operator= (const c2_function< float_type > &f)

    fill the container with a function
- void release_for_return () throw (c2_exception)

    release the function without destroying it, so it can be returned from a function
- void unset_function (void)

    clear the function
- ∼c2_const_ptr ()

    destructor
- const c2_function< float_type > & get () const throw (c2_exception)

    get a reference to our owned function
- const c2_function< float_type > ∗ get_ptr () const

    get an unchecked pointer to our owned function
- const c2_function< float_type > ∗ operator-> () const

    get a checked pointer to our owned function
- bool valid () const

    check if we have a valid function
- operator const c2_function< float_type > & () const

    type coercion operator which lets us use a pointer as if it were a const c2_function
- float_type operator() (float_type x) const throw (c2_exception)

    convenience operator to make us look like a function
- float_type operator() (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    convenience operator to make us look like a function
- c2_sum_p< float_type > & operator+ (const c2_function< float_type > &rhs) const throw (c2_exception)

    factory function to create a c2_sum_p from a regular algebraic expression.
- c2_diff_p< float_type > & operator- (const c2_function< float_type > &rhs) const throw (c2_exception)

    factory function to create a c2_diff_p from a regular algebraic expression.
- c2_product_p< float_type > & operator∗ (const c2_function< float_type > &rhs) const throw (c2_exception)

    factory function to create a c2_product_p from a regular algebraic expression.
- c2_ratio_p< float_type > & operator/ (const c2_function< float_type > &rhs) const throw (c2_exception)

    factory function to create a c2_ratio_p from a regular algebraic expression.
- c2_composed_function_p< float_type > & operator() (const c2_function< float_type > &inner) const throw (c2_exception)

    compose this function outside another.

Protected Attributes

- const c2_function< float_type > ∗ func

6.17.1 Detailed Description

template<typename float_type>
class c2_const_ptr< float_type >

create a container for a c2_function which handles the reference counting.

It is useful as a smart container to hold a c2_function and keep the reference count correct. The recommended way for a class to store a c2_function which is handed in from the outside is for it to have a c2_ptr member into which the passed-in function is stored. This way, when the class instance is deleted, it will automatically dereference any function which it was handed.

This class contains a copy constructor and operator=, to make it fairly easy to make a std::vector of these objects, and have it work as expected.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 template<typename float_type> c2_const_ptr< float_type >::c2_const_ptr ( ) [inline]

construct the container with no function

6.17.2.2 template<typename float_type> c2_const_ptr< float_type >::c2_const_ptr (const c2_function< float_type > & *f*) [inline]

construct the container with a pre-defined function

Parameters

f	the function to store

6.17.2.3 template<typename float_type> c2_const_ptr< float_type >::c2_const_ptr (const c2_const_ptr< float_type > & *src*) [inline]

copy constructor

Parameters

src	the container to copy

6.17.2.4 template<typename float_type> c2_const_ptr< float_type >::~c2_const_ptr ( ) [inline]

destructor

6.17.3 Member Function Documentation

6.17.3.1 template<typename float_type> const c2_function<float_type>& c2_const_ptr< float_type >::get ( ) const throw c2_exception) [inline]

get a reference to our owned function

6.17.3.2 template<typename float_type> const c2_function<float_type>* c2_const_ptr< float_type >::get_ptr ( ) const [inline]

get an unchecked pointer to our owned function

6.17.3.3 template<typename float_type> c2_const_ptr< float_type >::operator const c2_function< float_type > & ( ) const [inline]

type coercion operator which lets us use a pointer as if it were a const c2_function

6.17.3.4 template<typename float_type> float_type c2_const_ptr< float_type >::operator() (float_type *x*) const throw c2_exception) [inline]

convenience operator to make us look like a function

Parameters

x	the value at which to evaluate the contained function

Returns

the evaluated function

Note

If you using this repeatedly, do const c2_function<float_type> &func=ptr; and use func(x). Calling this operator wastes some time, since it checks the validity of the pointer every time.

6.17.3.5 template<typename float_type> float_type c2_const_ptr< float_type >::operator() (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline]

convenience operator to make us look like a function

Parameters

x	the value at which to evaluate the contained function
yprime	the derivative
yprime2	the second derivative

Returns

the evaluated function

Note

If you using this repeatedly, do const c2_function<float_type> &func=ptr; and use func(x). Calling this operator wastes some time, since it checks the validity of the pointer every time.

6.17.3.6 template<typename float_type> c2_composed_function_p<float_type>& c2_const_ptr< float_type >::operator() (const c2_function< float_type > & *inner*) const throw c2_exception) [inline]

compose this function outside another.

Parameters

inner	the inner function

Returns

the composed function

6.17.3.7 template<typename float_type> c2_product_p<float_type>& c2_const_ptr< float_type >::operator* (const c2_function< float_type > & *rhs*) const throw c2_exception) [inline]

factory function to create a c2_product_p from a regular algebraic expression.

Parameters

rhs	the right-hand term of the product

Returns

a new c2_function

6.17.3.8 template<typename float_type> c2_sum_p<float_type>& c2_const_ptr< float_type >::operator+ (const c2_function< float_type > & *rhs*) const throw c2_exception) [inline]

factory function to create a c2_sum_p from a regular algebraic expression.

Parameters

rhs	the right-hand term of the sum

Returns

a new c2_function

6.17.3.9 template<typename float_type> c2_diff_p<float_type>& c2_const_ptr< float_type >::operator- (const c2_function< float_type > & *rhs*) const throw c2_exception) [inline]

factory function to create a c2_diff_p from a regular algebraic expression.

Parameters

rhs	the right-hand term of the difference

Returns

a new c2_function

6.17.3.10 template<typename float_type> const c2_function<float_type>* c2_const_ptr< float_type >::operator-> ( ) const [inline]

get a checked pointer to our owned function

6.17.3.11 template<typename float_type> c2_ratio_p<float_type>& c2_const_ptr< float_type >::operator/ (const c2_function< float_type > & *rhs*) const throw c2_exception) [inline]

factory function to create a c2_ratio_p from a regular algebraic expression.

Parameters

rhs	the right-hand term of the ratio (the denominator)

Returns

a new c2_function

6.17.3.12 template<typename float_type> const c2_const_ptr<float_type>& c2_const_ptr< float_type >::operator= (const c2_const_ptr< float_type > & *f*) [inline]

fill the container from another container

Parameters

f	the container to copy

6.17.3.13 template<typename float_type> const c2_function<float_type>& c2_const_ptr< float_type >::operator= (const c2_function< float_type > & *f*) [inline]

fill the container with a function

Parameters

f	the function

6.17.3.14 template<typename float_type> void c2_const_ptr< float_type >::release_for_return ( ) throw c2_exception) [inline]

release the function without destroying it, so it can be returned from a function

This is usually the very last line of a function before the return statement, so that any exceptions that happen during execution of the function will cause proper cleanup. Once the function has been released from its container this way, it is an orhpaned object until the caller claims it, so it could get lost if an exception happens.

6.17.3.15 template<typename float_type> void c2_const_ptr< float_type >::set_function (const c2_function< float_type > * *f*) [inline]

fill the container with a new function, or clear it with a null pointer

Parameters

f	the function to store, releasing any previously held function

6.17.3.16 template<typename float_type> void c2_const_ptr< float_type >::unset_function (void) [inline]

clear the function

Any attempt to use this c2_plugin_function_p throws an exception if the saved function is cleared.

6.17.3.17 template<typename float_type> bool c2_const_ptr< float_type >::valid ( ) const [inline]

check if we have a valid function

6.17.4 Member Data Documentation

6.17.4.1 template<typename float_type> const c2_function<float_type>* c2_const_ptr< float_type >::func [protected]

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

- c2_function.hh

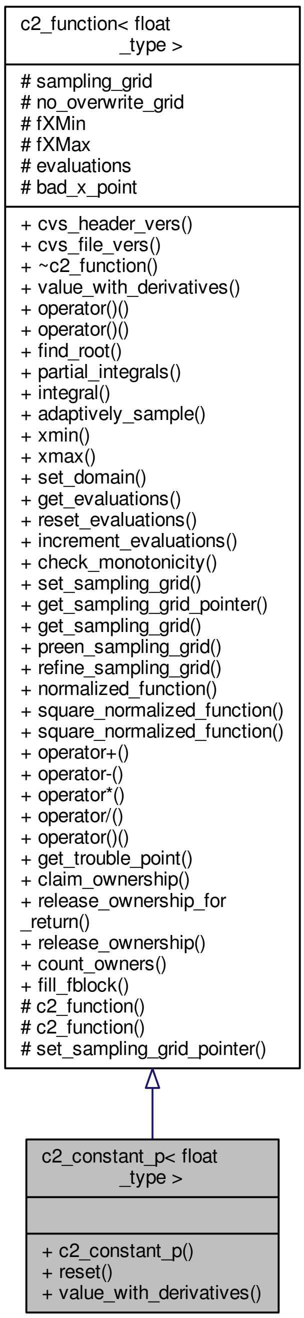
6.18 c2_constant_p< float_type > Class Template Reference

a c2_function which is constant

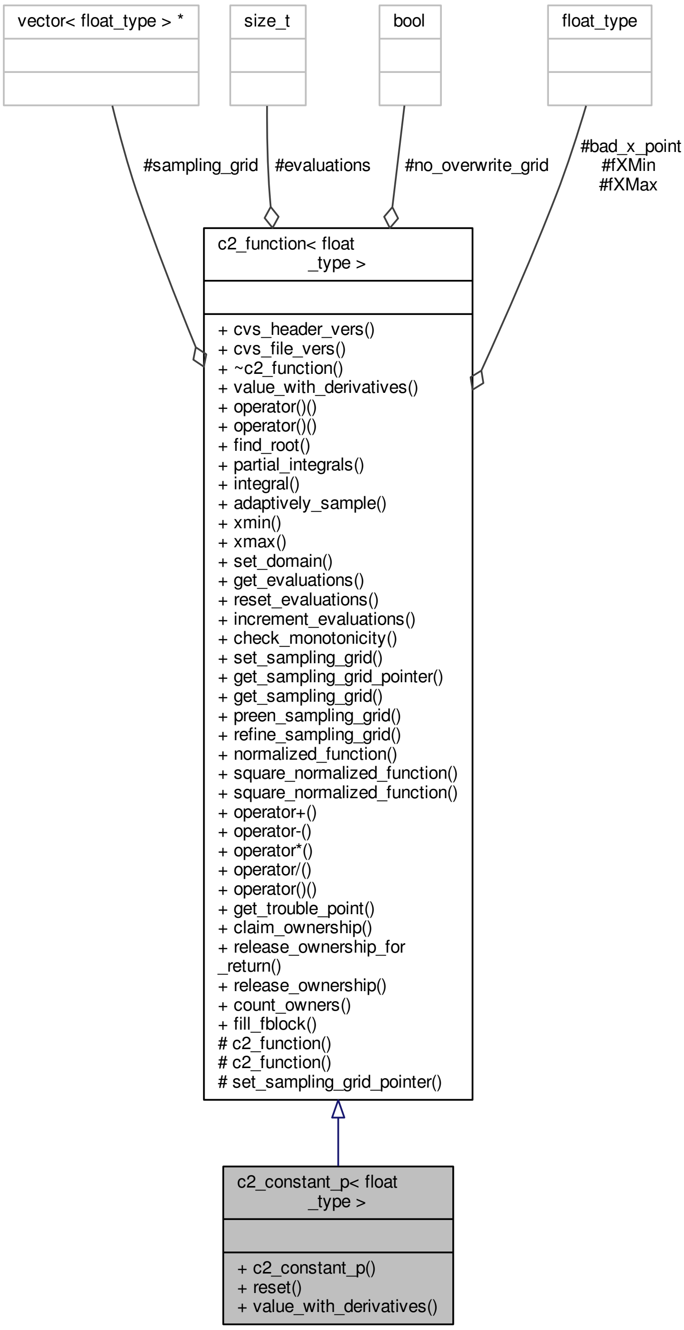
The factory function c2_factory::constant() creates *new c2_constant_p()

```
#include "c2_function.hh"
```

Inheritance diagram for c2_constant_p< float_type >:



Collaboration diagram for c2_constant_p< float_type >:



Public Member Functions

- c2_constant_p (float_type x)
- void reset (float_type val)
- virtual float_type value_with_derivatives (float_type, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.

Additional Inherited Members

6.18.1 Detailed Description

template<typename float_type>
class c2_constant_p< float_type >

a c2_function which is constant

The factory function c2_factory::constant() creates *new c2_constant_p()

6.18.2 Constructor & Destructor Documentation

6.18.2.1 template<typename float_type > c2_constant_p< float_type >::c2_constant_p (float_type *x*) [inline]

6.18.3 Member Function Documentation

6.18.3.1 template<typename float_type > void c2_constant_p< float_type >::reset (float_type *val*) [inline]

6.18.3.2 template<typename float_type > virtual float_type c2_constant_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

- c2_function.hh

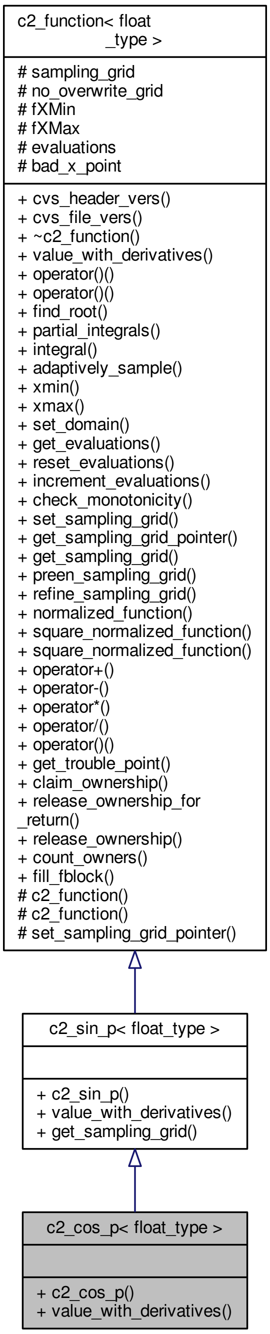
6.19 c2_cos_p< float_type > Class Template Reference

compute cos(x) with its derivatives.

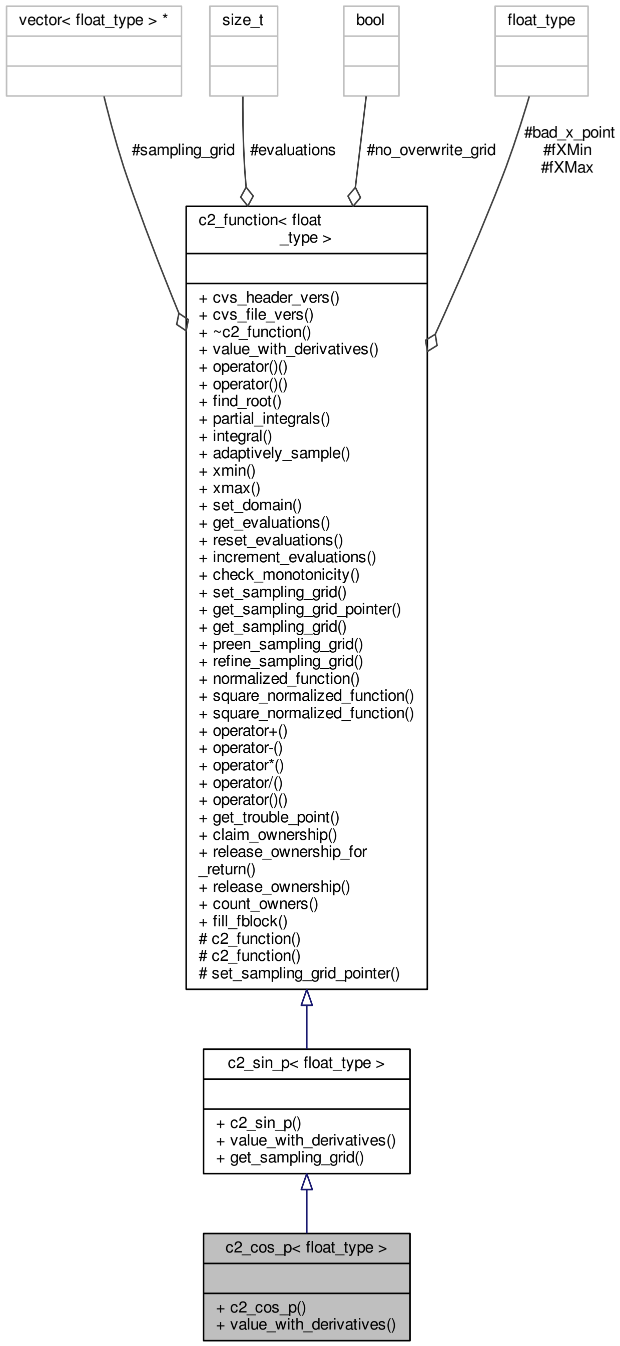
The factory function c2_factory::cos() creates ∗new c2_cos_p

```
#include "c2_function.hh"
```

Inheritance diagram for c2_cos_p< float_type >:



Collaboration diagram for c2_cos_p< float_type >:



Public Member Functions

- c2_cos_p ()

    constructor.

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.

Additional Inherited Members

6.19.1 Detailed Description

template<typename float_type = double>
class c2_cos_p< float_type >

compute cos(x) with its derivatives.

The factory function c2_factory::cos() creates *new c2_cos_p

6.19.2 Constructor & Destructor Documentation

6.19.2.1 template<typename float_type = double> c2_cos_p< float_type >::c2_cos_p ( ) [inline]

constructor.

6.19.3 Member Function Documentation

6.19.3.1 template<typename float_type = double> virtual float_type c2_cos_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Reimplemented from c2_sin_p< float_type >.

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

- c2_function.hh

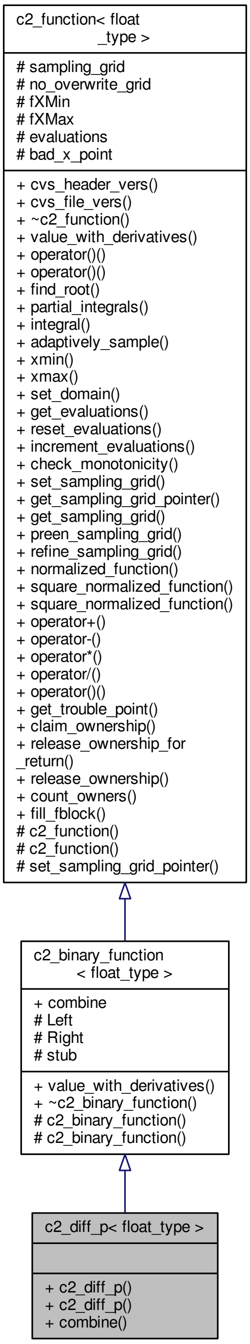
6.20 c2_diff_p< float_type > Class Template Reference

create a c2_function which is the difference of two other c2_functions.

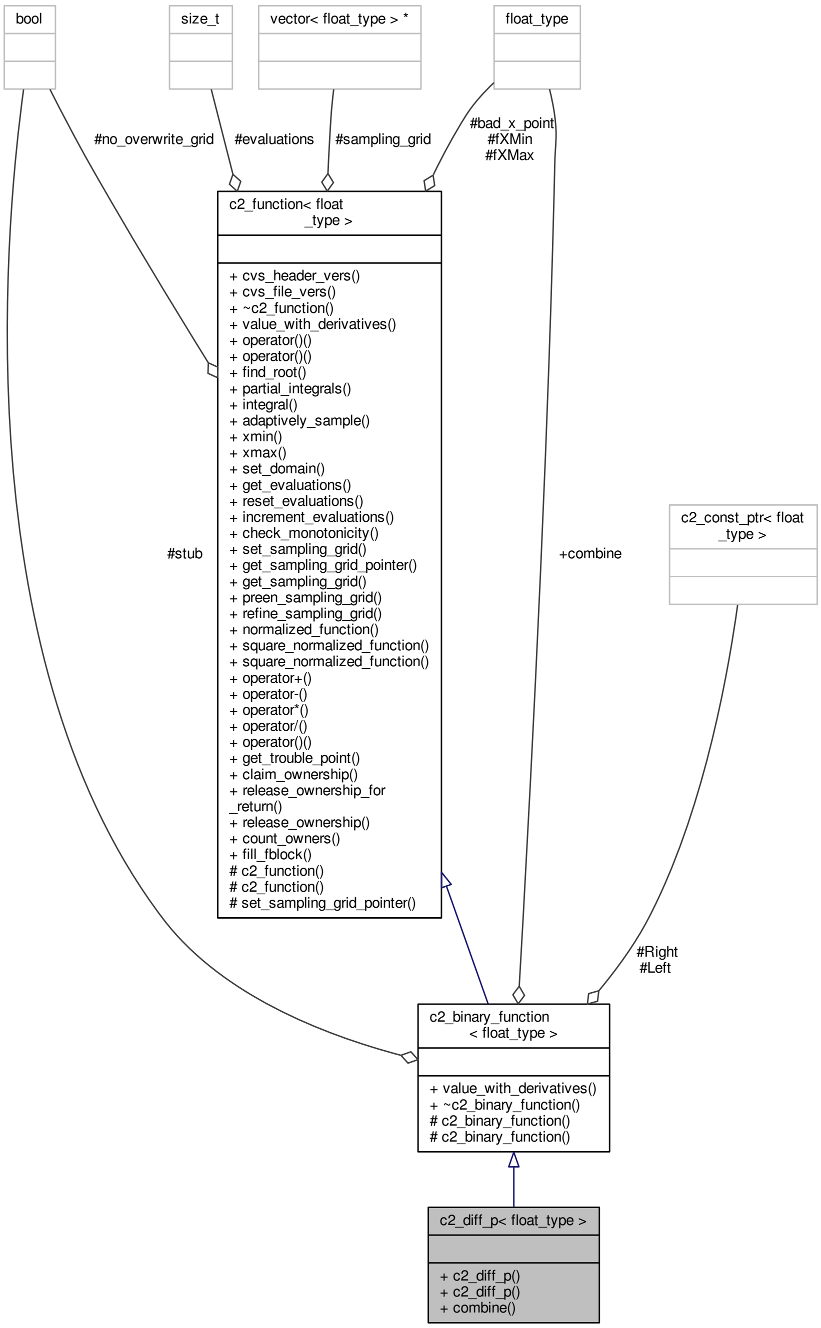
This should always be constructed using c2_function::operator-()

```
#include "c2_function.hh"
```

Inheritance diagram for c2_diff_p< float_type >:



Collaboration diagram for c2_diff_p< float_type >:



Public Member Functions

- c2_diff_p (const c2_function< float_type > &left, const c2_function< float_type > &right)

  construct left - right

- c2_diff_p ()

  Create a stub just for the combiner to avoid statics.

Static Public Member Functions

- static float_type combine (const c2_function< float_type > &left, const c2_function< float_type > &right, float_type x, float_type *yprime, float_type *yprime2) throw (c2_exception)

  execute math necessary to do subtraction

Additional Inherited Members

6.20.1 Detailed Description

template<typename float_type = double>
class c2_diff_p< float_type >

create a c2_function which is the difference of two other c2_functions.

This should always be constructed using c2_function::operator-()

6.20.2 Constructor & Destructor Documentation

6.20.2.1 template<typename float_type = double> c2_diff_p< float_type >::c2_diff_p (const c2_function< float_type > & *left*, const c2_function< float_type > & *right*) [inline]

construct *left* - *right*

Parameters

left	the left function
right	the right function

6.20.2.2 template<typename float_type = double> c2_diff_p< float_type >::c2_diff_p ( ) [inline]

Create a stub just for the combiner to avoid statics.

6.20.3 Member Function Documentation

6.20.3.1 template<typename float_type = double> static float_type c2_diff_p< float_type >::combine (const c2_function< float_type > & *left*, const c2_function< float_type > & *right*, float_type *x*, float_type * *yprime*, float_type * *yprime2*) throw c2_exception) [inline], [static]

execute math necessary to do subtraction

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

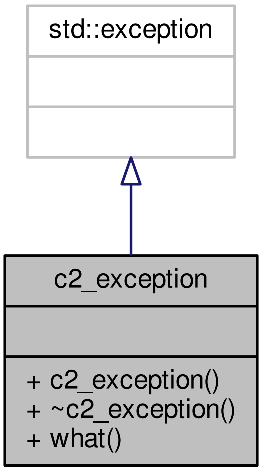
- c2_function.hh

6.21 c2_exception Class Reference

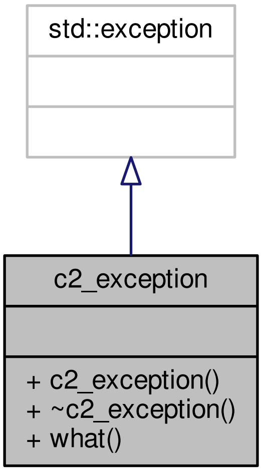
the exception class for c2_function operations.

```
#include "c2_function.hh"
```

Inheritance diagram for c2_exception:



Collaboration diagram for c2_exception:



Public Member Functions

- c2_exception (const char msgcode[ ])

  construct the exception with an error message
- virtual ∼c2_exception () throw ()
- virtual const char ∗ what () const throw ()

6.21.1 Detailed Description

the exception class for c2_function operations.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 c2_exception::c2_exception (const char *msgcode[ ]*) [inline]

construct the exception with an error message

Parameters

msgcode	the message

6.21.2.2 virtual c2_exception::∼c2_exception ( ) throw) [inline], [virtual]

6.21.3 Member Function Documentation

6.21.3.1 virtual const char∗ c2_exception::what ( ) const throw) [inline], [virtual]

Returns a C-style character string describing the general cause of the current error.

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

- c2_function.hh

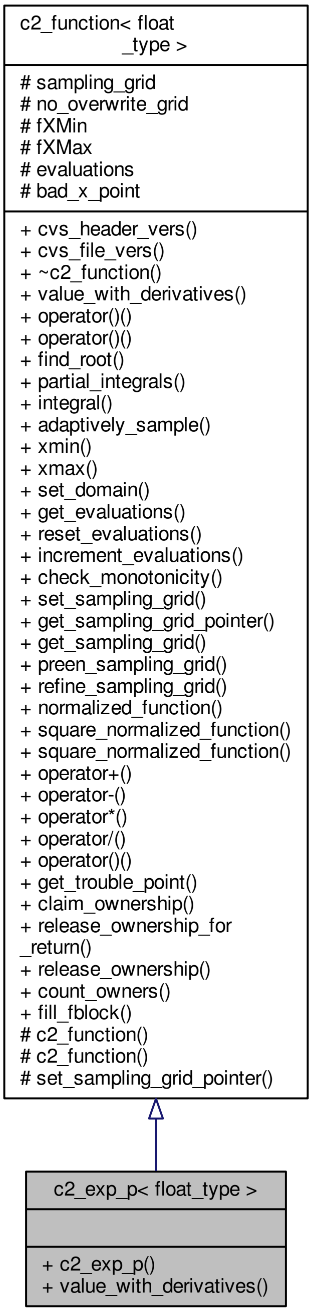
6.22 c2_exp_p< float_type > Class Template Reference

compute exp(x) with its derivatives.

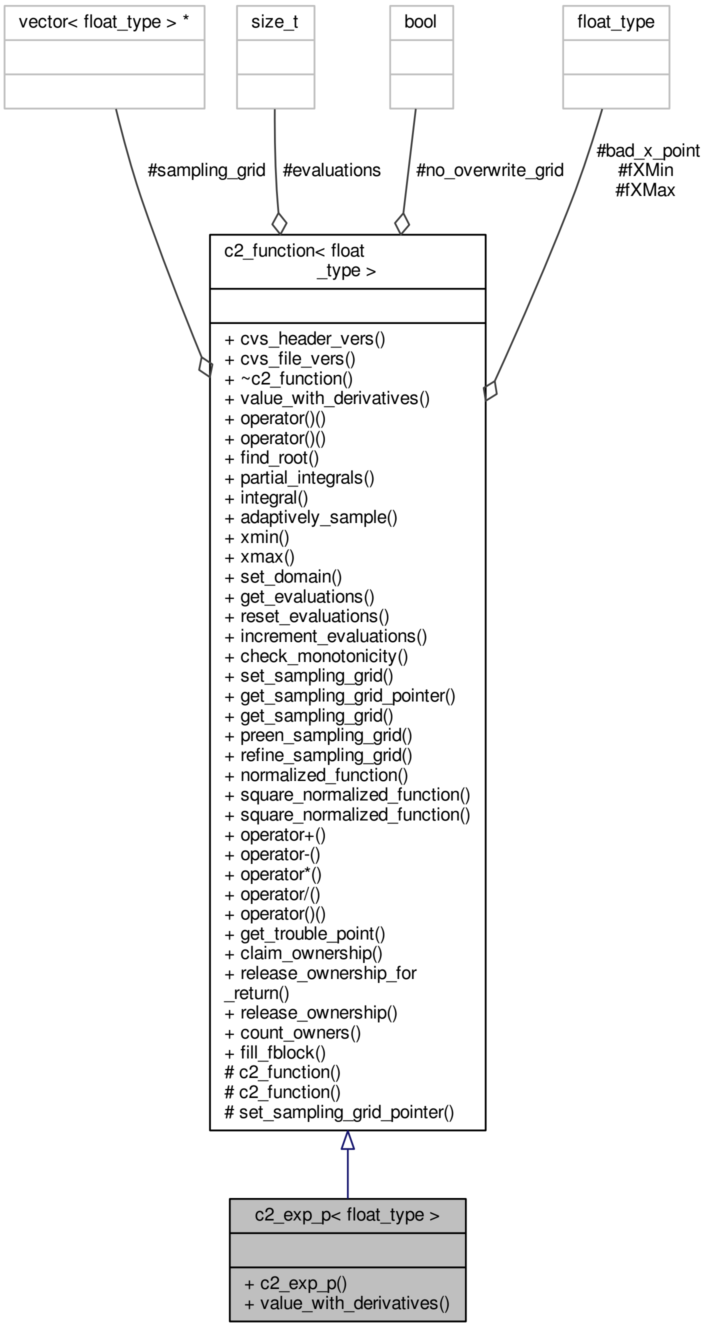
The factory function c2_factory::exp() creates ∗new c2_exp_p

```
#include "c2_function.hh"
```

Inheritance diagram for c2_exp_p< float_type >:



Collaboration diagram for c2_exp_p< float_type >:



Public Member Functions

- c2_exp_p ()

  constructor.
- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

  get the value and derivatives.

Additional Inherited Members

6.22.1 Detailed Description

template<typename float_type = double>
class c2_exp_p< float_type >

compute exp(x) with its derivatives.

The factory function c2_factory::exp() creates *new c2_exp_p

6.22.2 Constructor & Destructor Documentation

6.22.2.1 template<typename float_type = double> c2_exp_p< float_type >::c2_exp_p ( ) [inline]

constructor.

6.22.3 Member Function Documentation

6.22.3.1 template<typename float_type = double> virtual float_type c2_exp_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

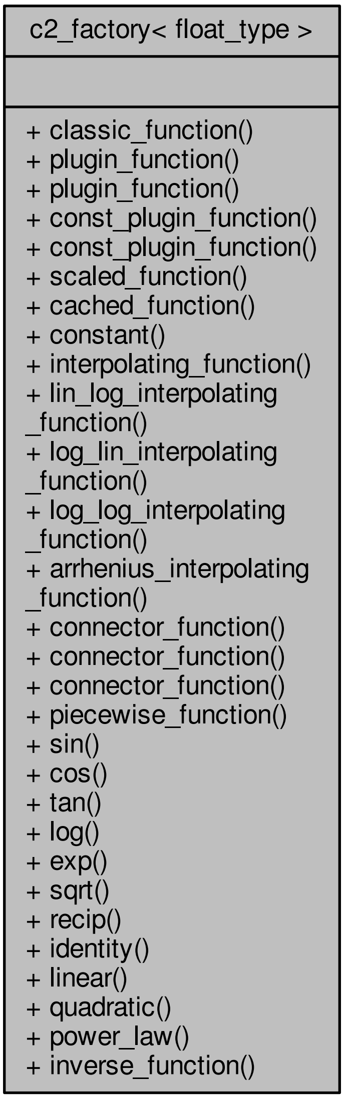
- c2_function.hh

6.23 c2_factory< float_type > Class Template Reference

a factory of pre-templated c2_function generators

```
#include "c2_factory.hh"
```

Collaboration diagram for c2_factory< float_type >:



Static Public Member Functions

- static c2_classic_function_p< float_type > & classic_function (float_type(∗c_func)(float_type))

    make a ∗new object
- static c2_plugin_function_p< float_type > & plugin_function ()

*make a *new object*
- static c2_plugin_function_p< float_type > & plugin_function (c2_function< float_type > &f)

  *make a *new object*
- static c2_const_plugin_function_p< float_type > & const_plugin_function ()

  *make a *new object*
- static c2_const_plugin_function_p< float_type > & const_plugin_function (const c2_function< float_type > &f)

  *make a *new object*
- static c2_scaled_function_p< float_type > & scaled_function (const c2_function< float_type > &outer, float_type scale)

  *make a *new object*
- static c2_cached_function_p< float_type > & cached_function (const c2_function< float_type > &func)

  *make a *new object*
- static c2_constant_p< float_type > & constant (float_type x)

  *make a *new object*
- static interpolating_function_p< float_type > & interpolating_function ()

  *make a *new object*
- static lin_log_interpolating_function_p< float_type > & lin_log_interpolating_function ()

  *make a *new object*
- static log_lin_interpolating_function_p< float_type > & log_lin_interpolating_function ()

  *make a *new object*
- static log_log_interpolating_function_p< float_type > & log_log_interpolating_function ()

  *make a *new object*
- static arrhenius_interpolating_function_p< float_type > & arrhenius_interpolating_function ()

  *make a *new object*
- static c2_connector_function_p< float_type > & connector_function (float_type x0, const c2_function< float_type > &f0, float_type x2, const c2_function< float_type > &f2, bool auto_center, float_type y1)

  *make a *new object*
- static c2_connector_function_p< float_type > & connector_function (const c2_fblock< float_type > &fb0, const c2_fblock< float_type > &fb2, bool auto_center, float_type y1)

  *make a *new object*
- static c2_connector_function_p< float_type > & connector_function (float_type x0, float_type y0, float_type yp0, float_type ypp0, float_type x2, float_type y2, float_type yp2, float_type ypp2, bool auto_center, float_type y1)

  *make a *new object*
- static c2_piecewise_function_p< float_type > & piecewise_function ()

  *make a *new object*
- static c2_sin_p< float_type > & sin ()

  *make a *new object*
- static c2_cos_p< float_type > & cos ()

  *make a *new object*
- static c2_tan_p< float_type > & tan ()

  *make a *new object*
- static c2_log_p< float_type > & log ()

  *make a *new object*
- static c2_exp_p< float_type > & exp ()

  *make a *new object*
- static c2_sqrt_p< float_type > & sqrt ()

  *make a *new object*
- static c2_recip_p< float_type > & recip (float_type scale=1)

  *make a *new object*

- static c2_identity_p< float_type > & identity ()
  make a ∗new object
- static c2_linear_p< float_type > & linear (float_type x0, float_type y0, float_type slope)
  make a ∗new object
- static c2_quadratic_p< float_type > & quadratic (float_type x0, float_type y0, float_type xcoef, float_type x2coef)
  make a ∗new object
- static c2_power_law_p< float_type > & power_law (float_type scale, float_type power)
  make a ∗new object
- static c2_inverse_function_p< float_type > & inverse_function (const c2_function< float_type > &source)
  make a ∗new object

6.23.1 Detailed Description

template<typename float_type>
class c2_factory< float_type >

a factory of pre-templated c2_function generators

do

```
typedef c2_ptr<double> c2_p;
static c2_factory<double> c2;
c2_p f=c2.sin();
```

Note

The factory class doesn't contain any data. It can be statically instantiated at the top of a file, and used everywhere inside, or even instantiated in your project's top-level include file.

See also

c2_math_factory

6.23.2 Member Function Documentation

6.23.2.1 template<typename float_type> static arrhenius_interpolating_function_p<float_type>& c2_factory< float_type >::arrhenius_interpolating_function ( ) [inline], [static]

make a ∗new object

6.23.2.2 template<typename float_type> static c2_cached_function_p<float_type>& c2_factory< float_type >::cached_function (const c2_function< float_type > & *func*) [inline], [static]

make a ∗new object

6.23.2.3 template<typename float_type> static c2_classic_function_p<float_type>& c2_factory< float_type >::classic_function (float_type(*)(float_type) *c_func*) [inline], [static]

make a *new object

6.23.2.4 template<typename float_type> static c2_connector_function_p<float_type>& c2_factory< float_type >::connector_function (float_type *x0,* const c2_function< float_type > & *f0,* float_type *x2,* const c2_function< float_type > & *f2,* bool *auto_center,* float_type *y1*) [inline], [static]

make a *new object

6.23.2.5 template<typename float_type> static c2_connector_function_p<float_type>& c2_factory< float_type >::connector_function (const c2_fblock< float_type > & *fb0,* const c2_fblock< float_type > & *fb2,* bool *auto_center,* float_type *y1*) [inline], [static]

make a *new object

6.23.2.6 template<typename float_type> static c2_connector_function_p<float_type>& c2_factory< float_type >::connector_function (float_type *x0,* float_type *y0,* float_type *yp0,* float_type *ypp0,* float_type *x2,* float_type *y2,* float_type *yp2,* float_type *ypp2,* bool *auto_center,* float_type *y1*) [inline], [static]

make a *new object

6.23.2.7 template<typename float_type> static c2_const_plugin_function_p<float_type>& c2_factory< float_type >::const_plugin_function ( ) [inline], [static]

make a *new object

6.23.2.8 template<typename float_type> static c2_const_plugin_function_p<float_type>& c2_factory< float_type >::const_plugin_function (const c2_function< float_type > & *f*) [inline], [static]

make a *new object

6.23.2.9 template<typename float_type> static c2_constant_p<float_type>& c2_factory< float_type >::constant (float_type *x*) [inline], [static]

make a *new object

6.23.2.10 template<typename float_type> static c2_cos_p<float_type>& c2_factory< float_type >::cos ( ) [inline], [static]

make a *new object

6.23.2.11 template<typename float_type> static c2_exp_p<float_type>& c2_factory< float_type >::exp ( ) [inline], [static]

make a *new object

6.23.2.12 template<typename float_type> static c2_identity_p<float_type>& c2_factory< float_type >::identity ( ) [inline], [static]

make a *new object

6.23.2.13 template<typename float_type> static interpolating_function_p<float_type>& c2_factory< float_type >::interpolating_function ( ) [inline], [static]

make a *new object

6.23.2.14 template<typename float_type> static c2_inverse_function_p<float_type>& c2_factory< float_type >::inverse_function (const c2_function< float_type > & *source*) [inline], [static]

make a *new object

6.23.2.15 template<typename float_type> static lin_log_interpolating_function_p<float_type>& c2_factory< float_type >::lin_log_interpolating_function ( ) [inline], [static]

make a *new object

6.23.2.16 template<typename float_type> static c2_linear_p<float_type>& c2_factory< float_type >::linear (float_type *x0*, float_type *y0*, float_type *slope*) [inline], [static]

make a *new object

6.23.2.17 template<typename float_type> static c2_log_p<float_type>& c2_factory< float_type >::log ( ) [inline], [static]

make a *new object

6.23.2.18 template<typename float_type> static log_lin_interpolating_function_p<float_type>& c2_factory< float_type >::log_lin_interpolating_function ( ) [inline], [static]

make a *new object

6.23.2.19 template<typename float_type> static log_log_interpolating_function_p<float_type>& c2_factory< float_type >::log_log_interpolating_function ( ) [inline], [static]

make a *new object

6.23.2.20 template<typename float_type> static c2_piecewise_function_p<float_type>& c2_factory< float_type >::piecewise_function ( ) [inline], [static]

make a *new object

6.23.2.21 template<typename float_type> static c2_plugin_function_p<float_type>& c2_factory< float_type >::plugin_function ( ) [inline], [static]

make a *new object

6.23.2.22 template<typename float_type> static c2_plugin_function_p<float_type>& c2_factory< float_type >::plugin_function (c2_function< float_type > & *f*) [inline], [static]

make a *new object

6.23.2.23 template<typename float_type> static c2_power_law_p<float_type>& c2_factory< float_type >::power_law (float_type *scale*, float_type *power*) [inline], [static]

make a *new object

6.23.2.24 template<typename float_type> static c2_quadratic_p<float_type>& c2_factory< float_type >::quadratic (float_type *x0*, float_type *y0*, float_type *xcoef*, float_type *x2coef*) [inline], [static]

make a *new object

6.23.2.25 template<typename float_type> static c2_recip_p<float_type>& c2_factory< float_type >::recip (float_type *scale* = 1) [inline], [static]

make a *new object

6.23.2.26 template<typename float_type> static c2_scaled_function_p<float_type>& c2_factory< float_type >::scaled_function (const c2_function< float_type > & *outer*, float_type *scale*) [inline], [static]

make a *new object

6.23.2.27 template<typename float_type> static c2_sin_p<float_type>& c2_factory< float_type >::sin ( ) [inline], [static]

make a *new object

6.23.2.28 template<typename float_type> static c2_sqrt_p<float_type>& c2_factory< float_type >::sqrt ( ) [inline], [static]

make a *new object

6.23.2.29 template<typename float_type> static c2_tan_p<float_type>& c2_factory< float_type >::tan ( ) [inline], [static]

make a ∗new object

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

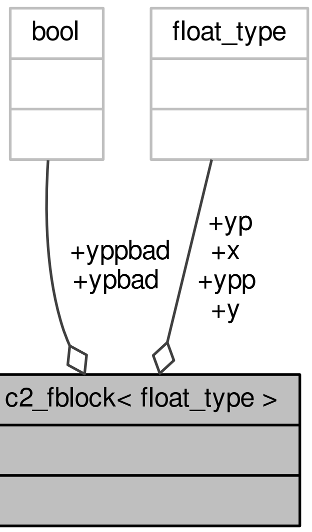
- c2_factory.hh

6.24 c2_fblock< float_type > Class Template Reference

structure used to hold evaluated function data at a point.

`#include "c2_function.hh"`

Collaboration diagram for c2_fblock< float_type >:



Public Attributes

- float_type x

    the abscissa
- float_type y

    the value of the function at x
- float_type yp

    the derivative at x
- float_type ypp

    the second derivative at x
- bool ypbad

    flag, filled in by c2_function::fill_fblock(), indicating the derivative is NaN of Inf
- bool yppbad

    flag, filled in by c2_function::fill_fblock(), indicating the second derivative is NaN of Inf

6.24.1 Detailed Description

template<typename float_type>
class c2_fblock< float_type >

structure used to hold evaluated function data at a point.

Contains all the information for the function at one point.

6.24.2 Member Data Documentation

6.24.2.1 template<typename float_type> float_type c2_fblock< float_type >::x

the abscissa

6.24.2.2 template<typename float_type> float_type c2_fblock< float_type >::y

the value of the function at *x*

6.24.2.3 template<typename float_type> float_type c2_fblock< float_type >::yp

the derivative at *x*

6.24.2.4 template<typename float_type> bool c2_fblock< float_type >::ypbad

flag, filled in by c2_function::fill_fblock(), indicating the derivative is NaN of Inf

6.24.2.5 template<typename float_type> float_type c2_fblock< float_type >::ypp

the second derivative at *x*

6.24.2.6 template<typename float_type> bool c2_fblock< float_type >::yppbad

flag, filled in by c2_function::fill_fblock(), indicating the second derivative is NaN of Inf

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

- c2_function.hh

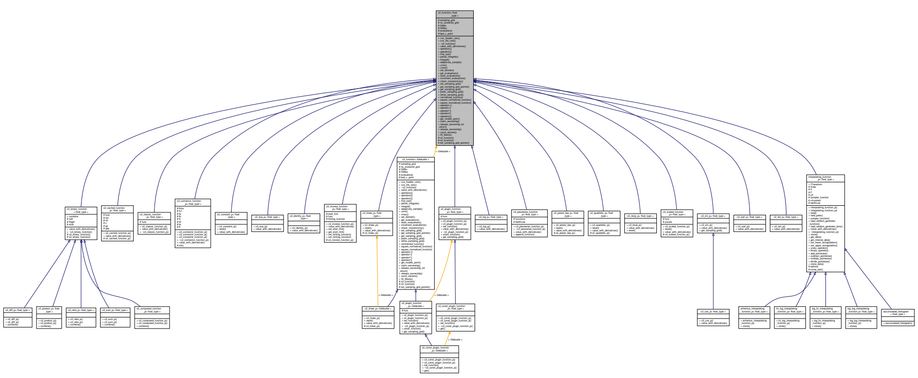
6.25 c2_function< float_type > Class Template Reference

the parent class for all c2_functions.

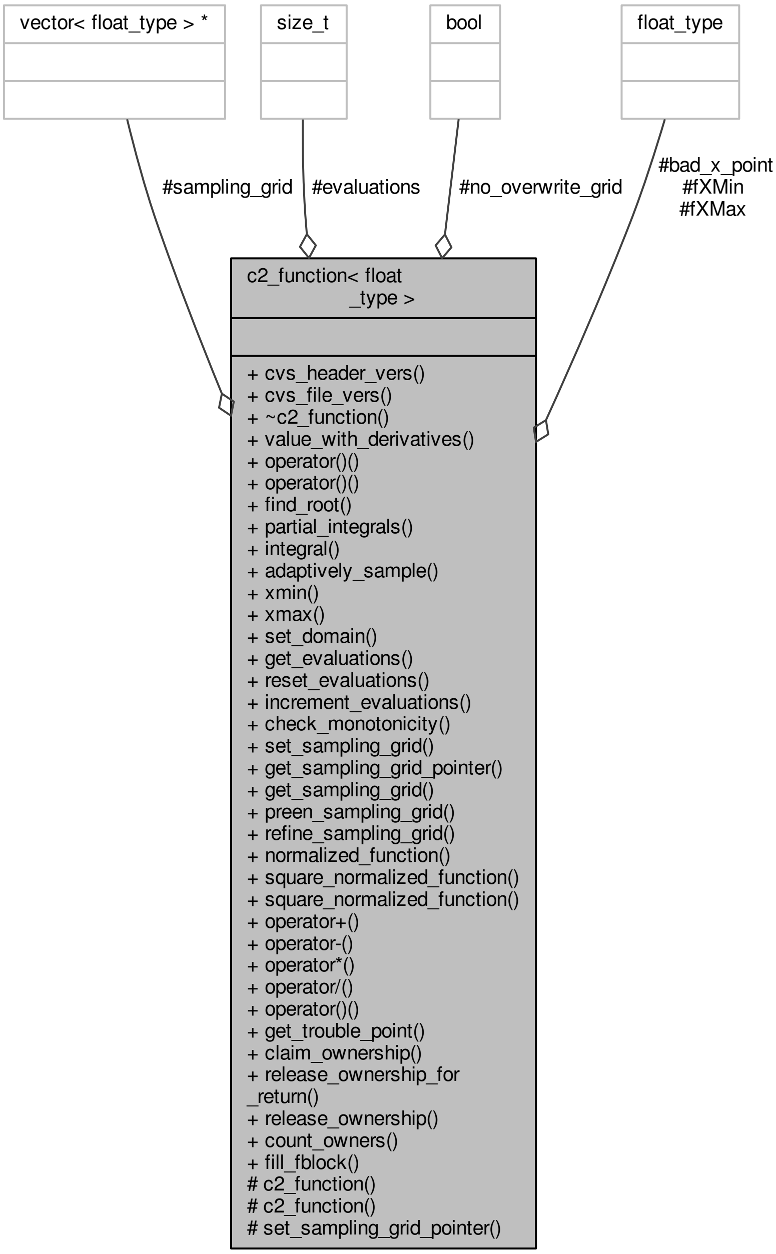
c2_functions know their value, first, and second derivative at almost every point. They can be efficiently combined with binary operators, via c2_binary_function, composed via c2_composed_function_, have their roots found via find_root(), and be adaptively integrated via partial_integrals() or integral(). They also can carry information with them about how to find 'interesting' points on the function. This information is set with set_sampling_grid() and extracted with get_sampling_grid().

```
#include "c2_function.hh"
```

Inheritance diagram for c2_function< float_type >:



Collaboration diagram for c2_function< float_type >:



Public Member Functions

- const std::string cvs_header_vers () const

  get versioning information for the header file
- const std::string cvs_file_vers () const

  get versioning information for the source file
- virtual ∼c2_function ()

  destructor

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const =0 throw (c2_exception)

  get the value and derivatives.
- float_type operator() (float_type x) const throw (c2_exception)

  evaluate the function in the classic way, ignoring derivatives.
- float_type operator() (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

  get the value and derivatives.
- float_type find_root (float_type lower_bracket, float_type upper_bracket, float_type start, float_type value, int ∗error=0, float_type ∗final_yprime=0, float_type ∗final_yprime2=0) const throw (c2_exception)

  solve f(x)==value very efficiently, with explicit knowledge of derivatives of the function
- float_type partial_integrals (std::vector< float_type > xgrid, std::vector< float_type > ∗partials=0, float_type abs_tol=1e-12, float_type rel_tol=1e-12, int derivs=2, bool adapt=true, bool extrapolate=true) const throw (c2_exception)

  for points in xgrid, adaptively return Integral[f(x),{x,xgrid[i],xgrid[i+1]}] and return in vector, along with sum
- float_type integral (float_type amin, float_type amax, std::vector< float_type > ∗partials=0, float_type abs↩_tol=1e-12, float_type rel_tol=1e-12, int derivs=2, bool adapt=true, bool extrapolate=true) const throw (c2_↩exception)

  a fully-automated integrator which uses the information provided by the get_sampling_grid() function to figure out what to do.
- c2_piecewise_function_p< float_type > ∗ adaptively_sample (float_type amin, float_type amax, float_type abs_tol=1e-12, float_type rel_tol=1e-12, int derivs=2, std::vector< float_type > ∗xvals=0, std::vector< float↩_type > ∗yvals=0) const throw (c2_exception)

  create a c2_piecewise_function_p from c2_connector_function_p segments which is a representation of the parent function to the specified accuracy, but maybe much cheaper to evaluate
- float_type xmin () const

  return the lower bound of the domain for this function as set by set_domain()
- float_type xmax () const

  return the upper bound of the domain for this function as set by set_domain()
- void set_domain (float_type amin, float_type amax)

  set the domain for this function.
- size_t get_evaluations () const

  this is a counter owned by the function but which can be used to monitor efficiency of algorithms.
- void reset_evaluations () const

  reset the counter
- void increment_evaluations () const

  count evaluations
- bool check_monotonicity (const std::vector< float_type > &data, const char message[ ]) const throw (c2_↩exception)

  check that a vector is monotonic, throw an exception if not, and return a flag if it is reversed
- virtual void set_sampling_grid (const std::vector< float_type > &grid) throw (c2_exception)

  establish a grid of 'interesting' points on the function.
- std::vector< float_type > ∗ get_sampling_grid_pointer () const

  get the sampling grid, which may be a null pointer
- virtual void get_sampling_grid (float_type amin, float_type amax, std::vector< float_type > &grid) const

  return the grid of 'interesting' points along this function which lie in the region requested
- void preen_sampling_grid (std::vector< float_type > ∗result) const

  clean up endpoints on a grid of points
- void refine_sampling_grid (std::vector< float_type > &grid, size_t refinement) const

  refine a grid by splitting each interval into more intervals
- c2_function< float_type > & normalized_function (float_type amin, float_type amax, float_type norm=1.0) const throw (c2_exception)

  create a new c2_function from this one which is normalized on the interval

- c2_function< float_type > & square_normalized_function (float_type amin, float_type amax, float_type norm=1.0) const throw (c2_exception)

    create a new c2_function from this one which is square-normalized on the interval
- c2_function< float_type > & square_normalized_function (float_type amin, float_type amax, const c2_function< float_type > &weight, float_type norm=1.0) const throw (c2_exception)

    create a new c2_function from this one which is square-normalized with the provided weight on the interval
- c2_sum_p< float_type > & operator+ (const c2_function< float_type > &rhs) const

    factory function to create a c2_sum_p from a regular algebraic expression.
- c2_diff_p< float_type > & operator- (const c2_function< float_type > &rhs) const

    factory function to create a c2_diff_p from a regular algebraic expression.
- c2_product_p< float_type > & operator* (const c2_function< float_type > &rhs) const

    factory function to create a c2_product_p from a regular algebraic expression.
- c2_ratio_p< float_type > & operator/ (const c2_function< float_type > &rhs) const

    factory function to create a c2_ratio_p from a regular algebraic expression.
- c2_composed_function_p< float_type > & operator() (const c2_function< float_type > &inner) const

    compose this function outside another.
- float_type get_trouble_point () const

    Find out where a calculation ran into trouble, if it got a nan. If the most recent computation did not return a nan, this is undefined.
- void claim_ownership () const

    increment our reference count. Destruction is only legal if the count is zero.
- size_t release_ownership_for_return () const throw (c2_exception)

    decrement our reference count. Do not destroy at zero.
- void release_ownership () const throw (c2_exception)

    decrement our reference count. If the count reaches zero, destroy ourself.
- size_t count_owners () const

    get the reference count, mostly for debugging
- void fill_fblock (c2_fblock< float_type > &fb) const throw (c2_exception)

    fill in a c2_fblock<float_type>... a shortcut for the integrator & sampler

Protected Member Functions

- c2_function (const c2_function< float_type > &src)
- c2_function ()
- virtual void set_sampling_grid_pointer (std::vector< float_type > &grid)

Protected Attributes

- std::vector< float_type > * sampling_grid
- bool no_overwrite_grid
- float_type fXMin
- float_type fXMax
- size_t evaluations
- float_type bad_x_point

    this point may be used to record where a calculation ran into trouble

6.25.1 Detailed Description

template<typename float_type = double>
class c2_function< float_type >

the parent class for all c2_functions.

c2_functions know their value, first, and second derivative at almost every point. They can be efficiently combined with binary operators, via c2_binary_function, composed via c2_composed_function_, have their roots found via find_root(), and be adaptively integrated via partial_integrals() or integral(). They also can carry information with them about how to find 'interesting' points on the function. This information is set with set_sampling_grid() and extracted with get_sampling_grid().

Particularly important subclasses are the interpolating functions classes, interpolating_function , lin_log_↩ interpolating_function, log_lin_interpolating_function, log_log_interpolating_function, and arrhenius_interpolating↩ _function, as well as the template functions inverse_integrated_density_function().

For a discussion of memory management, see memory_management

6.25.2 Constructor & Destructor Documentation

6.25.2.1 template<typename float_type = double> virtual c2_function< float_type >::∼c2_function ( ) [inline], [virtual]

destructor

6.25.2.2 template<typename float_type = double> c2_function< float_type >::c2_function (const c2_function< float_type > & *src*) [inline], [protected]

6.25.2.3 template<typename float_type = double> c2_function< float_type >::c2_function ( ) [inline], [protected]

6.25.3 Member Function Documentation

6.25.3.1 template<typename float_type = double> c2_piecewise_function_p<float_type>* c2_function< float_type >::adaptively_sample (float_type *amin*, float_type *amax*, float_type *abs_tol* = 1e-12, float_type *rel_tol* = 1e-12, int *derivs* = 2, std::vector< float_type > * *xvals* = 0, std::vector< float_type > * *yvals* = 0) const throw c2_exception)

create a c2_piecewise_function_p from c2_connector_function_p segments which is a representation of the parent function to the specified accuracy, but maybe much cheaper to evaluate

This method has three modes, depending on the *derivs* flag.

If *derivs* is 2, it computes a c2_piecewise_function_p representation of its parent function, which may be a much faster function to use in codes if the parent function is expensive. If *xvals* and *yvals* are non-null, it will also fill them in with the function values at each grid point the adaptive algorithm chooses.

If *derivs* is 1, this does not create the connectors, and returns an null pointer, but will fill in the *xvals* and *yvals* vectors with values of the function at points such that the linear interpolation error between the points is bounded by the tolerance values given. Because it uses derivative information from the function to manage the error control, it is almost completely free of issues with missing periods of oscillatory functions, even with no information provided in the sampling grid. This is typically useful for sampling a function for plotting.

If *derivs* is 0, this does something very like what it does if *derivs* = 1, but without derivatives. Instead, to compute the intermediate value of the function for error control, it just uses 3-point parabolic interpolation. This is useful amost exclusively for converting a non-c2_function, with no derivatives, but wrapped in a c2_classic_function wrapper, into a table of values to seed an interpolating_function_p. Note, however, that without derivatives, this is very susceptible to missing periods of oscillatory functions, so it is important to set a sampling grid which isn't too much coarser than the typical oscillations.

Note

the *sampling_grid* of the returned function matches the *sampling_grid* of its parent.

See also

Adaptive Sampling Examples

Parameters

	amin	lower bound of the domain for sampling
	amax	upper bound of the domain for sampling
	abs_tol	the absolute error bound for each segment
	rel_tol	the fractional error bound for each segment.
	derivs	if 0 or 1, return a useless function, but fill in the *xvals* and *yvals* vectors (if non-null). Also, if 0 or 1, tolerances refer to linear interpolation, not high-order interpolation. If 2, return a full piecewise collection of c2_connector_function_p segments. See discussion above.
`in,out`	*xvals*	vector of abscissas at which the function was actually sampled (if non-null)
`in,out`	*yvals*	vector of function values corresponding to *xvals* (if non-null)

Returns

a new, sampled representation, if *derivs* is 2. A null pointer if *derivs* is 0 or 1.

6.25.3.2 template<typename float_type = double> bool c2_function< float_type >::check_monotonicity (const std::vector< float_type > & *data*, const char *message[ ]*) const throw c2_exception)

check that a vector is monotonic, throw an exception if not, and return a flag if it is reversed

Parameters

data	a vector of data points which are expected to be monotonic.
message	an informative string to include in an exception if this throws c2_exception

Returns

true if in decreasing order, false if increasing

6.25.3.3 template<typename float_type = double> void c2_function< float_type >::claim_ownership ( ) const `[inline]`

increment our reference count. Destruction is only legal if the count is zero.

6.25.3.4 template<typename float_type = double> size_t c2_function< float_type >::count_owners ( ) const `[inline]`

get the reference count, mostly for debugging

Returns

the count

6.25.3.5 template<typename float_type = double> const std::string c2_function< float_type >::cvs_file_vers ( ) const

get versioning information for the source file

Returns

the CVS Id string

6.25.3.6 template<typename float_type = double> const std::string c2_function< float_type >::cvs_header_vers ( ) const [inline]

get versioning information for the header file

Returns

the CVS Id string

6.25.3.7 template<typename float_type = double> void c2_function< float_type >::fill_fblock (c2_fblock< float_type > & *fb*) const throw c2_exception) [inline]

fill in a c2_fblock<float_type>... a shortcut for the integrator & sampler

Parameters

in,out	*fb*	the block to fill in with information

6.25.3.8 template<typename float_type = double> float_type c2_function< float_type >::find_root (float_type *lower_bracket,* float_type *upper_bracket,* float_type *start,* float_type *value,* int ∗ *error =* 0*,* float_type ∗ *final_yprime =* 0*,* float_type ∗ *final_yprime2 =* 0) const throw c2_exception)

solve f(x)==value very efficiently, with explicit knowledge of derivatives of the function

find_root solves by iterated inverse quadratic extrapolation for a solution to f(x)=y. It includes checks against bad convergence, so it should never be able to fail. Unlike typical secant method or fancier Brent's method finders, this does not depend in any strong wasy on the brackets, unless the finder has to resort to successive approximations to close in on a root. Often, it is possible to make the brackets equal to the domain of the function, if there is any clue as to where the root lies, as given by the parameter *start*.

Parameters

	lower_bracket	the lower bound for the search
	upper_bracket	the upper bound for the search. Function sign must be opposite to that at *lower_bracket*
	start	starting value for the search

Parameters

	value	the value of the function being sought (solves f(x) = *value*)
out	*error*	If pointer is zero, errors raise exception. Otherwise, returns error here.
out	*final_yprime*	If pointer is not zero, return derivative of function at root
out	*final_yprime2*	If pointer is not zero, return second derivative of function at root

Returns

the position of the root.

See also

Root finding sample

6.25.3.9 template<typename float_type = double> size_t **c2_function**< float_type >::get_evaluations ( ) const [inline]

this is a counter owned by the function but which can be used to monitor efficiency of algorithms.

It is not maintained automatically in general! The root finder, integrator, and sampler do increment it.

Returns

number of evaluations logged since last reset.

6.25.3.10 template<typename float_type = double> virtual void **c2_function**< float_type >::get_sampling_grid (float_type *amin*, float_type *amax*, std::vector< float_type > & *grid*) const [virtual]

return the grid of 'interesting' points along this function which lie in the region requested

if a sampling grid is defined, work from there, otherwise return vector of (amin, amax)

Parameters

	amin	the lower bound for which the function is to be sampled
	amax	the upper bound for which the function is to be sampled
in,out	*grid*	filled vector containing the samplng grid.

Reimplemented in c2_sin_p< float_type >, c2_plugin_function_p< float_type >, and c2_plugin_function_p< G4double >.

6.25.3.11 template<typename float_type = double> std::vector<float_type>* **c2_function**< float_type >::get_sampling_grid_pointer ( ) const [inline]

get the sampling grid, which may be a null pointer

Returns

pointer to the sampling grid

6.25.3.12 template<typename float_type = double> float_type c2_function< float_type >::get_trouble_point ( ) const [inline]

Find out where a calculation ran into trouble, if it got a nan. If the most recent computation did not return a nan, this is undefined.

Returns

x value of point at which something went wrong, if integrator (or otherwise) returned a nan.

6.25.3.13 template<typename float_type = double> void c2_function< float_type >::increment_evaluations ( ) const [inline]

count evaluations

6.25.3.14 template<typename float_type = double> float_type c2_function< float_type >::integral (float_type *amin*, float_type *amax*, std::vector< float_type > * *partials* = 0, float_type *abs_tol* = 1e-12, float_type *rel_tol* = 1e-12, int *derivs* = 2, bool *adapt* = true, bool *extrapolate* = true) const throw c2_exception)

a fully-automated integrator which uses the information provided by the get_sampling_grid() function to figure out what to do.

It returns the integral of the function over the domain requested with error tolerances as specified. It is just a front-end to partial_integrals()

Parameters

amin	lower bound of the domain for integration
amax	upper bound of the domain for integration
partials	if non-NULL, a vector in which to receive the partial integrals. It will automatically be sized appropriately, if provided, to contain *n* - 1 elements where *n* is the length of *xgrid*
abs_tol	the absolute error bound for each segment
rel_tol	the fractional error bound for each segment. If the error is smaller than either the relative or absolute tolerance, the integration step is finished.
derivs	number of derivatives to trust, which sets the order of the integrator. The order is 3∗*derivs* + 4. *derivs* can be 0, 1, or 2.
adapt	if true, use recursive adaptation, otherwise do simple evaluation on the grid provided with no error checking.
extrapolate	if true, use simple Richardson extrapolation on the final 2 steps to reduce the error.

Returns

sum of partial integrals, which is the definite integral from the first value in *xgrid* to the last.

6.25.3.15 template<typename float_type = double> c2_function<float_type>& c2_function< float_type >::normalized_function (float_type *amin*, float_type *amax*, float_type *norm* = 1.0) const throw c2_exception)

create a new c2_function from this one which is normalized on the interval

Parameters

amin	lower bound of the domain for integration
amax	upper bound of the domain for integration
norm	the desired integral for the function over the region

Returns

a new c2_function with the desired *norm*.

6.25.3.16 template<typename float_type = double> float_type c2_function< float_type >::operator() (float_type *x*) const throw c2_exception) [inline]

evaluate the function in the classic way, ignoring derivatives.

Parameters

x	the point at which to evaluate

Returns

the value of the function

6.25.3.17 template<typename float_type = double> float_type c2_function< float_type >::operator() (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline]

get the value and derivatives.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

6.25.3.18 template<typename float_type = double> c2_composed_function_p<float_type>& c2_function< float_type >::operator() (const c2_function< float_type > & *inner*) const [inline]

compose this function outside another.

Parameters

inner	the inner function

Returns

the composed function

6.25.3.19 template<typename float_type = double> c2_product_p<float_type>& c2_function< float_type >::operator∗ (const c2_function< float_type > & *rhs*) const [inline]

factory function to create a c2_product_p from a regular algebraic expression.

Parameters

rhs	the right-hand term of the product

Returns

a new c2_function

6.25.3.20 template<typename float_type = double> c2_sum_p<float_type>& c2_function< float_type >::operator+ (const c2_function< float_type > & *rhs*) const [inline]

factory function to create a c2_sum_p from a regular algebraic expression.

Parameters

rhs	the right-hand term of the sum

Returns

a new c2_function

6.25.3.21 template<typename float_type = double> c2_diff_p<float_type>& c2_function< float_type >::operator- (const c2_function< float_type > & *rhs*) const [inline]

factory function to create a c2_diff_p from a regular algebraic expression.

Parameters

rhs	the right-hand term of the difference

Returns

a new c2_function

6.25.3.22 template< typename float_type = double > c2_ratio_p< float_type >& c2_function< float_type >::operator/ (const c2_function< float_type > & *rhs*) const [inline]

factory function to create a c2_ratio_p from a regular algebraic expression.

Parameters

rhs	the right-hand term of the ratio (the denominator)

Returns

a new c2_function

6.25.3.23 template< typename float_type = double > float_type c2_function< float_type >::partial_integrals (std::vector< float_type > *xgrid*, std::vector< float_type > * *partials* = 0, float_type *abs_tol* = 1e-12, float_type *rel_tol* = 1e-12, int *derivs* = 2, bool *adapt* = true, bool *extrapolate* = true) const throw c2_exception)

for points in xgrid, adaptively return Integral[f(x),{x,xgrid[i],xgrid[i+1]}] and return in vector, along with sum

partial_integrals uses a method with an error O(dx**10) with full information from the derivatives, and falls back to lower order methods if informed of incomplete derivatives. It uses exact midpoint splitting of the intervals for recursion, resulting in no recomputation of the function during recursive descent at previously computed points.

Parameters

xgrid	points between which to evaluate definite integrals.
partials	if non-NULL, a vector in which to receive the partial integrals. It will automatically be sized apprpropriately, if provided, to contain *n* - 1 elements where *n* is the length of *xgrid*
abs_tol	the absolute error bound for each segment
rel_tol	the fractional error bound for each segment. If the error is smaller than either the relative or absolute tolerance, the integration step is finished.
derivs	number of derivatives to trust, which sets the order of the integrator. The order is 3*derivs + 4. *derivs* can be 0, 1, or 2.
adapt	if true, use recursive adaptation, otherwise do simple evaluation on the grid provided with no error checking.
extrapolate	if true, use simple Richardson extrapolation on the final 2 steps to reduce the error.

Returns

sum of partial integrals, which is the definite integral from the first value in *xgrid* to the last.

6.25.3.24 template< typename float_type = double > void c2_function< float_type >::preen_sampling_grid (std::vector< float_type > * *result*) const

clean up endpoints on a grid of points

Parameters

in,out	*result*	the sampling grid with excessively closely space endpoints removed. The grid is modified in place.

6.25.3.25 template<typename float_type = double> void c2_function< float_type >::refine_sampling_grid (std::vector< float_type > & *grid,* size_t *refinement*) const

refine a grid by splitting each interval into more intervals

Parameters

in,out	*grid*	the grid to refine in place
	refinement	the number of new steps for each old step

6.25.3.26 template<typename float_type = double> void c2_function< float_type >::release_ownership ( ) const throw c2_exception) [inline]

decrement our reference count. If the count reaches zero, destroy ourself.

6.25.3.27 template<typename float_type = double> size_t c2_function< float_type >::release_ownership_for_return ( ) const throw c2_exception) [inline]

decrement our reference count. Do not destroy at zero.

Returns

final owner count, to check whether object should disappear.

6.25.3.28 template<typename float_type = double> void c2_function< float_type >::reset_evaluations ( ) const [inline]

reset the counter

6.25.3.29 template<typename float_type = double> void c2_function< float_type >::set_domain (float_type *amin,* float_type *amax*) [inline]

set the domain for this function.

6.25.3.30 template<typename float_type = double> virtual void c2_function< float_type >::set_sampling_grid (const std::vector< float_type > & *grid*) throw c2_exception) [virtual]

establish a grid of 'interesting' points on the function.

The sampling grid describes a reasonable initial set of points to look at the function. this should generally be set at a scale which is quite coarse, and sufficient for initializing adaptive integration or possibly root bracketing. For sampling a function to build a new interpolating function, one may want to refine this for accuracy. However, interpolating_functions themselves return their original X grid by default, so refining the grid in this case might be a bad idea.

Parameters

grid	a vector of abscissas. The contents is copied into an internal vector, so the *grid* can be discarded after passingin.

6.25.3.31 template<typename float_type = double> virtual void c2_function< float_type >::set_sampling_grid_pointer (std::vector< float_type > & *grid*) [inline], [protected], [virtual]

6.25.3.32 template<typename float_type = double> c2_function<float_type>& c2_function< float_type >::square_normalized_function (float_type *amin*, float_type *amax*, float_type *norm* = 1.0) const throw c2_exception)

create a new c2_function from this one which is square-normalized on the interval

Parameters

amin	lower bound of the domain for integration
amax	upper bound of the domain for integration
norm	the desired integral for the function over the region

Returns

a new c2_function with the desired *norm*.

6.25.3.33 template<typename float_type = double> c2_function<float_type>& c2_function< float_type >::square_normalized_function (float_type *amin*, float_type *amax*, const c2_function< float_type > & *weight*, float_type *norm* = 1.0) const throw c2_exception)

create a new c2_function from this one which is square-normalized with the provided *weight* on the interval

Parameters

amin	lower bound of the domain for integration
amax	upper bound of the domain for integration
weight	a c2_function providing the weight
norm	the desired integral for the function over the region

Returns

a new c2_function with the desired *norm*.

6.25.3.34 template<typename float_type = double> virtual float_type c2_function< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [pure virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implemented in c2_piecewise_function_p< float_type >, c2_connector_function_p< float_type >, c2_inverse_function_p< float_type >, c2_power_law_p< float_type >, c2_quadratic_p< float_type >, c2_linear_p< float_type >, c2_linear_p< G4double >, c2_identity_p< float_type >, c2_recip_p< float_type >, c2_sqrt_p< float_type >, c2_exp_p< float_type >, c2_log_p< float_type >, c2_tan_p< float_type >, c2_cos_p< float_type >, c2_sin_p< float_type >, interpolating_function_p< float_type >, c2_constant_p< float_type >, c2_cached_function_p< float_type >, c2_scaled_function_p< float_type >, c2_binary_function< float_type >, c2_plugin_function_p< float_type >, c2_plugin_function_p< G4double >, and c2_classic_function_p< float_type >.

6.25.3.35 template<typename float_type = double> float_type c2_function< float_type >::xmax ( ) const [inline]

return the upper bound of the domain for this function as set by set_domain()

6.25.3.36 template<typename float_type = double> float_type c2_function< float_type >::xmin ( ) const [inline]

return the lower bound of the domain for this function as set by set_domain()

6.25.4 Member Data Documentation

6.25.4.1 template<typename float_type = double> float_type c2_function< float_type >::bad_x_point [mutable], [protected]

this point may be used to record where a calculation ran into trouble

6.25.4.2 template<typename float_type = double> size_t c2_function< float_type >::evaluations [mutable], [protected]

6.25.4.3 template<typename float_type = double> float_type c2_function< float_type >::fXMax [protected]

6.25.4.4 template<typename float_type = double> float_type c2_function< float_type >::fXMin [protected]

6.25.4.5 template<typename float_type = double> bool c2_function< float_type >::no_overwrite_grid [protected]

6.25.4.6 template<typename float_type = double> std::vector<float_type>* c2_function< float_type >::sampling_grid [protected]

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

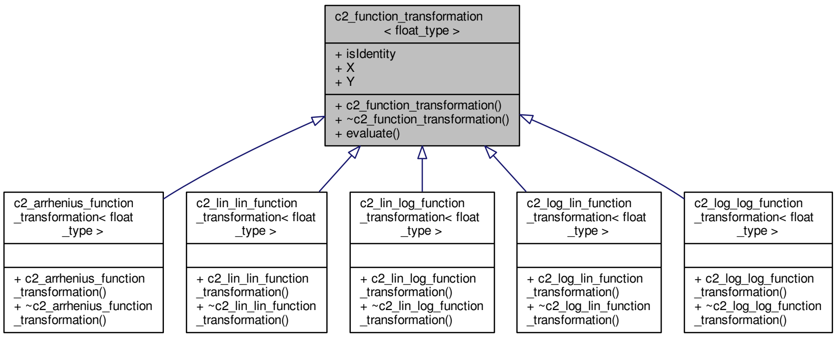
- c2_function.hh

6.26 c2_function_transformation< float_type > Class Template Reference

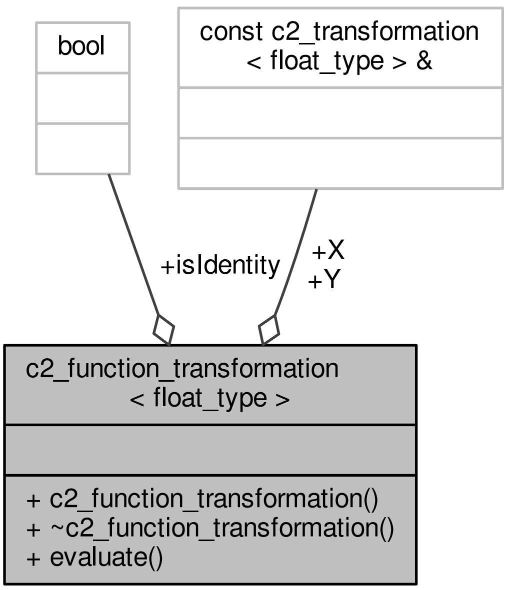
a transformation of a function in and out of a coordinate space, using 2 c2_transformations

```
#include "c2_function.hh"
```

Inheritance diagram for c2_function_transformation< float_type >:



Collaboration diagram for c2_function_transformation< float_type >:



Public Member Functions

- c2_function_transformation (const c2_transformation< float_type > &xx, const c2_transformation< float_type > &yy)

construct this from two c2_transformation instances

- virtual ~c2_function_transformation ()

  destructor

- virtual float_type evaluate (float_type xraw, float_type y, float_type yp0, float_type ypp0, float_type ∗yprime, float_type ∗yprime2) const

  evaluate the transformation from internal coordinates to external coordinates

Public Attributes

- const bool isIdentity

  flag indicating of the transform is the identity, and can be skipped for efficiency

- const c2_transformation< float_type > & X

  the X axis transform

- const c2_transformation< float_type > & Y

  the Y axis transform

6.26.1 Detailed Description

template<typename float_type>
class c2_function_transformation< float_type >

a transformation of a function in and out of a coordinate space, using 2 c2_transformations

This class is a container for two axis transforms, but also provides the critical evaluate() function which converts a result in internal coordinates (with derivatives) into the external representation

6.26.2 Constructor & Destructor Documentation

6.26.2.1 template<typename float_type> c2_function_transformation< float_type >::c2_function_transformation (const c2_transformation< float_type > & *xx*, const c2_transformation< float_type > & *yy*) [inline]

construct this from two c2_transformation instances

Parameters

xx	the X axis transform
yy	the Y axis transform

6.26.2.2 template<typename float_type> virtual c2_function_transformation< float_type >::~c2_function_transformation ( ) [inline], [virtual]

destructor

6.26.3 Member Function Documentation

6.26.3.1 template< typename float_type > virtual float_type c2_function_transformation< float_type >::evaluate (float_type *xraw*, float_type *y*, float_type *yp0*, float_type *ypp0*, float_type * *yprime*, float_type * *yprime2*) const [virtual]

evaluate the transformation from internal coordinates to external coordinates

Parameters

	xraw	the value of *x* in external cordinates at which the transform is taking place
	y	the value of the function in internal coordinates
	yp0	the derivative in internal coordinates
	ypp0	the second derivative in internal coordinates
out	*yprime*	pointer to the derivative, or NULL, in external coordinates
out	*yprime2*	pointer to the second derivative, or NULL, in external coordinates

Returns

the value of the function in external coordinates

6.26.4 Member Data Documentation

6.26.4.1 template< typename float_type > const bool c2_function_transformation< float_type >::isIdentity

flag indicating of the transform is the identity, and can be skipped for efficiency

6.26.4.2 template< typename float_type > const c2_transformation<float_type>& c2_function_transformation< float_type >::X

the X axis transform

6.26.4.3 template< typename float_type > const c2_transformation<float_type>& c2_function_transformation< float_type >::Y

the Y axis transform

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

- c2_function.hh

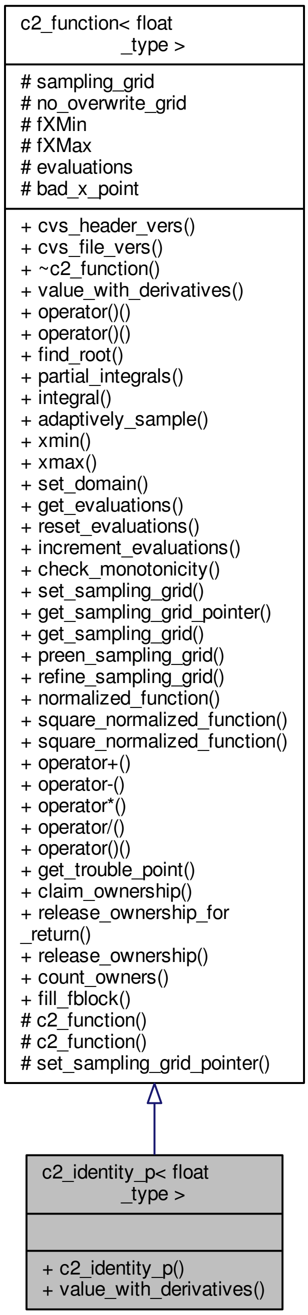
6.27 c2_identity_p< float_type > Class Template Reference

compute x with its derivatives.

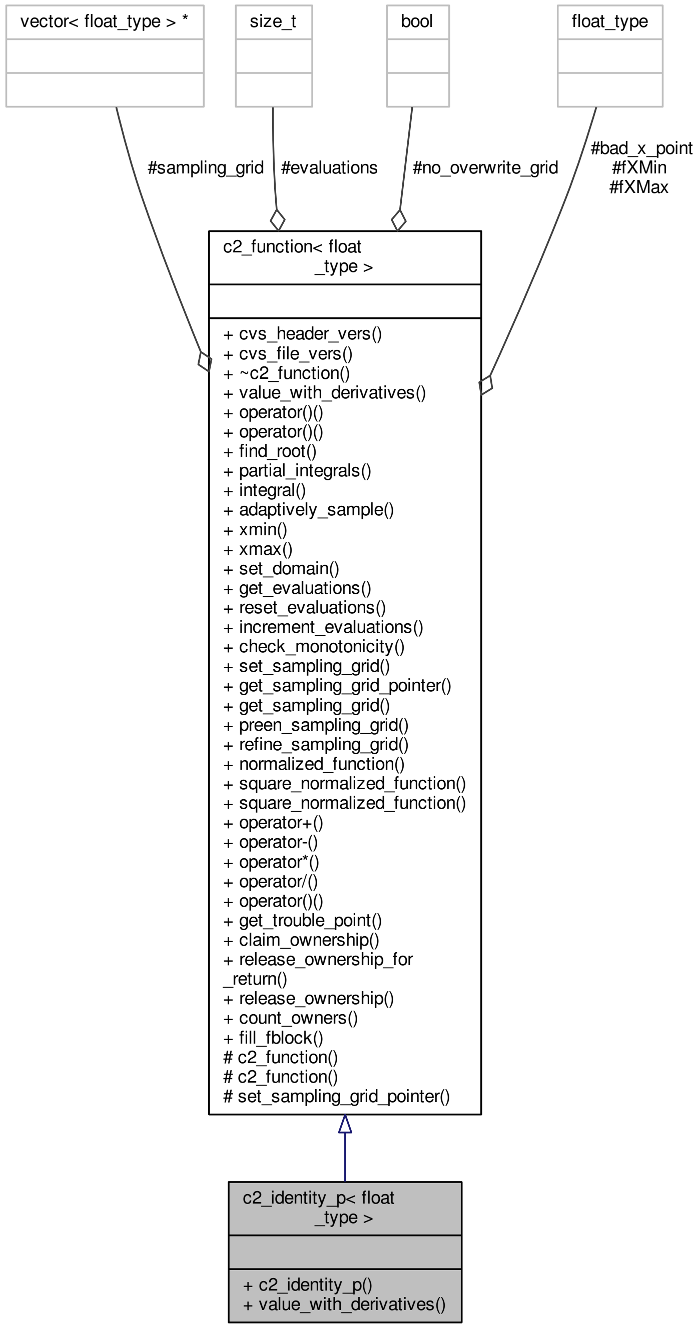
The factory function c2_factory::identity() creates *new c2_identity_p

```
#include "c2_function.hh"
```

Inheritance diagram for c2_identity_p< float_type >:



Collaboration diagram for c2_identity_p< float_type >:



Public Member Functions

- c2_identity_p ()

    constructor.

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.

Additional Inherited Members

6.27.1 Detailed Description

template<typename float_type = double>
class c2_identity_p< float_type >

compute x with its derivatives.

The factory function c2_factory::identity() creates ∗new c2_identity_p

6.27.2 Constructor & Destructor Documentation

6.27.2.1 template<typename float_type = double> c2_identity_p< float_type >::c2_identity_p ( ) [inline]

constructor.

6.27.3 Member Function Documentation

6.27.3.1 template<typename float_type = double> virtual float_type c2_identity_p< float_type >::value_with_derivatives (float_type *x*, float_type ∗ *yprime*, float_type ∗ *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

- c2_function.hh

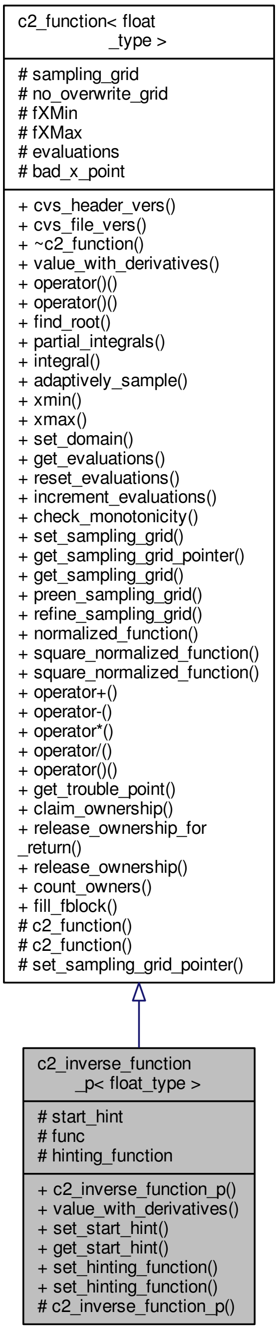
6.28 c2_inverse_function_p< float_type > Class Template Reference

create the formal inverse function of another function

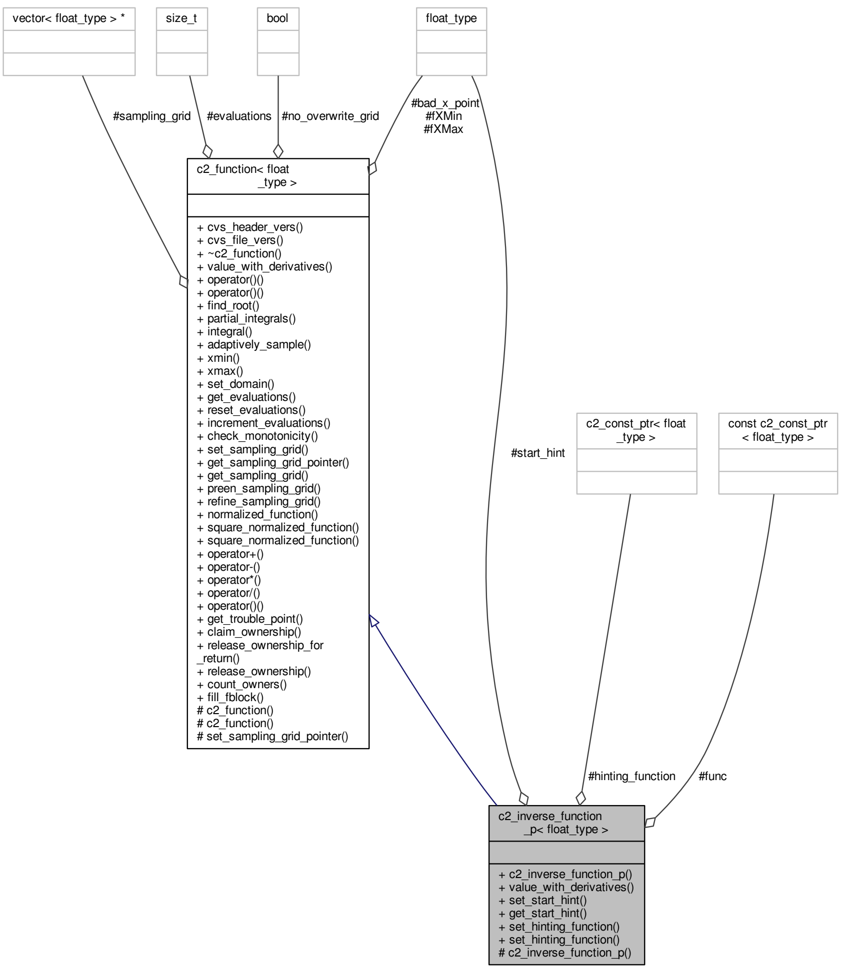
for example, given a c2_function *f*

```
#include "c2_function.hh"
```

Inheritance diagram for c2_inverse_function_p< float_type >:



Collaboration diagram for c2_inverse_function_p< float_type >:



Public Member Functions

- c2_inverse_function_p (const c2_function< float_type > &source)

    Construct the operator.
- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.
- void set_start_hint (float_type hint) const

    give the function a hint as to where to look for its inverse
- virtual float_type get_start_hint (float_type x) const

    get the starting hint.
- void set_hinting_function (const c2_function< float_type > ∗hint_func)

    set or unset the approximate function used to start the root finder
- void set_hinting_function (const c2_const_ptr< float_type > hint_func)

    set the hinting function from a pointer.

Protected Member Functions

- c2_inverse_function_p ()

Protected Attributes

- float_type start_hint
- const c2_const_ptr< float_type > func
- c2_const_ptr< float_type > hinting_function

6.28.1 Detailed Description

template<typename float_type = double>
class c2_inverse_function_p< float_type >

create the formal inverse function of another function

for example, given a c2_function *f*

```
c2_inverse_function<double> inv(f);
a=f(x);
x1=inv(a);
```

will return x1=x to machine precision. The important part of this is that the resulting function is a first-class c2_function, so it knows its derivatives, too, unlike the case of a simple root-finding inverse. This means it can be integrated (for example) quite efficiently.

See also

combined_inversion_hinting_sampling

The factory function c2_factory::inverse_function() creates *new c2_inverse_function_p

6.28.2 Constructor & Destructor Documentation

6.28.2.1 template<typename float_type = double> c2_inverse_function_p< float_type >::c2_inverse_function_p (const c2_function< float_type > & *source*)

Construct the operator.

Parameters

source	the function to be inverted

6.28.2.2 template<typename float_type = double> c2_inverse_function_p< float_type >::c2_inverse_function_p ( ) [inline], [protected]

6.28.3 Member Function Documentation

6.28.3.1 template<typename float_type = double> virtual float_type c2_inverse_function_p< float_type >::get_start_hint (float_type *x*) const [inline], [virtual]

get the starting hint.

This is virtual so if there is a better way, this can be easily overridden. It is used in value_with_derivatives() to guess where to start the root finder.

Parameters

x	the abscissa for which an estimate is needed

6.28.3.2 template<typename float_type = double> void **c2_inverse_function_p**< float_type >::set_hinting_function (const **c2_function**< float_type > * *hint_func*) [inline]

set or unset the approximate function used to start the root finder

A hinting function is mostly useful if the evaluation of this inverse is going to be carried out in very non-local order, so the root finder has to start over for each step. If most evaluations are going to be made in fairly localized clusters (scanning through the function, for example), the default mechanism used (which just remembers the last point) is almost certainly faster.

Typically, the hinting function is likely to be set up by creating the inverse function, and then adaptively sampling an interpolating function from it, and then using the result to hint it. Another way, if the parent function is already an interpolating function, is just to create a version of the parent with the x & y coordinates reversed.

See also

combined_inversion_hinting_sampling

Parameters

hint_func	the function that is an approximate inverse of the parent of this inverse_function

6.28.3.3 template<typename float_type = double> void **c2_inverse_function_p**< float_type >::set_hinting_function (const **c2_const_ptr**< float_type > *hint_func*) [inline]

set the hinting function from a pointer.

See discussion

Parameters

hint_func	the container holding the function

6.28.3.4 template<typename float_type = double> void **c2_inverse_function_p**< float_type >::set_start_hint (float_type *hint*) const [inline]

give the function a hint as to where to look for its inverse

Parameters

hint	the likely value of the inverse, which defaults to whatever the evaluation returned.

6.28.3.5 template< typename float_type = double > virtual float_type c2_inverse_function_p< float_type >::value_with_derivatives (float_type *x*, float_type ∗ *yprime*, float_type ∗ *yprime2*) const throw c2_exception) [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

6.28.4 Member Data Documentation

6.28.4.1 template< typename float_type = double > const c2_const_ptr<float_type> c2_inverse_function_p< float_type >::func [protected]

6.28.4.2 template< typename float_type = double > c2_const_ptr<float_type> c2_inverse_function_p< float_type >::hinting_function [protected]

6.28.4.3 template< typename float_type = double > float_type c2_inverse_function_p< float_type >::start_hint [mutable], [protected]

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

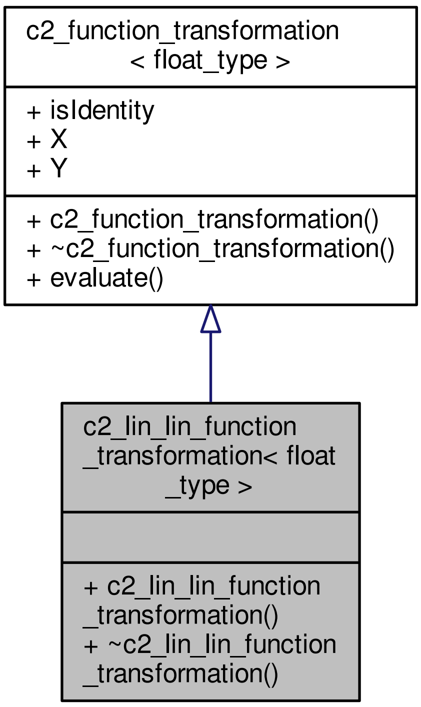
- c2_function.hh

6.29 c2_lin_lin_function_transformation< float_type > Class Template Reference

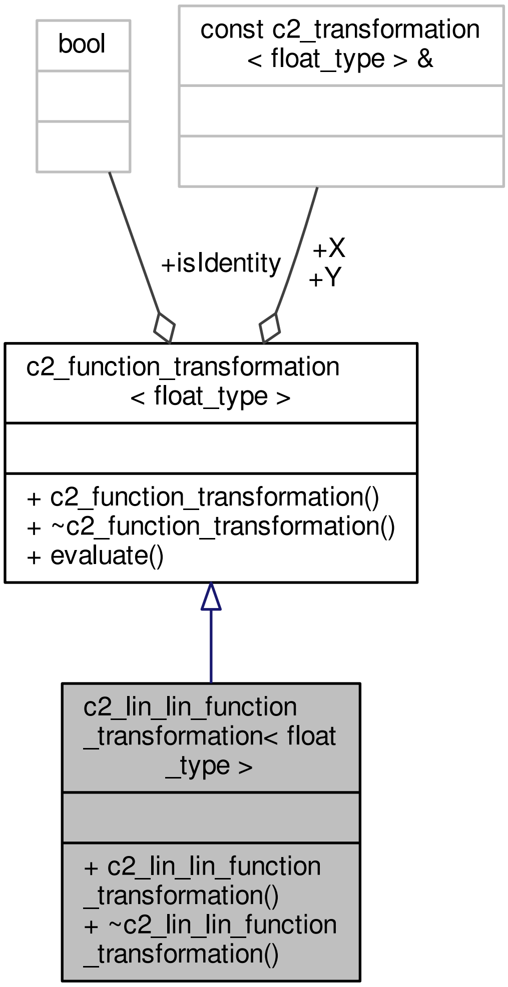
a transformation of a function in and out of lin-lin space

```
#include "c2_function.hh"
```

Inheritance diagram for c2_lin_lin_function_transformation< float_type >:



Collaboration diagram for c2_lin_lin_function_transformation< float_type >:



Public Member Functions

- c2_lin_lin_function_transformation ()
- virtual ∼c2_lin_lin_function_transformation ()

Additional Inherited Members

6.29.1 Detailed Description

template<typename float_type>
class c2_lin_lin_function_transformation< float_type >

a transformation of a function in and out of lin-lin space

6.29.2 Constructor & Destructor Documentation

6.29.2.1 template<typename float_type > c2_lin_lin_function_transformation< float_type >::c2_lin_lin_function_transformation ( ) [inline]

6.29.2.2 template<typename float_type > virtual c2_lin_lin_function_transformation< float_type >::~c2_lin_lin_function_transformation ( ) [inline], [virtual]

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

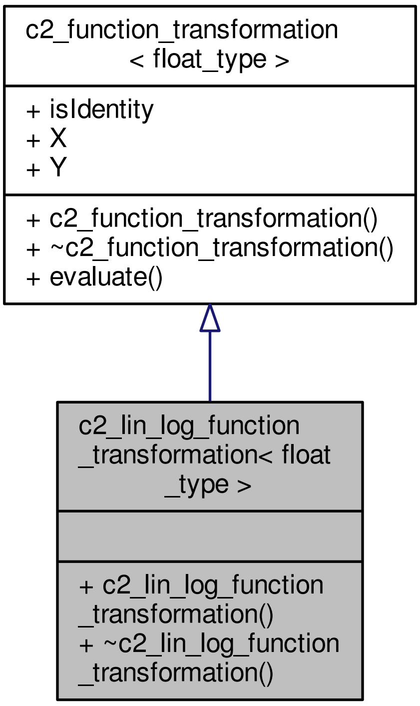
- c2_function.hh

6.30 c2_lin_log_function_transformation< float_type > Class Template Reference

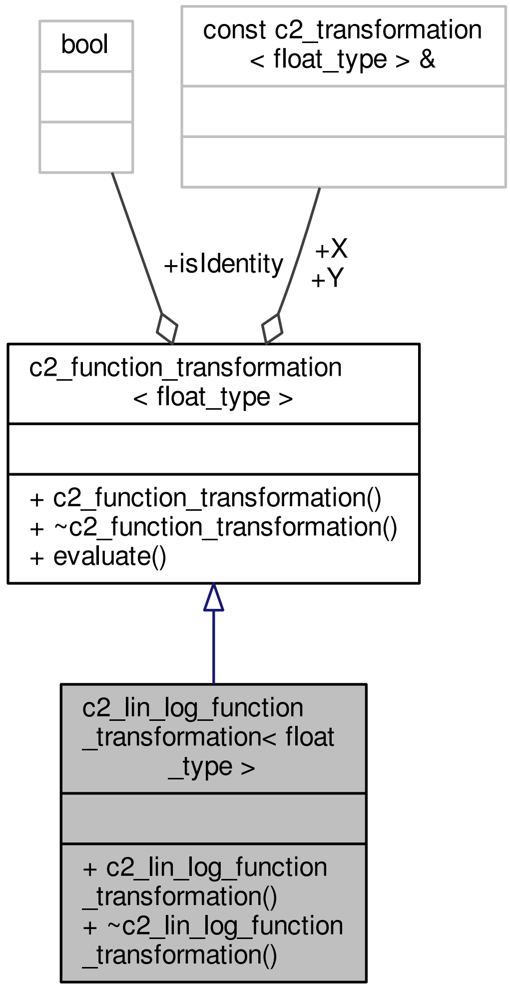
a transformation of a function in and out of lin-log space

`#include "c2_function.hh"`

Inheritance diagram for c2_lin_log_function_transformation< float_type >:



Collaboration diagram for c2_lin_log_function_transformation< float_type >:



Public Member Functions

- c2_lin_log_function_transformation ()
- virtual ∼c2_lin_log_function_transformation ()

Additional Inherited Members

6.30.1 Detailed Description

template<typename float_type>
class c2_lin_log_function_transformation< float_type >

a transformation of a function in and out of lin-log space

6.30.2 Constructor & Destructor Documentation

6.30.2.1 template<typename float_type > **c2_lin_log_function_transformation**< float_type >::**c2_lin_log_function_transformation** ( ) `[inline]`

6.30.2.2 template<typename float_type > virtual **c2_lin_log_function_transformation**< float_type >::~**c2_lin_log_function_transformation** ( ) `[inline]`, `[virtual]`

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

- c2_function.hh

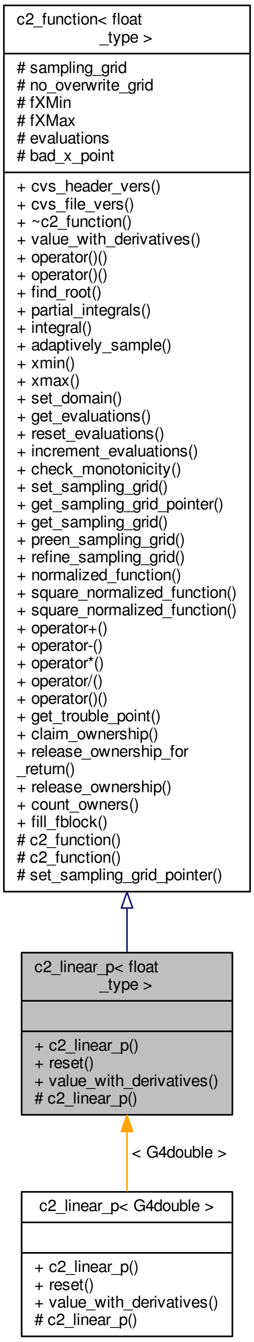
6.31 c2_linear_p< float_type > Class Template Reference

create a linear mapping of another function

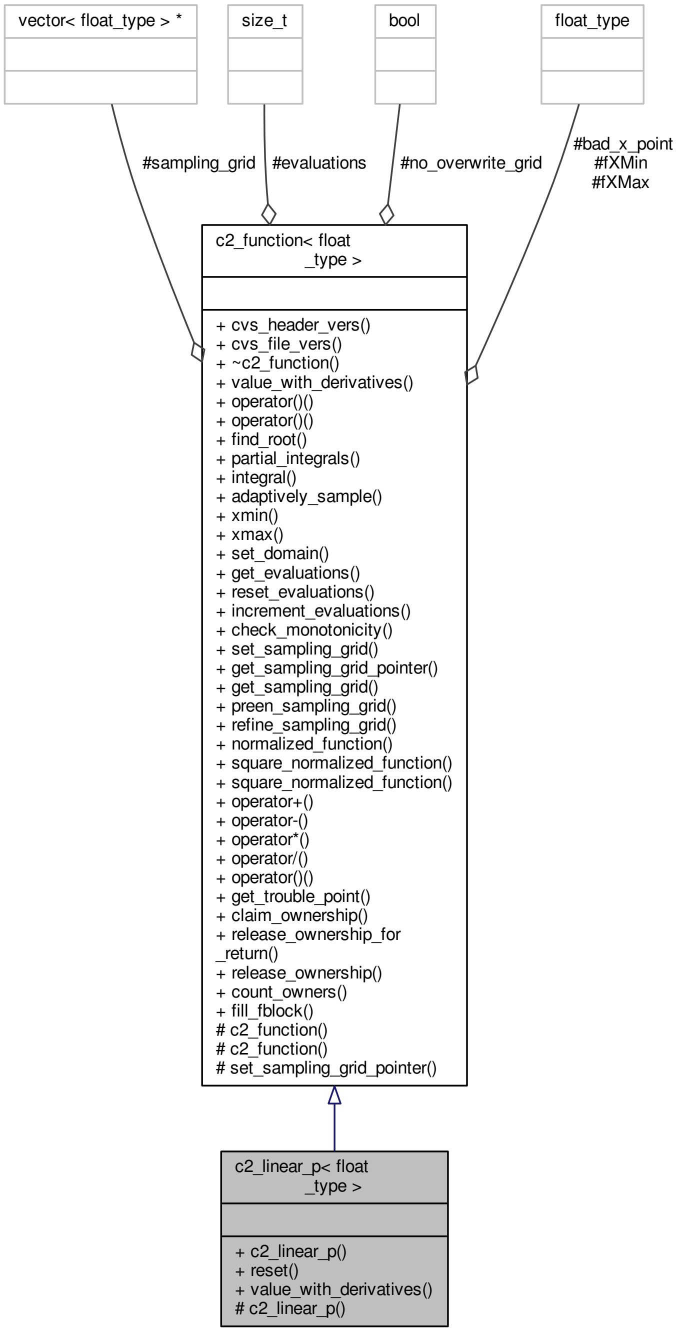
for example, given a c2_function *f*

```
#include "c2_function.hh"
```

Inheritance diagram for c2_linear_p< float_type >:



Collaboration diagram for c2_linear_p< float_type >:



Public Member Functions

- c2_linear_p (float_type x0, float_type y0, float_type slope)

    Construct the operator f=y0 + slope ∗ (x-x0)

- void reset (float_type x0, float_type y0, float_type slope)

    Change the slope and intercepts after construction.

- virtual float_type value_with_derivatives (float_type x, float_type *yprime, float_type *yprime2) const throw (c2_exception)

    get the value and derivatives.

Protected Member Functions

- c2_linear_p ()

Additional Inherited Members

6.31.1 Detailed Description

template<typename float_type = double>
class c2_linear_p< float_type >

create a linear mapping of another function

for example, given a c2_function *f*

```
c2_function<double> &F=c2_linear<double>(1.2, 2.0, 3.0)(f);
```

produces a new c2_function F=2.0+3.0*(*f* - 1.2)

The factory function c2_factory::linear() creates *new c2_linear_p

6.31.2 Constructor & Destructor Documentation

6.31.2.1 template<typename float_type = double> c2_linear_p< float_type >::c2_linear_p (float_type *x0*, float_type *y0*, float_type *slope*) [inline]

Construct the operator f=y0 + slope * (x-x0)

Parameters

x0	the x offset
y0	the y-intercept i.e. f(x0)
slope	the slope of the mapping

6.31.2.2 template<typename float_type = double> c2_linear_p< float_type >::c2_linear_p ( ) [inline], [protected]

6.31.3 Member Function Documentation

6.31.3.1 template<typename float_type = double> void c2_linear_p< float_type >::reset (float_type *x0*, float_type *y0*, float_type *slope*) [inline]

Change the slope and intercepts after construction.

Parameters

x0	the x offset
y0	the y-intercept
slope	the slope of the mapping

6.31.3.2 template<typename float_type = double> virtual float_type c2_linear_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

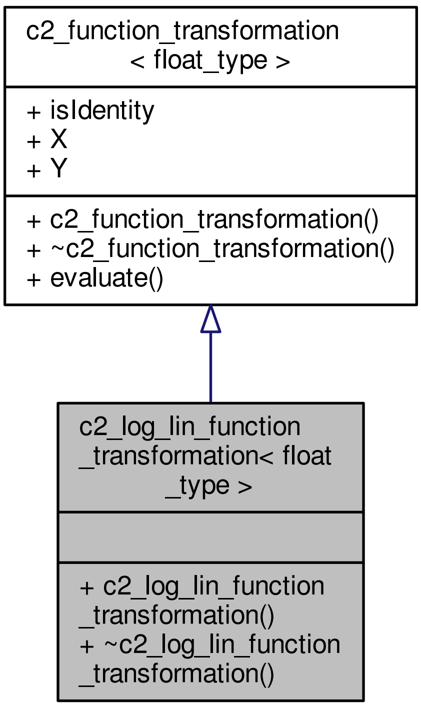
- c2_function.hh

6.32 c2_log_lin_function_transformation< float_type > Class Template Reference

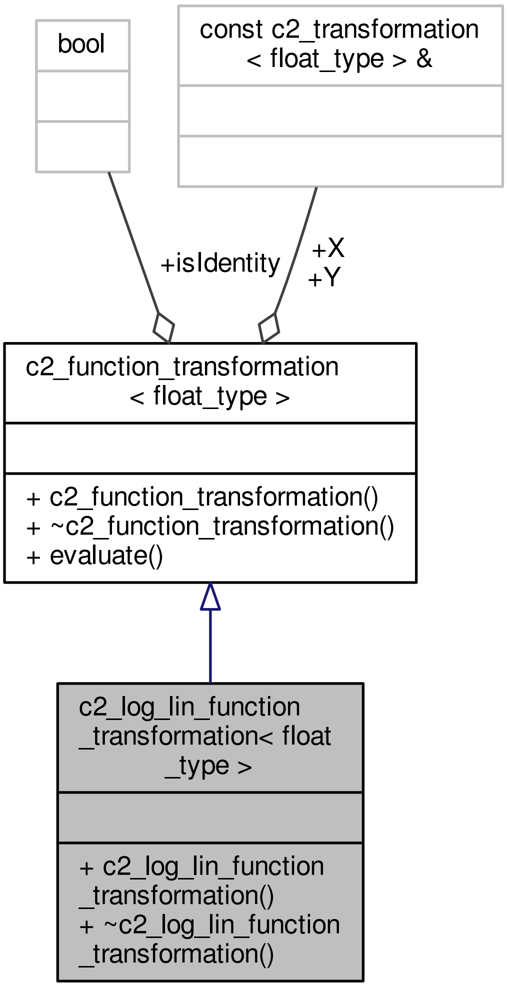
a transformation of a function in and out of log-lin space

`#include "c2_function.hh"`

Inheritance diagram for c2_log_lin_function_transformation< float_type >:



Collaboration diagram for c2_log_lin_function_transformation< float_type >:



Public Member Functions

- c2_log_lin_function_transformation ()
- virtual ∼c2_log_lin_function_transformation ()

Additional Inherited Members

6.32.1 Detailed Description

template<typename float_type>
class c2_log_lin_function_transformation< float_type >

a transformation of a function in and out of log-lin space

6.32.2 Constructor & Destructor Documentation

6.32.2.1 template<typename float_type > c2_log_lin_function_transformation< float_type >::c2_log_lin_function_transformation ( ) [inline]

6.32.2.2 template<typename float_type > virtual c2_log_lin_function_transformation< float_type >::~c2_log_lin_function_transformation ( ) [inline], [virtual]

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

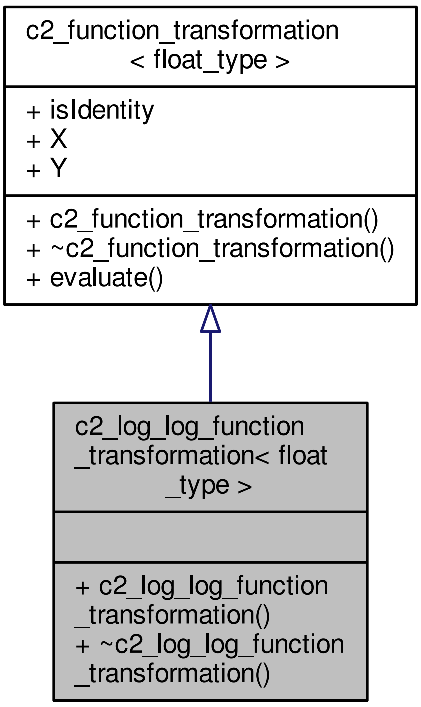
- c2_function.hh

6.33 c2_log_log_function_transformation< float_type > Class Template Reference

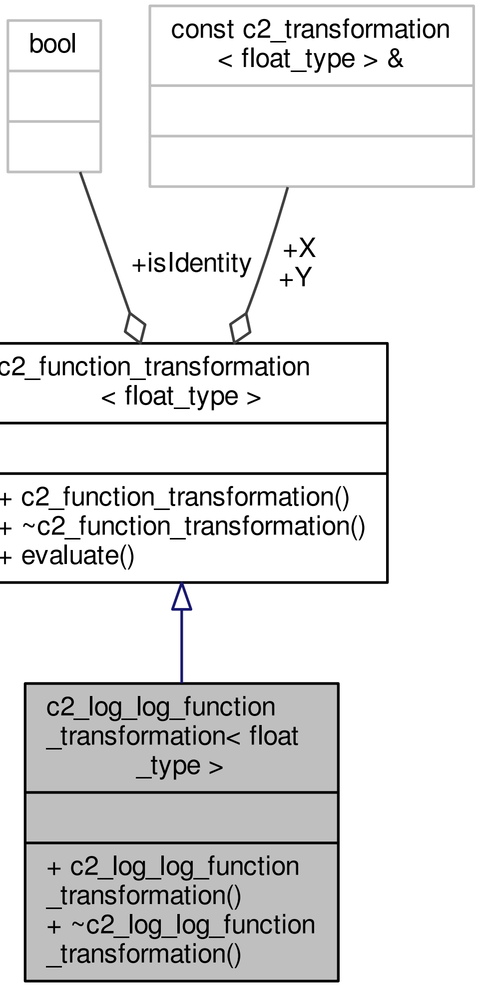
a transformation of a function in and out of log-log space

```
#include "c2_function.hh"
```

Inheritance diagram for c2_log_log_function_transformation< float_type >:



Collaboration diagram for c2_log_log_function_transformation< float_type >:



Public Member Functions

- c2_log_log_function_transformation ()
- virtual ∼c2_log_log_function_transformation ()

Additional Inherited Members

6.33.1 Detailed Description

template<typename float_type>
class c2_log_log_function_transformation< float_type >

a transformation of a function in and out of log-log space

6.33.2 Constructor & Destructor Documentation

6.33.2.1 template<typename float_type > **c2_log_log_function_transformation**< float_type >::**c2_log_log_function_transformation** ( ) [inline]

6.33.2.2 template<typename float_type > virtual **c2_log_log_function_transformation**< float_type >::∼**c2_log_log_function_transformation** ( ) [inline], [virtual]

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

- c2_function.hh

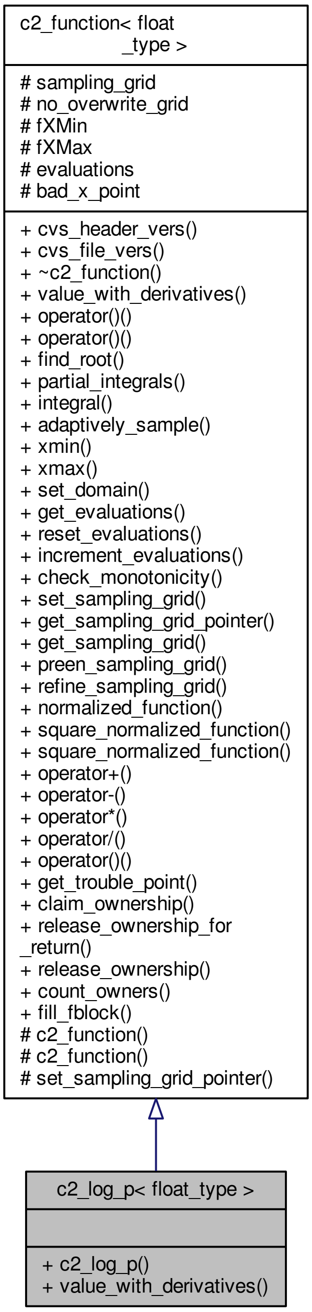
6.34 c2_log_p< float_type > Class Template Reference

compute log(x) with its derivatives.

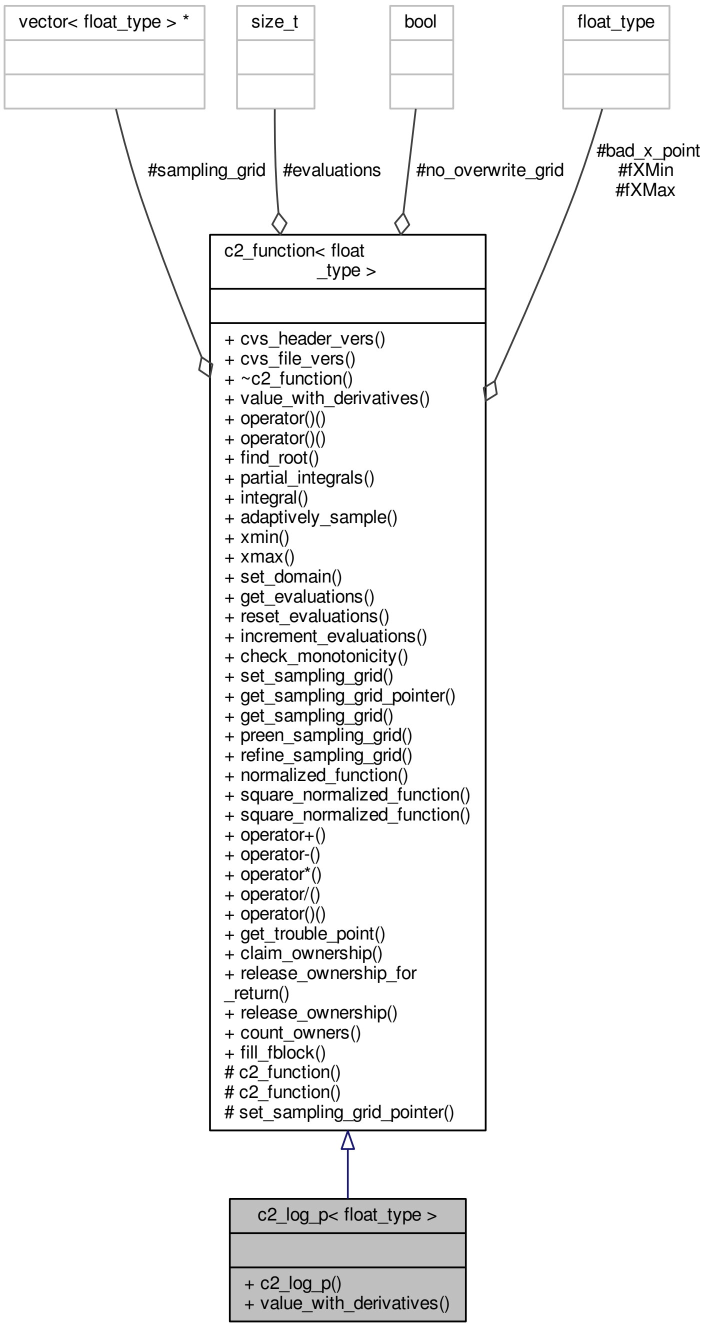
The factory function c2_factory::log() creates ∗new c2_log_p

```
#include "c2_function.hh"
```

Inheritance diagram for c2_log_p< float_type >:



Collaboration diagram for c2_log_p< float_type >:



Public Member Functions

- c2_log_p ()

    constructor.

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.

Additional Inherited Members

6.34.1 Detailed Description

template<typename float_type = double>
class c2_log_p< float_type >

compute log(x) with its derivatives.

The factory function c2_factory::log() creates *new c2_log_p

6.34.2 Constructor & Destructor Documentation

6.34.2.1 template<typename float_type = double> c2_log_p< float_type >::c2_log_p ( ) [inline]

constructor.

6.34.3 Member Function Documentation

6.34.3.1 template<typename float_type = double> virtual float_type c2_log_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

- c2_function.hh

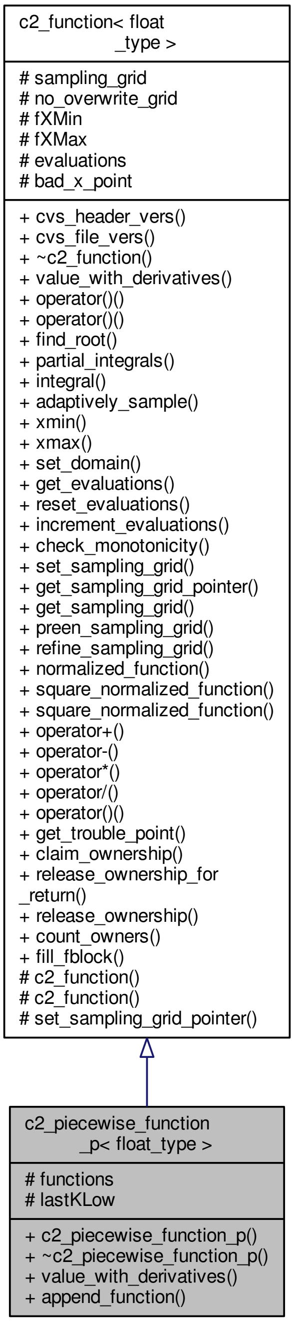
6.35 c2_piecewise_function_p< float_type > Class Template Reference

create a c2_function which is a piecewise assembly of other c2_functions.

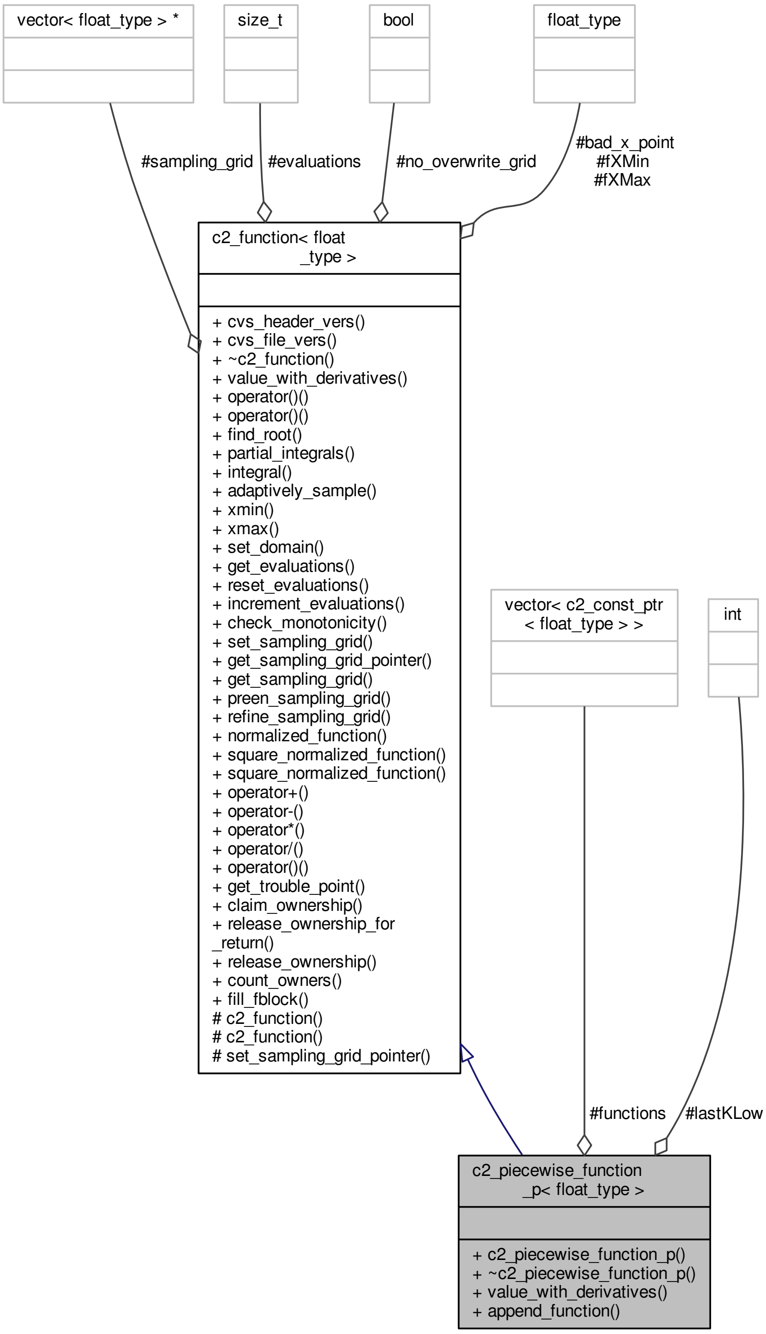
The functions must have increasing, non-overlapping domains. Any empty space between functions will be filled with a linear interpolation.

```
#include "c2_function.hh"
```

Inheritance diagram for c2_piecewise_function_p< float_type >:



Collaboration diagram for c2_piecewise_function_p< float_type >:



Public Member Functions

- c2_piecewise_function_p ()

  construct the container
- virtual ∼c2_piecewise_function_p ()

  destructor

- virtual float_type value_with_derivatives (float_type x, float_type *yprime, float_type *yprime2) const throw (c2_exception)

  get the value and derivatives.
- void append_function (const c2_function< float_type > &func) throw (c2_exception)

  append a new function to the sequence

Protected Attributes

- std::vector< c2_const_ptr< float_type > > functions
- int lastKLow

Additional Inherited Members

6.35.1 Detailed Description

template<typename float_type = double>
class c2_piecewise_function_p< float_type >

create a c2_function which is a piecewise assembly of other c2_functions.

The functions must have increasing, non-overlapping domains. Any empty space between functions will be filled with a linear interpolation.

Note

If you want a smooth connection, instead of the default linear interpolation, create a c2_connector_function_p to bridge the gap. The linear interpolation is intended to be a barely intelligent bridge, and may never get used by anyone.
The creation of the container results in the creation of an explicit sampling grid. If this is used with functions with a large domain, or which generate very dense sampling grids, it could eat a lot of memory. Do not abuse this by using functions which can generate gigantic grids.

See also

Sample Applications
c2_plugin_function_p page
c2_connector_function_p page
Adaptive sampling

The factory function c2_factory::piecewise_function() creates *new c2_piecewise_function_p

6.35.2 Constructor & Destructor Documentation

6.35.2.1 template<typename float_type = double> c2_piecewise_function_p< float_type >::c2_piecewise_function_p ( )

construct the container

6.35.2.2 template<typename float_type = double> virtual c2_piecewise_function_p< float_type >::~c2_piecewise_function_p ( ) [virtual]

destructor

6.35.3 Member Function Documentation

6.35.3.1 template<typename float_type = double> void c2_piecewise_function_p< float_type >::append_function (const c2_function< float_type > & *func*) throw c2_exception)

append a new function to the sequence

This takes a c2_function, and appends it onto the end of the piecewise collection. The domain of the function (which MUST be set) specifies the place it will be used in the final function. If the domain exactly abuts the domain of the previous function, it will be directly attached. If there is a gap, the gap will be filled in by linear interpolation.

Parameters

func	a c2_function with a defined domain to be appended to the collection

6.35.3.2 template<typename float_type = double> virtual float_type c2_piecewise_function_p< float_type >::value_with_derivatives (float_type *x*, float_type ∗ *yprime*, float_type ∗ *yprime2*) const throw c2_exception) [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

6.35.4 Member Data Documentation

6.35.4.1 template<typename float_type = double> std::vector<c2_const_ptr<float_type> > c2_piecewise_function_p< float_type >::functions [protected]

6.35.4.2 template<typename float_type = double> int c2_piecewise_function_p< float_type >::lastKLow [mutable], [protected]

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

- c2_function.hh

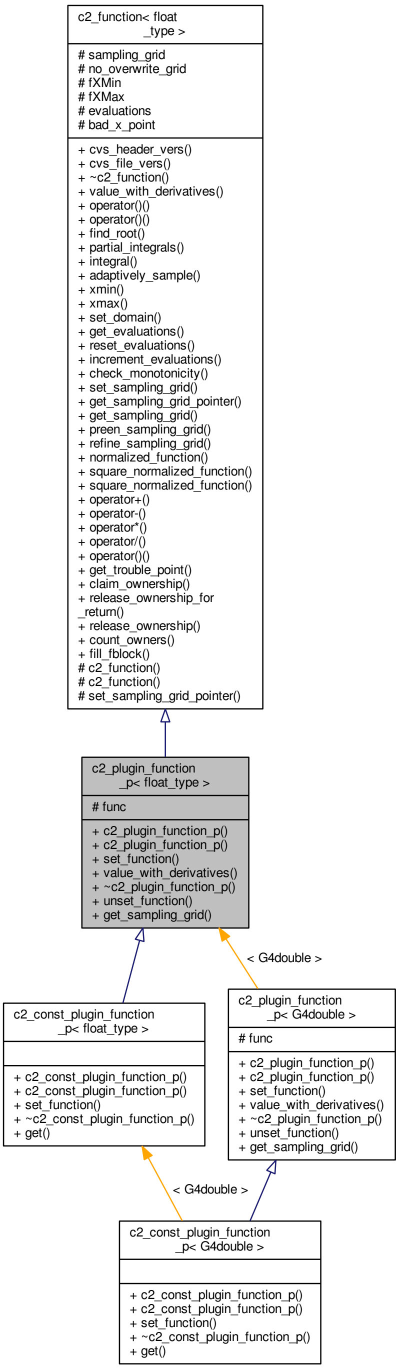
6.36 c2_plugin_function_p< float_type > Class Template Reference

a container into which any other c2_function can be dropped, to allow expressions with replacable components.

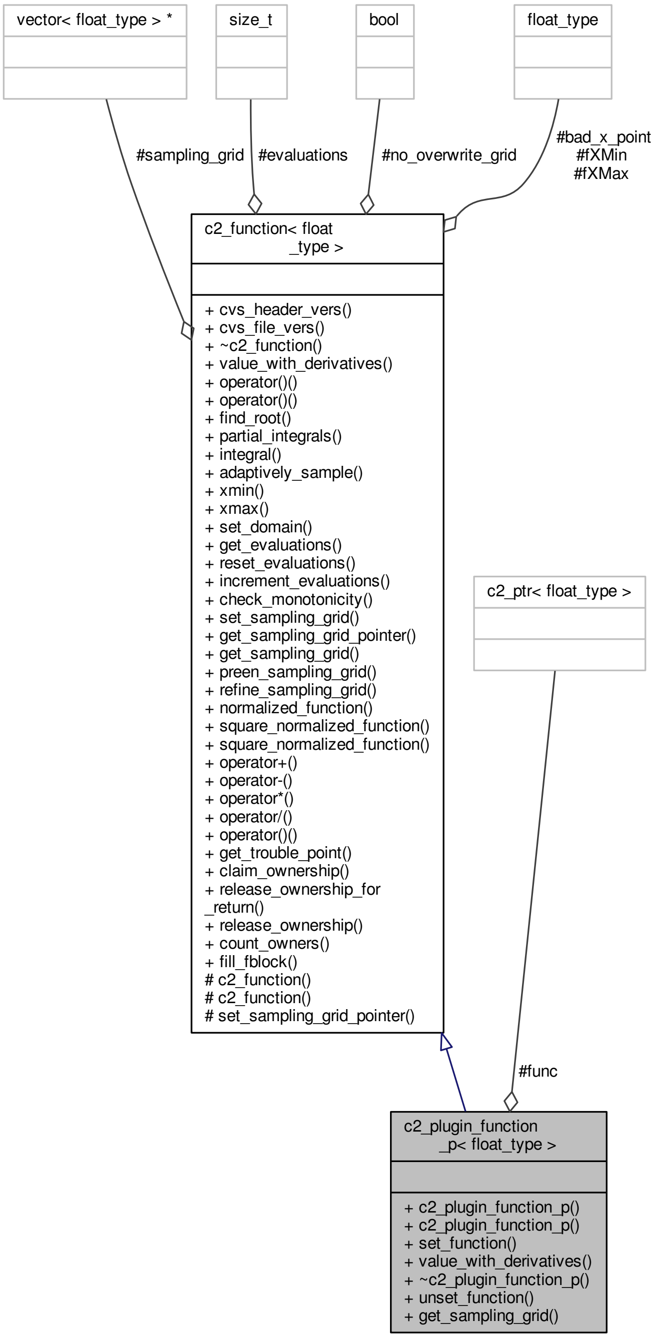
It is useful for plugging different InterpolatingFunctions into a c2_function expression. It saves a lot of effort in other places with casting away const declarations.

```
#include "c2_function.hh"
```

Inheritance diagram for c2_plugin_function_p< float_type >:



Collaboration diagram for c2_plugin_function_p< float_type >:



Public Member Functions

- c2_plugin_function_p ()

    construct the container with no function
- c2_plugin_function_p (c2_function< float_type > &f)

    construct the container with a pre-defined function
- void set_function (c2_function< float_type > *f)

fill the container with a new function, or clear it with a null pointer and copy our domain

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

  get the value and derivatives.
- virtual ~c2_plugin_function_p ()

  destructor
- void unset_function ()

  clear our function
- virtual void get_sampling_grid (float_type amin, float_type amax, std::vector< float_type > &grid) const

  return the grid of 'interesting' points along this function which lie in the region requested

Protected Attributes

- c2_ptr< float_type > func

Additional Inherited Members

6.36.1 Detailed Description

template<typename float_type = double>
class c2_plugin_function_p< float_type >

a container into which any other c2_function can be dropped, to allow expressions with replacable components.

It is useful for plugging different InterpolatingFunctions into a c2_function expression. It saves a lot of effort in other places with casting away const declarations.

It is also useful as a wrapper for a function if it is necessary to have a copy of a function which has a different domain or sampling grid than the parent function. This can be be used, for example, to patch badly-behaved functions with c2_piecewise_function_p by taking the parent function, creating two plugins of it with domains on each side of the nasty bit, and then inserting a nice function in the hole.

This can also be used as a fancier c2_ptr which allows direct evaluation instead of having to dereference the container first.

The factory function c2_factory::plugin_function() creates ∗new c2_plugin_function_p()

6.36.2 Constructor & Destructor Documentation

6.36.2.1 template<typename float_type = double> c2_plugin_function_p< float_type >::c2_plugin_function_p ( ) [inline]

construct the container with no function

6.36.2.2 template<typename float_type = double> c2_plugin_function_p< float_type >::c2_plugin_function_p (c2_function< float_type > & *f*) [inline]

construct the container with a pre-defined function

6.36.2.3 template<typename float_type = double> virtual c2_plugin_function_p< float_type >::~c2_plugin_function_p ( ) [inline], [virtual]

destructor

6.36.3 Member Function Documentation

6.36.3.1 template<typename float_type = double> virtual void c2_plugin_function_p< float_type >::get_sampling_grid (float_type *amin*, float_type *amax*, std::vector< float_type > & *grid*) const [inline], [virtual]

return the grid of 'interesting' points along this function which lie in the region requested

if a sampling grid is defined, work from there, otherwise return vector of (amin, amax)

Parameters

	amin	the lower bound for which the function is to be sampled
	amax	the upper bound for which the function is to be sampled
in,out	*grid*	filled vector containing the samplng grid.

Reimplemented from c2_function< float_type >.

6.36.3.2 template<typename float_type = double> void c2_plugin_function_p< float_type >::set_function (c2_function< float_type > * *f*) [inline]

fill the container with a new function, or clear it with a null pointer and copy our domain

6.36.3.3 template<typename float_type = double> void c2_plugin_function_p< float_type >::unset_function ( ) [inline]

clear our function

6.36.3.4 template<typename float_type = double> virtual float_type c2_plugin_function_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function Uses the internal function pointer set by set_function().

Implements c2_function< float_type >.

6.36.4 Member Data Documentation

6.36.4.1 template<typename float_type = double> c2_ptr<float_type> c2_plugin_function_p< float_type >::func [protected]

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

- c2_function.hh

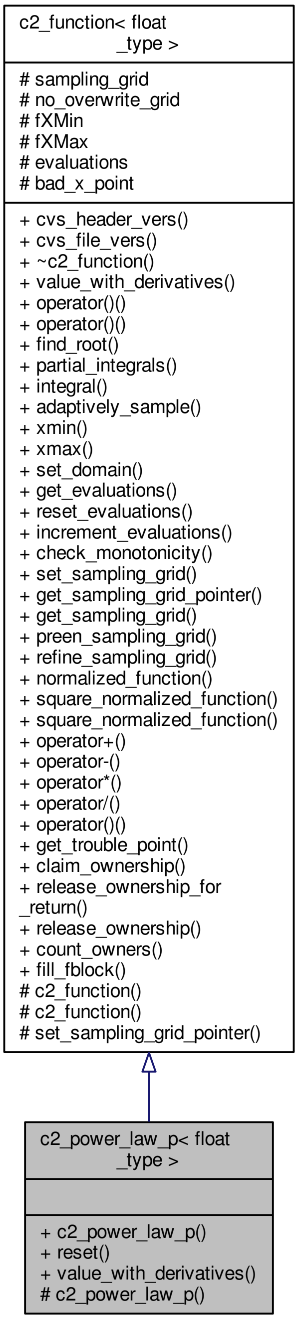
6.37 c2_power_law_p< float_type > Class Template Reference

create a power law mapping of another function

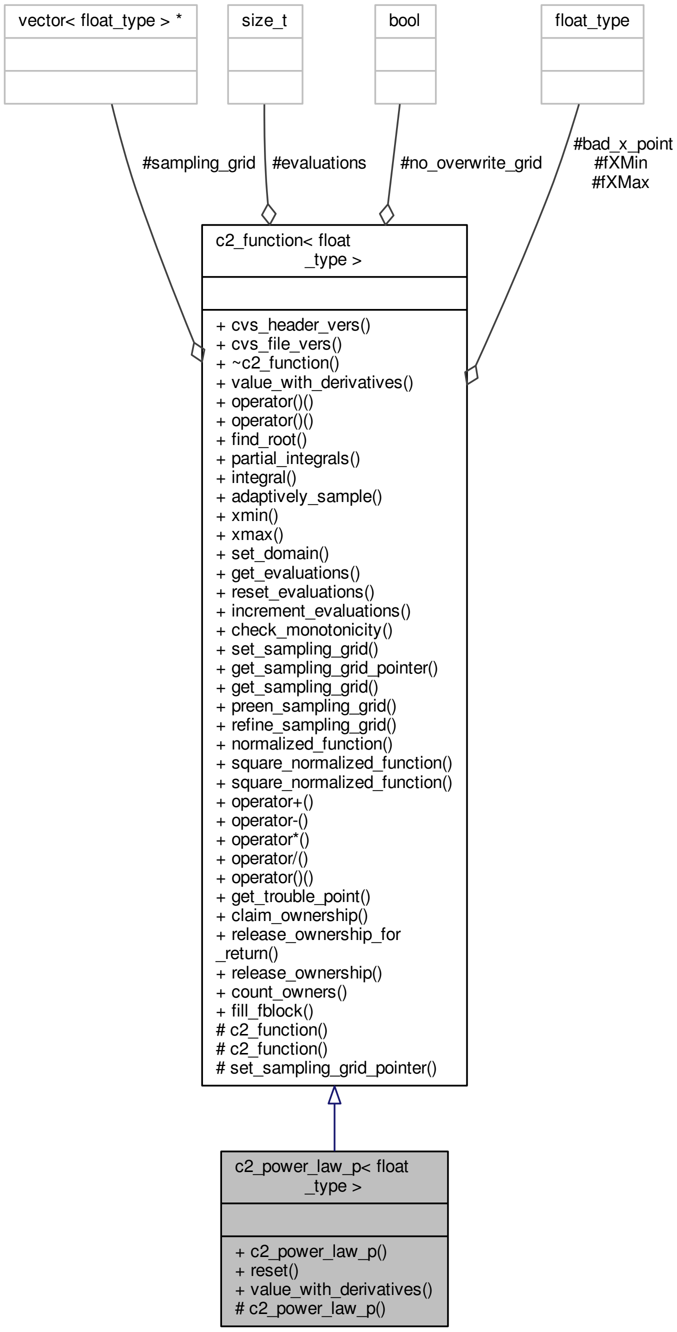
for example, given a c2_function *f*

```
#include "c2_function.hh"
```

Inheritance diagram for c2_power_law_p< float_type >:



Collaboration diagram for c2_power_law_p< float_type >:



Public Member Functions

- c2_power_law_p (float_type scale, float_type power)

    Construct the operator.
- void reset (float_type scale, float_type power)

    Modify the mapping after construction.

- virtual float_type value_with_derivatives (float_type x, float_type *yprime, float_type *yprime2) const throw (c2_exception)

    get the value and derivatives.

Protected Member Functions

- c2_power_law_p ()

Additional Inherited Members

6.37.1 Detailed Description

template<typename float_type = double>
class c2_power_law_p< float_type >

create a power law mapping of another function

for example, given a c2_function *f*

```
c2_power_law_p<double> PLaw(1.2, 2.5);
c2_composed_function_p<double> &F=PLaw(f);
```

produces a new c2_function F=1.2 * f^2.5

The factory function c2_factory::power_law() creates *new c2_power_law_p

6.37.2 Constructor & Destructor Documentation

6.37.2.1 template<typename float_type = double> c2_power_law_p< float_type >::c2_power_law_p (float_type *scale*, float_type *power*) [inline]

Construct the operator.

Parameters

scale	the multipler
power	the exponent

6.37.2.2 template<typename float_type = double> c2_power_law_p< float_type >::c2_power_law_p ( ) [inline], [protected]

6.37.3 Member Function Documentation

6.37.3.1 template<typename float_type = double> void c2_power_law_p< float_type >::reset (float_type *scale*, float_type *power*) [inline]

Modify the mapping after construction.

Parameters

scale	the new multipler
power	the new exponent

6.37.3.2 template<typename float_type = double> virtual float_type c2_power_law_p< float_type >::value_with_derivatives (float_type *x,* float_type ∗ *yprime,* float_type ∗ *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

- c2_function.hh

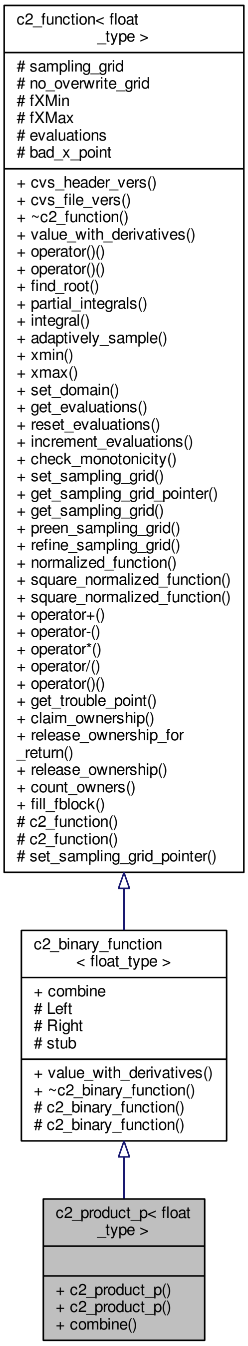
6.38 c2_product_p< float_type > Class Template Reference

create a c2_function which is the product of two other c2_functions.

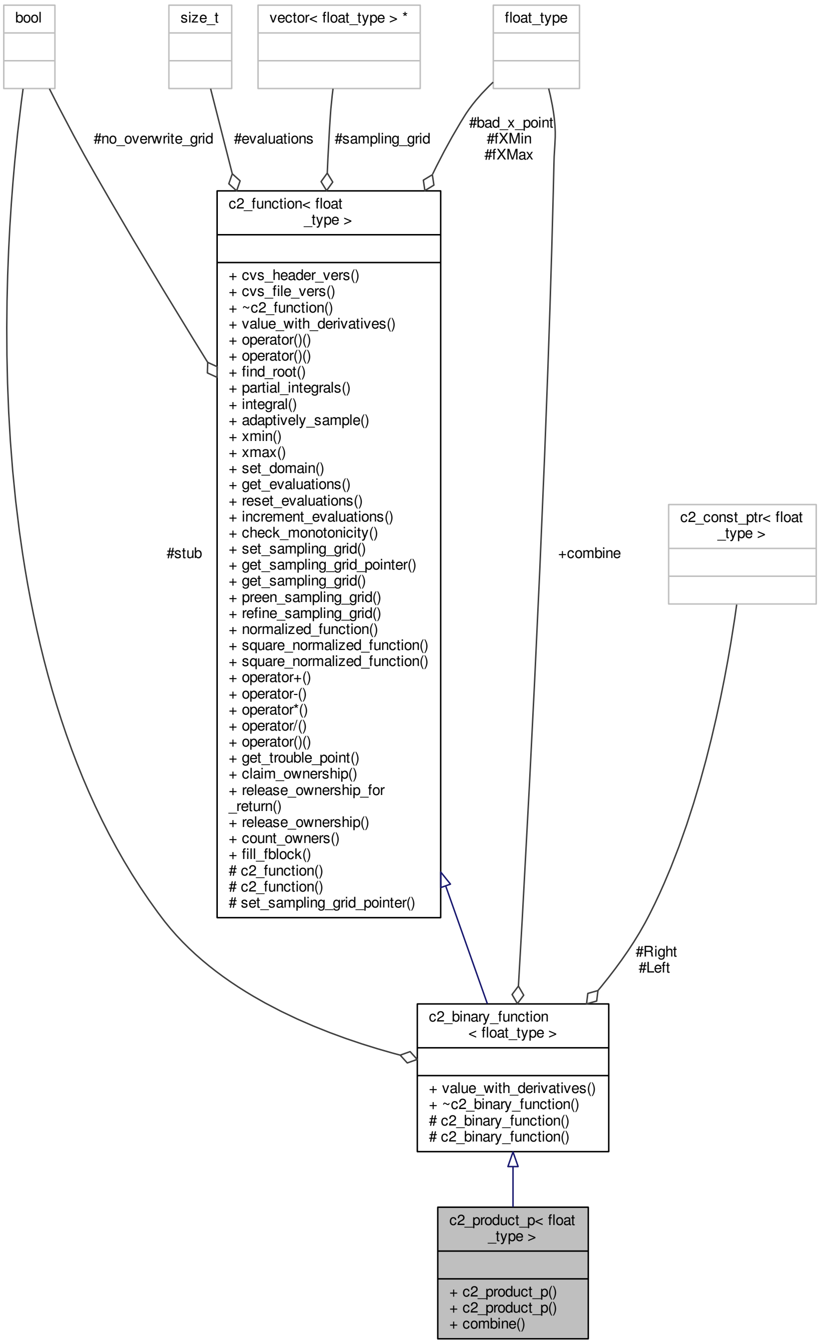
This should always be constructed using c2_function::operator∗()

```
#include "c2_function.hh"
```

Inheritance diagram for c2_product_p< float_type >:



Collaboration diagram for c2_product_p< float_type >:



Public Member Functions

- c2_product_p (const c2_function< float_type > &left, const c2_function< float_type > &right)

  construct left ∗ right

- c2_product_p ()

  Create a stub just for the combiner to avoid statics.

Static Public Member Functions

- static float_type combine (const c2_function< float_type > &left, const c2_function< float_type > &right, float_type x, float_type *yprime, float_type *yprime2) throw (c2_exception)

    execute math necessary to do multiplication

Additional Inherited Members

6.38.1 Detailed Description

template<typename float_type = double>
class c2_product_p< float_type >

create a c2_function which is the product of two other c2_functions.

This should always be constructed using c2_function::operator*()

6.38.2 Constructor & Destructor Documentation

6.38.2.1 template<typename float_type = double> c2_product_p< float_type >::c2_product_p (const c2_function< float_type > & *left*, const c2_function< float_type > & *right*) [inline]

construct *left* * *right*

Parameters

left	the left function
right	the right function

6.38.2.2 template<typename float_type = double> c2_product_p< float_type >::c2_product_p ( ) [inline]

Create a stub just for the combiner to avoid statics.

6.38.3 Member Function Documentation

6.38.3.1 template<typename float_type = double> static float_type c2_product_p< float_type >::combine (const c2_function< float_type > & *left*, const c2_function< float_type > & *right*, float_type *x*, float_type * *yprime*, float_type * *yprime2*) throw c2_exception) [inline], [static]

execute math necessary to do multiplication

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

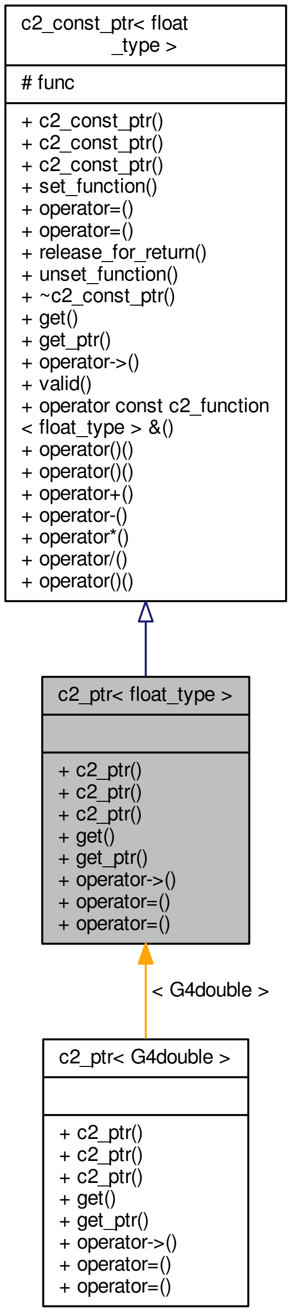
- c2_function.hh

6.39 c2_ptr< float_type > Class Template Reference

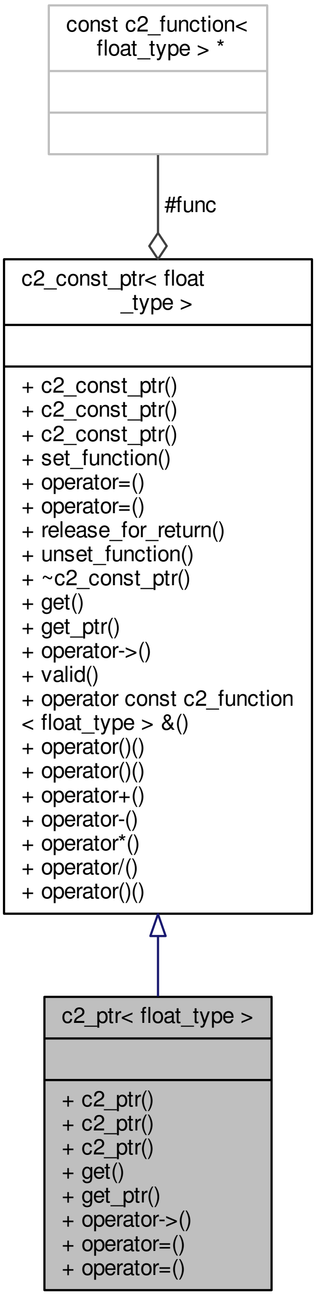
create a container for a c2_function which handles the reference counting.

```
#include "c2_function.hh"
```

Inheritance diagram for c2_ptr< float_type >:



Collaboration diagram for c2_ptr< float_type >:



Public Member Functions

- c2_ptr ()

    construct the container with no function
- c2_ptr (c2_function< float_type > &f)

    construct the container with a pre-defined function
- c2_ptr (const c2_ptr< float_type > &src)

copy constructor
- c2_function< float_type > & get () const throw (c2_exception)

  get a checked pointer to our owned function
- c2_function< float_type > ∗ get_ptr () const

  get an unchecked pointer to our owned function
- c2_function< float_type > ∗ operator-> () const

  get a checked pointer to our owned function
- const c2_ptr< float_type > & operator= (const c2_ptr< float_type > &f)

  fill the container from another container
- c2_function< float_type > & operator= (c2_function< float_type > &f)

  fill the container with a function

Additional Inherited Members

6.39.1 Detailed Description

template<typename float_type>
class c2_ptr< float_type >

create a container for a c2_function which handles the reference counting.

See also

c2_const_ptr and Use of c2_ptr for memory management

6.39.2 Constructor & Destructor Documentation

6.39.2.1 template<typename float_type> c2_ptr< float_type >::c2_ptr ( ) [inline]

construct the container with no function

6.39.2.2 template<typename float_type> c2_ptr< float_type >::c2_ptr (c2_function< float_type > & *f*) [inline]

construct the container with a pre-defined function

Parameters

f	the function to store

6.39.2.3 template<typename float_type> c2_ptr< float_type >::c2_ptr (const c2_ptr< float_type > & *src*) [inline]

copy constructor

Parameters

src	the container to copy

6.39.3 Member Function Documentation

6.39.3.1 template<typename float_type> c2_function<float_type>& c2_ptr< float_type >::get ( ) const throw c2_exception) [inline]

get a checked pointer to our owned function

6.39.3.2 template<typename float_type> c2_function<float_type>∗ c2_ptr< float_type >::get_ptr ( ) const [inline]

get an unchecked pointer to our owned function

6.39.3.3 template<typename float_type> c2_function<float_type>∗ c2_ptr< float_type >::operator-> ( ) const [inline]

get a checked pointer to our owned function

6.39.3.4 template<typename float_type> const c2_ptr<float_type>& c2_ptr< float_type >::operator= (const c2_ptr< float_type > & *f*) [inline]

fill the container from another container

Parameters

f	the container to copy

6.39.3.5 template<typename float_type> c2_function<float_type>& c2_ptr< float_type >::operator= (c2_function< float_type > & *f*) [inline]

fill the container with a function

Parameters

f	the function

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

- c2_function.hh

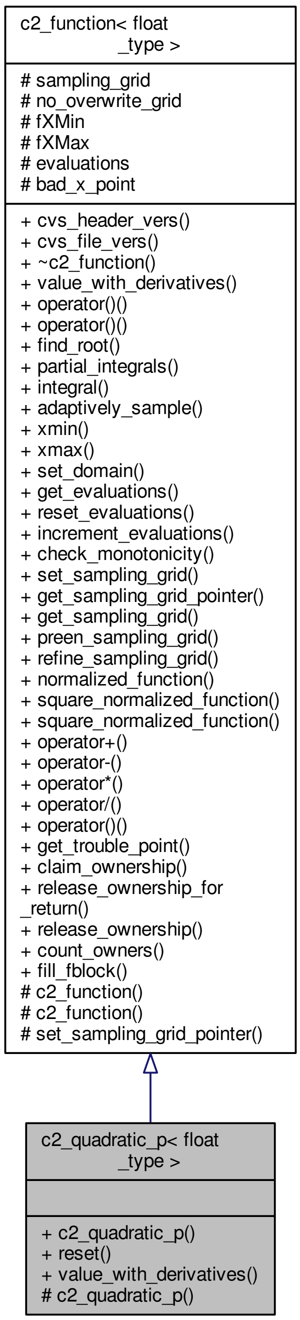
6.40 c2_quadratic_p< float_type > Class Template Reference

create a quadratic mapping of another function

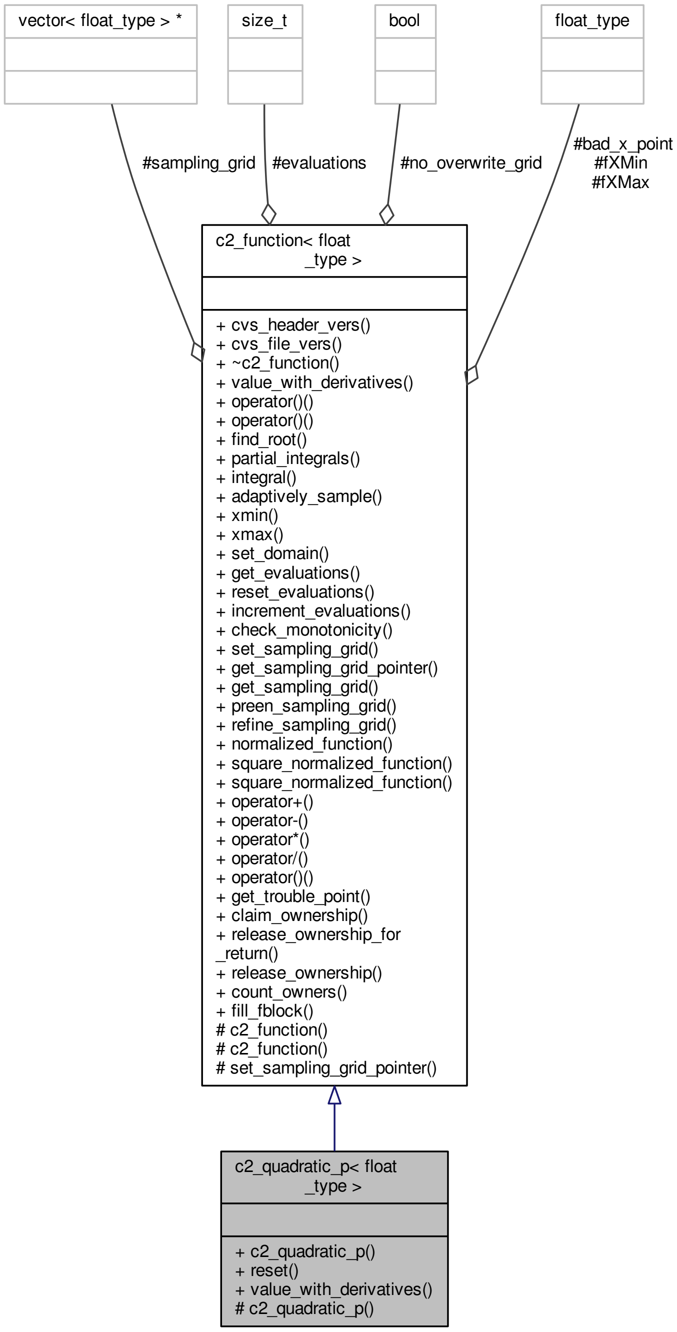
for example, given a c2_function *f*

```
#include "c2_function.hh"
```

Inheritance diagram for c2_quadratic_p< float_type >:



Collaboration diagram for c2_quadratic_p< float_type >:



Public Member Functions

- c2_quadratic_p (float_type x0, float_type y0, float_type xcoef, float_type x2coef)

  Construct the operator.
- void reset (float_type x0, float_type y0, float_type xcoef, float_type x2coef)

  Modify the coefficients after construction.

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

  get the value and derivatives.

Protected Member Functions

- c2_quadratic_p ()

Additional Inherited Members

6.40.1 Detailed Description

template<typename float_type = double>
class c2_quadratic_p< float_type >

create a quadratic mapping of another function

for example, given a c2_function *f*

```
c2_function<double> &F=c2_quadratic<double>(1.2, 2.0, 3.0, 4.0)(f);
```

produces a new c2_function F=2.0 + 3.0∗(f-1.2) + 4.0∗(f-1.2)^2

note that the parameters are overdetermined, but allows the flexibility of two different representations

The factory function c2_factory::quadratic() creates ∗new c2_quadratic_p

6.40.2 Constructor & Destructor Documentation

6.40.2.1 template<typename float_type = double> c2_quadratic_p< float_type >::c2_quadratic_p (float_type *x0*, float_type *y0*, float_type *xcoef*, float_type *x2coef*) [inline]

Construct the operator.

Parameters

x0	the center around which the powers are computed
y0	the value of the function at *x* = *x0*
xcoef	the scale on the (*x* - *x0*) term
x2coef	the scale on the (*x* - *x0*)^2 term

6.40.2.2 template<typename float_type = double> c2_quadratic_p< float_type >::c2_quadratic_p ( ) [inline], [protected]

6.40.3 Member Function Documentation

6.40.3.1 template< typename float_type = double > void c2_quadratic_p< float_type >::reset (float_type *x0*, float_type *y0*, float_type *xcoef*, float_type *x2coef*) [inline]

Modify the coefficients after construction.

Parameters

x0	the new center around which the powers are computed
y0	the new value of the function at *x* = *x0*
xcoef	the new scale on the (*x* - *x0*) term
x2coef	the new scale on the (*x* - *x0*)^2 term

6.40.3.2 template< typename float_type = double > virtual float_type c2_quadratic_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

- c2_function.hh

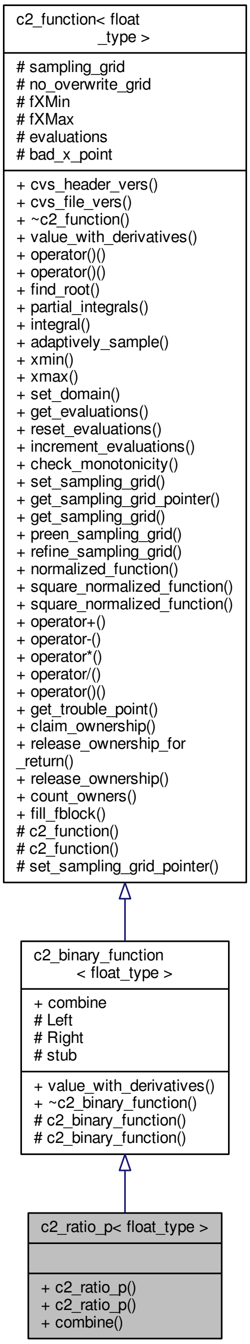
6.41 c2_ratio_p< float_type > Class Template Reference

create a c2_function which is the ratio of two other c2_functions.

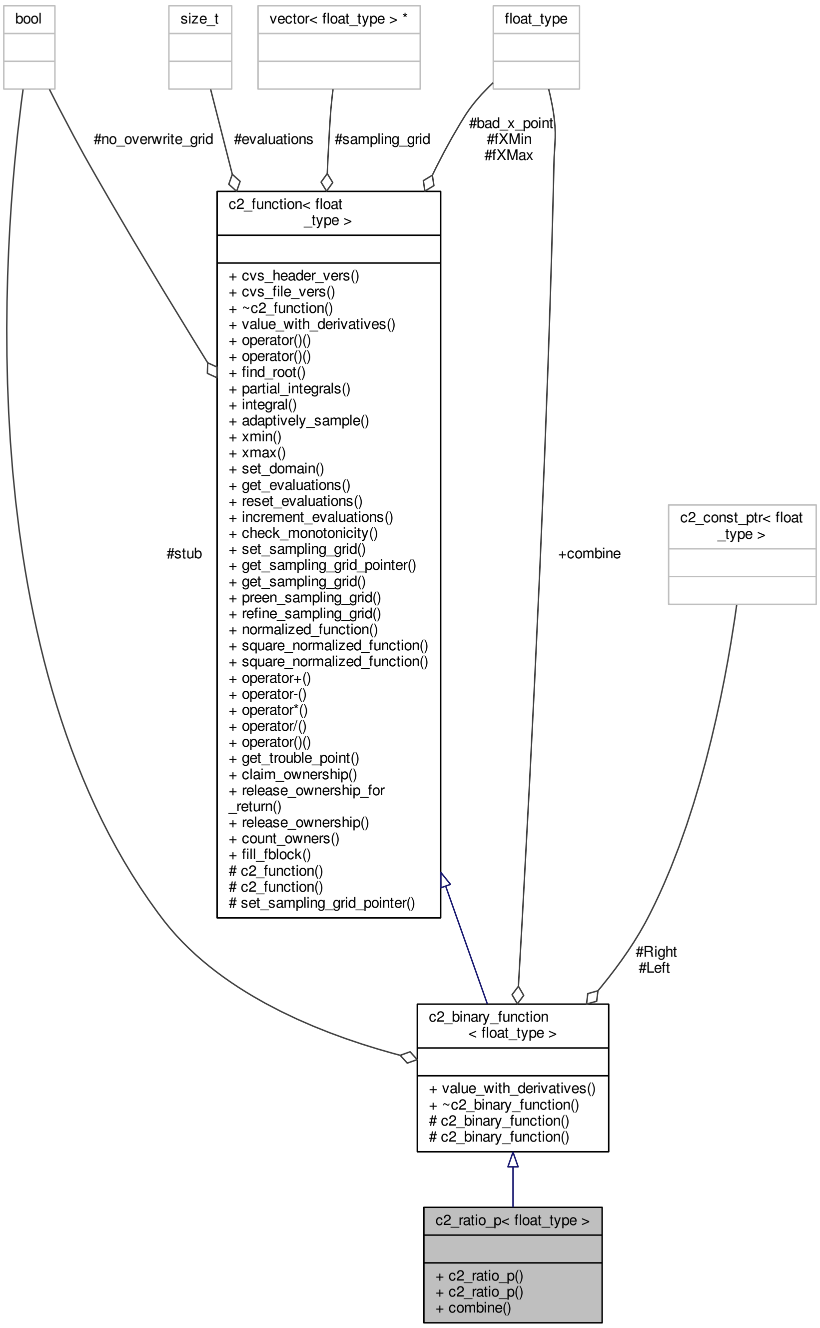
This should always be constructed using c2_function::operator/()

```
#include "c2_function.hh"
```

Inheritance diagram for c2_ratio_p< float_type >:



Collaboration diagram for c2_ratio_p< float_type >:



Public Member Functions

- c2_ratio_p (const c2_function< float_type > &left, const c2_function< float_type > &right)

    construct left / right

- c2_ratio_p ()

    Create a stub just for the combiner to avoid statics.

Static Public Member Functions

- static float_type combine (const c2_function< float_type > &left, const c2_function< float_type > &right, float_type x, float_type *yprime, float_type *yprime2) throw (c2_exception)

  execute math necessary to do division

Additional Inherited Members

6.41.1 Detailed Description

template<typename float_type = double>
class c2_ratio_p< float_type >

create a c2_function which is the ratio of two other c2_functions.

This should always be constructed using c2_function::operator/()

6.41.2 Constructor & Destructor Documentation

6.41.2.1 template<typename float_type = double> c2_ratio_p< float_type >::c2_ratio_p (const c2_function< float_type > & *left*, const c2_function< float_type > & *right*) [inline]

construct *left* / *right*

Parameters

left	the left function
right	the right function

6.41.2.2 template<typename float_type = double> c2_ratio_p< float_type >::c2_ratio_p ( ) [inline]

Create a stub just for the combiner to avoid statics.

6.41.3 Member Function Documentation

6.41.3.1 template<typename float_type = double> static float_type c2_ratio_p< float_type >::combine (const c2_function< float_type > & *left*, const c2_function< float_type > & *right*, float_type *x*, float_type * *yprime*, float_type * *yprime2*) throw c2_exception) [inline], [static]

execute math necessary to do division

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

- c2_function.hh

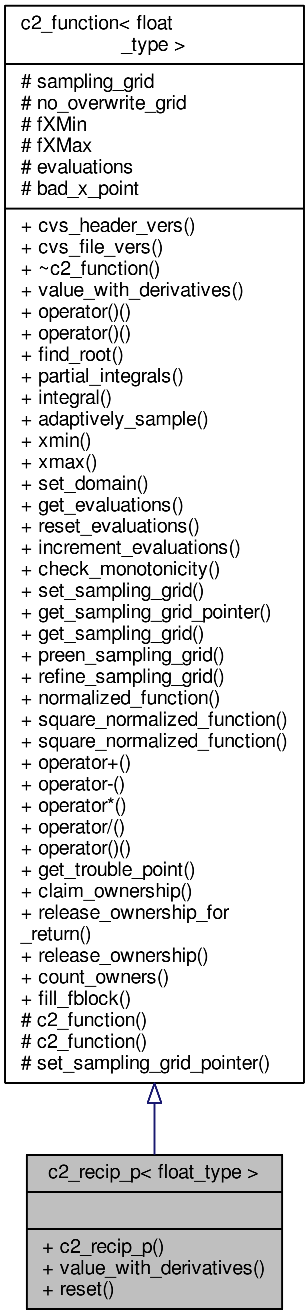
6.42 c2_recip_p< float_type > Class Template Reference

compute scale/x with its derivatives.

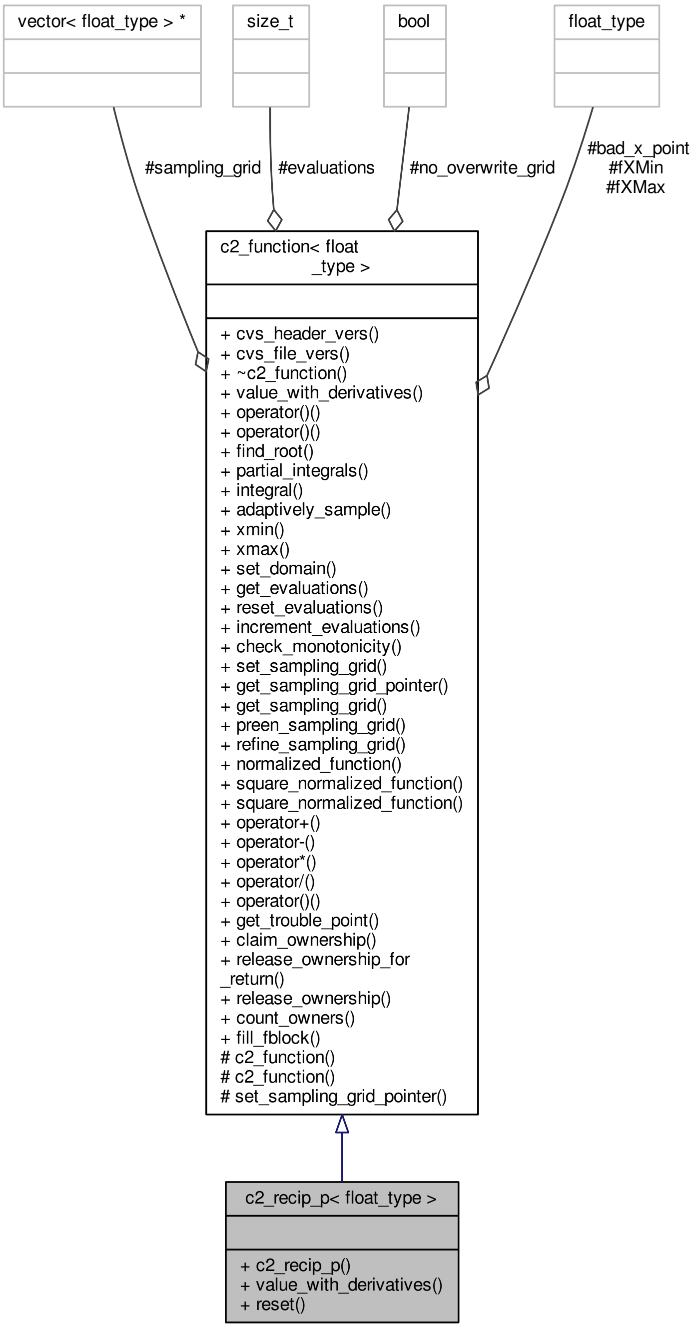
The factory function c2_factory::recip() creates ∗new c2_recip_p

```
#include "c2_function.hh"
```

Inheritance diagram for c2_recip_p< float_type >:



Collaboration diagram for c2_recip_p< float_type >:



Public Member Functions

- c2_recip_p (float_type scale)

    constructor.

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.

- void reset (float_type scale)

    reset the scale factor

Additional Inherited Members

6.42.1 Detailed Description

template<typename float_type = double>
class c2_recip_p< float_type >

compute scale/x with its derivatives.

The factory function c2_factory::recip() creates ∗new c2_recip_p

6.42.2 Constructor & Destructor Documentation

6.42.2.1 template<typename float_type = double> c2_recip_p< float_type >::c2_recip_p (float_type *scale*) `[inline]`

constructor.

6.42.3 Member Function Documentation

6.42.3.1 template<typename float_type = double> void c2_recip_p< float_type >::reset (float_type *scale*) `[inline]`

reset the scale factor

Parameters

scale	the new numerator

6.42.3.2 template<typename float_type = double> virtual float_type c2_recip_p< float_type >::value_with_derivatives (float_type *x*, float_type ∗ *yprime*, float_type ∗ *yprime2*) const throw c2_exception) `[inline]`, `[virtual]`

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

`in`	*x*	the point at which to evaluate the function
`out`	*yprime*	the first derivative (if pointer is non-null)
`out`	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

- c2_function.hh

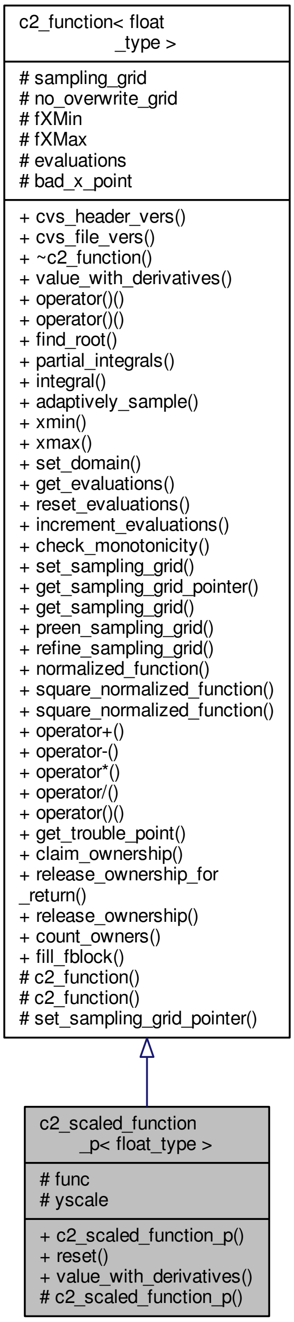
6.43 c2_scaled_function_p< float_type > Class Template Reference

Create a very lightweight method to return a scalar multiple of another function. \\

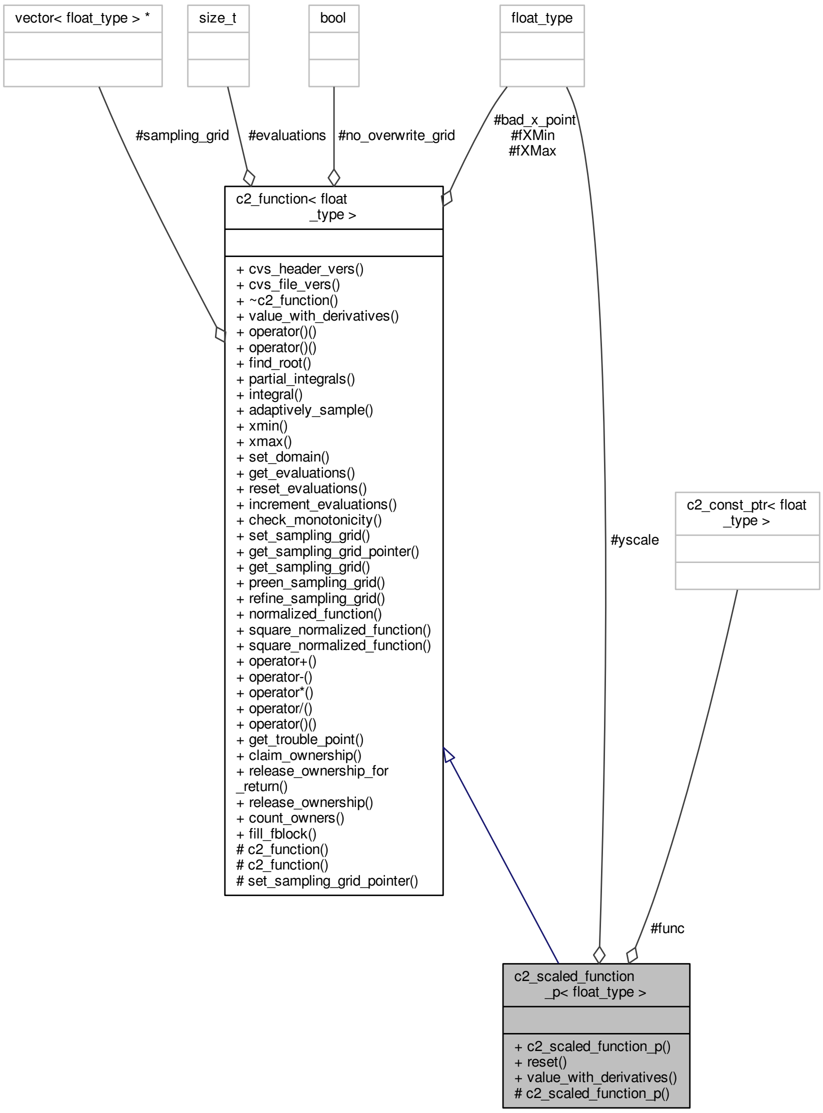
The factory function c2_factory::scaled_function() creates *new c2_scaled_function_p.

```
#include "c2_function.hh"
```

Inheritance diagram for c2_scaled_function_p< float_type >:



Collaboration diagram for c2_scaled_function_p< float_type >:



Public Member Functions

- c2_scaled_function_p (const c2_function< float_type > &outer, float_type scale)
  construct the function with its scale factor.
- void reset (float_type scale)
  set a new scale factor
- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)
  get the value and derivatives.

Protected Member Functions

- c2_scaled_function_p ()

Protected Attributes

- const c2_const_ptr< float_type > func

    the scaling factor for the function
- float_type yscale

6.43.1 Detailed Description

template<typename float_type = double>
class c2_scaled_function_p< float_type >

Create a very lightweight method to return a scalar multiple of another function. \\

The factory function c2_factory::scaled_function() creates *new c2_scaled_function_p.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 template<typename float_type = double> c2_scaled_function_p< float_type >::c2_scaled_function_p (const c2_function< float_type > & *outer*, float_type *scale*) [inline]

construct the function with its scale factor.

Parameters

outer	the function to be scaled
scale	the multiplicative scale factor

6.43.2.2 template<typename float_type = double> c2_scaled_function_p< float_type >::c2_scaled_function_p ( ) [inline], [protected]

6.43.3 Member Function Documentation

6.43.3.1 template<typename float_type = double> void c2_scaled_function_p< float_type >::reset (float_type *scale*) [inline]

set a new scale factor

Parameters

scale	the new factor

6.43.3.2 template<typename float_type = double> virtual float_type c2_scaled_function_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

provide our own value_with_derivatives which bypasses the combiner for quicker operation

Implements c2_function< float_type >.

6.43.4 Member Data Documentation

6.43.4.1 template<typename float_type = double> const c2_const_ptr<float_type> c2_scaled_function_p< float_type >::func `[protected]`

the scaling factor for the function

6.43.4.2 template<typename float_type = double> float_type c2_scaled_function_p< float_type >::yscale `[protected]`

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

- c2_function.hh

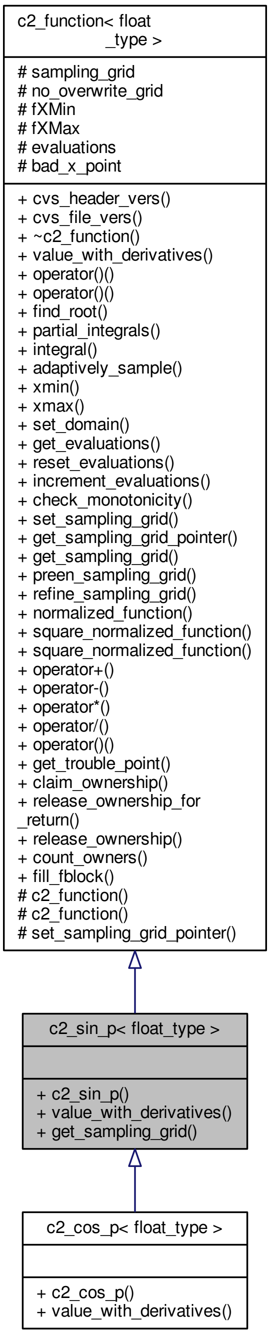
6.44 c2_sin_p< float_type > Class Template Reference

compute sin(x) with its derivatives.

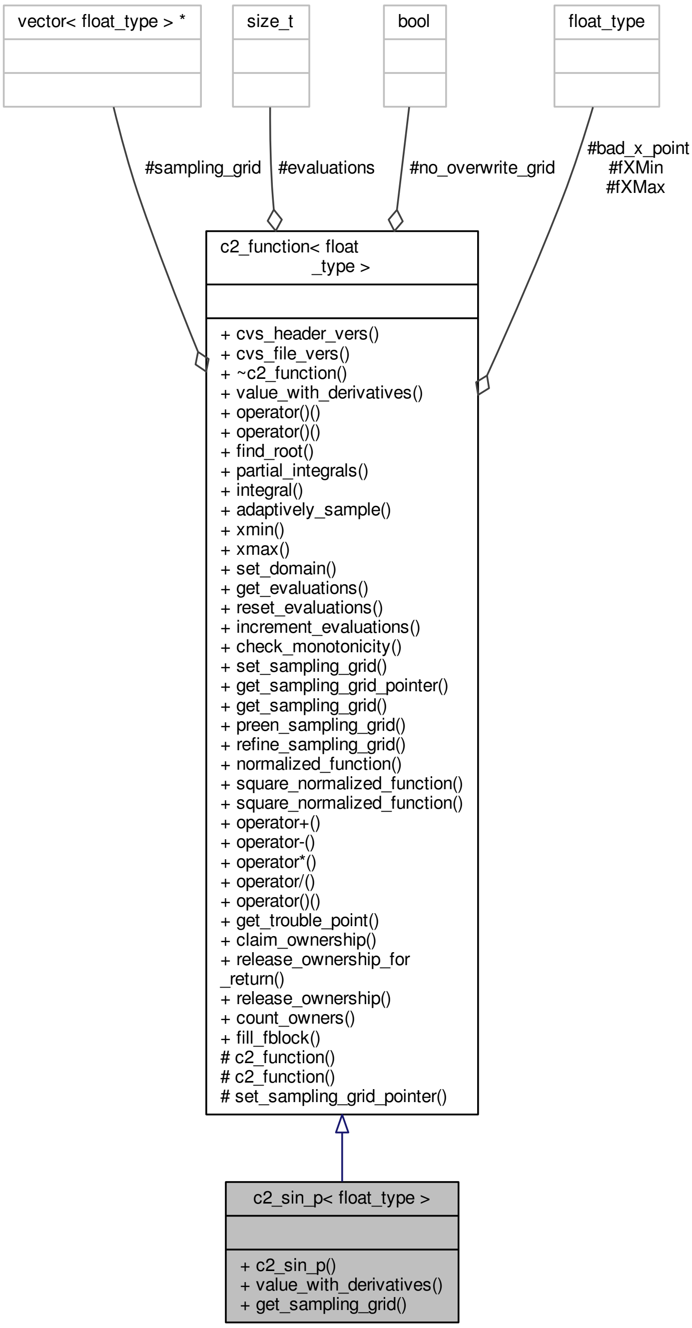
The factory function c2_factory::sin() creates *new c2_sin_p

```
#include "c2_function.hh"
```

Inheritance diagram for c2_sin_p< float_type >:



Collaboration diagram for c2_sin_p< float_type >:



Public Member Functions

- c2_sin_p ()

  constructor.
- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

  get the value and derivatives.

- virtual void get_sampling_grid (float_type amin, float_type amax, std::vector< float_type > &grid) const
  return a grid dynamically, suitable for use with trig functions with period 2∗pi

Additional Inherited Members

6.44.1 Detailed Description

template<typename float_type = double>
class c2_sin_p< float_type >

compute sin(x) with its derivatives.

The factory function c2_factory::sin() creates ∗new c2_sin_p

6.44.2 Constructor & Destructor Documentation

6.44.2.1 template<typename float_type = double> c2_sin_p< float_type >::c2_sin_p ( ) [inline]

constructor.

6.44.3 Member Function Documentation

6.44.3.1 template<typename float_type = double> virtual void c2_sin_p< float_type >::get_sampling_grid (float_type *amin*, float_type *amax*, std::vector< float_type > & *grid*) const [virtual]

return a grid dynamically, suitable for use with trig functions with period 2∗pi

Parameters

	amin	the lower bound for the grid
	amax	upper bound for the grid
in,out	*grid*	the sampling grid.

Reimplemented from c2_function< float_type >.

6.44.3.2 template<typename float_type = double> virtual float_type c2_sin_p< float_type >::value_with_derivatives (float_type *x*, float_type ∗ *yprime*, float_type ∗ *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

Reimplemented in c2_cos_p< float_type >.

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

- c2_function.hh

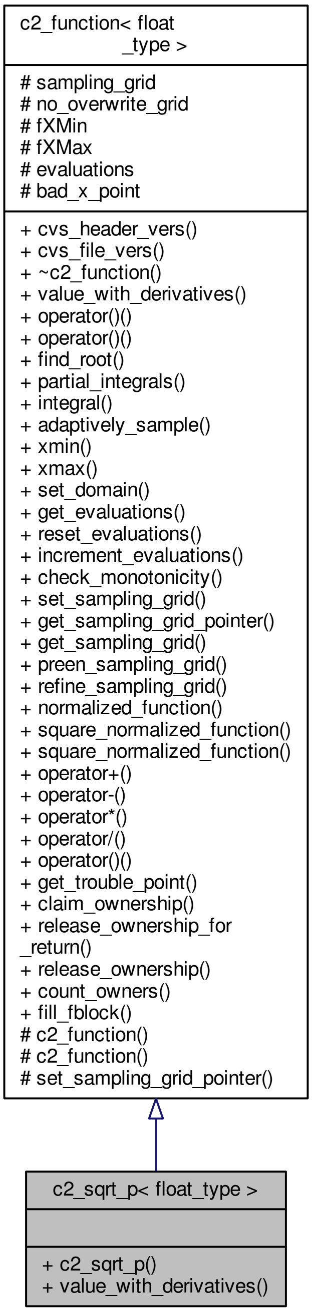
6.45 c2_sqrt_p< float_type > Class Template Reference

compute sqrt(x) with its derivatives.

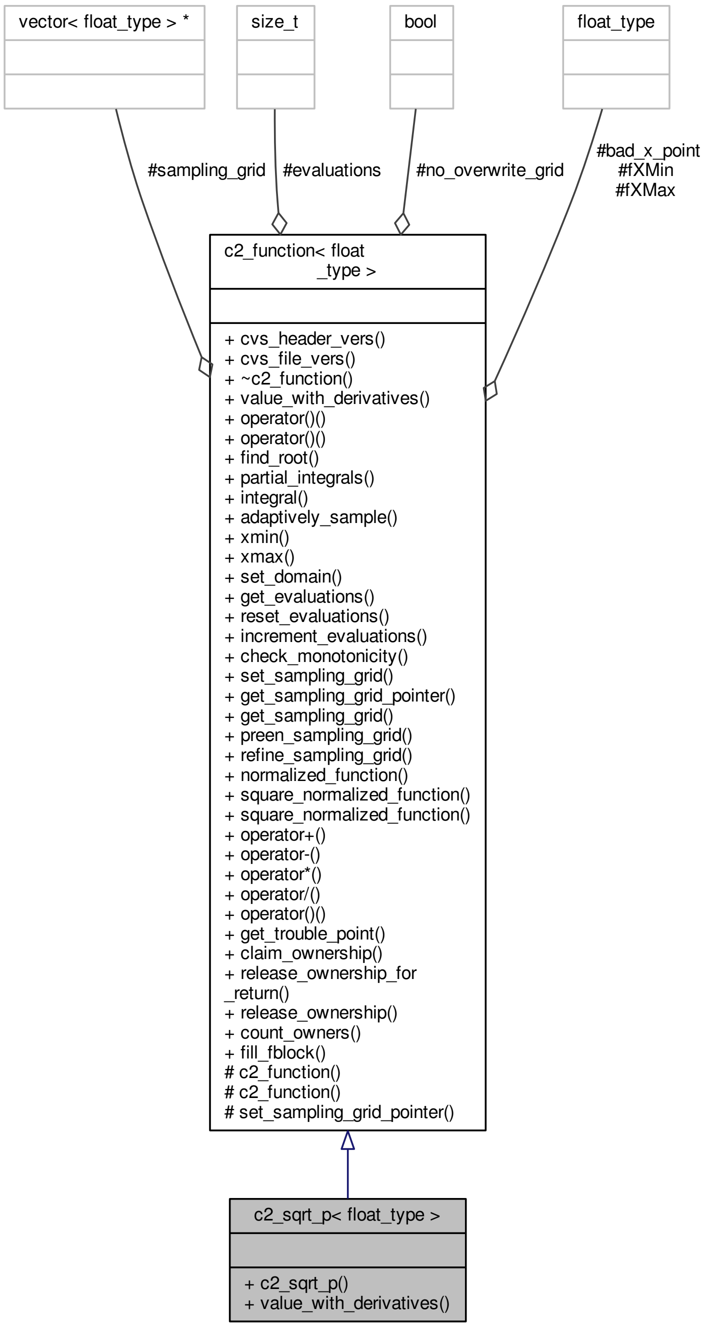
The factory function c2_factory::sqrt() creates *new c2_sqrt_p()

```
#include "c2_function.hh"
```

Inheritance diagram for c2_sqrt_p< float_type >:



Collaboration diagram for c2_sqrt_p< float_type >:



Public Member Functions

- c2_sqrt_p ()

    constructor.

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.

Additional Inherited Members

6.45.1 Detailed Description

template<typename float_type = double>
class c2_sqrt_p< float_type >

compute sqrt(x) with its derivatives.

The factory function c2_factory::sqrt() creates ∗new c2_sqrt_p()

6.45.2 Constructor & Destructor Documentation

6.45.2.1 template<typename float_type = double> c2_sqrt_p< float_type >::c2_sqrt_p ( ) [inline]

constructor.

6.45.3 Member Function Documentation

6.45.3.1 template<typename float_type = double> virtual float_type c2_sqrt_p< float_type >::value_with_derivatives (float_type *x*, float_type ∗ *yprime*, float_type ∗ *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

- c2_function.hh

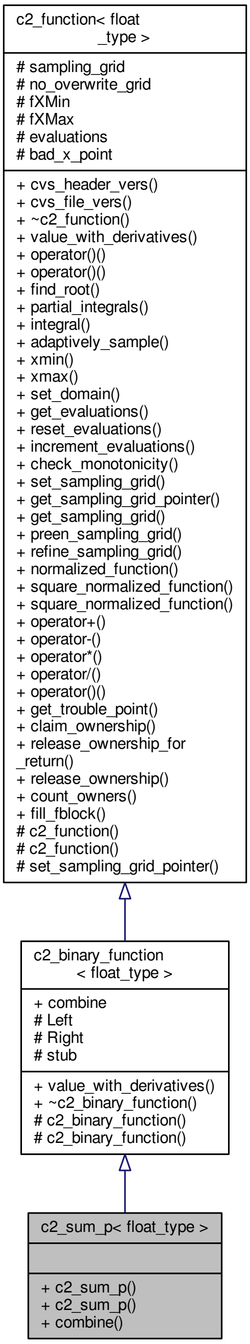
6.46 c2_sum_p< float_type > Class Template Reference

create a c2_function which is the sum of two other c2_function objects.

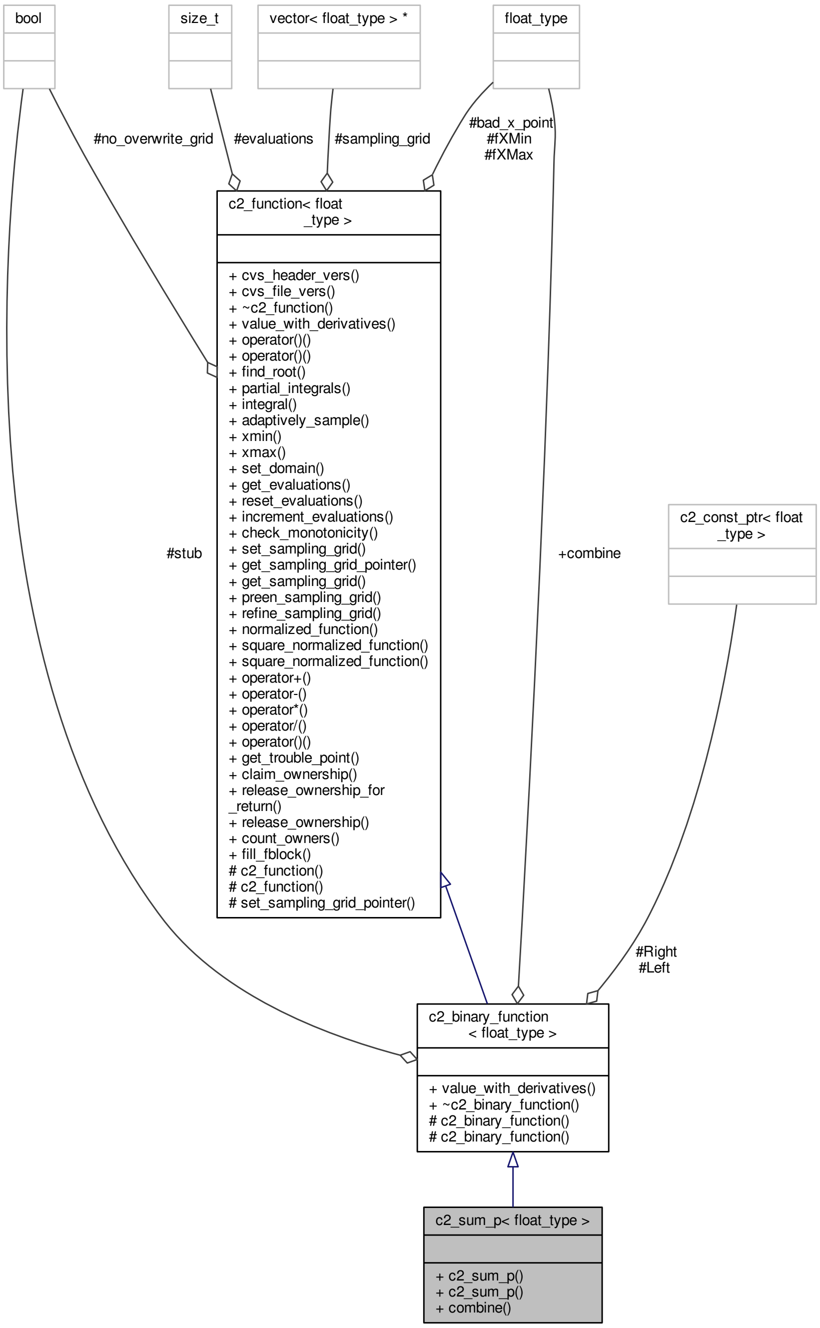
This should always be constructed using c2_function::operator+()

```
#include "c2_function.hh"
```

Inheritance diagram for c2_sum_p< float_type >:



Collaboration diagram for c2_sum_p< float_type >:



Public Member Functions

- c2_sum_p (const c2_function< float_type > &left, const c2_function< float_type > &right)

    construct left + right

- c2_sum_p ()

    Create a stub just for the combiner to avoid statics.

Static Public Member Functions

- static float_type combine (const c2_function< float_type > &left, const c2_function< float_type > &right, float_type x, float_type *yprime, float_type *yprime2) throw (c2_exception)

  execute math necessary to do addition

Additional Inherited Members

6.46.1 Detailed Description

template<typename float_type = double>
class c2_sum_p< float_type >

create a c2_function which is the sum of two other c2_function objects.

This should always be constructed using c2_function::operator+()

6.46.2 Constructor & Destructor Documentation

6.46.2.1 template<typename float_type = double> c2_sum_p< float_type >::c2_sum_p (const c2_function< float_type > & *left*, const c2_function< float_type > & *right*) [inline]

construct *left* + *right*

Parameters

left	the left function
right	the right function

6.46.2.2 template<typename float_type = double> c2_sum_p< float_type >::c2_sum_p ( ) [inline]

Create a stub just for the combiner to avoid statics.

6.46.3 Member Function Documentation

6.46.3.1 template<typename float_type = double> static float_type c2_sum_p< float_type >::combine (const c2_function< float_type > & *left*, const c2_function< float_type > & *right*, float_type *x*, float_type * *yprime*, float_type * *yprime2*) throw c2_exception) [inline], [static]

execute math necessary to do addition

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

- c2_function.hh

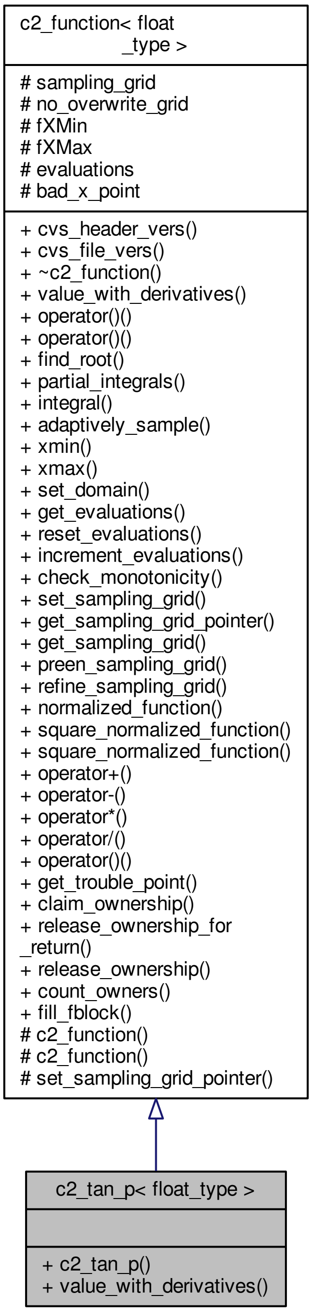
6.47 c2_tan_p$<$ float_type $>$ Class Template Reference

compute tan(x) with its derivatives.

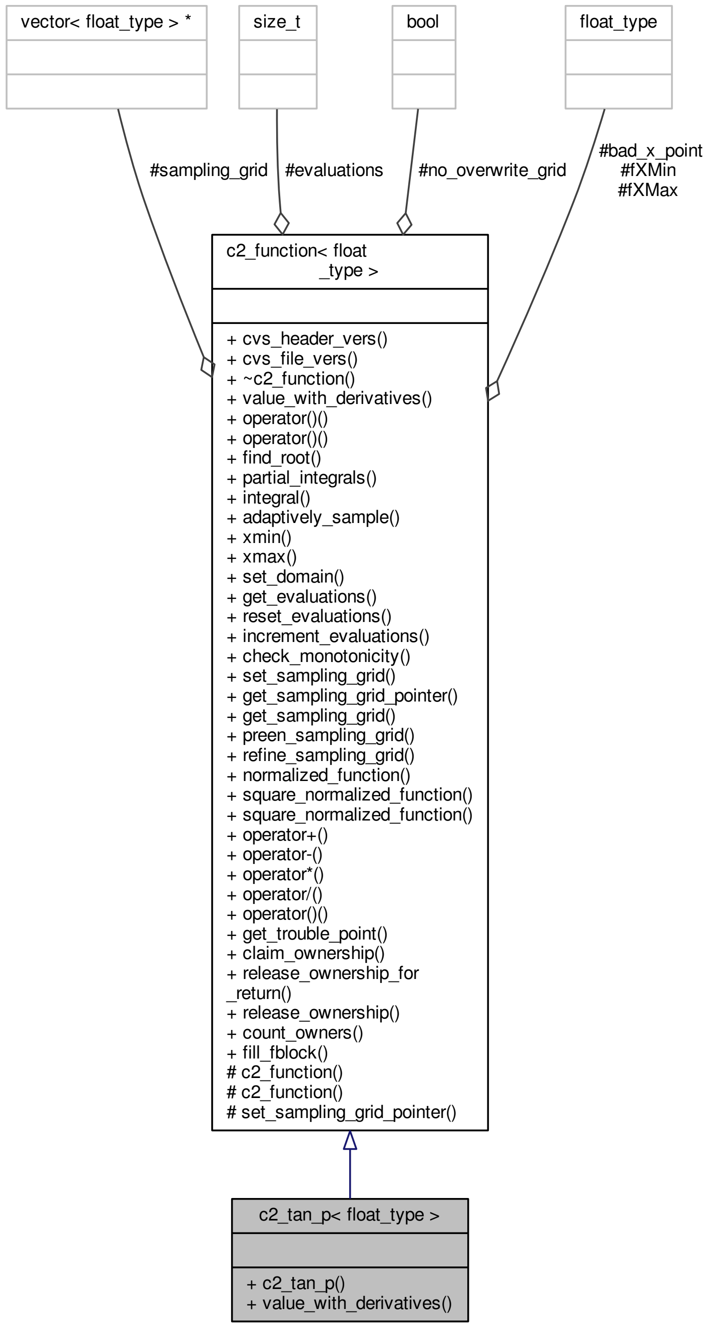
The factory function c2_factory::tan() creates ∗new c2_tan_p

`#include "c2_function.hh"`

Inheritance diagram for c2_tan_p$<$ float_type $>$:



Collaboration diagram for c2_tan_p< float_type >:



Public Member Functions

- c2_tan_p ()

    constructor.

- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.

Additional Inherited Members

6.47.1 Detailed Description

template<typename float_type = double>
class c2_tan_p< float_type >

compute tan(x) with its derivatives.

The factory function c2_factory::tan() creates *new c2_tan_p

6.47.2 Constructor & Destructor Documentation

6.47.2.1 template<typename float_type = double> c2_tan_p< float_type >::c2_tan_p ( ) [inline]

constructor.

6.47.3 Member Function Documentation

6.47.3.1 template<typename float_type = double> virtual float_type c2_tan_p< float_type >::value_with_derivatives (float_type *x*, float_type * *yprime*, float_type * *yprime2*) const throw c2_exception) [inline], [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

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

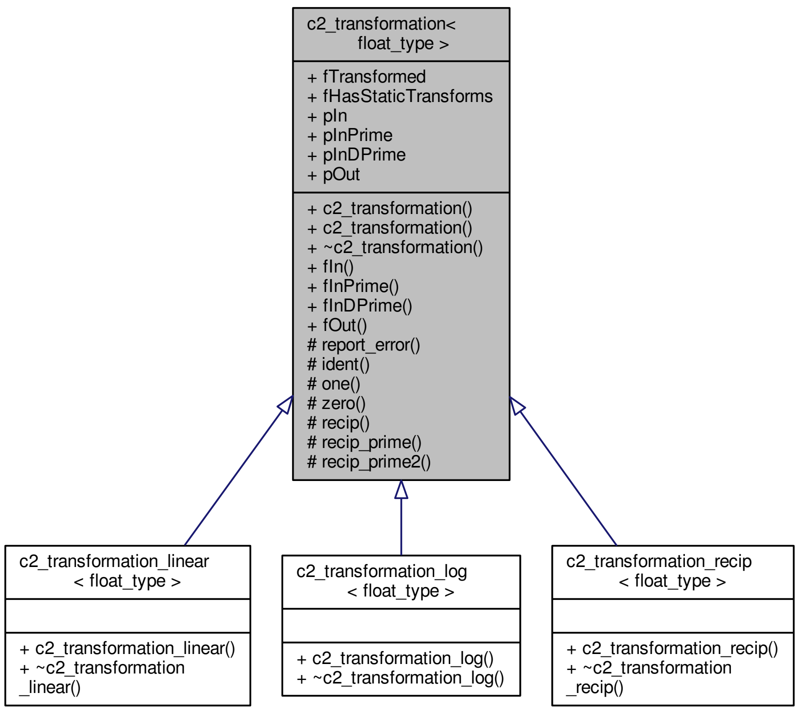
- c2_function.hh

6.48 c2_transformation< float_type > Class Template Reference

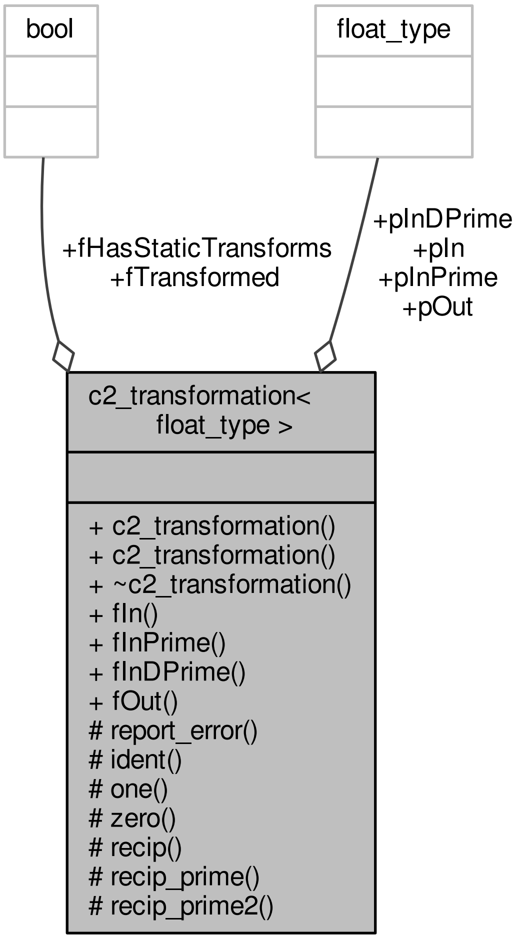
a transformation of a coordinate, including an inverse

`#include "c2_function.hh"`

Inheritance diagram for c2_transformation< float_type >:



Collaboration diagram for c2_transformation< float_type >:



Public Member Functions

- c2_transformation (bool transformed, float_type(∗xin)(float_type), float_type(∗xinp)(float_type), float_type(∗xinpp)(float_type), float_type(∗xout)(float_type))

  initialize all our function pointers
- c2_transformation (bool transformed)

  initialize all our function pointers so that only the (overridden) virtual functions can be called without an error
- virtual ∼c2_transformation ()

  the destructor
- virtual float_type fIn (float_type x) const

  virtual input X transform
- virtual float_type fInPrime (float_type x) const

  virtual input X transform derivative
- virtual float_type fInDPrime (float_type x) const

  virtual input X transform second derivative
- virtual float_type fOut (float_type x) const

  virtual output X transform

Public Attributes

- const bool fTransformed

  flag to indicate if this transform is not the identity
- const bool fHasStaticTransforms

  flag to indicate if the static function pointers can be used for efficiency
- float_type(∗const pIn)(float_type)

  non-virtual pointer to input X transform
- float_type(∗const pInPrime)(float_type)

  non-virtual pointer to input X transform derivative
- float_type(∗const pInDPrime)(float_type)

  non-virtual pointer to input X transform second derivative
- float_type(∗const pOut)(float_type)

  non-virtual pointer to output X transform

Static Protected Member Functions

- static float_type report_error (float_type x)

  utility function for unimplemented conversion
- static float_type ident (float_type x)

  utility function f(x)=x useful in axis transforms
- static float_type one (float_type)

  utility function f(x)=1 useful in axis transforms
- static float_type zero (float_type)

  utility function f(x)=0 useful in axis transforms
- static float_type recip (float_type x)

  utility function f(x)=1/x useful in axis transforms
- static float_type recip_prime (float_type x)

  utility function f(x)=-1/x∗∗2 useful in axis transforms
- static float_type recip_prime2 (float_type x)

  utility function f(x)=2/x∗∗3 useful in axis transforms

6.48.1 Detailed Description

template<typename float_type>
class c2_transformation< float_type >

a transformation of a coordinate, including an inverse

6.48.2 Constructor & Destructor Documentation

6.48.2.1 template<typename float_type> c2_transformation< float_type >::c2_transformation (bool *transformed*, float_type(∗)(float_type) *xin*, float_type(∗)(float_type) *xinp*, float_type(∗)(float_type) *xinpp*, float_type(∗)(float_type) *xout*) [inline]

initialize all our function pointers

Parameters

transformed	true if this function is not the identity
xin	input X transform
xinp	input X transform derivative
xinpp	input X transform second derivative
xout	output X transform, which MUST be the inverse of *xin*

6.48.2.2 template<typename float_type> c2_transformation< float_type >::c2_transformation (bool *transformed*) [inline]

initialize all our function pointers so that only the (overridden) virtual functions can be called without an error

Parameters

transformed	true if this function is nonlinear

6.48.2.3 template<typename float_type> virtual c2_transformation< float_type >::~c2_transformation ( ) [inline], [virtual]

the destructor

6.48.3 Member Function Documentation

6.48.3.1 template<typename float_type> virtual float_type c2_transformation< float_type >::fIn (float_type *x*) const [inline], [virtual]

virtual input X transform

6.48.3.2 template<typename float_type> virtual float_type c2_transformation< float_type >::fInDPrime (float_type *x*) const [inline], [virtual]

virtual input X transform second derivative

6.48.3.3 template<typename float_type> virtual float_type c2_transformation< float_type >::fInPrime (float_type *x*) const [inline], [virtual]

virtual input X transform derivative

6.48.3.4 template<typename float_type> virtual float_type c2_transformation< float_type >::fOut (float_type *x*) const [inline], [virtual]

virtual output X transform

6.48.3.5 template<typename float_type> static float_type c2_transformation< float_type >::ident (float_type *x*) [inline], [static], [protected]

utility function f(x)=x useful in axis transforms

6.48.3.6 template<typename float_type> static float_type c2_transformation< float_type >::one (float_type) [inline], [static], [protected]

utility function f(x)=1 useful in axis transforms

6.48.3.7 template<typename float_type> static float_type c2_transformation< float_type >::recip (float_type *x*) [inline], [static], [protected]

utility function f(x)=1/x useful in axis transforms

6.48.3.8 template<typename float_type> static float_type c2_transformation< float_type >::recip_prime (float_type *x*) [inline], [static], [protected]

utility function f(x)=-1/x**2 useful in axis transforms

6.48.3.9 template<typename float_type> static float_type c2_transformation< float_type >::recip_prime2 (float_type *x*) [inline], [static], [protected]

utility function f(x)=2/x**3 useful in axis transforms

6.48.3.10 template<typename float_type> static float_type c2_transformation< float_type >::report_error (float_type *x*) [inline], [static], [protected]

utility function for unimplemented conversion

6.48.3.11 template<typename float_type> static float_type c2_transformation< float_type >::zero (float_type) [inline], [static], [protected]

utility function f(x)=0 useful in axis transforms

6.48.4 Member Data Documentation

6.48.4.1 template<typename float_type> const bool c2_transformation< float_type >::fHasStaticTransforms

flag to indicate if the static function pointers can be used for efficiency

6.48.4.2 template<typename float_type> const bool c2_transformation< float_type >::fTransformed

flag to indicate if this transform is not the identity

6.48.4.3 template< typename float_type > float_type(∗ const c2_transformation< float_type >::pIn) (float_type)

non-virtual pointer to input X transform

Note

the pointers to functions allow highly optimized access when static functions are available. They are only used inside value_with_derivatives(), which is assumed to be the most critical routine.

6.48.4.4 template< typename float_type > float_type(∗ const c2_transformation< float_type >::pInDPrime) (float_type)

non-virtual pointer to input X transform second derivative

6.48.4.5 template< typename float_type > float_type(∗ const c2_transformation< float_type >::pInPrime) (float_type)

non-virtual pointer to input X transform derivative

6.48.4.6 template< typename float_type > float_type(∗ const c2_transformation< float_type >::pOut) (float_type)

non-virtual pointer to output X transform

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

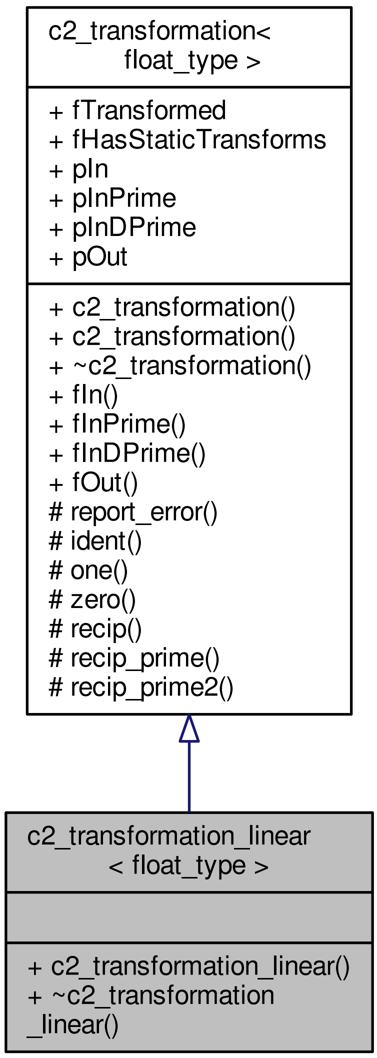
- c2_function.hh

6.49 c2_transformation_linear< float_type > Class Template Reference

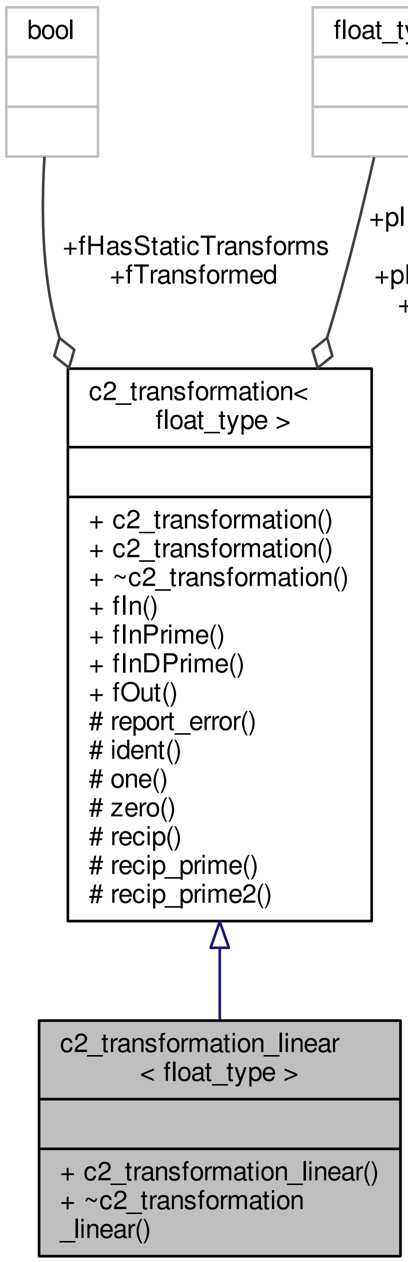
the identity transform

```
#include "c2_function.hh"
```

Inheritance diagram for c2_transformation_linear< float_type >:



Collaboration diagram for c2_transformation_linear< float_type >:



Public Member Functions

- c2_transformation_linear ()

    constructor
- virtual ∼c2_transformation_linear ()

    destructor

Additional Inherited Members

6.49.1 Detailed Description

template<typename float_type>
class c2_transformation_linear< float_type >

the identity transform

6.49.2 Constructor & Destructor Documentation

6.49.2.1 template<typename float_type > c2_transformation_linear< float_type >::c2_transformation_linear ( ) [inline]

constructor

6.49.2.2 template<typename float_type > virtual c2_transformation_linear< float_type >::~c2_transformation_linear ( ) [inline], [virtual]

destructor

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

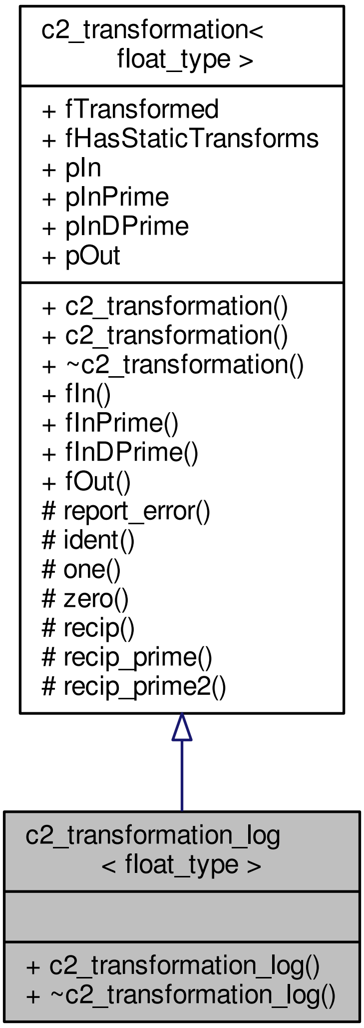
- c2_function.hh

6.50 c2_transformation_log< float_type > Class Template Reference

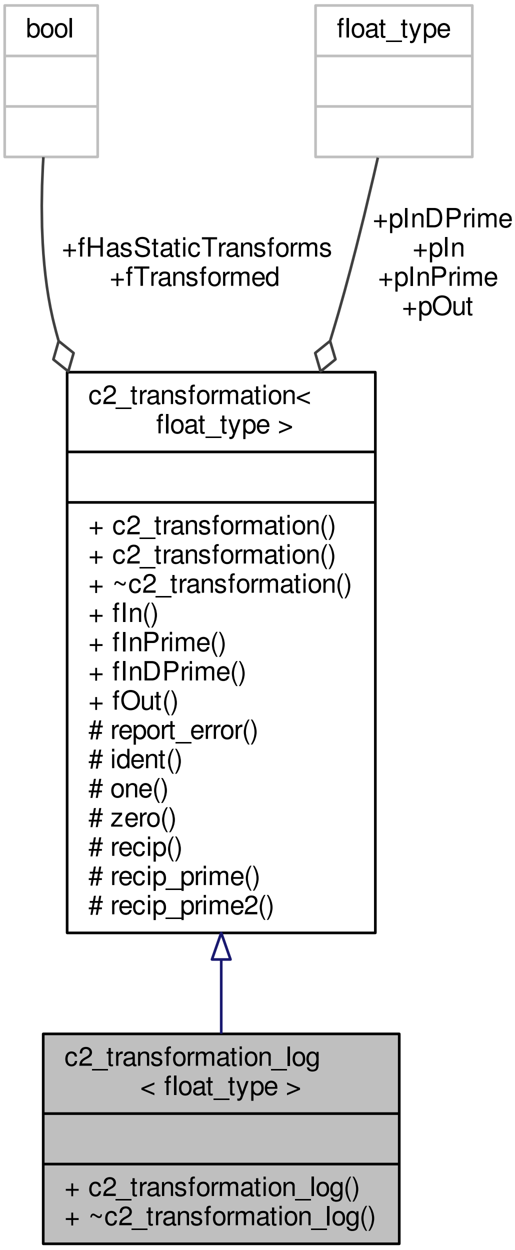
log axis transform

`#include "c2_function.hh"`

Inheritance diagram for c2_transformation_log< float_type >:



Collaboration diagram for c2_transformation_log< float_type >:



Public Member Functions

- c2_transformation_log ()

  constructor
- virtual ∼c2_transformation_log ()

  destructor

Additional Inherited Members

6.50.1 Detailed Description

template<typename float_type>
class c2_transformation_log< float_type >

log axis transform

6.50.2 Constructor & Destructor Documentation

6.50.2.1 template<typename float_type > c2_transformation_log< float_type >::c2_transformation_log ( ) [inline]

constructor

6.50.2.2 template<typename float_type > virtual c2_transformation_log< float_type >::~c2_transformation_log ( ) [inline], [virtual]

destructor

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

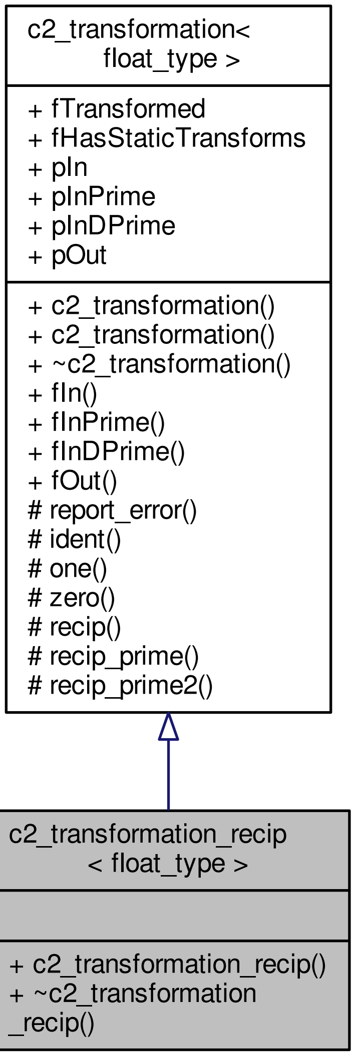
- c2_function.hh

6.51 c2_transformation_recip< float_type > Class Template Reference

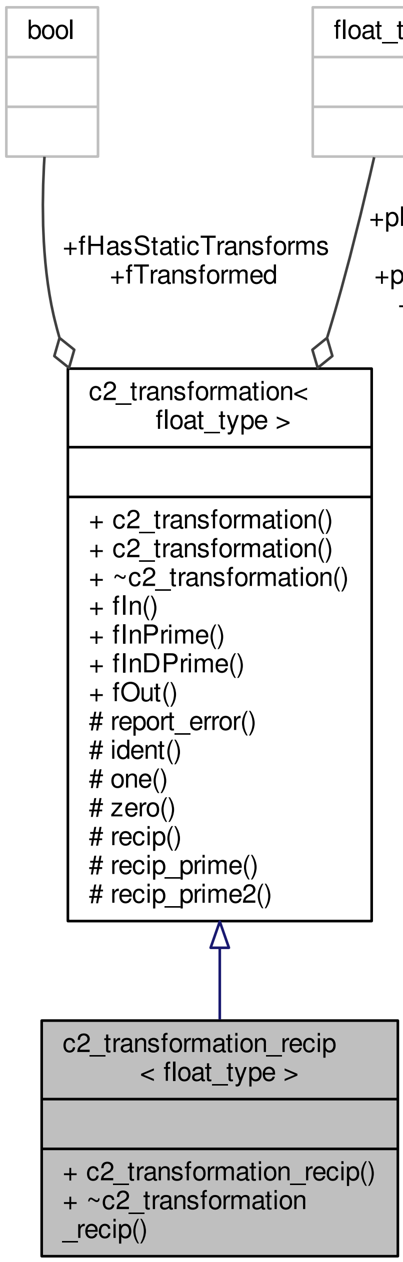
reciprocal axis transform

```
#include "c2_function.hh"
```

Inheritance diagram for c2_transformation_recip< float_type >:



Collaboration diagram for c2_transformation_recip< float_type >:



Public Member Functions

- c2_transformation_recip ()

  constructor
- virtual ∼c2_transformation_recip ()

  destructor

Additional Inherited Members

6.51.1 Detailed Description

template<typename float_type>
class c2_transformation_recip< float_type >

reciprocal axis transform

6.51.2 Constructor & Destructor Documentation

6.51.2.1 template<typename float_type > c2_transformation_recip< float_type >::c2_transformation_recip ( ) [inline]

constructor

6.51.2.2 template<typename float_type > virtual c2_transformation_recip< float_type >::~c2_transformation_recip ( ) [inline], [virtual]

destructor

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

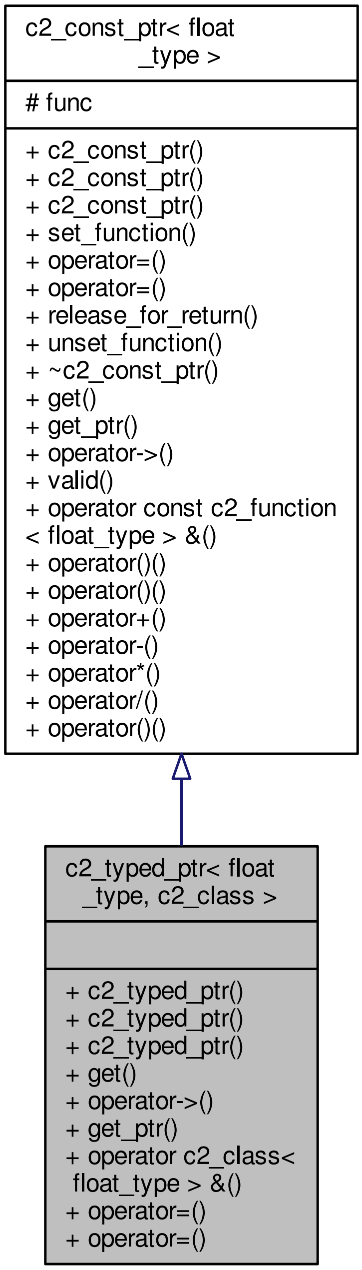
- c2_function.hh

6.52 c2_typed_ptr< float_type, c2_class > Class Template Reference

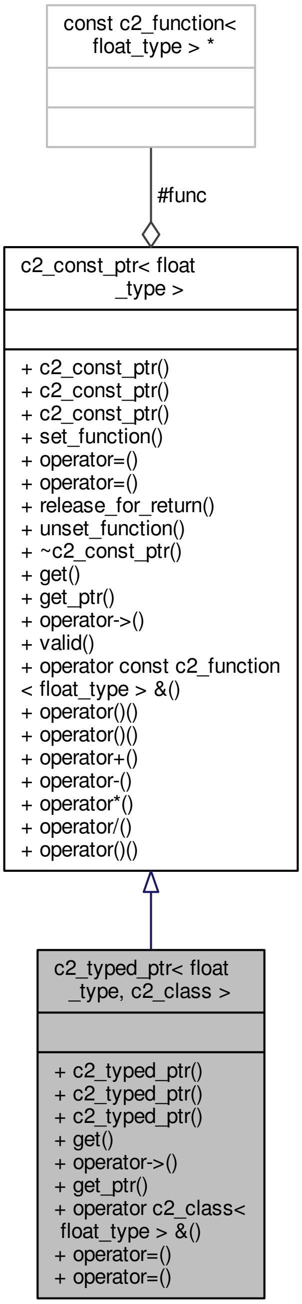
create a non-generic container for a c2_function which handles the reference counting.

```
#include "c2_function.hh"
```

Inheritance diagram for c2_typed_ptr< float_type, c2_class >:



Collaboration diagram for c2_typed_ptr< float_type, c2_class >:



Public Member Functions

- c2_typed_ptr ()

    construct the container with no function
- c2_typed_ptr (c2_class< float_type > &f)

    construct the container with a pre-defined function
- c2_typed_ptr (const c2_typed_ptr< float_type, c2_class > &src)

copy constructor
- c2_class< float_type > & get () const throw (c2_exception)

  get a reference to our owned function
- c2_class< float_type > ∗ operator-> () const

  get a checked pointer to our owned function
- c2_class< float_type > ∗ get_ptr () const

  get an unchecked pointer to our owned function
- operator c2_class< float_type > & () const

  type coercion operator which lets us use a pointer as if it were a c2_function
- void operator= (const c2_typed_ptr< float_type, c2_class > &f)

  fill the container from another container
- void operator= (c2_class< float_type > &f)

  fill the container with a function

Additional Inherited Members

6.52.1 Detailed Description

template<typename float_type, template< typename > class c2_class>
class c2_typed_ptr< float_type, c2_class >

create a non-generic container for a c2_function which handles the reference counting.

See also

c2_const_ptr and Use of c2_ptr for memory management

Note

Overuse of this class will generate massive bloat. Use c2_ptr and c2_const_ptr if you don't *really* need specific pointer types.

See also

Use of c2_ptr for memory management

6.52.2 Constructor & Destructor Documentation

6.52.2.1 template<typename float_type, template< typename > class c2_class> c2_typed_ptr< float_type, c2_class >::c2_typed_ptr ( ) [inline]

construct the container with no function

6.52.2.2 template<typename float_type, template< typename > class c2_class> c2_typed_ptr< float_type, c2_class >::c2_typed_ptr (c2_class< float_type > & *f*) [inline]

construct the container with a pre-defined function

Parameters

f	the function to store

6.52.2.3 template<typename float_type, template< typename > class c2_class> c2_typed_ptr< float_type, c2_class >::c2_typed_ptr (const c2_typed_ptr< float_type, c2_class > & *src*) [inline]

copy constructor

Parameters

src	the container to copy

6.52.3 Member Function Documentation

6.52.3.1 template<typename float_type, template< typename > class c2_class> c2_class<float_type>& c2_typed_ptr< float_type, c2_class >::get ( ) const throw c2_exception) [inline]

get a reference to our owned function

6.52.3.2 template<typename float_type, template< typename > class c2_class> c2_class<float_type>* c2_typed_ptr< float_type, c2_class >::get_ptr ( ) const [inline]

get an unchecked pointer to our owned function

6.52.3.3 template<typename float_type, template< typename > class c2_class> c2_typed_ptr< float_type, c2_class >::operator c2_class< float_type > & ( ) const [inline]

type coercion operator which lets us use a pointer as if it were a c2_function

6.52.3.4 template<typename float_type, template< typename > class c2_class> c2_class<float_type>* c2_typed_ptr< float_type, c2_class >::operator-> ( ) const [inline]

get a checked pointer to our owned function

6.52.3.5 template<typename float_type, template< typename > class c2_class> void c2_typed_ptr< float_type, c2_class >::operator= (const c2_typed_ptr< float_type, c2_class > & *f*) [inline]

fill the container from another container

Parameters

f	the container to copy

6.52.3.6 template<typename float_type, template< typename > class c2_class> void c2_typed_ptr< float_type, c2_class >::operator= (c2_class< float_type > & *f*) [inline]

fill the container with a function

Parameters

f	the function

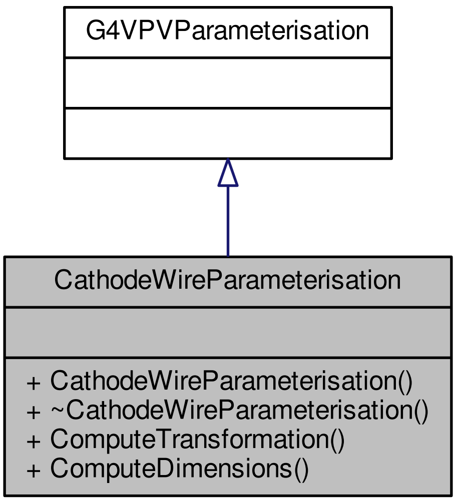
The documentation for this class was generated from the following file:

- c2_function.hh

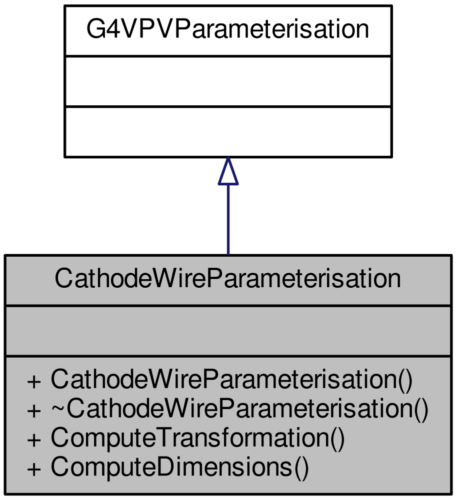
6.53 CathodeWireParameterisation Class Reference

```
#include "CathodeWireParameterisation.hh"
```

Inheritance diagram for CathodeWireParameterisation:



Collaboration diagram for CathodeWireParameterisation:



Public Member Functions

- CathodeWireParameterisation (G4int noWires, G4double startX, G4double spacing, G4double wireRadius, G4double wireLength)
- virtual ∼CathodeWireParameterisation ()
- void ComputeTransformation (const G4int copyNo, G4VPhysicalVolume ∗physVol) const
- void ComputeDimensions (G4Tubs &CathodeWire, const G4int copyNo, const G4VPhysicalVolume ∗physVol) const

6.53.1 Detailed Description

A parameterisation that describes a series of cylinders along X.

The cylinders have equal radius and length. They are spaced an equal distance apart, starting from given location.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 CathodeWireParameterisation::CathodeWireParameterisation (G4int *noWires,* G4double *startX,* G4double *spacing,* G4double *wireRadius,* G4double *wireLength*)

6.53.2.2 virtual CathodeWireParameterisation::∼CathodeWireParameterisation ( ) [virtual]

6.53.3 Member Function Documentation

6.53.3.1 void CathodeWireParameterisation::ComputeDimensions (G4Tubs & *CathodeWire,* const G4int *copyNo,* const G4VPhysicalVolume ∗ *physVol*) const

6.53.3.2 void CathodeWireParameterisation::ComputeTransformation (const G4int *copyNo,* G4VPhysicalVolume ∗ *physVol*) const

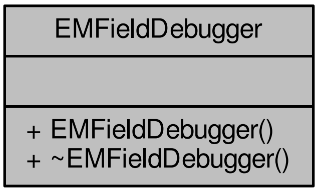
The documentation for this class was generated from the following file:

- CathodeWireParameterisation.hh

6.54 EMFieldDebugger Class Reference

```
#include "EMFieldDebugger.hh"
```

Collaboration diagram for EMFieldDebugger:



Public Member Functions

- EMFieldDebugger (int icomp)
- ∼EMFieldDebugger ()

6.54.1 Constructor & Destructor Documentation

6.54.1.1 EMFieldDebugger::EMFieldDebugger (int *icomp*)

6.54.1.2 EMFieldDebugger::∼EMFieldDebugger ( )

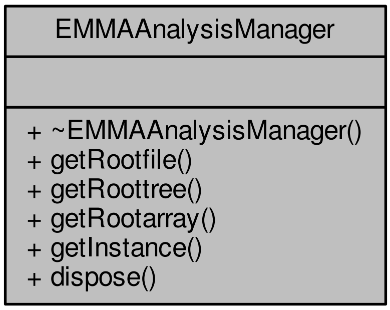
The documentation for this class was generated from the following file:

- EMFieldDebugger.hh

6.55 EMMAAnalysisManager Class Reference

`#include "EMMAAnalysisManager.hh"`

Collaboration diagram for EMMAAnalysisManager:



Public Member Functions

- virtual ∼EMMAAnalysisManager ()
- TFile ∗ getRootfile ()
- TTree ∗ getRoottree ()
- TObjArray ∗ getRootarray ()

Static Public Member Functions

- static EMMAAnalysisManager ∗ getInstance ()
- static void dispose ()

6.55.1 Constructor & Destructor Documentation

6.55.1.1 virtual EMMAAnalysisManager::∼EMMAAnalysisManager ( ) [virtual]

6.55.2 Member Function Documentation

6.55.2.1 static void EMMAAnalysisManager::dispose ( ) [static]

6.55.2.2 static EMMAAnalysisManager∗ EMMAAnalysisManager::getInstance ( ) [static]

6.55.2.3 TObjArray∗ EMMAAnalysisManager::getRootarray ( )

6.55.2.4 TFile∗ EMMAAnalysisManager::getRootfile ( )

6.55.2.5 TTree∗ EMMAAnalysisManager::getRoottree ( )

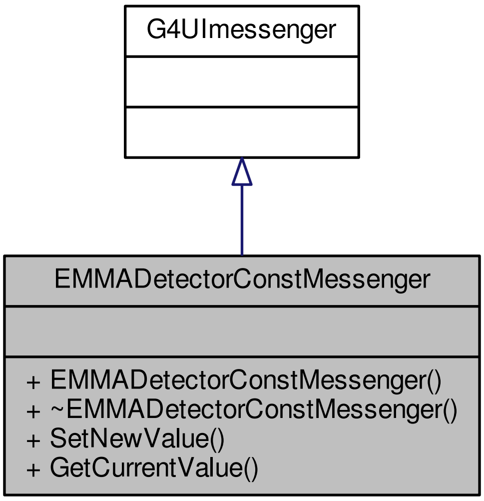
The documentation for this class was generated from the following file:

- EMMAAnalysisManager.hh

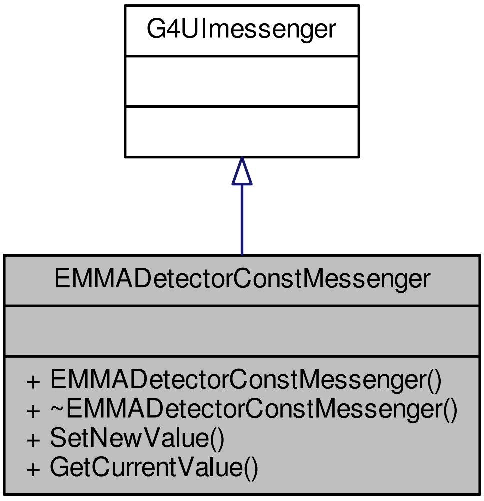
6.56 EMMADetectorConstMessenger Class Reference

```
#include "EMMADetectorConstMessenger.hh"
```

Inheritance diagram for EMMADetectorConstMessenger:



Collaboration diagram for EMMADetectorConstMessenger:



Public Member Functions

- EMMADetectorConstMessenger (EMMADetectorConstruction ∗mpga)
- ∼EMMADetectorConstMessenger ()
- void SetNewValue (G4UIcommand ∗command, G4String newValues)
- G4String GetCurrentValue (G4UIcommand ∗command)

6.56.1 Constructor & Destructor Documentation

6.56.1.1 EMMADetectorConstMessenger::EMMADetectorConstMessenger (EMMADetectorConstruction ∗ *mpga*)

6.56.1.2 EMMADetectorConstMessenger::∼EMMADetectorConstMessenger ( )

6.56.2 Member Function Documentation

6.56.2.1 G4String EMMADetectorConstMessenger::GetCurrentValue (G4UIcommand ∗ *command*)

6.56.2.2 void EMMADetectorConstMessenger::SetNewValue (G4UIcommand ∗ *command,* G4String *newValues*)

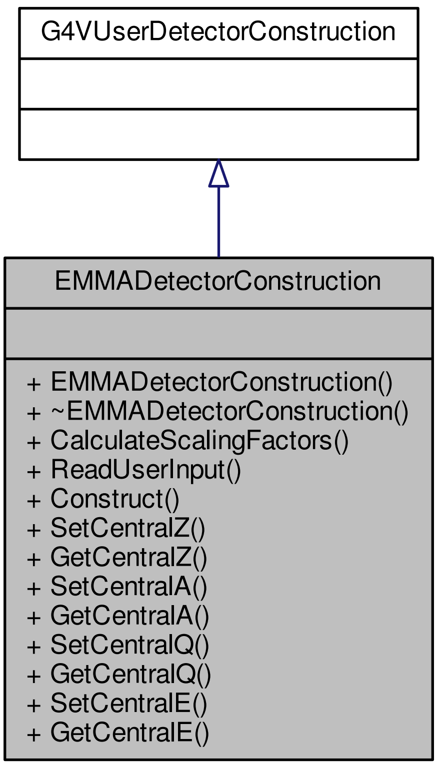
The documentation for this class was generated from the following file:

- EMMADetectorConstMessenger.hh

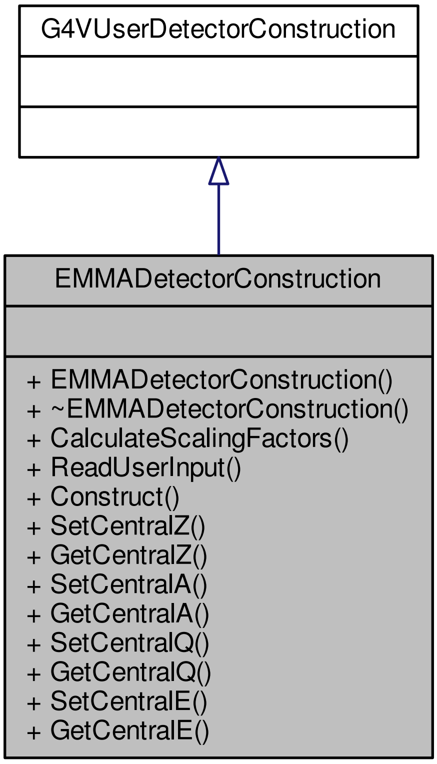
6.57 EMMADetectorConstruction Class Reference

```
#include "EMMADetectorConstruction.hh"
```

Inheritance diagram for EMMADetectorConstruction:



Collaboration diagram for EMMADetectorConstruction:



Public Member Functions

- EMMADetectorConstruction ()
- virtual ∼EMMADetectorConstruction ()
- void CalculateScalingFactors ()
- void ReadUserInput ()
- virtual G4VPhysicalVolume ∗ Construct ()
- void SetCentralZ (G4double val)
- G4double GetCentralZ () const
- void SetCentralA (G4double val)
- G4double GetCentralA () const
- void SetCentralQ (G4double val)
- G4double GetCentralQ () const
- void SetCentralE (G4double val)
- G4double GetCentralE () const

6.57.1 Constructor & Destructor Documentation

6.57.1.1 EMMADetectorConstruction::EMMADetectorConstruction ( )

6.57.1.2 virtual EMMADetectorConstruction::∼EMMADetectorConstruction ( ) [virtual]

6.57.2 Member Function Documentation

6.57.2.1 void EMMADetectorConstruction::CalculateScalingFactors ( )

6.57.2.2 virtual G4VPhysicalVolume∗ EMMADetectorConstruction::Construct ( ) [virtual]

6.57.2.3 G4double EMMADetectorConstruction::GetCentralA ( ) const [inline]

6.57.2.4 G4double EMMADetectorConstruction::GetCentralE ( ) const [inline]

6.57.2.5 G4double EMMADetectorConstruction::GetCentralQ ( ) const [inline]

6.57.2.6 G4double EMMADetectorConstruction::GetCentralZ ( ) const [inline]

6.57.2.7 void EMMADetectorConstruction::ReadUserInput ( )

6.57.2.8 void EMMADetectorConstruction::SetCentralA (G4double *val*) [inline]

6.57.2.9 void EMMADetectorConstruction::SetCentralE (G4double *val*) [inline]

6.57.2.10 void EMMADetectorConstruction::SetCentralQ (G4double *val*) [inline]

6.57.2.11 void EMMADetectorConstruction::SetCentralZ (G4double *val*) [inline]

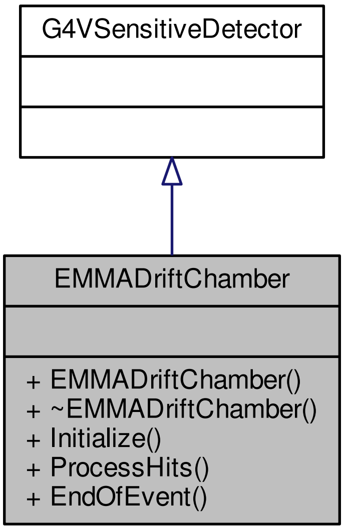
The documentation for this class was generated from the following file:

- EMMADetectorConstruction.hh

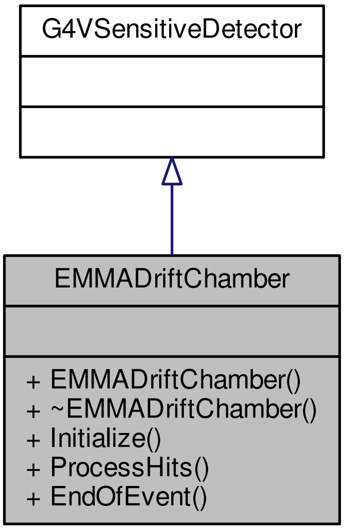
6.58 EMMADriftChamber Class Reference

```
#include "EMMADriftChamber.hh"
```

Inheritance diagram for EMMADriftChamber:



Collaboration diagram for EMMADriftChamber:



Public Member Functions

- EMMADriftChamber (G4String name)

- virtual ∼EMMADriftChamber ()
- virtual void Initialize (G4HCofThisEvent ∗HCE)
- virtual G4bool ProcessHits (G4Step ∗aStep, G4TouchableHistory ∗ROhist)
- virtual void EndOfEvent (G4HCofThisEvent ∗HCE)

6.58.1 Constructor & Destructor Documentation

6.58.1.1 EMMADriftChamber::EMMADriftChamber (G4String *name*)

6.58.1.2 virtual EMMADriftChamber::∼EMMADriftChamber ( ) [virtual]

6.58.2 Member Function Documentation

6.58.2.1 virtual void EMMADriftChamber::EndOfEvent (G4HCofThisEvent ∗ *HCE*) [virtual]

6.58.2.2 virtual void EMMADriftChamber::Initialize (G4HCofThisEvent ∗ *HCE*) [virtual]

6.58.2.3 virtual G4bool EMMADriftChamber::ProcessHits (G4Step ∗ *aStep,* G4TouchableHistory ∗ *ROhist*) [virtual]

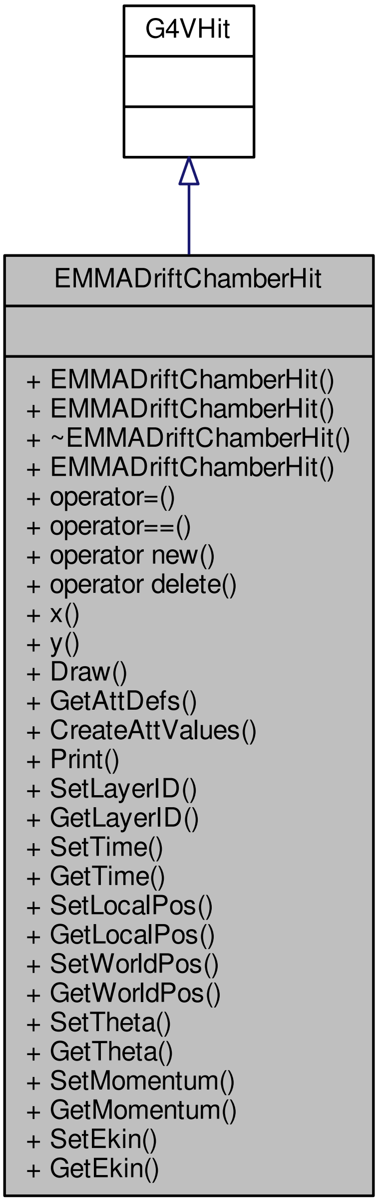
The documentation for this class was generated from the following file:

- EMMADriftChamber.hh

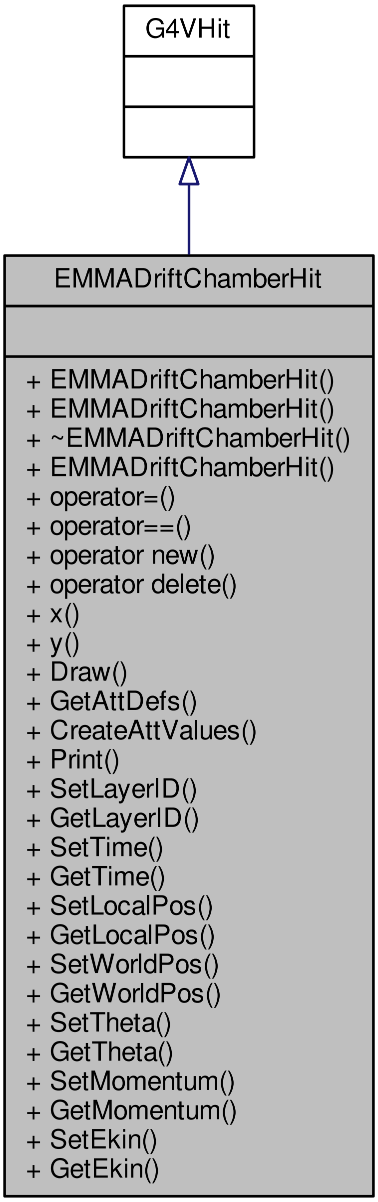
6.59 EMMADriftChamberHit Class Reference

```
#include "EMMADriftChamberHit.hh"
```

Inheritance diagram for EMMADriftChamberHit:



Collaboration diagram for EMMADriftChamberHit:



Public Member Functions

- EMMADriftChamberHit ()
- EMMADriftChamberHit (G4int z)
- virtual ∼EMMADriftChamberHit ()
- EMMADriftChamberHit (const EMMADriftChamberHit &right)
- const EMMADriftChamberHit & operator= (const EMMADriftChamberHit &right)
- int operator== (const EMMADriftChamberHit &right) const
- void ∗ operator new (size_t)

- void operator delete (void ∗)
- float x ()
- float y ()
- virtual void Draw ()
- virtual const std::map< G4String, G4AttDef > ∗ GetAttDefs () const
- virtual std::vector< G4AttValue > ∗ CreateAttValues () const
- virtual void Print ()
- void SetLayerID (G4int z)
- G4int GetLayerID () const
- void SetTime (G4double t)
- G4double GetTime () const
- void SetLocalPos (G4ThreeVector xyz)
- G4ThreeVector GetLocalPos () const
- void SetWorldPos (G4ThreeVector xyz)
- G4ThreeVector GetWorldPos () const
- void SetTheta (G4double thet)
- G4double GetTheta () const
- void SetMomentum (G4ThreeVector xyz)
- G4ThreeVector GetMomentum () const
- void SetEkin (G4double e)
- G4double GetEkin () const

6.59.1 Constructor & Destructor Documentation

6.59.1.1 EMMADriftChamberHit::EMMADriftChamberHit ( )

6.59.1.2 EMMADriftChamberHit::EMMADriftChamberHit (G4int *z*)

6.59.1.3 virtual EMMADriftChamberHit::∼EMMADriftChamberHit ( ) [virtual]

6.59.1.4 EMMADriftChamberHit::EMMADriftChamberHit (const EMMADriftChamberHit & *right*)

6.59.2 Member Function Documentation

6.59.2.1 virtual std::vector<G4AttValue>∗ EMMADriftChamberHit::CreateAttValues ( ) const [virtual]

6.59.2.2 virtual void EMMADriftChamberHit::Draw ( ) [virtual]

6.59.2.3 virtual const std::map<G4String,G4AttDef>∗ EMMADriftChamberHit::GetAttDefs ( ) const [virtual]

6.59.2.4 G4double EMMADriftChamberHit::GetEkin ( ) const [inline]

6.59.2.5 G4int EMMADriftChamberHit::GetLayerID ( ) const [inline]

6.59.2.6 G4ThreeVector EMMADriftChamberHit::GetLocalPos ( ) const [inline]

6.59.2.7 G4ThreeVector EMMADriftChamberHit::GetMomentum ( ) const [inline]

6.59.2.8 G4double EMMADriftChamberHit::GetTheta ( ) const [inline]

6.59.2.9 G4double EMMADriftChamberHit::GetTime ( ) const [inline]

6.59.2.10 G4ThreeVector EMMADriftChamberHit::GetWorldPos ( ) const [inline]

6.59.2.11 void EMMADriftChamberHit::operator delete (void * *aHit*) [inline]

6.59.2.12 void * EMMADriftChamberHit::operator new (size_t) [inline]

6.59.2.13 const EMMADriftChamberHit& EMMADriftChamberHit::operator= (const EMMADriftChamberHit & *right*)

6.59.2.14 int EMMADriftChamberHit::operator== (const EMMADriftChamberHit & *right*) const

6.59.2.15 virtual void EMMADriftChamberHit::Print ( ) [virtual]

6.59.2.16 void EMMADriftChamberHit::SetEkin (G4double *e*) [inline]

6.59.2.17 void EMMADriftChamberHit::SetLayerID (G4int *z*) [inline]

6.59.2.18 void EMMADriftChamberHit::SetLocalPos (G4ThreeVector *xyz*) [inline]

6.59.2.19 void EMMADriftChamberHit::SetMomentum (G4ThreeVector *xyz*) [inline]

6.59.2.20 void EMMADriftChamberHit::SetTheta (G4double *thet*) [inline]

6.59.2.21 void EMMADriftChamberHit::SetTime (G4double *t*) [inline]

6.59.2.22 void EMMADriftChamberHit::SetWorldPos (G4ThreeVector *xyz*) [inline]

6.59.2.23 float EMMADriftChamberHit::x ( ) [inline]

6.59.2.24 float EMMADriftChamberHit::y ( ) [inline]

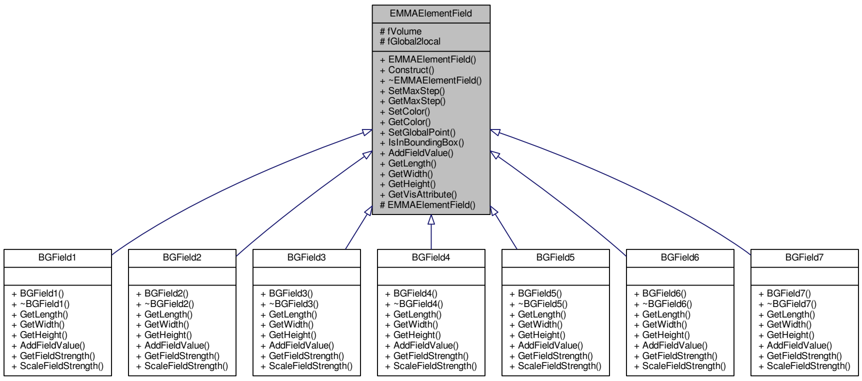
The documentation for this class was generated from the following file:

- EMMADriftChamberHit.hh

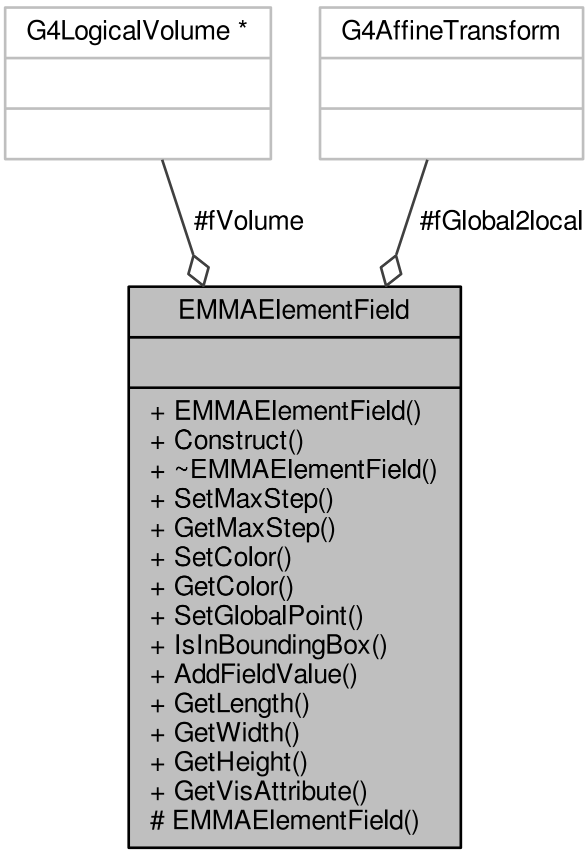
6.60 EMMAElementField Class Reference

```
#include "EMMAElementField.hh"
```

Inheritance diagram for EMMAElementField:



Collaboration diagram for EMMAElementField:



Public Member Functions

- EMMAElementField (const G4ThreeVector, G4LogicalVolume ∗)

  Constructor.
- void Construct ()

  the actual implementation constructs the F04ElementField
- virtual ∼EMMAElementField ()

  Destructor.
- void SetMaxStep (G4double stp)

  SetMaxStep(G4double) sets the max. step size.
- G4double GetMaxStep ()

  GetMaxStep() returns the max. step size.
- void SetColor (G4String c)

  SetColor(G4String) sets the color.
- G4String GetColor ()

  GetColor() returns the color.
- void SetGlobalPoint (const G4double point[4])
- bool IsInBoundingBox (const G4double point[4]) const
- virtual void AddFieldValue (const G4double point[4], G4double field[6]) const =0
- virtual G4double GetLength ()=0
- virtual G4double GetWidth ()=0
- virtual G4double GetHeight ()=0

Static Public Member Functions

- static G4VisAttributes ∗ GetVisAttribute (G4String color)

  GetVisAttribute() returns the appropriate G4VisAttributes.

Protected Member Functions

- EMMAElementField (const EMMAElementField &)

Protected Attributes

- G4LogicalVolume ∗ fVolume
- G4AffineTransform fGlobal2local

6.60.1 Constructor & Destructor Documentation

6.60.1.1 EMMAElementField::EMMAElementField (const G4ThreeVector , G4LogicalVolume ∗)

Constructor.

6.60.1.2 virtual EMMAElementField::∼EMMAElementField ( ) [inline], [virtual]

Destructor.

6.60.1.3 EMMAElementField::EMMAElementField (const EMMAElementField &) [protected]

6.60.2 Member Function Documentation

6.60.2.1 virtual void EMMAElementField::AddFieldValue (const G4double *point[4]*, G4double *field[6]*) const [pure virtual]

AddFieldValue() will add the field value for this element to field[]. Implementations must be sure to verify that point[] is within the field region, and do nothing if not. point[] is in global coordinates and geant4 units; x,y,z,t. field[] is in geant4 units; Bx,By,Bz,Ex,Ey,Ez. For efficiency, the caller may (but need not) call IsInBoundingBox(point), and only call this function if that returns true.

6.60.2.2 void EMMAElementField::Construct ( )

the actual implementation constructs the F04ElementField

6.60.2.3 G4String EMMAElementField::GetColor ( ) [inline]

GetColor() returns the color.

6.60.2.4 virtual G4double EMMAElementField::GetHeight ( ) [pure virtual]

Implemented in BGField4, BGField2, BGField1, BGField3, BGField5, BGField6, and BGField7.

6.60.2.5 virtual G4double EMMAElementField::GetLength ( ) [pure virtual]

Implemented in BGField4, BGField2, BGField1, BGField3, BGField5, BGField6, and BGField7.

6.60.2.6 G4double EMMAElementField::GetMaxStep ( ) [inline]

GetMaxStep() returns the max. step size.

6.60.2.7 static G4VisAttributes∗ EMMAElementField::GetVisAttribute (G4String *color*) [static]

GetVisAttribute() returns the appropriate G4VisAttributes.

6.60.2.8 virtual G4double EMMAElementField::GetWidth ( ) [pure virtual]

Implemented in BGField4, BGField2, BGField1, BGField3, BGField5, BGField6, and BGField7.

6.60.2.9 bool EMMAElementField::IsInBoundingBox (const G4double *point[4]*) const [inline]

IsInBoundingBox() returns true if the point is within the global bounding box - global coordinates.

6.60.2.10 void EMMAElementField::SetColor (G4String *c*) [inline]

SetColor(G4String) sets the color.

6.60.2.11 void EMMAElementField::SetGlobalPoint (const G4double *point[4]*) [inline]

SetGlobalPoint() ensures that the point is within the global bounding box of this ElementField's global coordinates. Normally called 8 times for the corners of the local bounding box, after a local->global coordinate transform. If never called, the global bounding box is infinite. BEWARE: if called only once, the bounding box is just a point.

6.60.2.12 void EMMAElementField::SetMaxStep (G4double *stp*) [inline]

SetMaxStep(G4double) sets the max. step size.

6.60.3 Member Data Documentation

6.60.3.1 G4AffineTransform EMMAElementField::fGlobal2local [protected]

6.60.3.2 G4LogicalVolume* EMMAElementField::fVolume [protected]

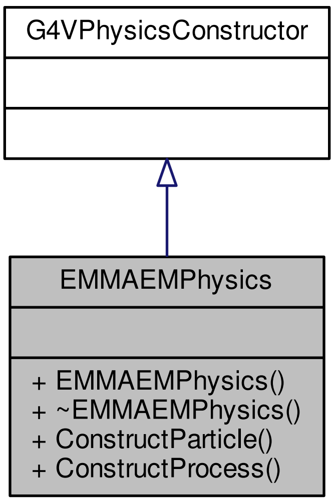
The documentation for this class was generated from the following file:

- EMMAElementField.hh

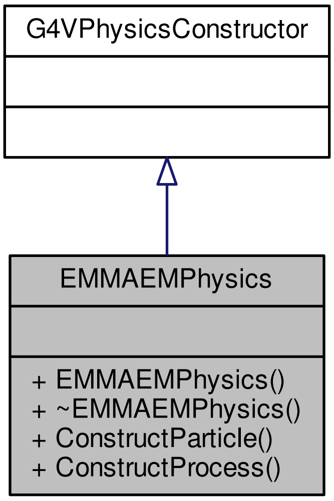
6.61 EMMAEMPhysics Class Reference

`#include "EMMAEMPhysics.hh"`

Inheritance diagram for EMMAEMPhysics:



Collaboration diagram for EMMAEMPhysics:



Public Member Functions

- EMMAEMPhysics (const G4String &name="EM")
- virtual ∼EMMAEMPhysics ()
- virtual void ConstructParticle ()
- virtual void ConstructProcess ()

6.61.1 Constructor & Destructor Documentation

6.61.1.1 EMMAEMPhysics::EMMAEMPhysics (const G4String & *name* = "EM")

6.61.1.2 virtual EMMAEMPhysics::∼EMMAEMPhysics ( ) [virtual]

6.61.2 Member Function Documentation

6.61.2.1 virtual void EMMAEMPhysics::ConstructParticle ( ) [inline], [virtual]

6.61.2.2 virtual void EMMAEMPhysics::ConstructProcess ( ) [virtual]

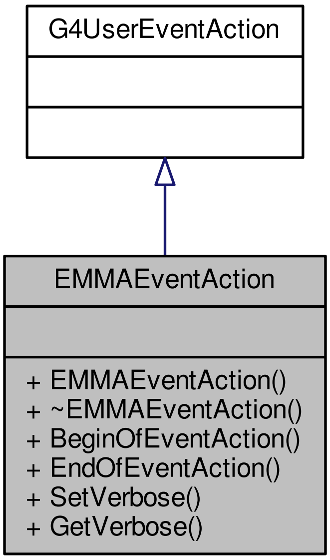
The documentation for this class was generated from the following file:

- EMMAEMPhysics.hh

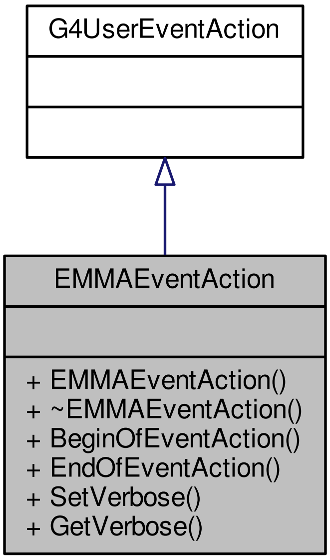
6.62 EMMAEventAction Class Reference

`#include "EMMAEventAction.hh"`

Inheritance diagram for EMMAEventAction:



Collaboration diagram for EMMAEventAction:



Public Member Functions

- EMMAEventAction ()
- virtual ∼EMMAEventAction ()
- virtual void BeginOfEventAction (const G4Event ∗)
- virtual void EndOfEventAction (const G4Event ∗)
- void SetVerbose (G4int val)
- G4int GetVerbose () const

6.62.1 Constructor & Destructor Documentation

6.62.1.1 EMMAEventAction::EMMAEventAction ( )

6.62.1.2 virtual EMMAEventAction::∼EMMAEventAction ( ) [virtual]

6.62.2 Member Function Documentation

6.62.2.1 virtual void EMMAEventAction::BeginOfEventAction (const G4Event ∗) [virtual]

6.62.2.2 virtual void EMMAEventAction::EndOfEventAction (const G4Event ∗) [virtual]

6.62.2.3 G4int EMMAEventAction::GetVerbose ( ) const [inline]

6.62.2.4 void EMMAEventAction::SetVerbose (G4int *val*) [inline]

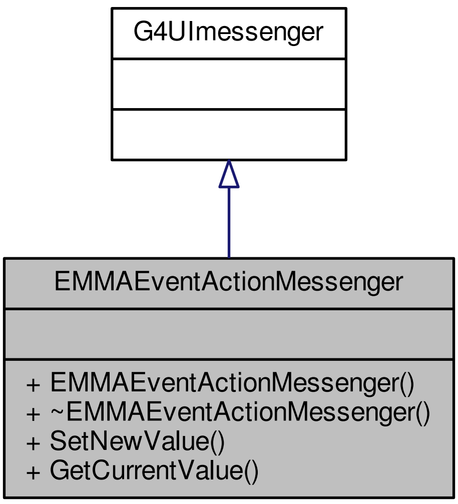
The documentation for this class was generated from the following file:

- EMMAEventAction.hh

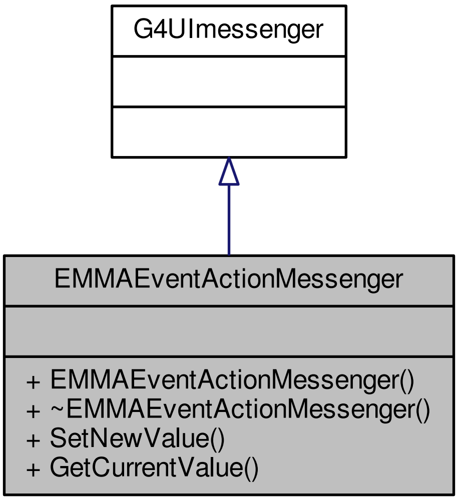
6.63 EMMAEventActionMessenger Class Reference

`#include "EMMAEventActionMessenger.hh"`

Inheritance diagram for EMMAEventActionMessenger:



Collaboration diagram for EMMAEventActionMessenger:



Public Member Functions

- EMMAEventActionMessenger (EMMAEventAction ∗mpga)
- ∼EMMAEventActionMessenger ()
- void SetNewValue (G4UIcommand ∗command, G4String newValues)
- G4String GetCurrentValue (G4UIcommand ∗command)

6.63.1 Constructor & Destructor Documentation

6.63.1.1 EMMAEventActionMessenger::EMMAEventActionMessenger (EMMAEventAction ∗ *mpga*)

6.63.1.2 EMMAEventActionMessenger::∼EMMAEventActionMessenger ( )

6.63.2 Member Function Documentation

6.63.2.1 G4String EMMAEventActionMessenger::GetCurrentValue (G4UIcommand ∗ *command*)

6.63.2.2 void EMMAEventActionMessenger::SetNewValue (G4UIcommand ∗ *command,* G4String *newValues*)

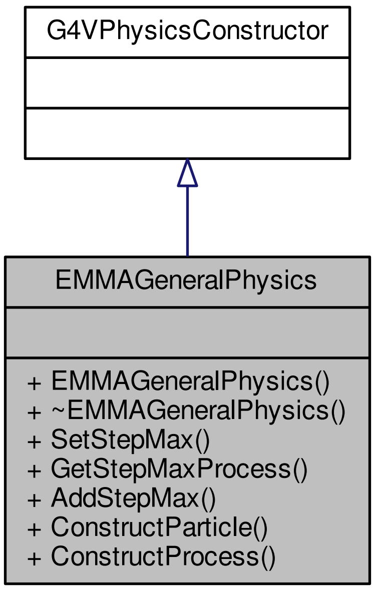
The documentation for this class was generated from the following file:

- EMMAEventActionMessenger.hh

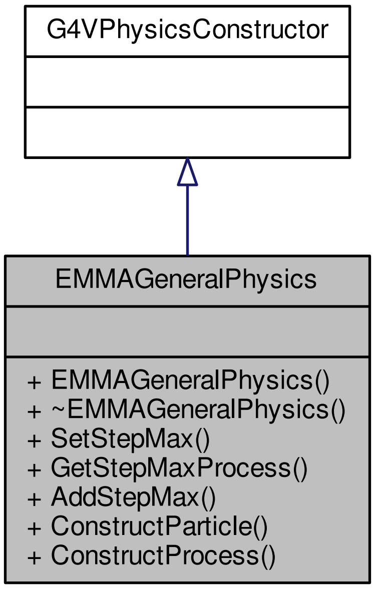
6.64 EMMAGeneralPhysics Class Reference

```
#include "EMMAGeneralPhysics.hh"
```

Inheritance diagram for EMMAGeneralPhysics:



Collaboration diagram for EMMAGeneralPhysics:



Public Member Functions

- EMMAGeneralPhysics (const G4String &name="general")
- virtual ∼EMMAGeneralPhysics ()
- void SetStepMax (G4double)
- F04StepMax ∗ GetStepMaxProcess ()
- void AddStepMax ()
- virtual void ConstructParticle ()
- virtual void ConstructProcess ()

6.64.1 Constructor & Destructor Documentation

6.64.1.1 EMMAGeneralPhysics::EMMAGeneralPhysics (const G4String & *name* = "general")

6.64.1.2 virtual EMMAGeneralPhysics::∼EMMAGeneralPhysics ( ) [virtual]

6.64.2 Member Function Documentation

6.64.2.1 void EMMAGeneralPhysics::AddStepMax ( )

6.64.2.2 virtual void EMMAGeneralPhysics::ConstructParticle ( ) [virtual]

6.64.2.3 virtual void EMMAGeneralPhysics::ConstructProcess ( ) [virtual]

6.64.2.4 F04StepMax* EMMAGeneralPhysics::GetStepMaxProcess ( )

6.64.2.5 void EMMAGeneralPhysics::SetStepMax (G4double)

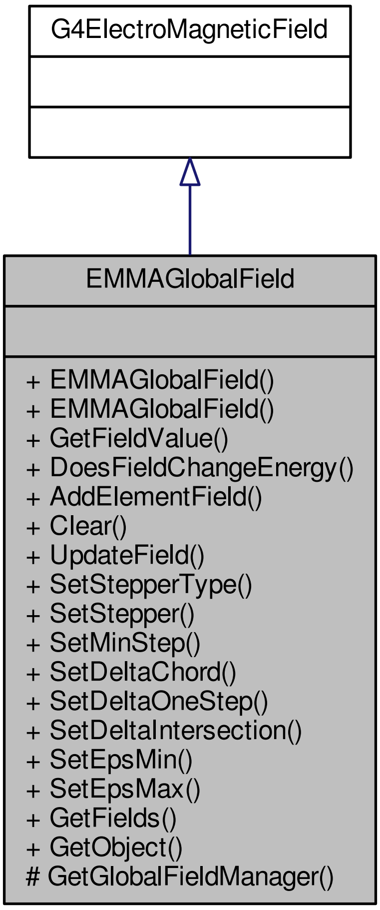
The documentation for this class was generated from the following file:

- EMMAGeneralPhysics.hh

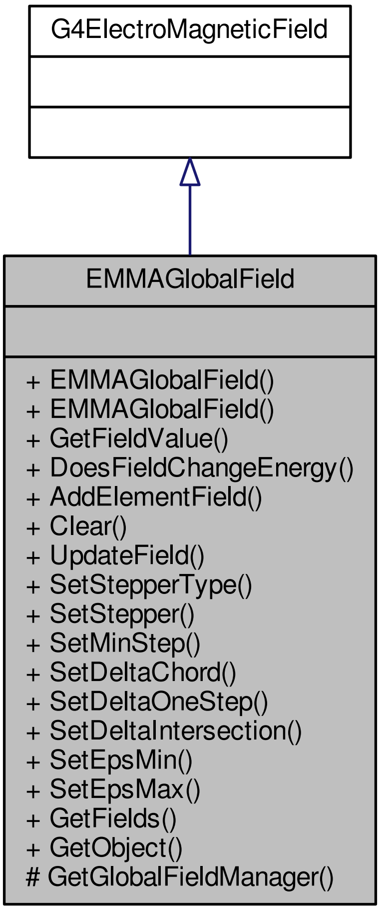
6.65 EMMAGlobalField Class Reference

```
#include "EMMAGlobalField.hh"
```

Inheritance diagram for EMMAGlobalField:



Collaboration diagram for EMMAGlobalField:



Public Member Functions

- EMMAGlobalField ()
- EMMAGlobalField (const EMMAGlobalField &)
- virtual void GetFieldValue (const G4double ∗point, G4double ∗field) const
- virtual G4bool DoesFieldChangeEnergy () const

    DoesFieldChangeEnergy() returns true.
- void AddElementField (EMMAElementField ∗f)
- void Clear ()
- void UpdateField ()

    updates all field tracking objects and Clear()
- void SetStepperType (G4int i)

    Set the Stepper types.
- void SetStepper ()

    Set the Stepper.
- void SetMinStep (G4double stp)

Set the minimum step length.
- void SetDeltaChord (G4double dcr)

  Set the delta chord length.
- void SetDeltaOneStep (G4double stp)

  Set the delta one step length.
- void SetDeltaIntersection (G4double its)

  Set the delta intersection length.
- void SetEpsMin (G4double eps)

  Set the minimum eps length.
- void SetEpsMax (G4double eps)

  Set the maximum eps length.
- FieldList ∗ GetFields ()

  Return the list of Element Fields.

Static Public Member Functions

- static EMMAGlobalField ∗ GetObject ()

Protected Member Functions

- G4FieldManager ∗ GetGlobalFieldManager ()

  Get the global field manager.

6.65.1 Constructor & Destructor Documentation

6.65.1.1 EMMAGlobalField::EMMAGlobalField ( )

6.65.1.2 EMMAGlobalField::EMMAGlobalField (const EMMAGlobalField &)

6.65.2 Member Function Documentation

6.65.2.1 void EMMAGlobalField::AddElementField (EMMAElementField ∗ *f*) [inline]

AddElementField() adds the ElementField object for a single element to the global field.

6.65.2.2 void EMMAGlobalField::Clear ( )

Clear() removes all ElementField-s from the global object, and destroys them. Used before the geometry is completely re-created.

6.65.2.3 virtual G4bool EMMAGlobalField::DoesFieldChangeEnergy ( ) const [inline], [virtual]

DoesFieldChangeEnergy() returns true.

6.65.2.4 FieldList∗ EMMAGlobalField::GetFields ( ) [inline]

Return the list of Element Fields.

6.65.2.5 virtual void EMMAGlobalField::GetFieldValue (const G4double ∗ *point*, G4double ∗ *field*) const [virtual]

GetFieldValue() returns the field value at a given point[]. field is really field[6]: Bx,By,Bz,Ex,Ey,Ez. point[] is in global coordinates: x,y,z,t.

6.65.2.6 G4FieldManager∗ EMMAGlobalField::GetGlobalFieldManager ( ) [protected]

Get the global field manager.

6.65.2.7 static EMMAGlobalField∗ EMMAGlobalField::GetObject ( ) [static]

GetObject() returns the single EMMAGlobalField object. It is constructed, if necessary.

6.65.2.8 void EMMAGlobalField::SetDeltaChord (G4double *dcr*) [inline]

Set the delta chord length.

6.65.2.9 void EMMAGlobalField::SetDeltaIntersection (G4double *its*) [inline]

Set the delta intersection length.

6.65.2.10 void EMMAGlobalField::SetDeltaOneStep (G4double *stp*) [inline]

Set the delta one step length.

6.65.2.11 void EMMAGlobalField::SetEpsMax (G4double *eps*) [inline]

Set the maximum eps length.

6.65.2.12 void EMMAGlobalField::SetEpsMin (G4double *eps*) [inline]

Set the minimum eps length.

6.65.2.13 void EMMAGlobalField::SetMinStep (G4double *stp*) [inline]

Set the minimum step length.

6.65.2.14 void EMMAGlobalField::SetStepper ( )

Set the Stepper.

6.65.2.15 void EMMAGlobalField::SetStepperType (G4int *i*) [inline]

Set the Stepper types.

6.65.2.16 void EMMAGlobalField::UpdateField ( )

updates all field tracking objects and Clear()

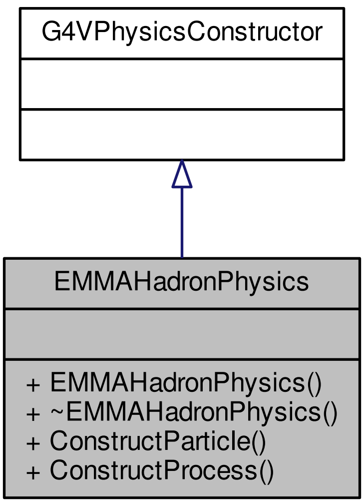
The documentation for this class was generated from the following file:

- EMMAGlobalField.hh

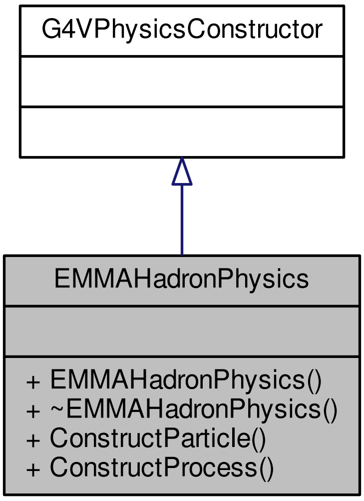
6.66 EMMAHadronPhysics Class Reference

`#include "EMMAHadronPhysics.hh"`

Inheritance diagram for EMMAHadronPhysics:



Collaboration diagram for EMMAHadronPhysics:



Public Member Functions

- EMMAHadronPhysics (const G4String &name="hadron")
- virtual ∼EMMAHadronPhysics ()
- virtual void ConstructParticle ()
- virtual void ConstructProcess ()

6.66.1 Constructor & Destructor Documentation

6.66.1.1 EMMAHadronPhysics::EMMAHadronPhysics (const G4String & *name* = "hadron")

6.66.1.2 virtual EMMAHadronPhysics::∼EMMAHadronPhysics ( ) [virtual]

6.66.2 Member Function Documentation

6.66.2.1 virtual void EMMAHadronPhysics::ConstructParticle ( ) [inline], [virtual]

6.66.2.2 virtual void EMMAHadronPhysics::ConstructProcess ( ) [virtual]

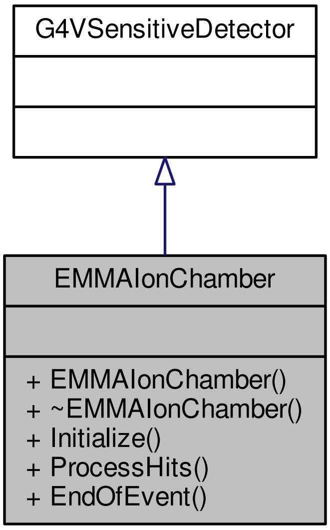
The documentation for this class was generated from the following file:

- EMMAHadronPhysics.hh

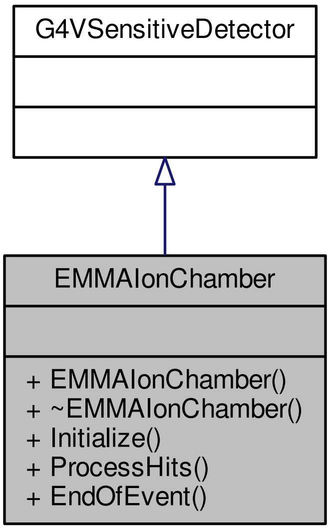
6.67 EMMAIonChamber Class Reference

```
#include "EMMAIonChamber.hh"
```

Inheritance diagram for EMMAIonChamber:



Collaboration diagram for EMMAIonChamber:



Public Member Functions

- EMMAIonChamber (const G4String &name, const G4String &hitsCollectionName, G4int nofCells)
- virtual ∼EMMAIonChamber ()
- virtual void Initialize (G4HCofThisEvent ∗hitCollection)
- virtual G4bool ProcessHits (G4Step ∗step, G4TouchableHistory ∗history)
- virtual void EndOfEvent (G4HCofThisEvent ∗hitCollection)

6.67.1 Detailed Description

Calorimeter sensitive detector class

In Initialize(), it creates one hit for each calorimeter layer and one more hit for accounting the total quantities in all layers.

The values are accounted in hits in ProcessHits() function which is called by Geant4 kernel at each step.

6.67.2 Constructor & Destructor Documentation

6.67.2.1 EMMAIonChamber::EMMAIonChamber (const G4String & *name,* const G4String & *hitsCollectionName,* G4int *nofCells*)

6.67.2.2 virtual EMMAIonChamber::∼EMMAIonChamber ( ) [virtual]

6.67.3 Member Function Documentation

6.67.3.1 virtual void EMMAIonChamber::EndOfEvent (G4HCofThisEvent ∗ *hitCollection*) [virtual]

6.67.3.2 virtual void EMMAIonChamber::Initialize (G4HCofThisEvent ∗ *hitCollection*) [virtual]

6.67.3.3 virtual G4bool EMMAIonChamber::ProcessHits (G4Step ∗ *step,* G4TouchableHistory ∗ *history*) [virtual]

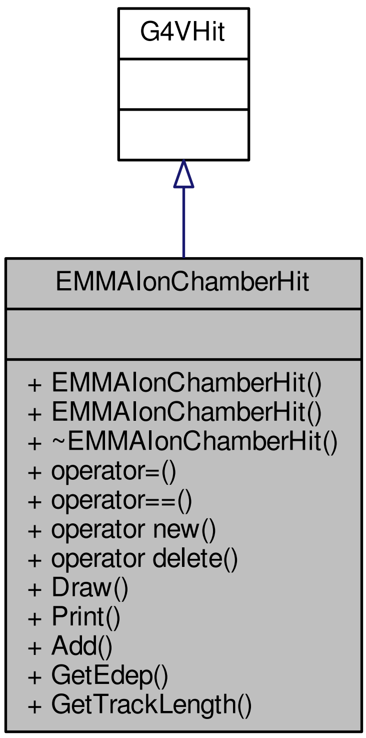
The documentation for this class was generated from the following file:

- EMMAIonChamber.hh

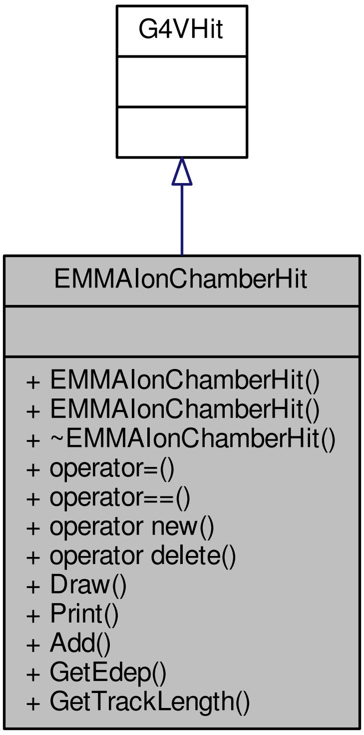
6.68 EMMAIonChamberHit Class Reference

```
#include "EMMAIonChamberHit.hh"
```

Inheritance diagram for EMMAIonChamberHit:



Collaboration diagram for EMMAIonChamberHit:



Public Member Functions

- EMMAIonChamberHit ()
- EMMAIonChamberHit (const EMMAIonChamberHit &)
- virtual ∼EMMAIonChamberHit ()
- const EMMAIonChamberHit & operator= (const EMMAIonChamberHit &)
- G4int operator== (const EMMAIonChamberHit &) const
- void ∗ operator new (size_t)
- void operator delete (void ∗)
- virtual void Draw ()
- virtual void Print ()
- void Add (G4double de, G4double dl)
- G4double GetEdep () const
- G4double GetTrackLength () const

6.68.1 Detailed Description

Calorimeter hit class

It defines data members to store the the energy deposit and track lengths of charged particles in a selected volume:

- fEdep, fTrackLength

6.68.2 Constructor & Destructor Documentation

6.68.2.1 EMMAIonChamberHit::EMMAIonChamberHit ( )

6.68.2.2 EMMAIonChamberHit::EMMAIonChamberHit (const EMMAIonChamberHit &)

6.68.2.3 virtual EMMAIonChamberHit::∼EMMAIonChamberHit ( ) [virtual]

6.68.3 Member Function Documentation

6.68.3.1 void EMMAIonChamberHit::Add (G4double *de*, G4double *dl*) [inline]

6.68.3.2 virtual void EMMAIonChamberHit::Draw ( ) [inline], [virtual]

6.68.3.3 G4double EMMAIonChamberHit::GetEdep ( ) const [inline]

6.68.3.4 G4double EMMAIonChamberHit::GetTrackLength ( ) const [inline]

6.68.3.5 void EMMAIonChamberHit::operator delete (void ∗ *hit*) [inline]

6.68.3.6 void ∗ EMMAIonChamberHit::operator new (size_t) [inline]

6.68.3.7 const EMMAIonChamberHit& EMMAIonChamberHit::operator= (const EMMAIonChamberHit &)

6.68.3.8 G4int EMMAIonChamberHit::operator== (const EMMAIonChamberHit &) const

6.68.3.9 virtual void EMMAIonChamberHit::Print ( ) [virtual]

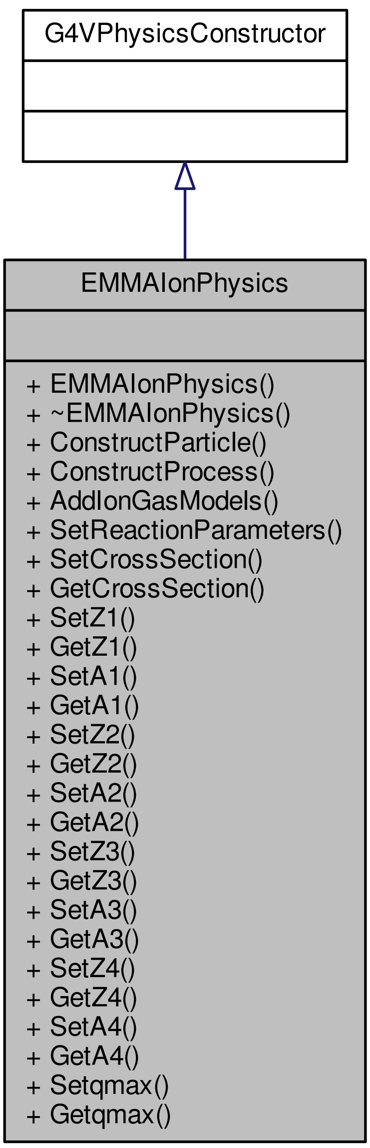
The documentation for this class was generated from the following file:

- EMMAIonChamberHit.hh

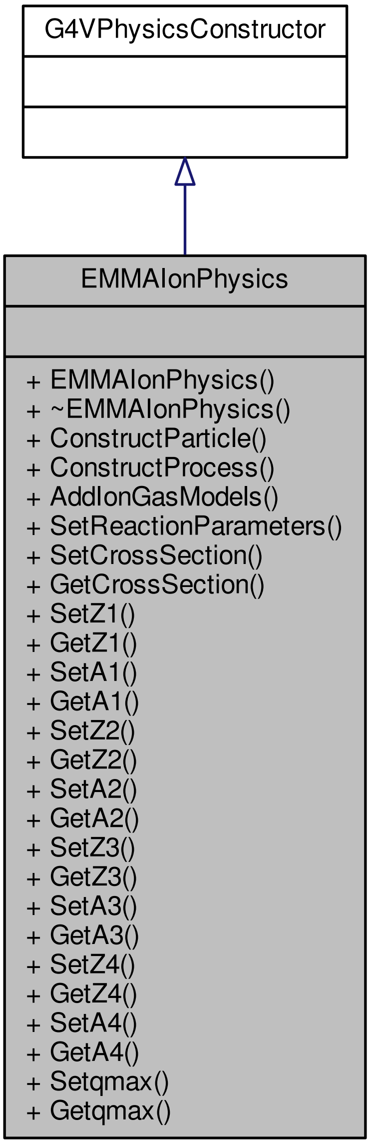
6.69 EMMAIonPhysics Class Reference

```
#include "EMMAIonPhysics.hh"
```

Inheritance diagram for EMMAIonPhysics:



Collaboration diagram for EMMAIonPhysics:



Public Member Functions

- EMMAIonPhysics (const G4String &name="ion")
- virtual ∼EMMAIonPhysics ()
- virtual void ConstructParticle ()
- virtual void ConstructProcess ()
- void AddIonGasModels ()
- void SetReactionParameters ()
- void SetCrossSection (G4double val)
- G4double GetCrossSection () const

- void SetZ1 (G4double val)
- G4double GetZ1 () const
- void SetA1 (G4double val)
- G4double GetA1 () const
- void SetZ2 (G4double val)
- G4double GetZ2 () const
- void SetA2 (G4double val)
- G4double GetA2 () const
- void SetZ3 (G4double val)
- G4double GetZ3 () const
- void SetA3 (G4double val)
- G4double GetA3 () const
- void SetZ4 (G4double val)
- G4double GetZ4 () const
- void SetA4 (G4double val)
- G4double GetA4 () const
- void Setqmax (G4double val)
- G4double Getqmax () const

6.69.1 Constructor & Destructor Documentation

6.69.1.1 EMMAIonPhysics::EMMAIonPhysics (const G4String & *name* = "ion")

6.69.1.2 virtual EMMAIonPhysics::∼EMMAIonPhysics ( ) [virtual]

6.69.2 Member Function Documentation

6.69.2.1 void EMMAIonPhysics::AddIonGasModels ( )

6.69.2.2 virtual void EMMAIonPhysics::ConstructParticle ( ) [inline], [virtual]

6.69.2.3 virtual void EMMAIonPhysics::ConstructProcess ( ) [virtual]

6.69.2.4 G4double EMMAIonPhysics::GetA1 ( ) const [inline]

6.69.2.5 G4double EMMAIonPhysics::GetA2 ( ) const [inline]

6.69.2.6 G4double EMMAIonPhysics::GetA3 ( ) const [inline]

6.69.2.7 G4double EMMAIonPhysics::GetA4 ( ) const [inline]

6.69.2.8 G4double EMMAIonPhysics::GetCrossSection ( ) const [inline]

6.69.2.9 G4double EMMAIonPhysics::Getqmax ( ) const [inline]

6.69.2.10 G4double EMMAIonPhysics::GetZ1 ( ) const [inline]

6.69.2.11 G4double EMMAIonPhysics::GetZ2 ( ) const [inline]

6.69.2.12 G4double EMMAIonPhysics::GetZ3 ( ) const [inline]

6.69.2.13 G4double EMMAIonPhysics::GetZ4 ( ) const [inline]

6.69.2.14 void EMMAIonPhysics::SetA1 (G4double *val*) [inline]

6.69.2.15 void EMMAIonPhysics::SetA2 (G4double *val*) [inline]

6.69.2.16 void EMMAIonPhysics::SetA3 (G4double *val*) [inline]

6.69.2.17 void EMMAIonPhysics::SetA4 (G4double *val*) [inline]

6.69.2.18 void EMMAIonPhysics::SetCrossSection (G4double *val*) [inline]

6.69.2.19 void EMMAIonPhysics::Setqmax (G4double *val*) [inline]

6.69.2.20 void EMMAIonPhysics::SetReactionParameters ( )

6.69.2.21 void EMMAIonPhysics::SetZ1 (G4double *val*) [inline]

6.69.2.22 void EMMAIonPhysics::SetZ2 (G4double *val*) [inline]

6.69.2.23 void EMMAIonPhysics::SetZ3 (G4double *val*) [inline]

6.69.2.24 void EMMAIonPhysics::SetZ4 (G4double *val*) [inline]

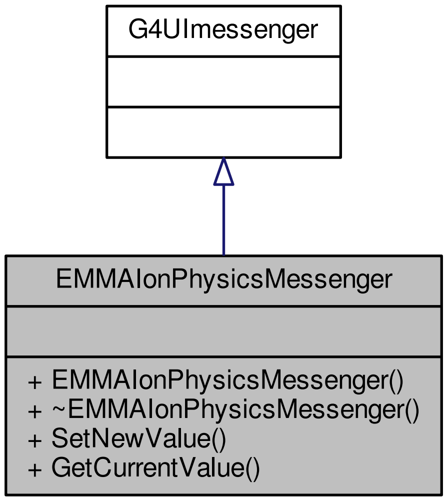
The documentation for this class was generated from the following file:

- EMMAIonPhysics.hh

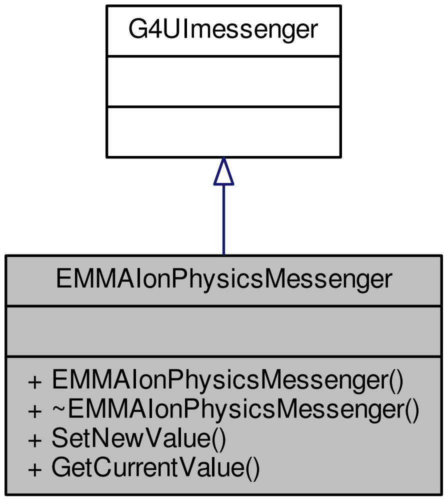
6.70 EMMAIonPhysicsMessenger Class Reference

```
#include "EMMAIonPhysicsMessenger.hh"
```

Inheritance diagram for EMMAIonPhysicsMessenger:



Collaboration diagram for EMMAIonPhysicsMessenger:



Public Member Functions

- EMMAIonPhysicsMessenger (EMMAIonPhysics ∗ionphys)
- ∼EMMAIonPhysicsMessenger ()
- void SetNewValue (G4UIcommand ∗command, G4String newValues)
- G4String GetCurrentValue (G4UIcommand ∗command)

6.70.1 Constructor & Destructor Documentation

6.70.1.1 EMMAIonPhysicsMessenger::EMMAIonPhysicsMessenger (EMMAIonPhysics ∗ *ionphys*)

6.70.1.2 EMMAIonPhysicsMessenger::∼EMMAIonPhysicsMessenger ( )

6.70.2 Member Function Documentation

6.70.2.1 G4String EMMAIonPhysicsMessenger::GetCurrentValue (G4UIcommand ∗ *command*)

6.70.2.2 void EMMAIonPhysicsMessenger::SetNewValue (G4UIcommand ∗ *command,* G4String *newValues*)

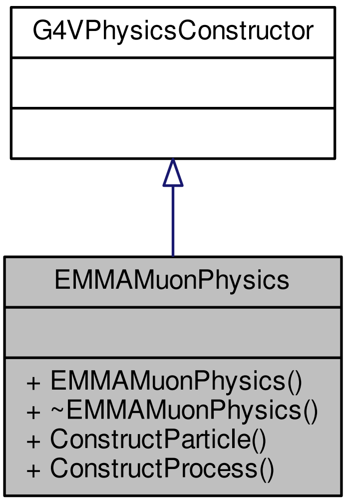
The documentation for this class was generated from the following file:

- EMMAIonPhysicsMessenger.hh

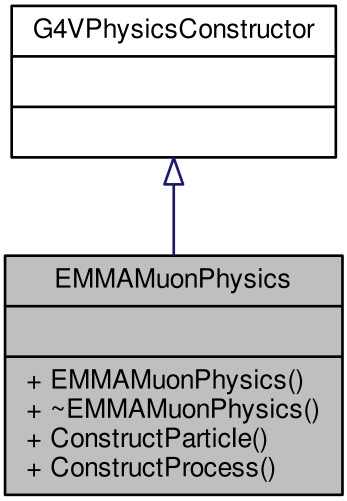
6.71 EMMAMuonPhysics Class Reference

`#include "EMMAMuonPhysics.hh"`

Inheritance diagram for EMMAMuonPhysics:



Collaboration diagram for EMMAMuonPhysics:



Public Member Functions

- EMMAMuonPhysics (const G4String &name="muon")
- virtual ∼EMMAMuonPhysics ()
- virtual void ConstructParticle ()
- virtual void ConstructProcess ()

6.71.1 Constructor & Destructor Documentation

6.71.1.1 EMMAMuonPhysics::EMMAMuonPhysics (const G4String & *name* = "muon")

6.71.1.2 virtual EMMAMuonPhysics::∼EMMAMuonPhysics ( ) [virtual]

6.71.2 Member Function Documentation

6.71.2.1 virtual void EMMAMuonPhysics::ConstructParticle ( ) [inline], [virtual]

6.71.2.2 virtual void EMMAMuonPhysics::ConstructProcess ( ) [virtual]

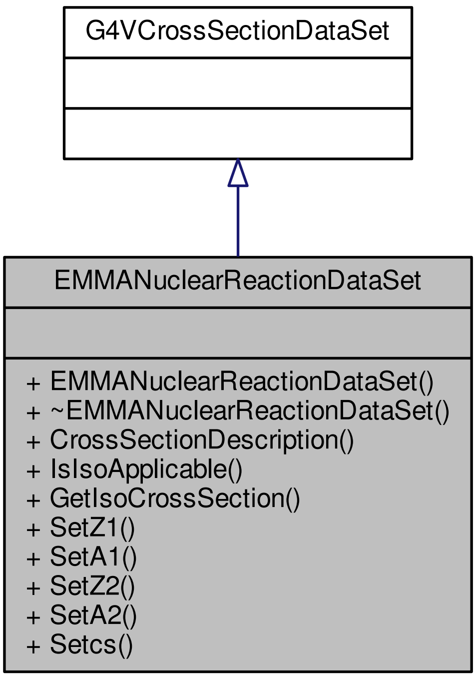
The documentation for this class was generated from the following file:

- EMMAMuonPhysics.hh

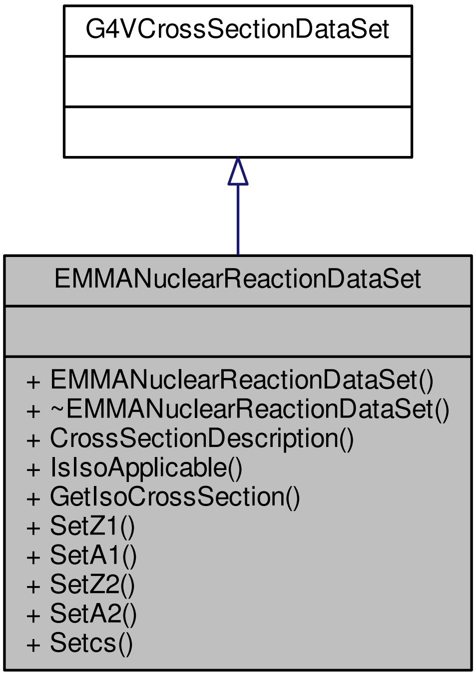
6.72 EMMANuclearReactionDataSet Class Reference

```
#include "EMMANuclearReactionDataSet.hh"
```

Inheritance diagram for EMMANuclearReactionDataSet:



Collaboration diagram for EMMANuclearReactionDataSet:



Public Member Functions

- EMMANuclearReactionDataSet (const G4String &name="EMMANuclearReactionDataSet", G4double Z1=0, G4double A1=0, G4double Z2=0, G4double A2=0, G4double cs=0)
- virtual ∼EMMANuclearReactionDataSet ()
- virtual void CrossSectionDescription (std::ostream &) const
- virtual G4bool IsIsoApplicable (const G4DynamicParticle ∗, G4int Z, G4int A, const G4Element ∗elm=0, const G4Material ∗mat=0)
- virtual G4double GetIsoCrossSection (const G4DynamicParticle ∗, G4int Z, G4int A, const G4Isotope ∗iso=0, const G4Element ∗elm=0, const G4Material ∗mat=0)
- void SetZ1 (G4double x)
- void SetA1 (G4double x)
- void SetZ2 (G4double x)
- void SetA2 (G4double x)
- void Setcs (G4double x)

6.72.1 Constructor & Destructor Documentation

6.72.1.1 EMMANuclearReactionDataSet::EMMANuclearReactionDataSet (const G4String & *name* = "EMMANuclearReactionDataSet", G4double *Z1* = 0, G4double *A1* = 0, G4double *Z2* = 0, G4double *A2* = 0, G4double *cs* = 0)

6.72.1.2 virtual EMMANuclearReactionDataSet::∼EMMANuclearReactionDataSet ( ) [virtual]

6.72.2 Member Function Documentation

6.72.2.1 virtual void EMMANuclearReactionDataSet::CrossSectionDescription (std::ostream &) const [virtual]

6.72.2.2 virtual G4double EMMANuclearReactionDataSet::GetIsoCrossSection (const G4DynamicParticle ∗ , G4int *Z*, G4int *A*, const G4Isotope ∗ *iso* = 0, const G4Element ∗ *elm* = 0, const G4Material ∗ *mat* = 0) [virtual]

6.72.2.3 virtual G4bool EMMANuclearReactionDataSet::IsIsoApplicable (const G4DynamicParticle ∗ , G4int *Z*, G4int *A*, const G4Element ∗ *elm* = 0, const G4Material ∗ *mat* = 0) [virtual]

6.72.2.4 void EMMANuclearReactionDataSet::SetA1 (G4double *x*) [inline]

6.72.2.5 void EMMANuclearReactionDataSet::SetA2 (G4double *x*) [inline]

6.72.2.6 void EMMANuclearReactionDataSet::Setcs (G4double *x*) [inline]

6.72.2.7 void EMMANuclearReactionDataSet::SetZ1 (G4double *x*) [inline]

6.72.2.8 void EMMANuclearReactionDataSet::SetZ2 (G4double *x*) [inline]

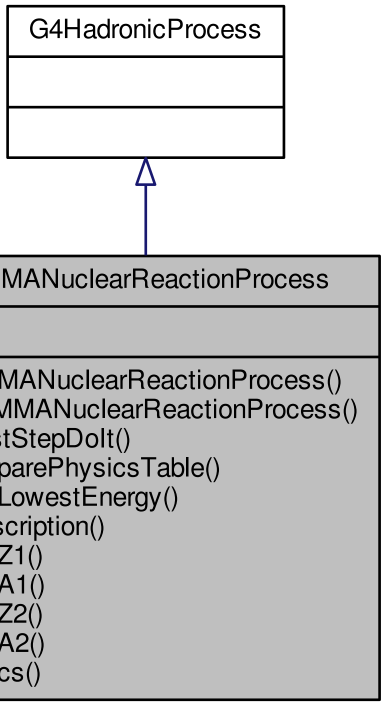
The documentation for this class was generated from the following file:

- EMMANuclearReactionDataSet.hh

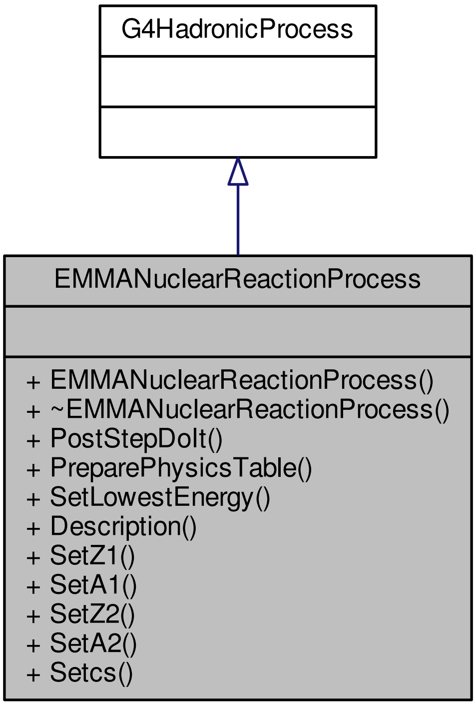
6.73 EMMANuclearReactionProcess Class Reference

```
#include "EMMANuclearReactionProcess.hh"
```

Inheritance diagram for EMMANuclearReactionProcess:



Collaboration diagram for EMMANuclearReactionProcess:



Public Member Functions

- EMMANuclearReactionProcess (const G4String &procName="EMMANuclearReactionProcess", G4double Z1=0, G4double A1=0, G4double Z2=0, G4double A2=0, G4double cs=0)
- virtual ∼EMMANuclearReactionProcess ()
- virtual G4VParticleChange ∗ PostStepDoIt (const G4Track &aTrack, const G4Step &aStep)
- virtual void PreparePhysicsTable (const G4ParticleDefinition &)
- virtual void SetLowestEnergy (G4double)
- virtual void Description () const
- void SetZ1 (G4double x)
- void SetA1 (G4double x)
- void SetZ2 (G4double x)
- void SetA2 (G4double x)
- void Setcs (G4double x)

6.73.1 Constructor & Destructor Documentation

6.73.1.1 EMMANuclearReactionProcess::EMMANuclearReactionProcess (const G4String & *procName* = "EMMANuclearReactionProcess", G4double *Z1* = 0, G4double *A1* = 0, G4double *Z2* = 0, G4double *A2* = 0, G4double *cs* = 0)

6.73.1.2 virtual EMMANuclearReactionProcess::∼EMMANuclearReactionProcess ( ) [virtual]

6.73.2 Member Function Documentation

6.73.2.1 virtual void EMMANuclearReactionProcess::Description ( ) const [virtual]

6.73.2.2 virtual G4VParticleChange∗ EMMANuclearReactionProcess::PostStepDoIt (const G4Track & *aTrack*, const G4Step & *aStep*) [virtual]

6.73.2.3 virtual void EMMANuclearReactionProcess::PreparePhysicsTable (const G4ParticleDefinition &) [virtual]

6.73.2.4 void EMMANuclearReactionProcess::SetA1 (G4double *x*) [inline]

6.73.2.5 void EMMANuclearReactionProcess::SetA2 (G4double *x*) [inline]

6.73.2.6 void EMMANuclearReactionProcess::Setcs (G4double *x*) [inline]

6.73.2.7 virtual void EMMANuclearReactionProcess::SetLowestEnergy (G4double) [virtual]

6.73.2.8 void EMMANuclearReactionProcess::SetZ1 (G4double *x*) [inline]

6.73.2.9 void EMMANuclearReactionProcess::SetZ2 (G4double *x*) [inline]

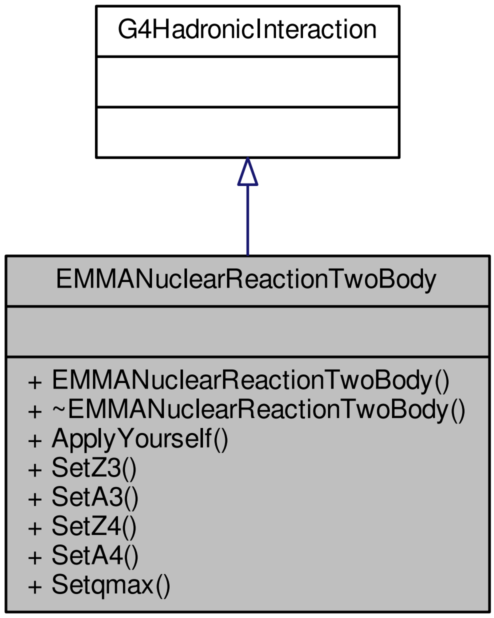
The documentation for this class was generated from the following file:

- EMMANuclearReactionProcess.hh

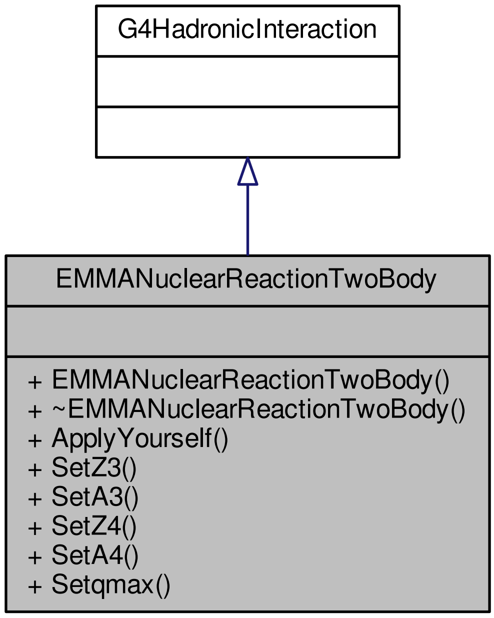
6.74 EMMANuclearReactionTwoBody Class Reference

```
#include "EMMANuclearReactionTwoBody.hh"
```

Inheritance diagram for EMMANuclearReactionTwoBody:



Collaboration diagram for EMMANuclearReactionTwoBody:



Public Member Functions

- EMMANuclearReactionTwoBody (const G4String &name="EMMANuclearReactionTwoBody", G4double Z3=0, G4double A3=0, G4double Z4=0, G4double A4=0, G4double qmax=180.)
- ∼EMMANuclearReactionTwoBody ()
- G4HadFinalState ∗ ApplyYourself (const G4HadProjectile &aTrack, G4Nucleus &targetNucleus)
- void SetZ3 (G4double x)
- void SetA3 (G4double x)
- void SetZ4 (G4double x)
- void SetA4 (G4double x)
- void Setqmax (G4double x)

6.74.1 Constructor & Destructor Documentation

6.74.1.1 EMMANuclearReactionTwoBody::EMMANuclearReactionTwoBody (const G4String & *name* = "EMMANuclearReactionTwoBody", G4double *Z3* = 0, G4double *A3* = 0, G4double *Z4* = 0, G4double *A4* = 0, G4double *qmax* = 180.)

6.74.1.2 EMMANuclearReactionTwoBody::∼EMMANuclearReactionTwoBody ( ) [inline]

6.74.2 Member Function Documentation

6.74.2.1 G4HadFinalState∗ EMMANuclearReactionTwoBody::ApplyYourself (const G4HadProjectile & *aTrack*, G4Nucleus & *targetNucleus*)

6.74.2.2 void EMMANuclearReactionTwoBody::SetA3 (G4double *x*) [inline]

6.74.2.3 void EMMANuclearReactionTwoBody::SetA4 (G4double *x*) [inline]

6.74.2.4 void EMMANuclearReactionTwoBody::Setqmax (G4double *x*) [inline]

6.74.2.5 void EMMANuclearReactionTwoBody::SetZ3 (G4double *x*) [inline]

6.74.2.6 void EMMANuclearReactionTwoBody::SetZ4 (G4double *x*) [inline]

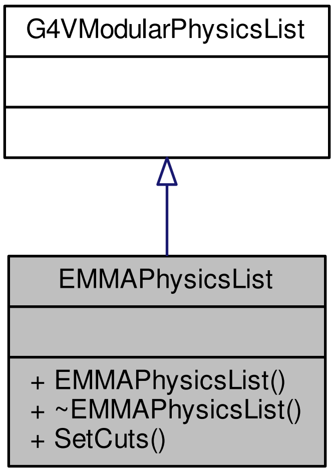
The documentation for this class was generated from the following file:

- EMMANuclearReactionTwoBody.hh

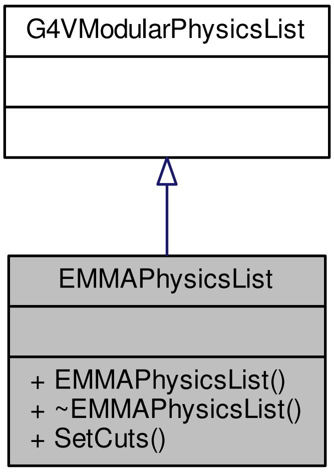
6.75 EMMAPhysicsList Class Reference

```
#include "EMMAPhysicsList.hh"
```

Inheritance diagram for EMMAPhysicsList:



Collaboration diagram for EMMAPhysicsList:



Public Member Functions

- EMMAPhysicsList ()
- virtual ∼EMMAPhysicsList ()
- virtual void SetCuts ()

6.75.1 Constructor & Destructor Documentation

6.75.1.1 EMMAPhysicsList::EMMAPhysicsList ( )

6.75.1.2 virtual EMMAPhysicsList::∼EMMAPhysicsList ( ) [virtual]

6.75.2 Member Function Documentation

6.75.2.1 virtual void EMMAPhysicsList::SetCuts ( ) [virtual]

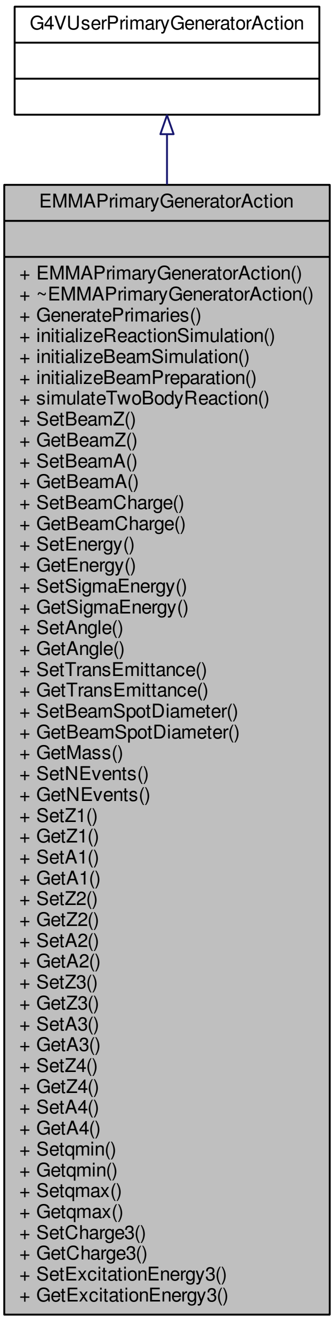
The documentation for this class was generated from the following file:

- EMMAPhysicsList.hh

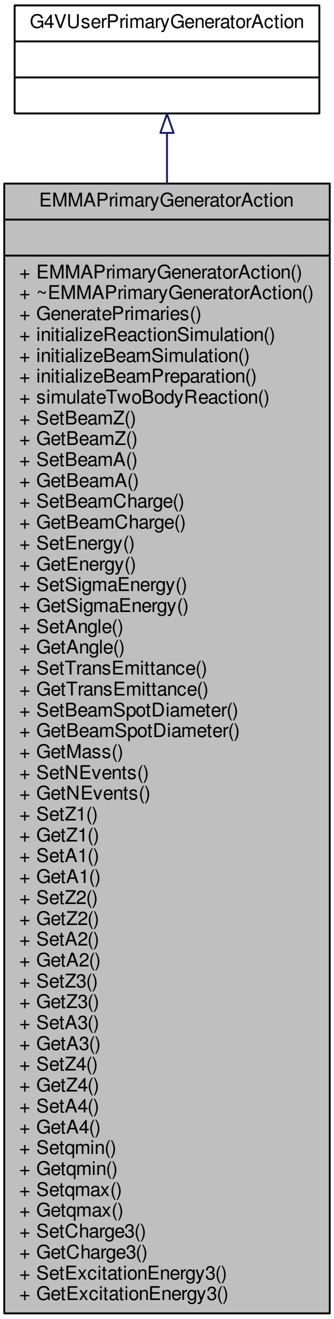
6.76 EMMAPrimaryGeneratorAction Class Reference

```
#include "EMMAPrimaryGeneratorAction.hh"
```

Inheritance diagram for EMMAPrimaryGeneratorAction:



Collaboration diagram for EMMAPrimaryGeneratorAction:



Public Member Functions

- EMMAPrimaryGeneratorAction ()
- virtual ~EMMAPrimaryGeneratorAction ()
- virtual void GeneratePrimaries (G4Event ∗)
- void initializeReactionSimulation ()
- void initializeBeamSimulation ()

- void initializeBeamPreparation ()
- void simulateTwoBodyReaction (G4double &Ebeam, G4ThreeVector &dir, G4double &Eejc, G4ThreeVector &dir2)
- void SetBeamZ (G4double val)
- G4double GetBeamZ () const
- void SetBeamA (G4double val)
- G4double GetBeamA () const
- void SetBeamCharge (G4double val)
- G4double GetBeamCharge () const
- void SetEnergy (G4double val)
- G4double GetEnergy () const
- void SetSigmaEnergy (G4double val)
- G4double GetSigmaEnergy () const
- void SetAngle (G4double val)
- G4double GetAngle () const
- void SetTransEmittance (G4double val)
- G4double GetTransEmittance () const
- void SetBeamSpotDiameter (G4double val)
- G4double GetBeamSpotDiameter () const
- G4double GetMass () const
- void SetNEvents (G4int val)
- G4int GetNEvents () const
- void SetZ1 (G4double val)
- G4double GetZ1 () const
- void SetA1 (G4double val)
- G4double GetA1 () const
- void SetZ2 (G4double val)
- G4double GetZ2 () const
- void SetA2 (G4double val)
- G4double GetA2 () const
- void SetZ3 (G4double val)
- G4double GetZ3 () const
- void SetA3 (G4double val)
- G4double GetA3 () const
- void SetZ4 (G4double val)
- G4double GetZ4 () const
- void SetA4 (G4double val)
- G4double GetA4 () const
- void Setqmin (G4double val)
- G4double Getqmin () const
- void Setqmax (G4double val)
- G4double Getqmax () const
- void SetCharge3 (G4double val)
- G4double GetCharge3 () const
- void SetExcitationEnergy3 (G4double val)
- G4double GetExcitationEnergy3 () const

6.76.1 Constructor & Destructor Documentation

6.76.1.1 EMMAPrimaryGeneratorAction::EMMAPrimaryGeneratorAction ( )

6.76.1.2 virtual EMMAPrimaryGeneratorAction::~EMMAPrimaryGeneratorAction ( ) [virtual]

6.76.2 Member Function Documentation

6.76.2.1 virtual void EMMAPrimaryGeneratorAction::GeneratePrimaries (G4Event *) [virtual]

6.76.2.2 G4double EMMAPrimaryGeneratorAction::GetA1 ( ) const [inline]

6.76.2.3 G4double EMMAPrimaryGeneratorAction::GetA2 ( ) const [inline]

6.76.2.4 G4double EMMAPrimaryGeneratorAction::GetA3 ( ) const [inline]

6.76.2.5 G4double EMMAPrimaryGeneratorAction::GetA4 ( ) const [inline]

6.76.2.6 G4double EMMAPrimaryGeneratorAction::GetAngle ( ) const [inline]

6.76.2.7 G4double EMMAPrimaryGeneratorAction::GetBeamA ( ) const [inline]

6.76.2.8 G4double EMMAPrimaryGeneratorAction::GetBeamCharge ( ) const [inline]

6.76.2.9 G4double EMMAPrimaryGeneratorAction::GetBeamSpotDiameter ( ) const [inline]

6.76.2.10 G4double EMMAPrimaryGeneratorAction::GetBeamZ ( ) const [inline]

6.76.2.11 G4double EMMAPrimaryGeneratorAction::GetCharge3 ( ) const [inline]

6.76.2.12 G4double EMMAPrimaryGeneratorAction::GetEnergy ( ) const [inline]

6.76.2.13 G4double EMMAPrimaryGeneratorAction::GetExcitationEnergy3 ( ) const [inline]

6.76.2.14 G4double EMMAPrimaryGeneratorAction::GetMass ( ) const [inline]

6.76.2.15 G4int EMMAPrimaryGeneratorAction::GetNEvents ( ) const [inline]

6.76.2.16 G4double EMMAPrimaryGeneratorAction::Getqmax ( ) const [inline]

6.76.2.17 G4double EMMAPrimaryGeneratorAction::Getqmin ( ) const [inline]

6.76.2.18 G4double EMMAPrimaryGeneratorAction::GetSigmaEnergy ( ) const [inline]

6.76.2.19 G4double EMMAPrimaryGeneratorAction::GetTransEmittance ( ) const [inline]

6.76.2.20 G4double EMMAPrimaryGeneratorAction::GetZ1 ( ) const [inline]

6.76.2.21 G4double EMMAPrimaryGeneratorAction::GetZ2 ( ) const [inline]

6.76.2.22 G4double EMMAPrimaryGeneratorAction::GetZ3 ( ) const [inline]

6.76.2.23 G4double EMMAPrimaryGeneratorAction::GetZ4 ( ) const [inline]

6.76.2.24 void EMMAPrimaryGeneratorAction::initializeBeamPreparation ( )

6.76.2.25 void EMMAPrimaryGeneratorAction::initializeBeamSimulation ( )

6.76.2.26 void EMMAPrimaryGeneratorAction::initializeReactionSimulation ( )

6.76.2.27 void EMMAPrimaryGeneratorAction::SetA1 (G4double *val*) [inline]

6.76.2.28 void EMMAPrimaryGeneratorAction::SetA2 (G4double *val*) [inline]

6.76.2.29 void EMMAPrimaryGeneratorAction::SetA3 (G4double *val*) [inline]

6.76.2.30 void EMMAPrimaryGeneratorAction::SetA4 (G4double *val*) [inline]

6.76.2.31 void EMMAPrimaryGeneratorAction::SetAngle (G4double *val*) [inline]

6.76.2.32 void EMMAPrimaryGeneratorAction::SetBeamA (G4double *val*) [inline]

6.76.2.33 void EMMAPrimaryGeneratorAction::SetBeamCharge (G4double *val*) [inline]

6.76.2.34 void EMMAPrimaryGeneratorAction::SetBeamSpotDiameter (G4double *val*) [inline]

6.76.2.35 void EMMAPrimaryGeneratorAction::SetBeamZ (G4double *val*) [inline]

6.76.2.36 void EMMAPrimaryGeneratorAction::SetCharge3 (G4double *val*) [inline]

6.76.2.37 void EMMAPrimaryGeneratorAction::SetEnergy (G4double *val*) [inline]

6.76.2.38 void EMMAPrimaryGeneratorAction::SetExcitationEnergy3 (G4double *val*) [inline]

6.76.2.39 void EMMAPrimaryGeneratorAction::SetNEvents (G4int *val*) [inline]

6.76.2.40 void EMMAPrimaryGeneratorAction::Setqmax (G4double *val*) [inline]

6.76.2.41 void EMMAPrimaryGeneratorAction::Setqmin (G4double *val*) [inline]

6.76.2.42 void EMMAPrimaryGeneratorAction::SetSigmaEnergy (G4double *val*) [inline]

6.76.2.43 void EMMAPrimaryGeneratorAction::SetTransEmittance (G4double *val*) [inline]

6.76.2.44 void EMMAPrimaryGeneratorAction::SetZ1 (G4double *val*) [inline]

6.76.2.45 void EMMAPrimaryGeneratorAction::SetZ2 (G4double *val*) [inline]

6.76.2.46 void EMMAPrimaryGeneratorAction::SetZ3 (G4double *val*) [inline]

6.76.2.47 void EMMAPrimaryGeneratorAction::SetZ4 (G4double *val*) [inline]

6.76.2.48 void EMMAPrimaryGeneratorAction::simulateTwoBodyReaction (G4double & *Ebeam,* G4ThreeVector & *dir,* G4double & *Eejc,* G4ThreeVector & *dir2*)

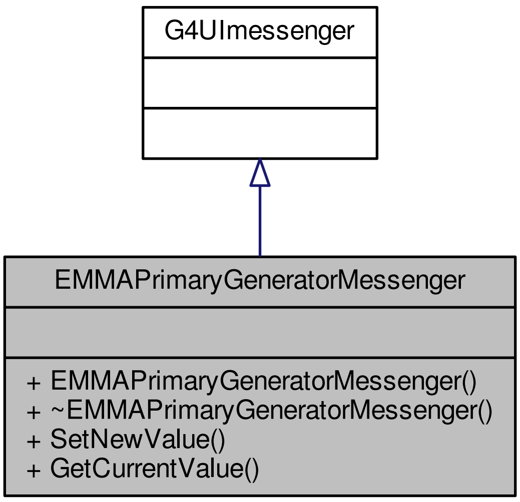
The documentation for this class was generated from the following file:

- EMMAPrimaryGeneratorAction.hh

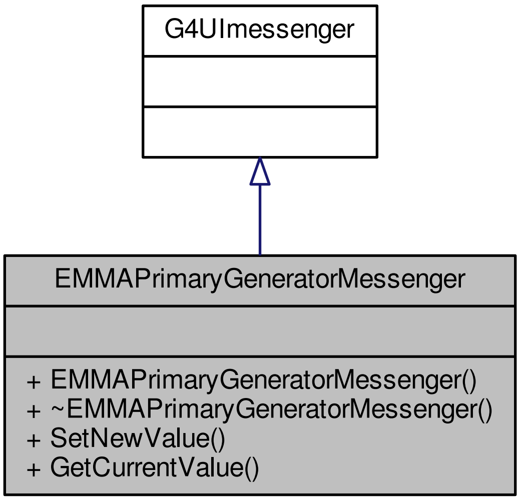
6.77 EMMAPrimaryGeneratorMessenger Class Reference

```
#include "EMMAPrimaryGeneratorMessenger.hh"
```

Inheritance diagram for EMMAPrimaryGeneratorMessenger:



Collaboration diagram for EMMAPrimaryGeneratorMessenger:



Public Member Functions

- EMMAPrimaryGeneratorMessenger (EMMAPrimaryGeneratorAction ∗mpga)
- ∼EMMAPrimaryGeneratorMessenger ()
- void SetNewValue (G4UIcommand ∗command, G4String newValues)
- G4String GetCurrentValue (G4UIcommand ∗command)

6.77.1 Constructor & Destructor Documentation

6.77.1.1 EMMAPrimaryGeneratorMessenger::EMMAPrimaryGeneratorMessenger (EMMAPrimaryGeneratorAction ∗ *mpga*)

6.77.1.2 EMMAPrimaryGeneratorMessenger::∼EMMAPrimaryGeneratorMessenger ( )

6.77.2 Member Function Documentation

6.77.2.1 G4String EMMAPrimaryGeneratorMessenger::GetCurrentValue (G4UIcommand ∗ *command*)

6.77.2.2 void EMMAPrimaryGeneratorMessenger::SetNewValue (G4UIcommand ∗ *command,* G4String *newValues*)

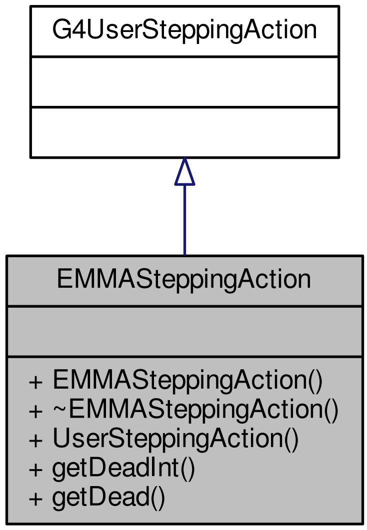
The documentation for this class was generated from the following file:

- EMMAPrimaryGeneratorMessenger.hh

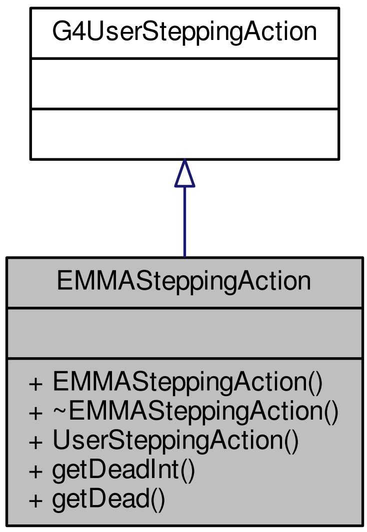
6.78 EMMASteppingAction Class Reference

```
#include "EMMASteppingAction.hh"
```

Inheritance diagram for EMMASteppingAction:



Collaboration diagram for EMMASteppingAction:



Public Member Functions

- EMMASteppingAction ()
- virtual ∼EMMASteppingAction ()
- virtual void UserSteppingAction (const G4Step ∗theStep)
- G4int getDeadInt ()
- G4bool getDead ()

6.78.1 Constructor & Destructor Documentation

6.78.1.1 EMMASteppingAction::EMMASteppingAction ( )

6.78.1.2 virtual EMMASteppingAction::∼EMMASteppingAction ( ) [virtual]

6.78.2 Member Function Documentation

6.78.2.1 G4bool EMMASteppingAction::getDead ( ) [inline]

6.78.2.2 G4int EMMASteppingAction::getDeadInt ( ) [inline]

6.78.2.3 virtual void EMMASteppingAction::UserSteppingAction (const G4Step ∗ *theStep*) [virtual]

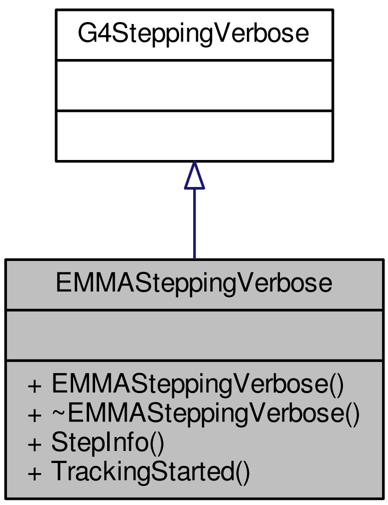
The documentation for this class was generated from the following file:

- EMMASteppingAction.hh

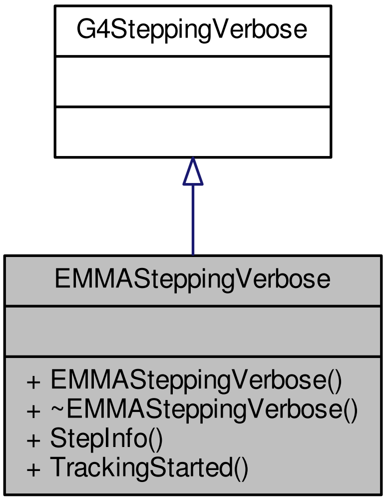
6.79 EMMASteppingVerbose Class Reference

`#include "EMMASteppingVerbose.hh"`

Inheritance diagram for EMMASteppingVerbose:



Collaboration diagram for EMMASteppingVerbose:



Public Member Functions

- EMMASteppingVerbose ()
- ∼EMMASteppingVerbose ()
- void StepInfo ()
- void TrackingStarted ()

6.79.1 Constructor & Destructor Documentation

6.79.1.1 EMMASteppingVerbose::EMMASteppingVerbose ( )

6.79.1.2 EMMASteppingVerbose::∼EMMASteppingVerbose ( )

6.79.2 Member Function Documentation

6.79.2.1 void EMMASteppingVerbose::StepInfo ( )

6.79.2.2 void EMMASteppingVerbose::TrackingStarted ( )

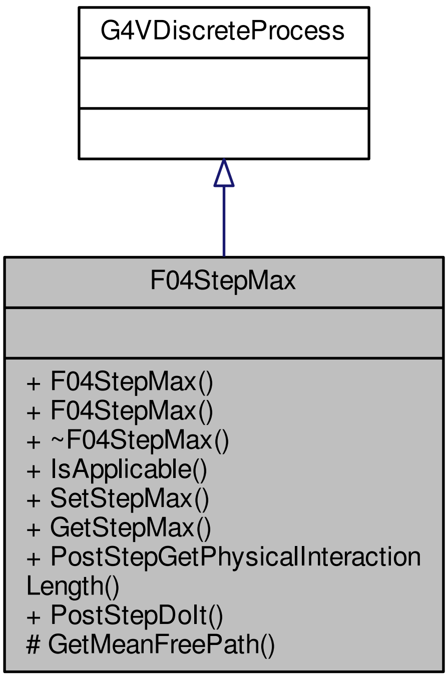
The documentation for this class was generated from the following file:

- EMMASteppingVerbose.hh

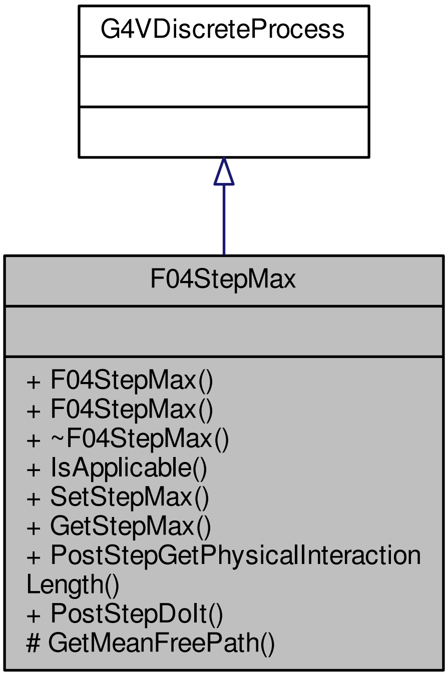
6.80 F04StepMax Class Reference

```
#include "F04StepMax.hh"
```

Inheritance diagram for F04StepMax:



Collaboration diagram for F04StepMax:



Public Member Functions

- F04StepMax (const G4String &processName="UserStepMax")
- F04StepMax (F04StepMax &)
- ∼F04StepMax ()
- G4bool IsApplicable (const G4ParticleDefinition &)
- void SetStepMax (G4double)
- G4double GetStepMax ()
- G4double PostStepGetPhysicalInteractionLength (const G4Track &track, G4double previousStepSize, G4↩ ForceCondition ∗condition)
- G4VParticleChange ∗ PostStepDoIt (const G4Track &, const G4Step &)

Protected Member Functions

- G4double GetMeanFreePath (const G4Track &, G4double, G4ForceCondition ∗)

6.80.1 Constructor & Destructor Documentation

6.80.1.1 F04StepMax::F04StepMax (const G4String & *processName* = "UserStepMax")

6.80.1.2 F04StepMax::F04StepMax (F04StepMax &)

6.80.1.3 F04StepMax::∼F04StepMax ( )

6.80.2 Member Function Documentation

6.80.2.1 G4double F04StepMax::GetMeanFreePath (const G4Track & , G4double , G4ForceCondition ∗) [protected]

6.80.2.2 G4double F04StepMax::GetStepMax ( ) [inline]

6.80.2.3 G4bool F04StepMax::IsApplicable (const G4ParticleDefinition &)

6.80.2.4 G4VParticleChange∗ F04StepMax::PostStepDoIt (const G4Track & , const G4Step &)

6.80.2.5 G4double F04StepMax::PostStepGetPhysicalInteractionLength (const G4Track & *track,* G4double *previousStepSize,* G4ForceCondition ∗ *condition*)

6.80.2.6 void F04StepMax::SetStepMax (G4double)

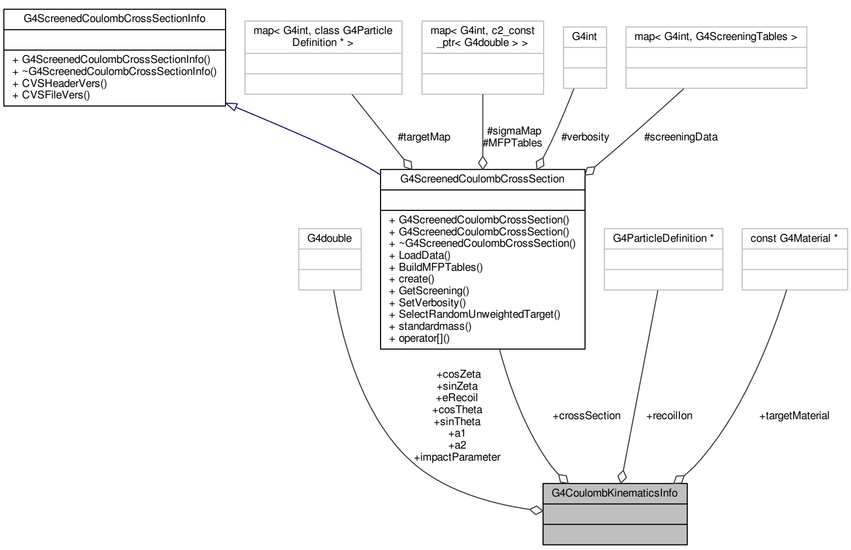
The documentation for this class was generated from the following file:

- F04StepMax.hh

6.81 G4CoulombKinematicsInfo Struct Reference

`#include "G4ScreenedNuclearRecoil.hh"`

Collaboration diagram for G4CoulombKinematicsInfo:



Public Attributes

- G4double impactParameter
- G4ScreenedCoulombCrossSection ∗ crossSection
- G4double a1
- G4double a2
- G4double sinTheta
- G4double cosTheta
- G4double sinZeta
- G4double cosZeta
- G4double eRecoil
- G4ParticleDefinition ∗ recoilIon
- const G4Material ∗ targetMaterial

6.81.1 Member Data Documentation

6.81.1.1 G4double G4CoulombKinematicsInfo::a1

6.81.1.2 G4double G4CoulombKinematicsInfo::a2

6.81.1.3 G4double G4CoulombKinematicsInfo::cosTheta

6.81.1.4 G4double G4CoulombKinematicsInfo::cosZeta

6.81.1.5 G4ScreenedCoulombCrossSection∗ G4CoulombKinematicsInfo::crossSection

6.81.1.6 G4double G4CoulombKinematicsInfo::eRecoil

6.81.1.7 G4double G4CoulombKinematicsInfo::impactParameter

6.81.1.8 G4ParticleDefinition∗ G4CoulombKinematicsInfo::recoilIon

6.81.1.9 G4double G4CoulombKinematicsInfo::sinTheta

6.81.1.10 G4double G4CoulombKinematicsInfo::sinZeta

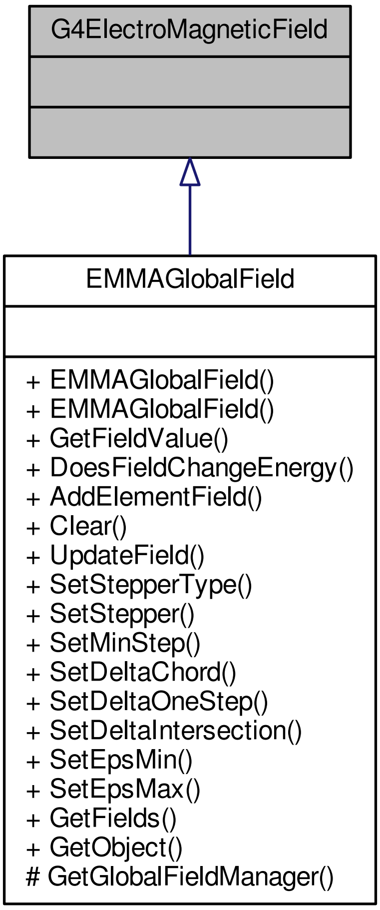
6.81.1.11 const G4Material∗ G4CoulombKinematicsInfo::targetMaterial

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

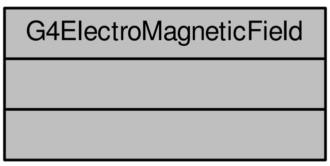
- G4ScreenedNuclearRecoil.hh

6.82 G4ElectroMagneticField Class Reference

Inheritance diagram for G4ElectroMagneticField:



Collaboration diagram for G4ElectroMagneticField:

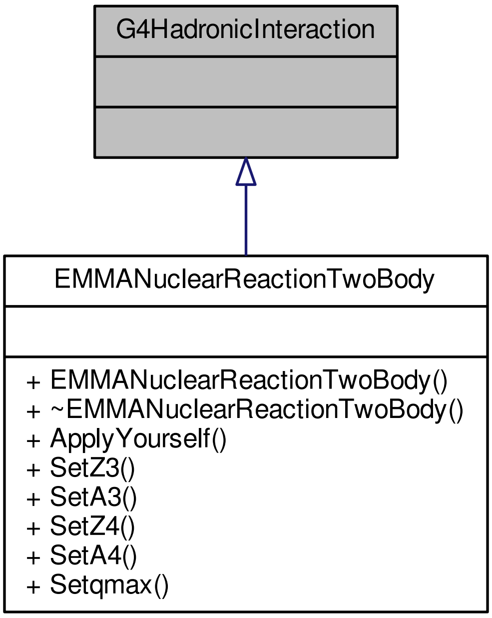


The documentation for this class was generated from the following file:

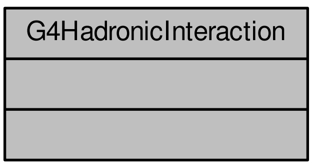
- EMMAGlobalField.hh

6.83 G4HadronicInteraction Class Reference

Inheritance diagram for G4HadronicInteraction:



Collaboration diagram for G4HadronicInteraction:

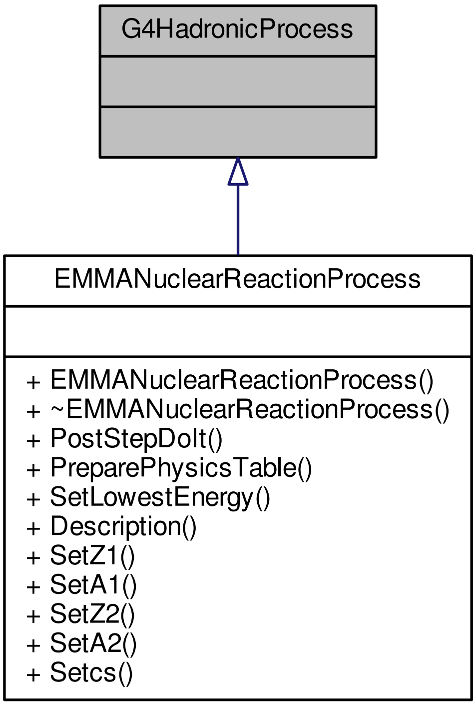


The documentation for this class was generated from the following file:

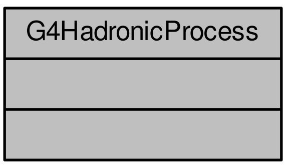
- EMMANuclearReactionTwoBody.hh

6.84 G4HadronicProcess Class Reference

Inheritance diagram for G4HadronicProcess:



Collaboration diagram for G4HadronicProcess:



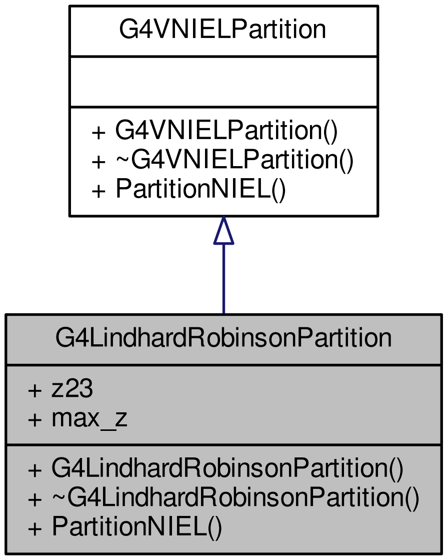
The documentation for this class was generated from the following file:

- EMMANuclearReactionProcess.hh

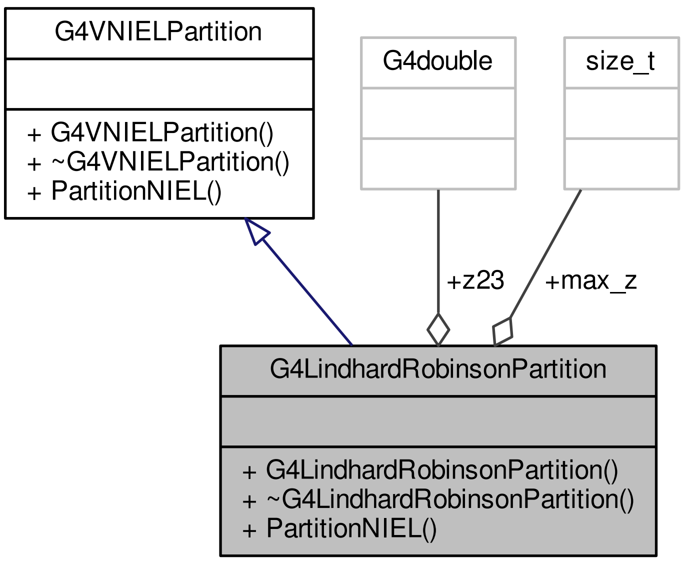
6.85 G4LindhardRobinsonPartition Class Reference

`#include "G4LindhardPartition.hh"`

Inheritance diagram for G4LindhardRobinsonPartition:



Collaboration diagram for G4LindhardRobinsonPartition:



Public Member Functions

- G4LindhardRobinsonPartition ()
- virtual ∼G4LindhardRobinsonPartition ()
- virtual G4double PartitionNIEL (G4int z1, G4double a1, const G4Material ∗material, G4double energy) const

Public Attributes

- G4double z23 [120]
- size_t max_z

6.85.1 Constructor & Destructor Documentation

6.85.1.1 G4LindhardRobinsonPartition::G4LindhardRobinsonPartition ( )

6.85.1.2 virtual G4LindhardRobinsonPartition::∼G4LindhardRobinsonPartition ( ) [inline], [virtual]

6.85.2 Member Function Documentation

6.85.2.1 virtual G4double G4LindhardRobinsonPartition::PartitionNIEL (G4int *z1*, G4double *a1*, const G4Material ∗ *material*, G4double *energy*) const [virtual]

Implements G4VNIELPartition.

6.85.3 Member Data Documentation

6.85.3.1 size_t G4LindhardRobinsonPartition::max_z

6.85.3.2 G4double G4LindhardRobinsonPartition::z23[120]

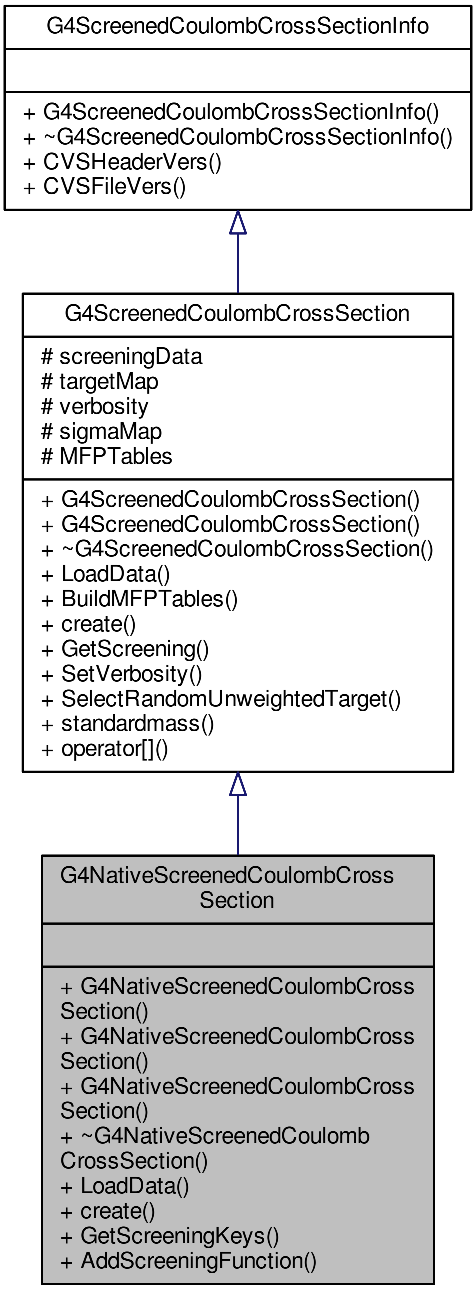
The documentation for this class was generated from the following file:

- G4LindhardPartition.hh

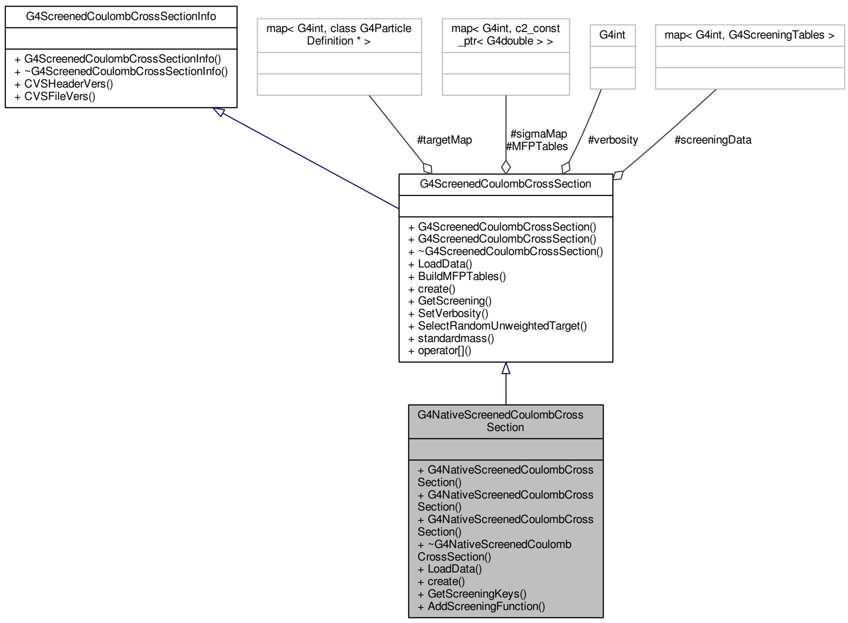
6.86 G4NativeScreenedCoulombCrossSection Class Reference

```
#include "G4ScreenedNuclearRecoil.hh"
```

Inheritance diagram for G4NativeScreenedCoulombCrossSection:



Collaboration diagram for G4NativeScreenedCoulombCrossSection:



Public Types

- typedef G4_c2_function &(∗ ScreeningFunc) (G4int z1, G4int z2, size_t nPoints, G4double rMax, G4double ∗au)

Public Member Functions

- G4NativeScreenedCoulombCrossSection ()
- G4NativeScreenedCoulombCrossSection (const G4NativeScreenedCoulombCrossSection &src)
- G4NativeScreenedCoulombCrossSection (const G4ScreenedCoulombCrossSection &src)
- virtual ∼G4NativeScreenedCoulombCrossSection ()
- virtual void LoadData (G4String screeningKey, G4int z1, G4double m1, G4double recoilCutoff)
- virtual G4ScreenedCoulombCrossSection ∗ create ()
- std::vector< G4String > GetScreeningKeys () const
- void AddScreeningFunction (G4String name, ScreeningFunc fn)

Additional Inherited Members

6.86.1 Member Typedef Documentation

6.86.1.1 typedef G4_c2_function&(∗ G4NativeScreenedCoulombCrossSection::ScreeningFunc) (G4int z1, G4int z2, size_t nPoints, G4double rMax, G4double ∗au)

6.86.2 Constructor & Destructor Documentation

6.86.2.1 G4NativeScreenedCoulombCrossSection::G4NativeScreenedCoulombCrossSection ( )

6.86.2.2 G4NativeScreenedCoulombCrossSection::G4NativeScreenedCoulombCrossSection (const G4NativeScreenedCoulombCrossSection & *src*) [inline]

6.86.2.3 G4NativeScreenedCoulombCrossSection::G4NativeScreenedCoulombCrossSection (const G4ScreenedCoulombCrossSection & *src*) [inline]

6.86.2.4 virtual G4NativeScreenedCoulombCrossSection::∼G4NativeScreenedCoulombCrossSection ( ) [virtual]

6.86.3 Member Function Documentation

6.86.3.1 void G4NativeScreenedCoulombCrossSection::AddScreeningFunction (G4String *name*, ScreeningFunc *fn*) [inline]

6.86.3.2 virtual G4ScreenedCoulombCrossSection∗ G4NativeScreenedCoulombCrossSection::create ( ) [inline], [virtual]

Implements G4ScreenedCoulombCrossSection.

6.86.3.3 std::vector<G4String> G4NativeScreenedCoulombCrossSection::GetScreeningKeys ( ) const

6.86.3.4 virtual void G4NativeScreenedCoulombCrossSection::LoadData (G4String *screeningKey*, G4int *z1*, G4double *m1*, G4double *recoilCutoff*) [virtual]

Implements G4ScreenedCoulombCrossSection.

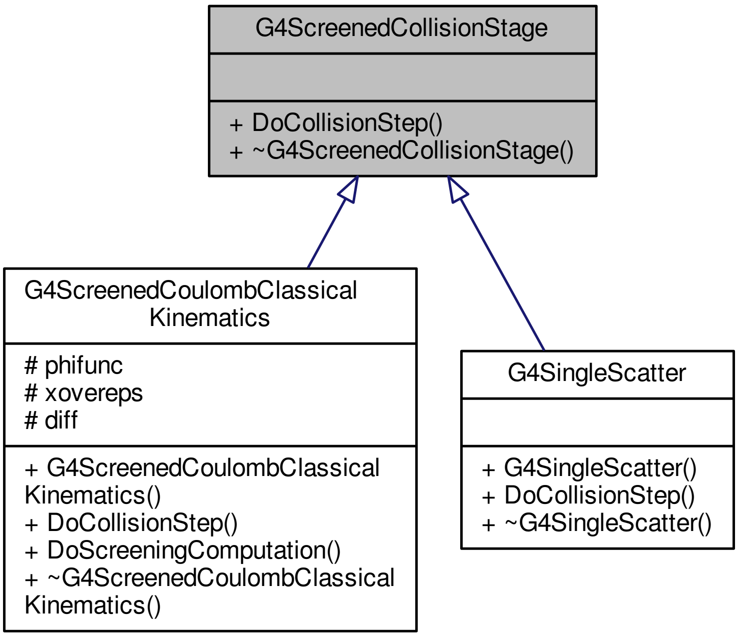
The documentation for this class was generated from the following file:

- G4ScreenedNuclearRecoil.hh

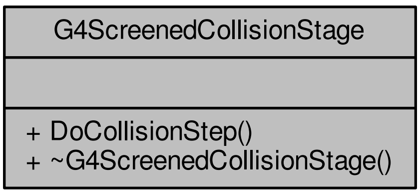
6.87 G4ScreenedCollisionStage Class Reference

```
#include "G4ScreenedNuclearRecoil.hh"
```

Inheritance diagram for G4ScreenedCollisionStage:



Collaboration diagram for G4ScreenedCollisionStage:



Public Member Functions

- virtual void DoCollisionStep (class G4ScreenedNuclearRecoil ∗master, const class G4Track &aTrack, const class G4Step &aStep)=0
- virtual ∼G4ScreenedCollisionStage ()

6.87.1 Constructor & Destructor Documentation

6.87.1.1 virtual G4ScreenedCollisionStage::~G4ScreenedCollisionStage ( ) [inline], [virtual]

6.87.2 Member Function Documentation

6.87.2.1 virtual void G4ScreenedCollisionStage::DoCollisionStep (class **G4ScreenedNuclearRecoil** * *master,* const class G4Track & *aTrack,* const class G4Step & *aStep*) [pure virtual]

Implemented in G4SingleScatter, and G4ScreenedCoulombClassicalKinematics.

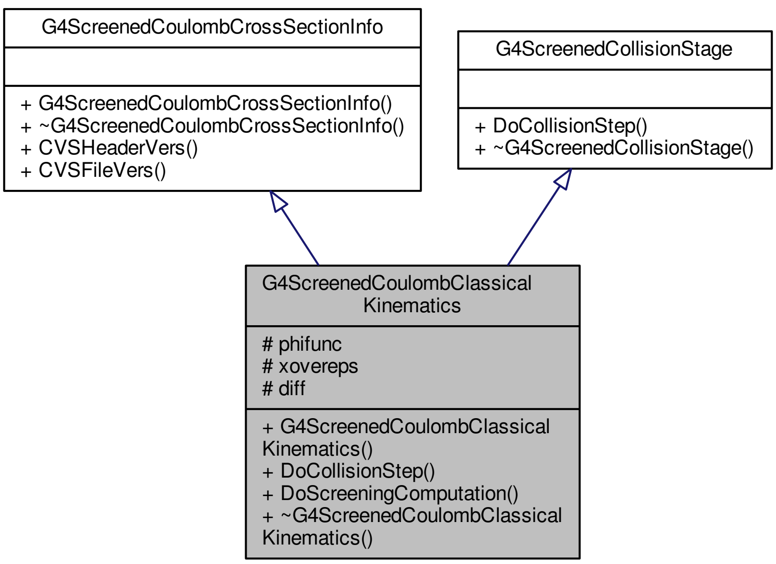
The documentation for this class was generated from the following file:

- G4ScreenedNuclearRecoil.hh

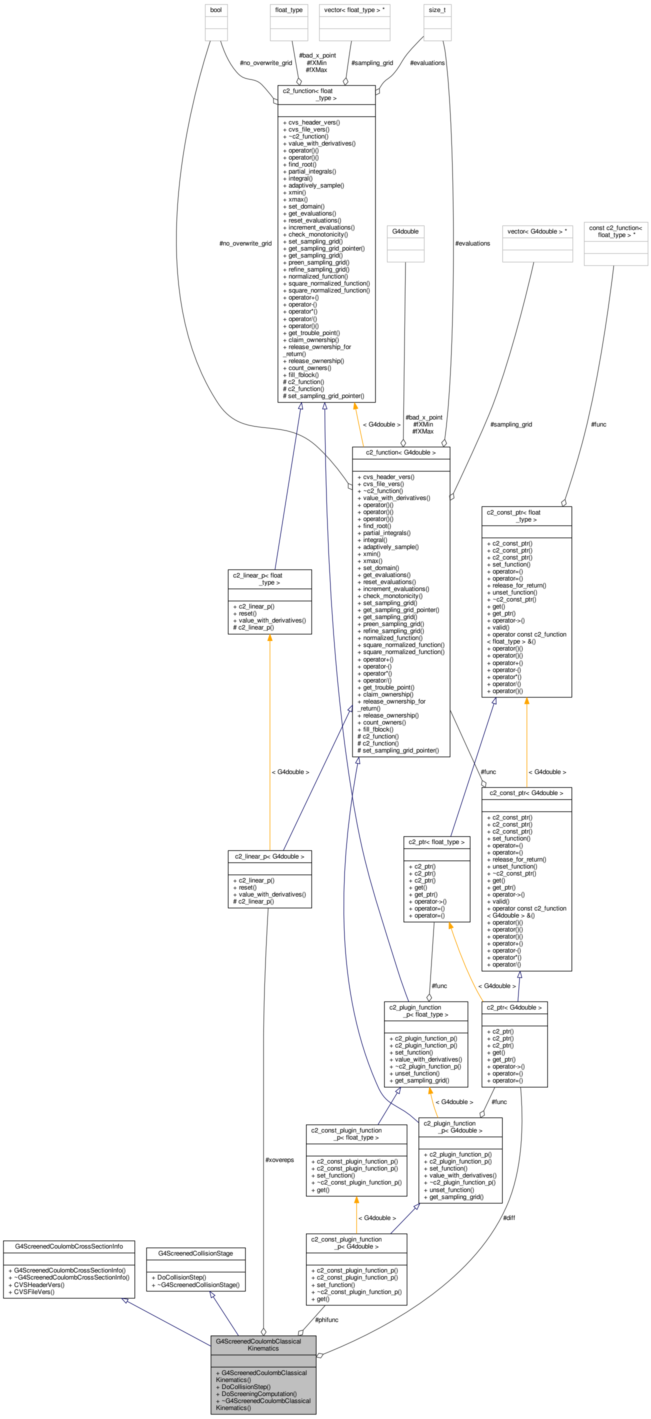
6.88 G4ScreenedCoulombClassicalKinematics Class Reference

`#include "G4ScreenedNuclearRecoil.hh"`

Inheritance diagram for G4ScreenedCoulombClassicalKinematics:



Collaboration diagram for G4ScreenedCoulombClassicalKinematics:



Public Member Functions

- G4ScreenedCoulombClassicalKinematics ()
- virtual void DoCollisionStep (class G4ScreenedNuclearRecoil ∗master, const class G4Track &aTrack, const class G4Step &aStep)
- G4bool DoScreeningComputation (class G4ScreenedNuclearRecoil ∗master, const G4ScreeningTables ∗screen, G4double eps, G4double beta)
- virtual ∼G4ScreenedCoulombClassicalKinematics ()

Protected Attributes

- c2_const_plugin_function_p< G4double > & phifunc
- c2_linear_p< G4double > & xovereps
- G4_c2_ptr diff

Additional Inherited Members

6.88.1 Constructor & Destructor Documentation

6.88.1.1 G4ScreenedCoulombClassicalKinematics::G4ScreenedCoulombClassicalKinematics ( )

6.88.1.2 virtual G4ScreenedCoulombClassicalKinematics::∼G4ScreenedCoulombClassicalKinematics ( ) [inline], [virtual]

6.88.2 Member Function Documentation

6.88.2.1 virtual void G4ScreenedCoulombClassicalKinematics::DoCollisionStep (class G4ScreenedNuclearRecoil ∗ *master*, const class G4Track & *aTrack*, const class G4Step & *aStep*) [virtual]

Implements G4ScreenedCollisionStage.

6.88.2.2 G4bool G4ScreenedCoulombClassicalKinematics::DoScreeningComputation (class G4ScreenedNuclearRecoil ∗ *master*, const G4ScreeningTables ∗ *screen*, G4double *eps*, G4double *beta*)

6.88.3 Member Data Documentation

6.88.3.1 G4_c2_ptr G4ScreenedCoulombClassicalKinematics::diff [protected]

6.88.3.2 c2_const_plugin_function_p<G4double>& G4ScreenedCoulombClassicalKinematics::phifunc [protected]

6.88.3.3 c2_linear_p<G4double>& G4ScreenedCoulombClassicalKinematics::xovereps [protected]

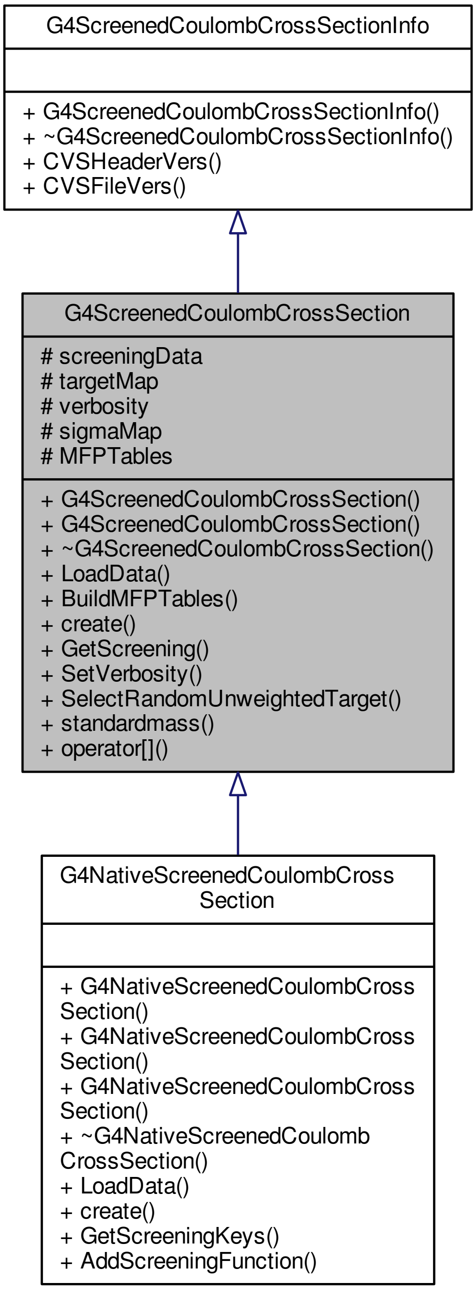
The documentation for this class was generated from the following file:

- G4ScreenedNuclearRecoil.hh

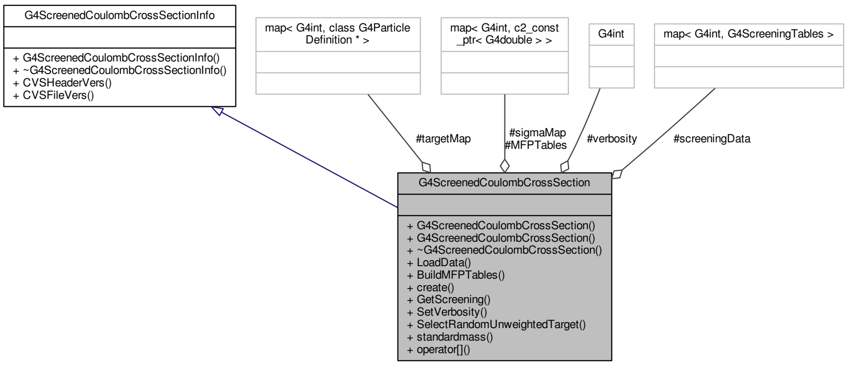
6.89 G4ScreenedCoulombCrossSection Class Reference

```
#include "G4ScreenedNuclearRecoil.hh"
```

Inheritance diagram for G4ScreenedCoulombCrossSection:



Collaboration diagram for G4ScreenedCoulombCrossSection:



Public Types

- enum { nMassMapElements =116 }
- typedef std::map< G4int, G4ScreeningTables > ScreeningMap
- typedef std::map< G4int, class G4ParticleDefinition ∗ > ParticleCache

Public Member Functions

- G4ScreenedCoulombCrossSection ()
- G4ScreenedCoulombCrossSection (const G4ScreenedCoulombCrossSection &src)
- virtual ∼G4ScreenedCoulombCrossSection ()
- virtual void LoadData (G4String screeningKey, G4int z1, G4double m1, G4double recoilCutoff)=0
- void BuildMFPTables (void)
- virtual G4ScreenedCoulombCrossSection ∗ create ()=0
- const G4ScreeningTables ∗ GetScreening (G4int Z)
- void SetVerbosity (G4int v)
- G4ParticleDefinition ∗ SelectRandomUnweightedTarget (const G4MaterialCutsCouple ∗couple)
- G4double standardmass (G4int z1)
- const G4_c2_function ∗ operator[ ] (G4int materialIndex)

Protected Attributes

- ScreeningMap screeningData
- ParticleCache targetMap
- G4int verbosity
- std::map< G4int, G4_c2_const_ptr > sigmaMap
- std::map< G4int, G4_c2_const_ptr > MFPTables

Additional Inherited Members

6.89.1 Member Typedef Documentation

6.89.1.1 typedef std::map<G4int, class G4ParticleDefinition ∗> G4ScreenedCoulombCrossSection::ParticleCache

6.89.1.2 typedef std::map<G4int, G4ScreeningTables> G4ScreenedCoulombCrossSection::ScreeningMap

6.89.2 Member Enumeration Documentation

6.89.2.1 anonymous enum

Enumerator

nMassMapElements

6.89.3 Constructor & Destructor Documentation

6.89.3.1 G4ScreenedCoulombCrossSection::G4ScreenedCoulombCrossSection ( ) [inline]

6.89.3.2 G4ScreenedCoulombCrossSection::G4ScreenedCoulombCrossSection (const G4ScreenedCoulombCross←Section & *src*) [inline]

6.89.3.3 virtual G4ScreenedCoulombCrossSection::∼G4ScreenedCoulombCrossSection ( ) [virtual]

6.89.4 Member Function Documentation

6.89.4.1 void G4ScreenedCoulombCrossSection::BuildMFPTables (void)

6.89.4.2 virtual G4ScreenedCoulombCrossSection∗ G4ScreenedCoulombCrossSection::create ( ) [pure virtual]

Implemented in G4NativeScreenedCoulombCrossSection.

6.89.4.3 const G4ScreeningTables∗ G4ScreenedCoulombCrossSection::GetScreening (G4int *Z*) [inline]

6.89.4.4 virtual void G4ScreenedCoulombCrossSection::LoadData (G4String *screeningKey,* G4int *z1,* G4double *m1,* G4double *recoilCutoff*) [pure virtual]

Implemented in G4NativeScreenedCoulombCrossSection.

6.89.4.5 const G4_c2_function* G4ScreenedCoulombCrossSection::operator[] (G4int *materialIndex*) [inline]

6.89.4.6 G4ParticleDefinition* G4ScreenedCoulombCrossSection::SelectRandomUnweightedTarget (const G4MaterialCutsCouple * *couple*)

6.89.4.7 void G4ScreenedCoulombCrossSection::SetVerbosity (G4int *v*) [inline]

6.89.4.8 G4double G4ScreenedCoulombCrossSection::standardmass (G4int *z1*) [inline]

6.89.5 Member Data Documentation

6.89.5.1 std::map<G4int, G4_c2_const_ptr > G4ScreenedCoulombCrossSection::MFPTables [protected]

6.89.5.2 ScreeningMap G4ScreenedCoulombCrossSection::screeningData [protected]

6.89.5.3 std::map<G4int, G4_c2_const_ptr > G4ScreenedCoulombCrossSection::sigmaMap [protected]

6.89.5.4 ParticleCache G4ScreenedCoulombCrossSection::targetMap [protected]

6.89.5.5 G4int G4ScreenedCoulombCrossSection::verbosity [protected]

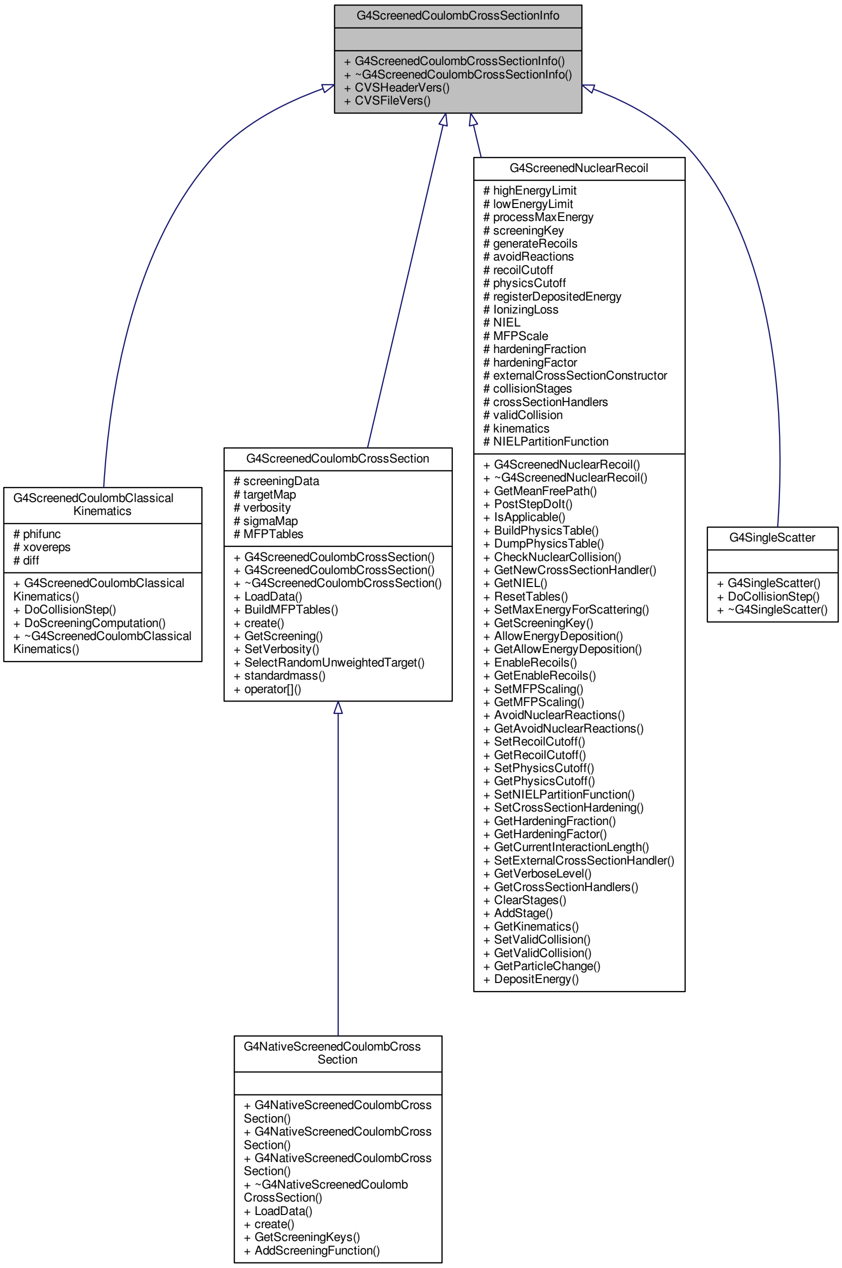
The documentation for this class was generated from the following file:

- G4ScreenedNuclearRecoil.hh

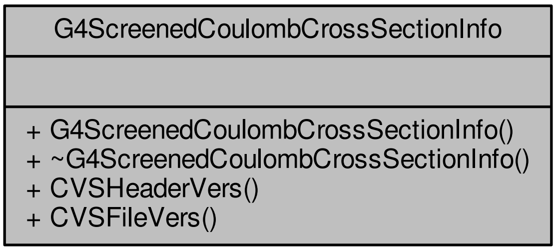
6.90 G4ScreenedCoulombCrossSectionInfo Class Reference

```
#include "G4ScreenedNuclearRecoil.hh"
```

Inheritance diagram for G4ScreenedCoulombCrossSectionInfo:



Collaboration diagram for G4ScreenedCoulombCrossSectionInfo:



Public Member Functions

- G4ScreenedCoulombCrossSectionInfo ()
- ∼G4ScreenedCoulombCrossSectionInfo ()

Static Public Member Functions

- static const char ∗ CVSHeaderVers ()
- static const char ∗ CVSFileVers ()

6.90.1 Constructor & Destructor Documentation

6.90.1.1 G4ScreenedCoulombCrossSectionInfo::G4ScreenedCoulombCrossSectionInfo ( ) [inline]

6.90.1.2 G4ScreenedCoulombCrossSectionInfo::∼G4ScreenedCoulombCrossSectionInfo ( ) [inline]

6.90.2 Member Function Documentation

6.90.2.1 static const char∗ G4ScreenedCoulombCrossSectionInfo::CVSFileVers ( ) [static]

6.90.2.2 static const char∗ G4ScreenedCoulombCrossSectionInfo::CVSHeaderVers ( ) [inline], [static]

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

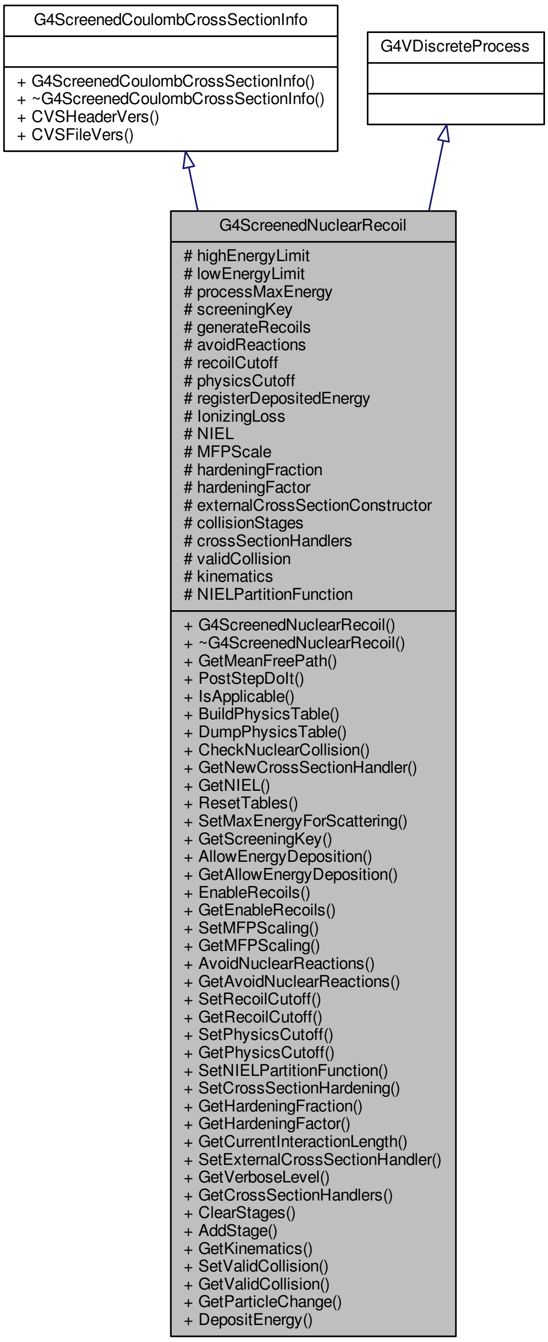
- G4ScreenedNuclearRecoil.hh

6.91 G4ScreenedNuclearRecoil Class Reference

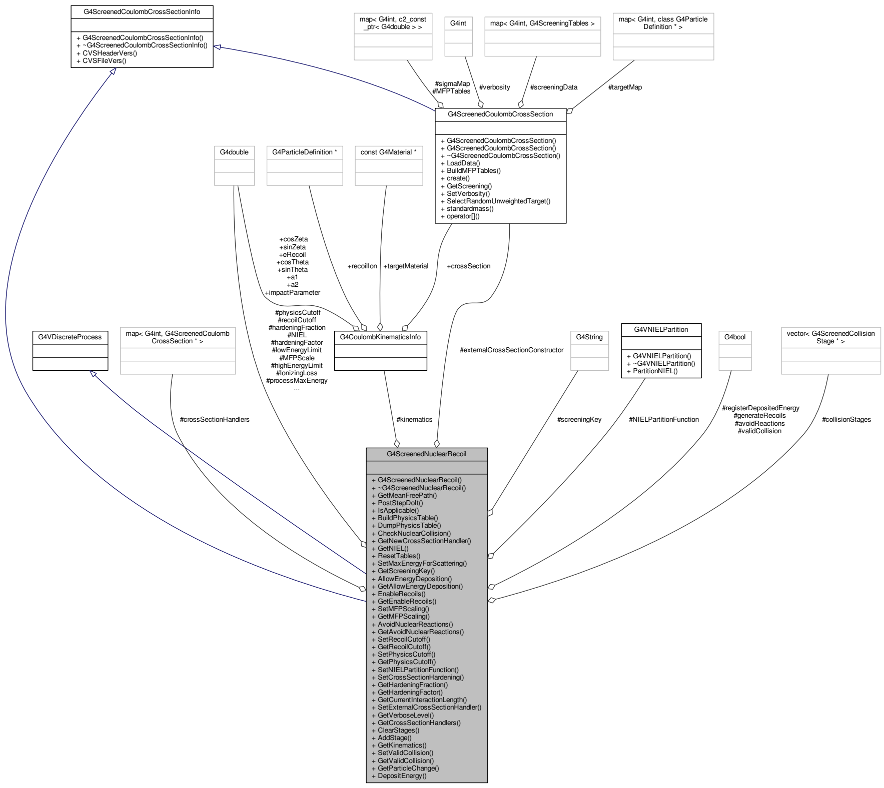
A process which handles screened Coulomb collisions between nuclei.

```
#include "G4ScreenedNuclearRecoil.hh"
```

Inheritance diagram for G4ScreenedNuclearRecoil:



Collaboration diagram for G4ScreenedNuclearRecoil:



Public Member Functions

- G4ScreenedNuclearRecoil (const G4String &processName="ScreenedElastic", const G4String &Screening↩ Key="zbl", G4bool GenerateRecoils=1, G4double RecoilCutoff=100.0 ∗CLHEP::eV, G4double Physics↩ Cutoff=10.0 ∗CLHEP::eV)

  Construct the process and set some physics parameters for it.
- virtual ∼G4ScreenedNuclearRecoil ()

  destructor
- virtual G4double GetMeanFreePath (const G4Track &, G4double, G4ForceCondition ∗)

  used internally by Geant4 machinery
- virtual G4VParticleChange ∗ PostStepDoIt (const G4Track &aTrack, const G4Step &aStep)

  used internally by Geant4 machinery
- virtual G4bool IsApplicable (const G4ParticleDefinition &aParticleType)

  test if a prticle of type aParticleType can use this process
- virtual void BuildPhysicsTable (const G4ParticleDefinition &)

  Build physics tables in advance. Not Implemented.
- virtual void DumpPhysicsTable (const G4ParticleDefinition &aParticleType)

  Export physics tables for persistency. Not Implemented.
- virtual G4bool CheckNuclearCollision (G4double A, G4double A1, G4double apsis)

  deterine if the moving particle is within the strong force range of the selected nucleus
- virtual G4ScreenedCoulombCrossSection ∗ GetNewCrossSectionHandler (void)
- G4double GetNIEL () const

  Get non-ionizing energy loss for last step.
- void ResetTables ()

clear precomputed screening tables

- void SetMaxEnergyForScattering (G4double energy)

  set the upper energy beyond which this process has no cross section
- std::string GetScreeningKey () const

  find out what screening funciton we are using
- void AllowEnergyDeposition (G4bool flag)

  enable or disable all energy deposition by this process
- G4bool GetAllowEnergyDeposition () const

  get flag indicating whether deposition is enabled
- void EnableRecoils (G4bool flag)

  enable or disable the generation of recoils. If recoils are disabled, the energy they would have received is just deposited.
- G4bool GetEnableRecoils () const

  find out if generation of recoils is enabled.
- void SetMFPScaling (G4double scale)

  set the mean free path scaling as specified
- G4double GetMFPScaling () const

  get the MFPScaling parameter
- void AvoidNuclearReactions (G4bool flag)

  enable or disable whether this process will skip collisions which are close enough they need hadronic phsyics. Default is true (skip close collisions). Disabling this results in excess nuclear stopping power.
- G4bool GetAvoidNuclearReactions () const

  get the flag indicating whether hadronic collisions are ignored.
- void SetRecoilCutoff (G4double energy)

  set the minimum energy (per nucleon) at which recoils can be generated, and the energy (per nucleon) below which all ions are stopped.
- G4double GetRecoilCutoff () const

  get the recoil cutoff
- void SetPhysicsCutoff (G4double energy)

  set the energy to which screening tables are computed. Typically, this is 10 eV or so, and not often changed.
- G4double GetPhysicsCutoff () const

  get the physics cutoff energy.
- void SetNIELPartitionFunction (const G4VNIELPartition ∗part)

  set the pointer to a class for paritioning energy into NIEL
- void SetCrossSectionHardening (G4double fraction, G4double HardeningFactor)

  set the cross section boost to provide faster computation of backscattering
- G4double GetHardeningFraction () const

  get the fraction of particles which will have boosted scattering
- G4double GetHardeningFactor () const

  get the boost factor in use.
- G4double GetCurrentInteractionLength () const

  the the interaciton length used in the last scattering.
- void SetExternalCrossSectionHandler (G4ScreenedCoulombCrossSection ∗cs)

  set a function to compute screening tables, if the user needs non-standard behavior.
- G4int GetVerboseLevel () const

  get the verbosity.
- std::map< G4int, G4ScreenedCoulombCrossSection ∗ > & GetCrossSectionHandlers ()
- void ClearStages (void)
- void AddStage (G4ScreenedCollisionStage ∗stage)
- G4CoulombKinematicsInfo & GetKinematics ()
- void SetValidCollision (G4bool flag)
- G4bool GetValidCollision () const

- class G4ParticleChange & GetParticleChange ()

  get the pointer to our ParticleChange object. for internal use, primarily.
- void DepositEnergy (G4int z1, G4double a1, const G4Material ∗material, G4double energy)

  take the given energy, and use the material information to partition it into NIEL and ionizing energy.

Protected Attributes

- G4double highEnergyLimit

  the energy per nucleon above which the MFP is constant
- G4double lowEnergyLimit

  the energy per nucleon below which the MFP is zero
- G4double processMaxEnergy

  the energy per nucleon beyond which the cross section is zero, to cross over to G4MSC
- G4String screeningKey
- G4bool generateRecoils
- G4bool avoidReactions
- G4double recoilCutoff
- G4double physicsCutoff
- G4bool registerDepositedEnergy
- G4double IonizingLoss
- G4double NIEL
- G4double MFPScale
- G4double hardeningFraction
- G4double hardeningFactor
- G4ScreenedCoulombCrossSection ∗ externalCrossSectionConstructor
- std::vector< G4ScreenedCollisionStage ∗ > collisionStages
- std::map< G4int, G4ScreenedCoulombCrossSection ∗ > crossSectionHandlers
- G4bool validCollision
- G4CoulombKinematicsInfo kinematics
- const G4VNIELPartition ∗ NIELPartitionFunction

Friends

- class G4ScreenedCollisionStage

Additional Inherited Members

6.91.1 Detailed Description

A process which handles screened Coulomb collisions between nuclei.

6.91.2 Constructor & Destructor Documentation

6.91.2.1 G4ScreenedNuclearRecoil::G4ScreenedNuclearRecoil (const G4String & *processName* = "ScreenedElastic", const G4String & *ScreeningKey* = "zbl", G4bool *GenerateRecoils* = 1, G4double *RecoilCutoff* = 100.0 ∗CLHEP::eV, G4double *PhysicsCutoff* = 10.0 ∗CLHEP::eV)

Construct the process and set some physics parameters for it.

Parameters

processName	the name to assign the process
ScreeningKey	the name of a screening function to use. The default functions are "zbl" (recommended for soft scattering), "lj" (recommended for backscattering) and "mol" (Moliere potential)
GenerateRecoils	if frue, ions struck by primary are converted into new moving particles. If false, energy is deposited, but no new moving ions are created.
RecoilCutoff	energy below which no new moving particles will be created, even if *GenerateRecoils* is true. Also, a moving primary particle will be stopped if its energy falls below this limit.
PhysicsCutoff	the energy transfer to which screening tables are calucalted. There is no really compelling reason to change it from the 10.0 eV default. However, see the paper on running this in thin targets for further discussion, and its interaction with SetMFPScaling()

6.91.2.2 virtual G4ScreenedNuclearRecoil::∼G4ScreenedNuclearRecoil ( ) [virtual]

destructor

6.91.3 Member Function Documentation

6.91.3.1 void G4ScreenedNuclearRecoil::AddStage (G4ScreenedCollisionStage ∗ *stage*) [inline]

6.91.3.2 void G4ScreenedNuclearRecoil::AllowEnergyDeposition (G4bool *flag*) [inline]

enable or disable all energy deposition by this process

Parameters

flag	if true, enable deposition of energy (the default). If false, disable deposition.

6.91.3.3 void G4ScreenedNuclearRecoil::AvoidNuclearReactions (G4bool *flag*) [inline]

enable or disable whether this process will skip collisions which are close enough they need hadronic phsyics. Default is true (skip close collisions). Disabling this results in excess nuclear stopping power.

Parameters

flag	true results in hard collisions being skipped. false allows hard collisions.

6.91.3.4 virtual void G4ScreenedNuclearRecoil::BuildPhysicsTable (const G4ParticleDefinition &) [inline], [virtual]

Build physics tables in advance. Not Implemented.

Parameters

aParticleType	the type of particle to build tables for

6.91.3.5 virtual G4bool G4ScreenedNuclearRecoil::CheckNuclearCollision (G4double *A*, G4double *A1*, G4double *apsis*) [virtual]

deterine if the moving particle is within the strong force range of the selected nucleus

Parameters

A	the nucleon number of the beam
A1	the nucleon number of the target
apsis	the distance of closest approach

6.91.3.6 void G4ScreenedNuclearRecoil::ClearStages (void)

6.91.3.7 void G4ScreenedNuclearRecoil::DepositEnergy (G4int *z1*, G4double *a1*, const G4Material ∗ *material*, G4double *energy*)

take the given energy, and use the material information to partition it into NIEL and ionizing energy.

6.91.3.8 virtual void G4ScreenedNuclearRecoil::DumpPhysicsTable (const G4ParticleDefinition & *aParticleType*) [virtual]

Export physics tables for persistency. Not Implemented.

Parameters

aParticleType	the type of particle to build tables for

6.91.3.9 void G4ScreenedNuclearRecoil::EnableRecoils (G4bool *flag*) [inline]

enable or disable the generation of recoils. If recoils are disabled, the energy they would have received is just deposited.

Parameters

flag	if true, create recoil ions in cases in which the energy is above the recoilCutoff. If false, just deposit the energy.

6.91.3.10 G4bool G4ScreenedNuclearRecoil::GetAllowEnergyDeposition ( ) const [inline]

get flag indicating whether deposition is enabled

6.91.3.11 G4bool G4ScreenedNuclearRecoil::GetAvoidNuclearReactions ( ) const [inline]

get the flag indicating whether hadronic collisions are ignored.

6.91.3.12 std::map<G4int, G4ScreenedCoulombCrossSection∗>& G4ScreenedNuclearRecoil::GetCrossSection↩ Handlers ( ) [inline]

6.91.3.13 G4double G4ScreenedNuclearRecoil::GetCurrentInteractionLength ( ) const [inline]

the the interaciton length used in the last scattering.

6.91.3.14 G4bool G4ScreenedNuclearRecoil::GetEnableRecoils ( ) const [inline]

find out if generation of recoils is enabled.

6.91.3.15 G4double G4ScreenedNuclearRecoil::GetHardeningFactor ( ) const [inline]

get the boost factor in use.

6.91.3.16 G4double G4ScreenedNuclearRecoil::GetHardeningFraction ( ) const [inline]

get the fraction of particles which will have boosted scattering

6.91.3.17 G4CoulombKinematicsInfo& G4ScreenedNuclearRecoil::GetKinematics ( ) [inline]

6.91.3.18 virtual G4double G4ScreenedNuclearRecoil::GetMeanFreePath (const G4Track & , G4double , G4ForceCondition ∗) [virtual]

used internally by Geant4 machinery

6.91.3.19 G4double G4ScreenedNuclearRecoil::GetMFPScaling ( ) const [inline]

get the MFPScaling parameter

6.91.3.20 virtual G4ScreenedCoulombCrossSection∗ G4ScreenedNuclearRecoil::GetNewCrossSectionHandler (void) [virtual]

6.91.3.21 G4double G4ScreenedNuclearRecoil::GetNIEL ( ) const [inline]

Get non-ionizing energy loss for last step.

6.91.3.22 class G4ParticleChange& G4ScreenedNuclearRecoil::GetParticleChange ( ) [inline]

get the pointer to our ParticleChange object. for internal use, primarily.

6.91.3.23 G4double G4ScreenedNuclearRecoil::GetPhysicsCutoff ( ) const [inline]

get the physics cutoff energy.

6.91.3.24 G4double G4ScreenedNuclearRecoil::GetRecoilCutoff ( ) const [inline]

get the recoil cutoff

6.91.3.25 std::string G4ScreenedNuclearRecoil::GetScreeningKey ( ) const [inline]

find out what screening funciton we are using

6.91.3.26 G4bool G4ScreenedNuclearRecoil::GetValidCollision ( ) const [inline]

6.91.3.27 G4int G4ScreenedNuclearRecoil::GetVerboseLevel ( ) const [inline]

get the verbosity.

6.91.3.28 virtual G4bool G4ScreenedNuclearRecoil::IsApplicable (const G4ParticleDefinition & *aParticleType*) [virtual]

test if a prticle of type *aParticleType* can use this process

Parameters

aParticleType	the particle to test

6.91.3.29 virtual G4VParticleChange* G4ScreenedNuclearRecoil::PostStepDoIt (const G4Track & *aTrack,* const G4Step & *aStep*) [virtual]

used internally by Geant4 machinery

6.91.3.30 void G4ScreenedNuclearRecoil::ResetTables ( )

clear precomputed screening tables

6.91.3.31 void G4ScreenedNuclearRecoil::SetCrossSectionHardening (G4double *fraction,* G4double *HardeningFactor*) [inline]

set the cross section boost to provide faster computation of backscattering

Parameters

fraction	the fraction of particles to have their cross section boosted.
HardeningFactor	the factor by which to boost the scattering cross section.

6.91.3.32 void G4ScreenedNuclearRecoil::SetExternalCrossSectionHandler (G4ScreenedCoulombCrossSection * *cs*) [inline]

set a function to compute screening tables, if the user needs non-standard behavior.

Parameters

cs	a class which constructs the screening tables.

6.91.3.33 void G4ScreenedNuclearRecoil::SetMaxEnergyForScattering (G4double *energy*) [inline]

set the upper energy beyond which this process has no cross section

This funciton is used to coordinate this process with G4MSC. Typically, G4MSC should not be allowed to operate in a range which overlaps that of this process. The criterion which is most reasonable is that the transition should be somewhere in the modestly relativistic regime (500 MeV/u for example).

Parameters

energy	energy per nucleon for the cutoff

6.91.3.34 void G4ScreenedNuclearRecoil::SetMFPScaling (G4double *scale*) [inline]

set the mean free path scaling as specified

Parameters

scale	the factor by which the default MFP will be scaled. Set to less than 1 for very thin films, typically, to sample multiple scattering, or to greater than 1 for quick simulaitons with a very long flight path.

6.91.3.35 void G4ScreenedNuclearRecoil::SetNIELPartitionFunction (const G4VNIELPartition * *part*)

set the pointer to a class for paritioning energy into NIEL

part the pointer to the class.

6.91.3.36 void G4ScreenedNuclearRecoil::SetPhysicsCutoff (G4double *energy*) [inline]

set the energy to which screening tables are computed. Typically, this is 10 eV or so, and not often changed.

Parameters

energy	the cutoff energy

6.91.3.37 void G4ScreenedNuclearRecoil::SetRecoilCutoff (G4double *energy*) [inline]

set the minimum energy (per nucleon) at which recoils can be generated, and the energy (per nucleon) below which all ions are stopped.

Parameters

energy	energy per nucleon

6.91.3.38 void G4ScreenedNuclearRecoil::SetValidCollision (G4bool *flag*) [inline]

6.91.4 Friends And Related Function Documentation

6.91.4.1 friend class G4ScreenedCollisionStage [friend]

6.91.5 Member Data Documentation

6.91.5.1 G4bool G4ScreenedNuclearRecoil::avoidReactions [protected]

6.91.5.2 std::vector<G4ScreenedCollisionStage *> G4ScreenedNuclearRecoil::collisionStages [protected]

6.91.5.3 std::map<G4int, G4ScreenedCoulombCrossSection*> G4ScreenedNuclearRecoil::crossSectionHandlers [protected]

6.91.5.4 G4ScreenedCoulombCrossSection* G4ScreenedNuclearRecoil::externalCrossSectionConstructor [protected]

6.91.5.5 G4bool G4ScreenedNuclearRecoil::generateRecoils [protected]

6.91.5.6 G4double G4ScreenedNuclearRecoil::hardeningFactor [protected]

6.91.5.7 G4double G4ScreenedNuclearRecoil::hardeningFraction [protected]

6.91.5.8 G4double G4ScreenedNuclearRecoil::highEnergyLimit [protected]

the energy per nucleon above which the MFP is constant

6.91.5.9 G4double G4ScreenedNuclearRecoil::IonizingLoss [protected]

6.91.5.10 G4CoulombKinematicsInfo G4ScreenedNuclearRecoil::kinematics [protected]

6.91.5.11 G4double G4ScreenedNuclearRecoil::lowEnergyLimit [protected]

the energy per nucleon below which the MFP is zero

6.91.5.12 G4double G4ScreenedNuclearRecoil::MFPScale [protected]

6.91.5.13 G4double G4ScreenedNuclearRecoil::NIEL [protected]

6.91.5.14 const G4VNIELPartition* G4ScreenedNuclearRecoil::NIELPartitionFunction [protected]

6.91.5.15 G4double G4ScreenedNuclearRecoil::physicsCutoff [protected]

6.91.5.16 G4double G4ScreenedNuclearRecoil::processMaxEnergy [protected]

the energy per nucleon beyond which the cross section is zero, to cross over to G4MSC

6.91.5.17 G4double G4ScreenedNuclearRecoil::recoilCutoff [protected]

6.91.5.18 G4bool G4ScreenedNuclearRecoil::registerDepositedEnergy [protected]

6.91.5.19 G4String G4ScreenedNuclearRecoil::screeningKey [protected]

6.91.5.20 G4bool G4ScreenedNuclearRecoil::validCollision [protected]

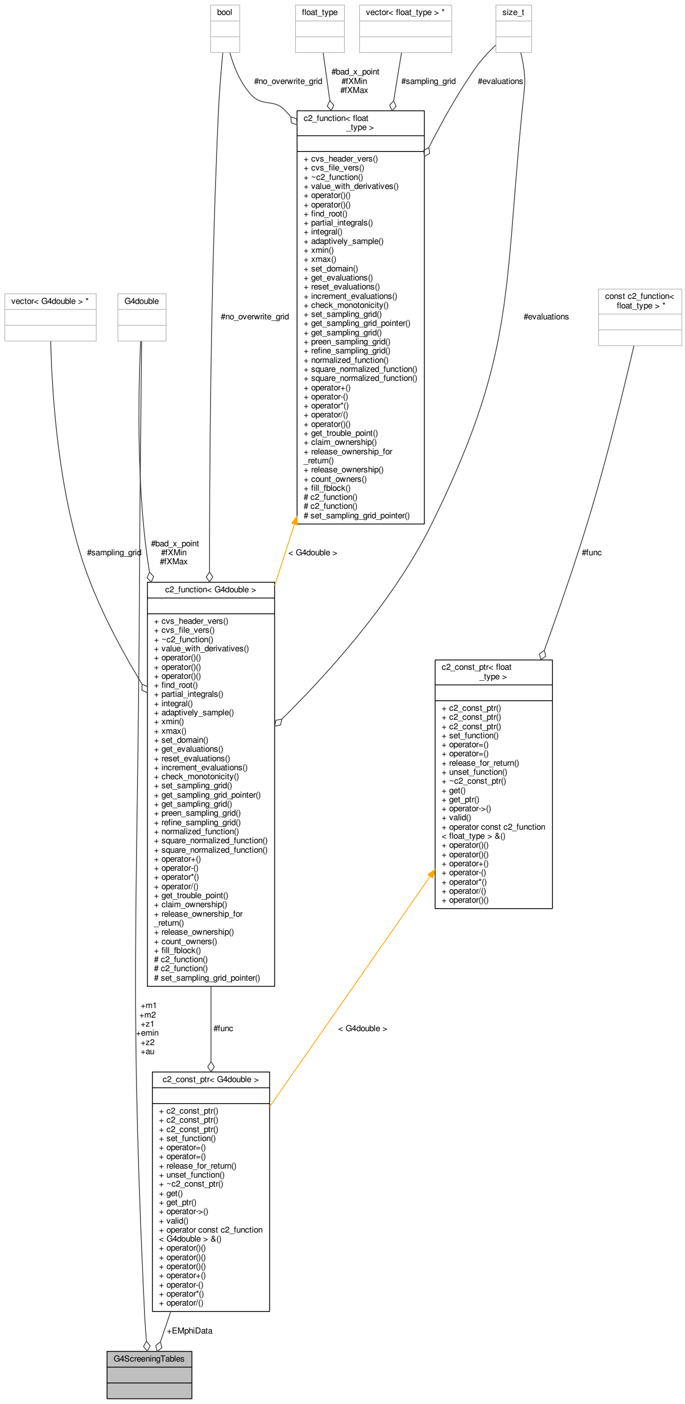
The documentation for this class was generated from the following file:

- G4ScreenedNuclearRecoil.hh

6.92 G4ScreeningTables Struct Reference

```
#include "G4ScreenedNuclearRecoil.hh"
```

Collaboration diagram for G4ScreeningTables:



Public Attributes

- G4double z1

- G4double z2
- G4double m1
- G4double m2
- G4double au
- G4double emin
- G4_c2_const_ptr EMphiData

6.92.1 Member Data Documentation

6.92.1.1 G4double G4ScreeningTables::au

6.92.1.2 G4double G4ScreeningTables::emin

6.92.1.3 G4_c2_const_ptr G4ScreeningTables::EMphiData

6.92.1.4 G4double G4ScreeningTables::m1

6.92.1.5 G4double G4ScreeningTables::m2

6.92.1.6 G4double G4ScreeningTables::z1

6.92.1.7 G4double G4ScreeningTables::z2

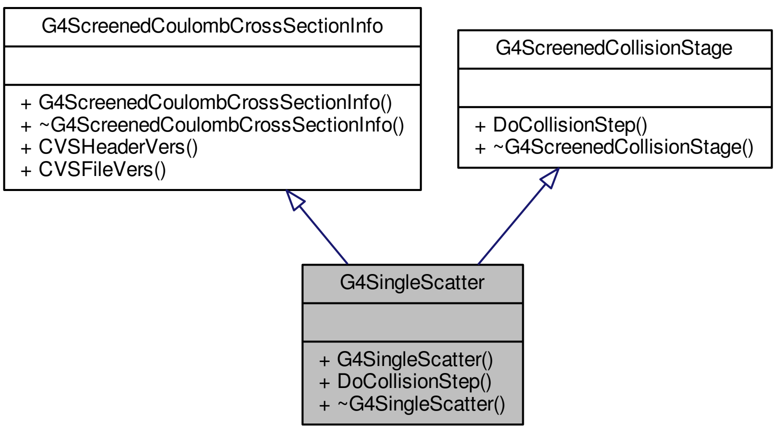
The documentation for this struct was generated from the following file:

- G4ScreenedNuclearRecoil.hh

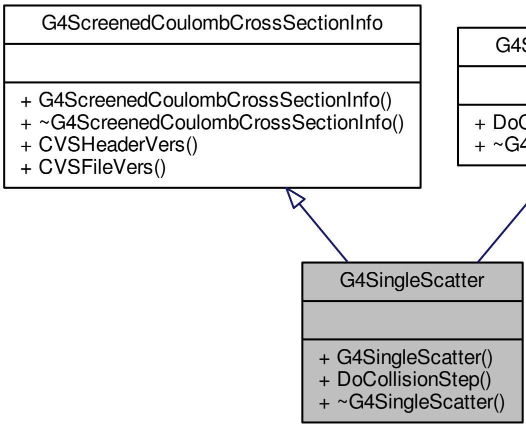
6.93 G4SingleScatter Class Reference

`#include "G4ScreenedNuclearRecoil.hh"`

Inheritance diagram for G4SingleScatter:



Collaboration diagram for G4SingleScatter:



Public Member Functions

- G4SingleScatter ()
- virtual void DoCollisionStep (class G4ScreenedNuclearRecoil ∗master, const class G4Track &aTrack, const class G4Step &aStep)
- virtual ∼G4SingleScatter ()

Additional Inherited Members

6.93.1 Constructor & Destructor Documentation

6.93.1.1 G4SingleScatter::G4SingleScatter ( ) [inline]

6.93.1.2 virtual G4SingleScatter::∼G4SingleScatter ( ) [inline], [virtual]

6.93.2 Member Function Documentation

6.93.2.1 virtual void G4SingleScatter::DoCollisionStep (class **G4ScreenedNuclearRecoil** ∗ *master*, const class G4Track & *aTrack*, const class G4Step & *aStep*) [virtual]

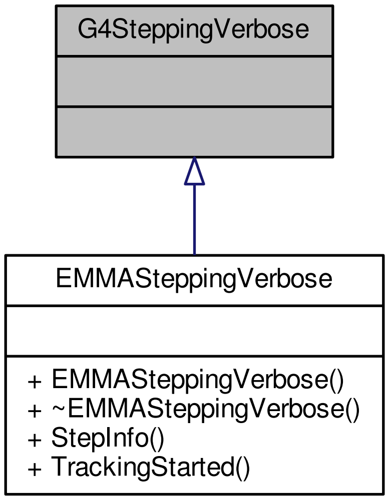
Implements G4ScreenedCollisionStage.

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

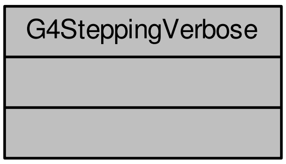
- G4ScreenedNuclearRecoil.hh

6.94 G4SteppingVerbose Class Reference

Inheritance diagram for G4SteppingVerbose:



Collaboration diagram for G4SteppingVerbose:

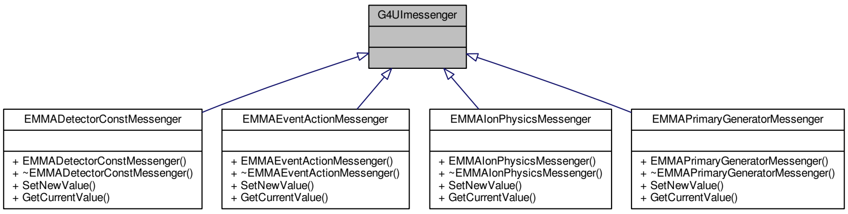


The documentation for this class was generated from the following file:

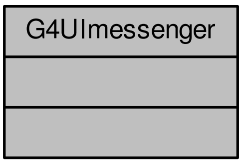
- EMMASteppingVerbose.hh

6.95 G4UImessenger Class Reference

Inheritance diagram for G4UImessenger:



Collaboration diagram for G4UImessenger:

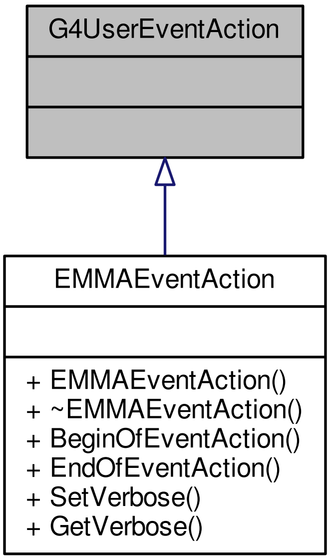


The documentation for this class was generated from the following file:

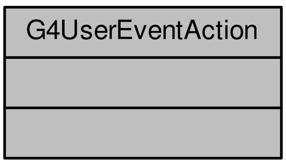
- EMMADetectorConstMessenger.hh

6.96 G4UserEventAction Class Reference

Inheritance diagram for G4UserEventAction:



Collaboration diagram for G4UserEventAction:

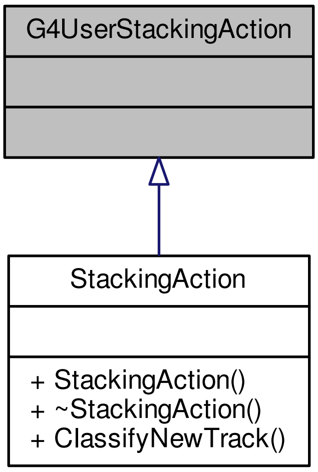


The documentation for this class was generated from the following file:

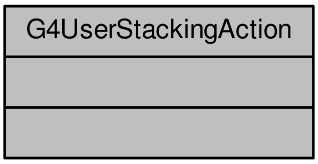
- EMMAEventAction.hh

6.97 G4UserStackingAction Class Reference

Inheritance diagram for G4UserStackingAction:



Collaboration diagram for G4UserStackingAction:

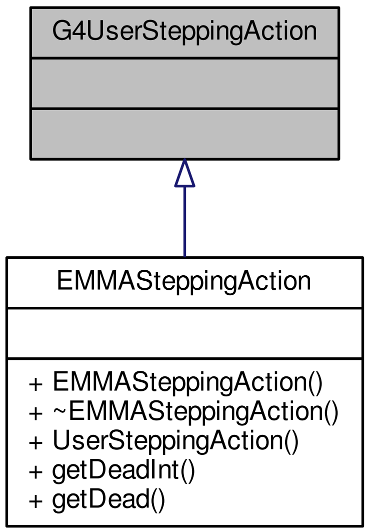


The documentation for this class was generated from the following file:

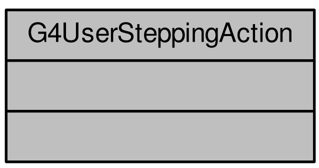
- StackingAction.hh

6.98 G4UserSteppingAction Class Reference

Inheritance diagram for G4UserSteppingAction:



Collaboration diagram for G4UserSteppingAction:

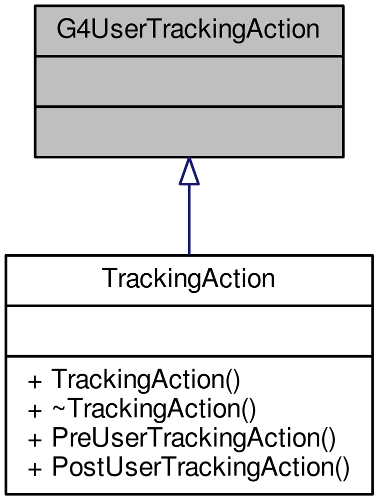


The documentation for this class was generated from the following file:

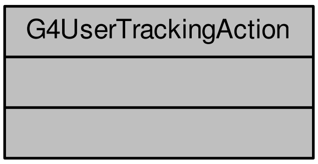
- EMMASteppingAction.hh

6.99 G4UserTrackingAction Class Reference

Inheritance diagram for G4UserTrackingAction:



Collaboration diagram for G4UserTrackingAction:

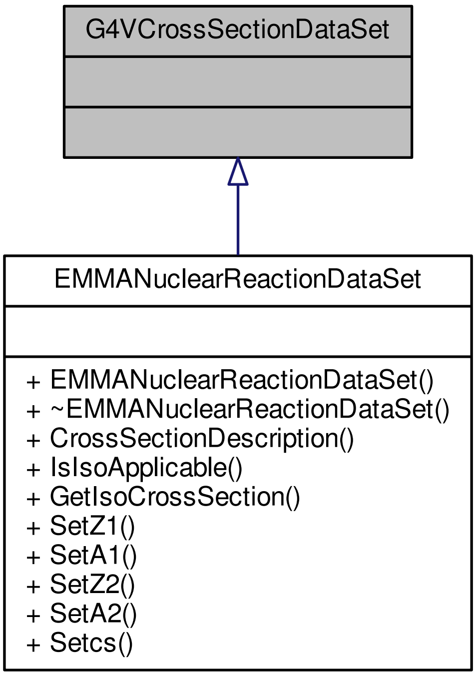


The documentation for this class was generated from the following file:

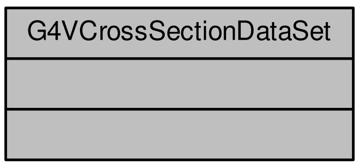
- TrackingAction.hh

6.100 G4VCrossSectionDataSet Class Reference

Inheritance diagram for G4VCrossSectionDataSet:



Collaboration diagram for G4VCrossSectionDataSet:

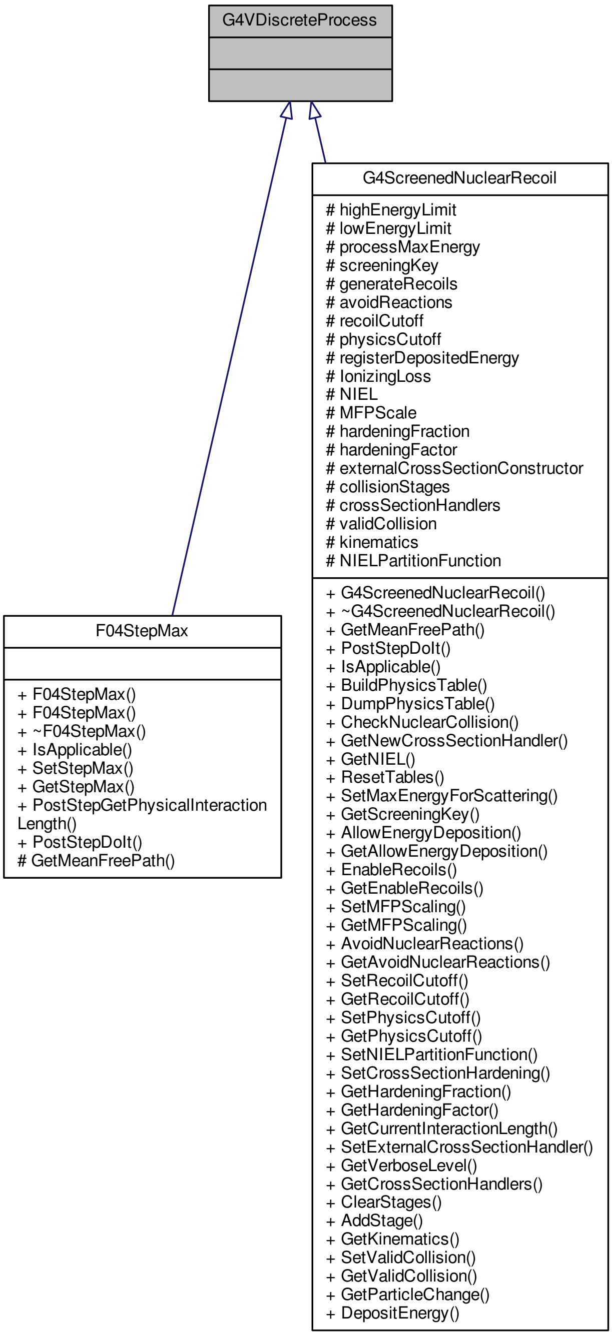


The documentation for this class was generated from the following file:

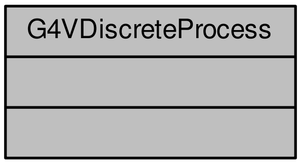
- EMMANuclearReactionDataSet.hh

6.101 G4VDiscreteProcess Class Reference

Inheritance diagram for G4VDiscreteProcess:



Collaboration diagram for G4VDiscreteProcess:

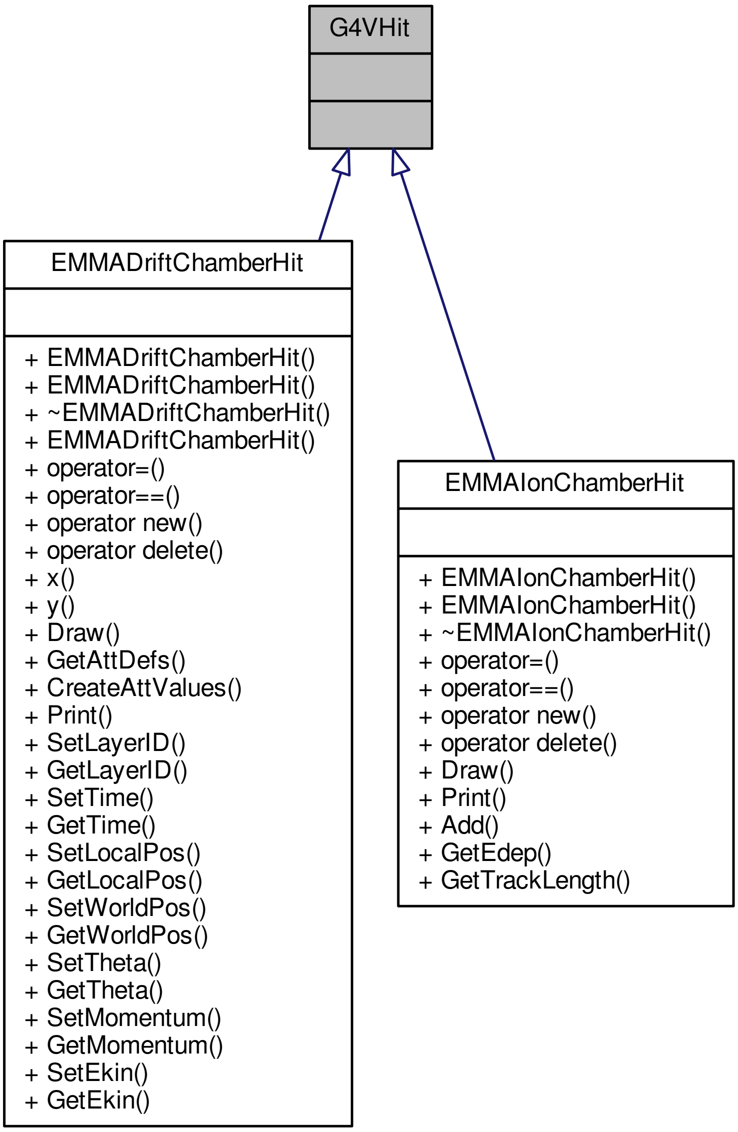


The documentation for this class was generated from the following file:

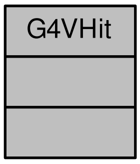
- F04StepMax.hh

6.102 G4VHit Class Reference

Inheritance diagram for G4VHit:



Collaboration diagram for G4VHit:

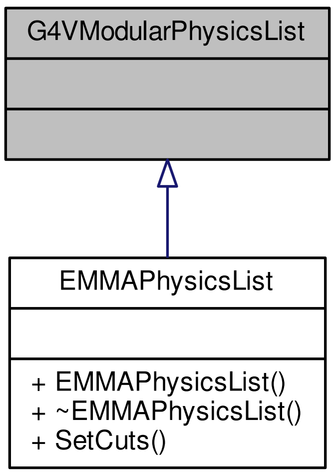


The documentation for this class was generated from the following file:

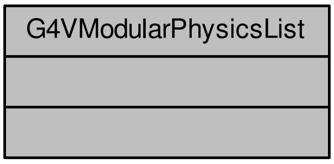
- EMMAIonChamberHit.hh

6.103 G4VModularPhysicsList Class Reference

Inheritance diagram for G4VModularPhysicsList:



Collaboration diagram for G4VModularPhysicsList:



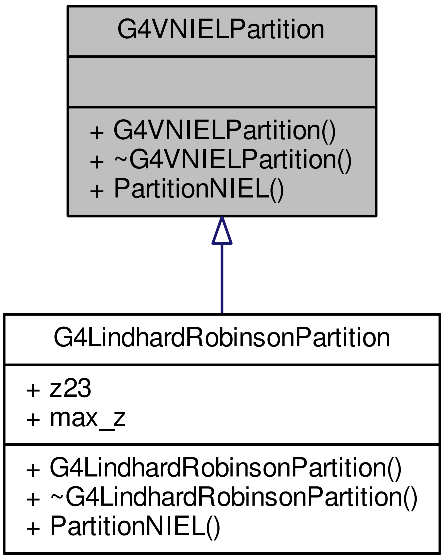
The documentation for this class was generated from the following file:

- EMMAPhysicsList.hh

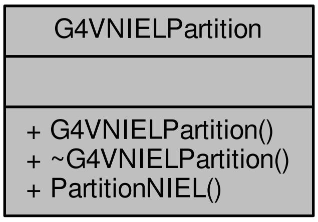
6.104 G4VNIELPartition Class Reference

```
#include "G4LindhardPartition.hh"
```

Inheritance diagram for G4VNIELPartition:



Collaboration diagram for G4VNIELPartition:



Public Member Functions

- G4VNIELPartition ()
- virtual ∼G4VNIELPartition ()
- virtual G4double PartitionNIEL (G4int z1, G4double a1, const G4Material ∗material, G4double energy) const =0

6.104.1 Constructor & Destructor Documentation

6.104.1.1 G4VNIELPartition::G4VNIELPartition ( ) [inline]

6.104.1.2 virtual G4VNIELPartition::∼G4VNIELPartition ( ) [inline], [virtual]

6.104.2 Member Function Documentation

6.104.2.1 virtual G4double G4VNIELPartition::PartitionNIEL (G4int *z1*, G4double *a1*, const G4Material ∗ *material*, G4double *energy*) const [pure virtual]

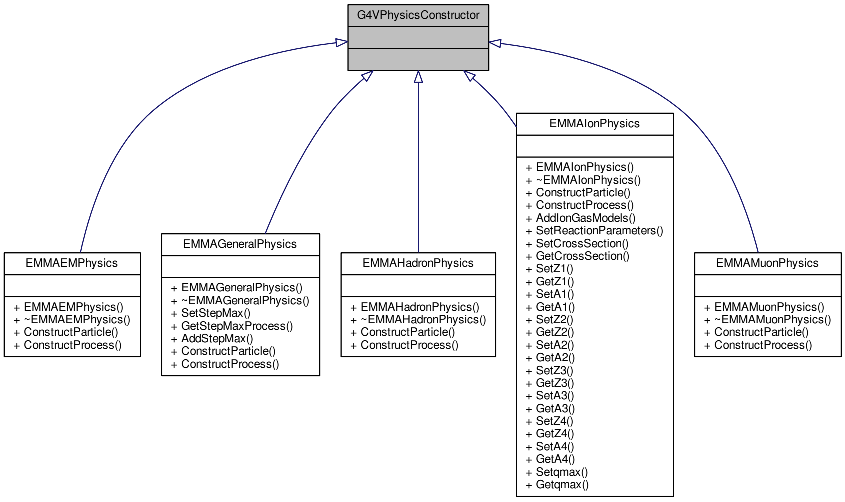
Implemented in G4LindhardRobinsonPartition.

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

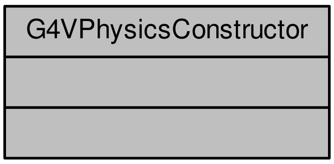
- G4LindhardPartition.hh

6.105 G4VPhysicsConstructor Class Reference

Inheritance diagram for G4VPhysicsConstructor:



Collaboration diagram for G4VPhysicsConstructor:

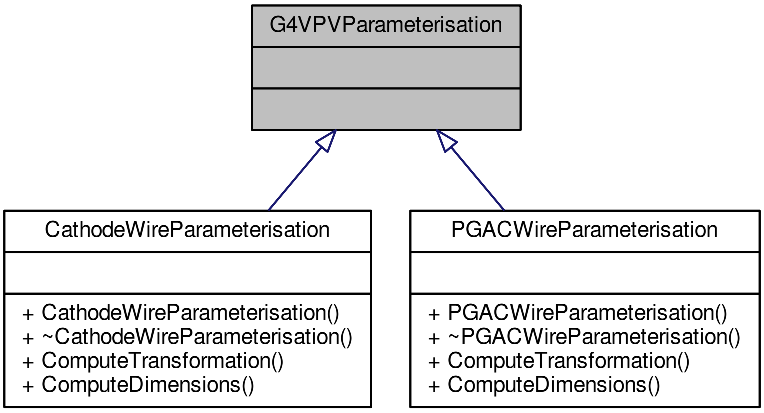


The documentation for this class was generated from the following file:

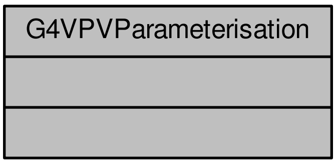
- EMMAEMPhysics.hh

6.106 G4VPVParameterisation Class Reference

Inheritance diagram for G4VPVParameterisation:



Collaboration diagram for G4VPVParameterisation:

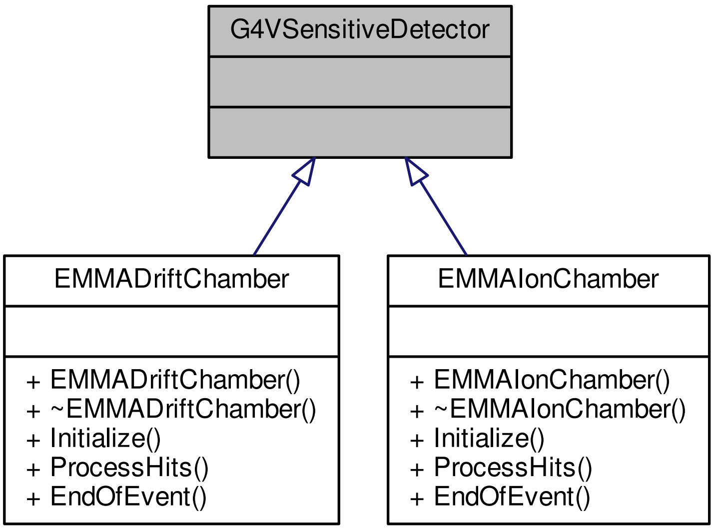


The documentation for this class was generated from the following file:

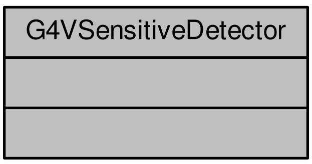
- CathodeWireParameterisation.hh

6.107 G4VSensitiveDetector Class Reference

Inheritance diagram for G4VSensitiveDetector:



Collaboration diagram for G4VSensitiveDetector:

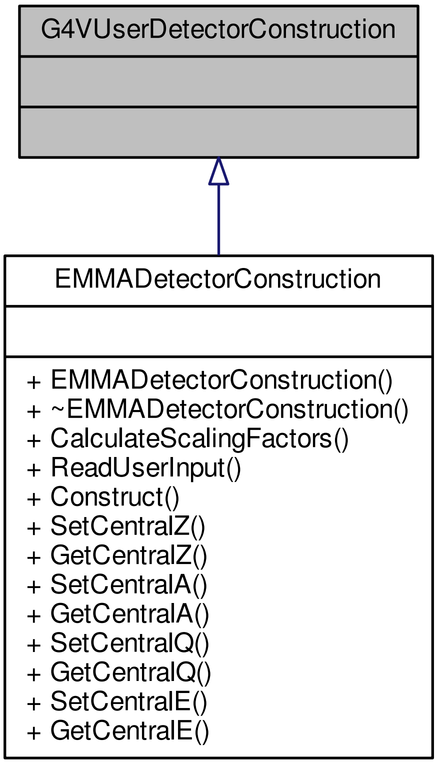


The documentation for this class was generated from the following file:

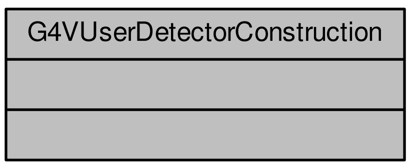
- EMMAIonChamber.hh

6.108 G4VUserDetectorConstruction Class Reference

Inheritance diagram for G4VUserDetectorConstruction:



Collaboration diagram for G4VUserDetectorConstruction:

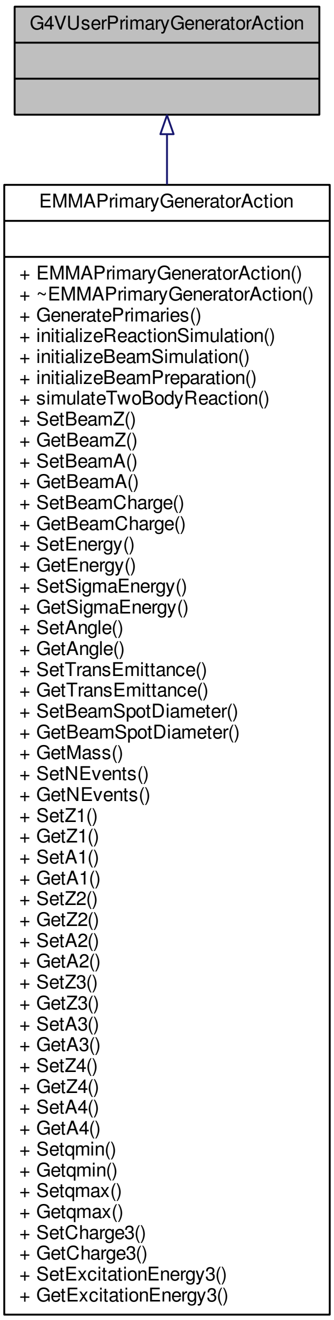


The documentation for this class was generated from the following file:

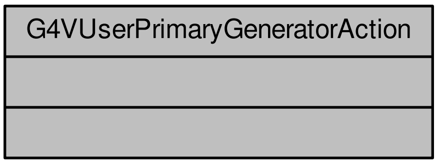
- EMMADetectorConstruction.hh

6.109 G4VUserPrimaryGeneratorAction Class Reference

Inheritance diagram for G4VUserPrimaryGeneratorAction:



Collaboration diagram for G4VUserPrimaryGeneratorAction:



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

- EMMAPrimaryGeneratorAction.hh

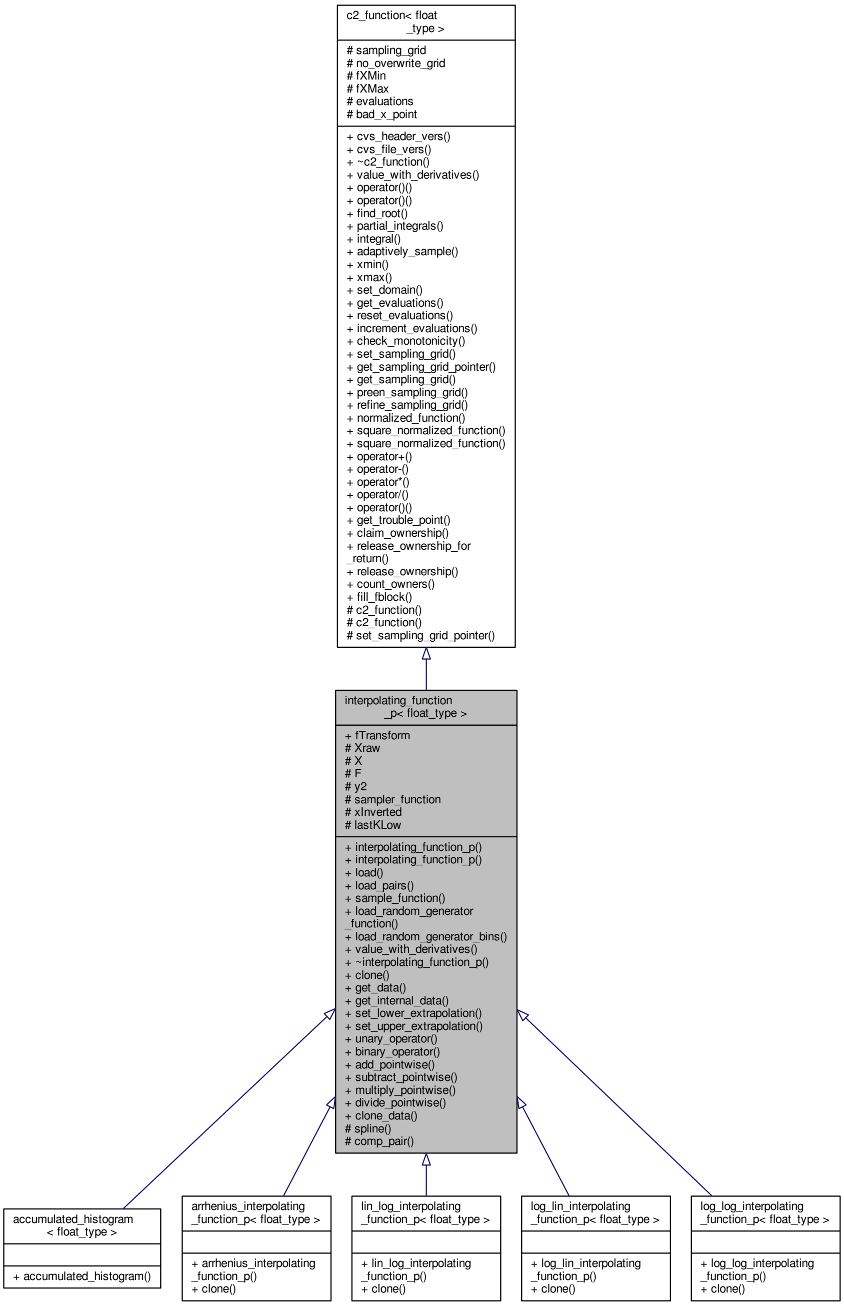
6.110 interpolating_function_p< float_type > Class Template Reference

create a cubic spline interpolation of a set of (x,y) pairs

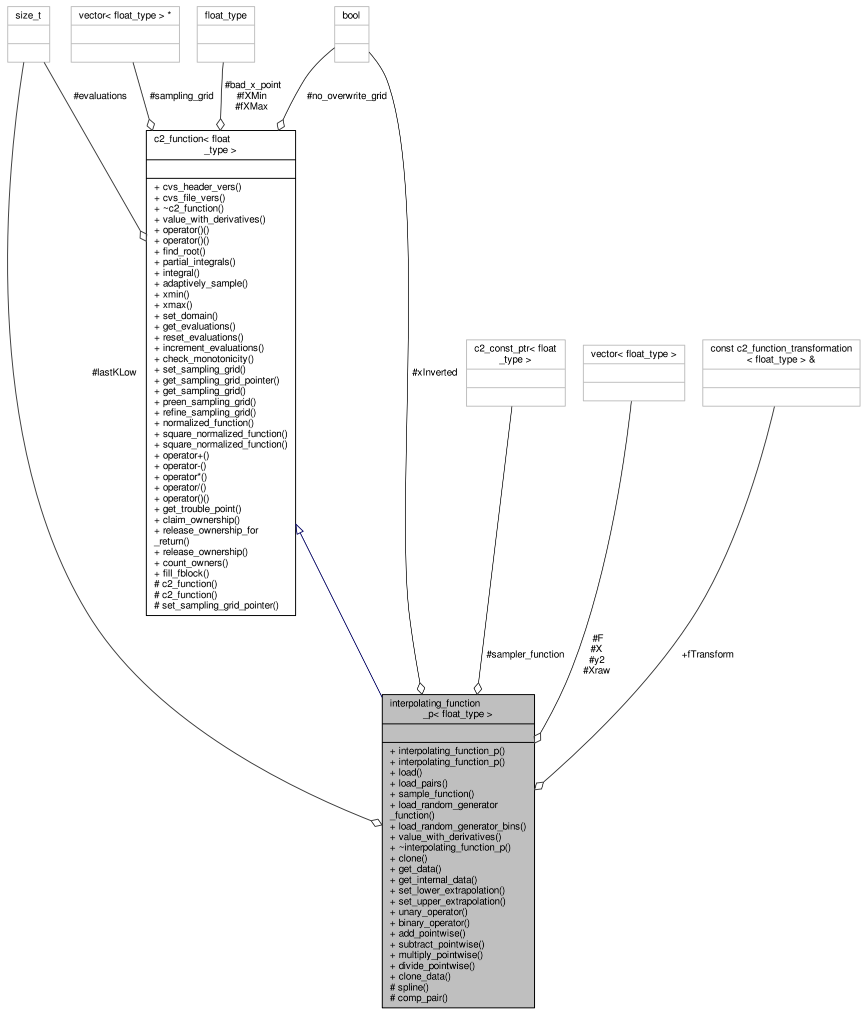
This is one of the main reasons for c2_function objects to exist.

```
#include "c2_function.hh"
```

Inheritance diagram for interpolating_function_p< float_type >:



Collaboration diagram for interpolating_function_p< float_type >:



Public Member Functions

- interpolating_function_p ()

    an empty linear-linear cubic-spline interpolating_function_p

- interpolating_function_p (const c2_function_transformation< float_type > &transform)

    an empty cubic-spline interpolating_function_p with a specific transform

- interpolating_function_p< float_type > & load (const std::vector< float_type > &x, const std::vector< float_type > &f, bool lowerSlopeNatural, float_type lowerSlope, bool upperSlopeNatural, float_type upperSlope, bool splined=true) throw (c2_exception)

    do the dirty work of constructing the spline from a function.

- interpolating_function_p< float_type > & load_pairs (std::vector< std::pair< float_type, float_type > > &data, bool lowerSlopeNatural, float_type lowerSlope, bool upperSlopeNatural, float_type upperSlope, bool splined=true) throw (c2_exception)

    do the dirty work of constructing the spline from a function.

- interpolating_function_p< float_type > & sample_function (const c2_function< float_type > &func, float_type amin, float_type amax, float_type abs_tol, float_type rel_tol, bool lowerSlopeNatural, float_type lowerSlope, bool upperSlopeNatural, float_type upperSlope) throw (c2_exception)

    do the dirty work of constructing the spline from a function.

- interpolating_function_p< float_type > & load_random_generator_function (const std::vector< float_type > &bincenters, const c2_function< float_type > &binheights) throw (c2_exception)

    initialize from a grid of points and a c2_function (un-normalized) to an interpolator which, when evaluated with a uniform random variate on [0,1] returns random numbers distributed as the input function.
- interpolating_function_p< float_type > & load_random_generator_bins (const std::vector< float_type > &bins, const std::vector< float_type > &binheights, bool splined=true) throw (c2_exception)

    initialize from a grid of points and an std::vector of probability densities (un-normalized) to an interpolator which, when evaluated with a uniform random variate on [0,1] returns random numbers distributed as the input histogram.
- virtual float_type value_with_derivatives (float_type x, float_type ∗yprime, float_type ∗yprime2) const throw (c2_exception)

    get the value and derivatives.
- virtual ∼interpolating_function_p ()

    destructor
- virtual interpolating_function_p< float_type > & clone () const throw (c2_exception)

    create a new, empty interpolating function of this type (virtual constructor)
- void get_data (std::vector< float_type > &xvals, std::vector< float_type > &yvals) const throw ()

    retrieve copies of the x & y tables from which this was built
- void get_internal_data (std::vector< float_type > &xvals, std::vector< float_type > &yvals, std::vector< float_type > &y2vals) const

    retrieve copies of the transformed x, y and y2 tables from which this was built
- void set_lower_extrapolation (float_type bound)

    enable extrapolation of the function below the tabulated data.
- void set_upper_extrapolation (float_type bound)

    enable extrapolation of the function above the tabulated data.
- interpolating_function_p< float_type > & unary_operator (const c2_function< float_type > &source) const

    create a new interpolating_function_p which is the source function applied to every point in the interpolating tables
- interpolating_function_p< float_type > & binary_operator (const c2_function< float_type > &rhs, const c2_binary_function< float_type > ∗combining_stub) const

    create a new interpolating_function_p which is the parent interpolating_function_p combined with rhs using combiner at every point in the interpolating tables
- interpolating_function_p< float_type > & add_pointwise (const c2_function< float_type > &rhs) const

    produce a newly resampled interpolating_function_p which is the specified sum.
- interpolating_function_p< float_type > & subtract_pointwise (const c2_function< float_type > &rhs) const

    produce a newly resampled interpolating_function_p which is the specified difference.
- interpolating_function_p< float_type > & multiply_pointwise (const c2_function< float_type > &rhs) const

    produce a newly resampled interpolating_function_p which is the specified product.
- interpolating_function_p< float_type > & divide_pointwise (const c2_function< float_type > &rhs) const

    produce a newly resampled interpolating_function_p which is the specified ratio.
- void clone_data (const interpolating_function_p< float_type > &rhs)

    copy data from another interpolating function. This only makes sense if the source

Public Attributes

- const c2_function_transformation< float_type > & fTransform

Protected Member Functions

- void spline (bool lowerSlopeNatural, float_type lowerSlope, bool upperSlopeNatural, float_type upperSlope) throw (c2_exception)

    create the spline coefficients

Static Protected Member Functions

- static bool comp_pair (std::pair< float_type, float_type > const &i, std::pair< float_type, float_type > const &j)

Protected Attributes

- std::vector< float_type > Xraw
- std::vector< float_type > X
- std::vector< float_type > F
- std::vector< float_type > y2
- c2_const_ptr< float_type > sampler_function
- bool xInverted
- size_t lastKLow

6.110.1 Detailed Description

template<typename float_type = double>
class interpolating_function_p< float_type >

create a cubic spline interpolation of a set of (x,y) pairs

This is one of the main reasons for c2_function objects to exist.

It provides support for cubic spline interpolation of data provides from tables of *x*, *y* pairs. It supports automatic, transparent linearization of the data before storing in its tables (through subclasses such as log_lin_interpolating_function, lin_log_interpolating_function, and log_log_interpolating_function) to permit very high accuracy representations of data which have a suitable structure. It provides utility functions LinearInterpolatingGrid() and LogLogInterpolatingGrid() to create grids for mapping other functions onto a arithmetic or geometric grid.

In its simplest form, an untransformed cubic spline of a data set, using natural boundary conditions (vanishing second derivative), is created as:

```
    c2_ptr<double> c2p;
    c2_factory<double> c2;
std::vector<double> xvals(10), yvals(10);
// < fill in xvals and yvals >
c2p myfunc=c2.interpolating_function().load(xvals, yvals,true,0,true,0);
// and it can be evaluated at a point for its value only by:
double y=myfunc(x);
// or it can be evaluated with its derivatives by
double yprime, yprime2;
double y=myfunc(x,&yprime, &yprime2);
```

`The factory function c2_factory::interpolating_function() creates *new interpolating_function_p()`

6.110.2 Constructor & Destructor Documentation

6.110.2.1 template<typename float_type = double> interpolating_function_p< float_type >::interpolating_function_p ( ) [inline]

an empty linear-linear cubic-spline interpolating_function_p

lots to say here, but see Numerical Recipes for a discussion of cubic splines.

6.110.2.2 template<typename float_type = double> **interpolating_function_p**< float_type >::**interpolating_function_p** (const **c2_function_transformation**< float_type > & *transform*) [inline]

an empty cubic-spline interpolating_function_p with a specific transform

6.110.2.3 template<typename float_type = double> virtual **interpolating_function_p**< float_type >::~**interpolating_function_p** ( ) [inline], [virtual]

destructor

6.110.3 Member Function Documentation

6.110.3.1 template<typename float_type = double> **interpolating_function_p**<float_type>& **interpolating_function_p**< float_type >::add_pointwise (const **c2_function**< float_type > & *rhs*) const [inline]

produce a newly resampled interpolating_function_p which is the specified sum.

Parameters

rhs	the function to add, pointwise

Returns

a new interpolating_function_p

6.110.3.2 template<typename float_type = double> **interpolating_function_p**<float_type>& **interpolating_function_p**< float_type >::binary_operator (const **c2_function**< float_type > & *rhs,* const **c2_binary_function**< float_type > * *combining_stub*) const

create a new interpolating_function_p which is the parent interpolating_function_p combined with *rhs* using *combiner* at every point in the interpolating tables

This carefully manages the derivative of the composed function at the two ends.

Parameters

rhs	the function to apply
combining_stub	a function which defines which binary operation to use.

Returns

a new interpolating_function_p with the same mappings for x and y

6.110.3.3 template<typename float_type = double> virtual interpolating_function_p<float_type>& interpolating_function_p< float_type >::clone ( ) const throw c2_exception) [inline], [virtual]

create a new, empty interpolating function of this type (virtual constructor)

Reimplemented in arrhenius_interpolating_function_p< float_type >, log_log_interpolating_function_p< float_type >, lin_log_interpolating_function_p< float_type >, and log_lin_interpolating_function_p< float_type >.

6.110.3.4 template<typename float_type = double> void interpolating_function_p< float_type >::clone_data (const interpolating_function_p< float_type > & *rhs*) [inline]

copy data from another interpolating function. This only makes sense if the source

function has the same transforms as the destination.

Parameters

rhs	interpolating_function_p to copy from

6.110.3.5 template<typename float_type = double> static bool interpolating_function_p< float_type >::comp_pair (std::pair< float_type, float_type > const & *i*, std::pair< float_type, float_type > const & *j*) [inline], [static], [protected]

6.110.3.6 template<typename float_type = double> interpolating_function_p<float_type>& interpolating_function_p< float_type >::divide_pointwise (const c2_function< float_type > & *rhs*) const [inline]

produce a newly resampled interpolating_function_p which is the specified ratio.

Parameters

rhs	the function to divide, pointwise

Returns

a new interpolating_function_p

6.110.3.7 template<typename float_type = double> void interpolating_function_p< float_type >::get_data (std::vector< float_type > & *xvals*, std::vector< float_type > & *yvals*) const throw)

retrieve copies of the x & y tables from which this was built

This is often useful in the creation of new interpolating functions with transformed data. The vectors will have their sizes set correctly on return.

Parameters

in,out	*xvals*	the abscissas
in,out	*yvals*	the ordinates

6.110.3.8 template<typename float_type = double> void interpolating_function_p< float_type >::get_internal_data (std::vector< float_type > & *xvals*, std::vector< float_type > & *yvals*, std::vector< float_type > & *y2vals*) const [inline]

retrieve copies of the transformed x, y and y2 tables from which this was built

The vectors will have their sizes set correctly on return.

Parameters

in,out	*xvals*	the transformed abscissas
in,out	*yvals*	the transformed ordinates
in,out	*y2vals*	the second derivatives

6.110.3.9 template<typename float_type = double> interpolating_function_p<float_type>& interpolating_function_p< float_type >::load (const std::vector< float_type > & *x*, const std::vector< float_type > & *f*, bool *lowerSlopeNatural*, float_type *lowerSlope*, bool *upperSlopeNatural*, float_type *upperSlope*, bool *splined* = true) throw c2_exception)

do the dirty work of constructing the spline from a function.

Parameters

x	the list of abscissas. Must be either strictly increasing or strictly decreasing. Strictly increasing is preferred, as less memory is used since a copy is not required for the sampling grid.
f	the list of function values.
lowerSlopeNatural	if true, set y''(first point)=0, otherwise compute it from *lowerSope*
lowerSlope	derivative of the function at the lower bound, used only if *lowerSlopeNatural* is false
upperSlopeNatural	if true, set y''(last point)=0, otherwise compute it from *upperSope*
upperSlope	derivative of the function at the upper bound, used only if *upperSlopeNatural* is false
splined	if true (default), use cubic spline, if false, use linear interpolation.

Returns

the same interpolating function, filled

6.110.3.10 template<typename float_type = double> interpolating_function_p<float_type>& interpolating_function_p< float_type >::load_pairs (std::vector< std::pair< float_type, float_type > > & *data*, bool *lowerSlopeNatural*, float_type *lowerSlope*, bool *upperSlopeNatural*, float_type *upperSlope*, bool *splined* = true) throw c2_exception)

do the dirty work of constructing the spline from a function.

Parameters

data	std::vector of std::pairs of x,y. Will be sorted into x increasing order in place.
lowerSlopeNatural	if true, set y''(first point)=0, otherwise compute it from *lowerSope*
lowerSlope	derivative of the function at the lower bound, used only if *lowerSlopeNatural* is false
upperSlopeNatural	if true, set y''(last point)=0, otherwise compute it from *upperSope*
upperSlope	derivative of the function at the upper bound, used only if *upperSlopeNatural* is false
splined	if true (default), use cubic spline, if false, use linear interpolation.

Returns

the same interpolating function, filled

6.110.3.11 template< typename float_type = double> **interpolating_function_p**<float_type>& **interpolating_function_p**< float_type >::load_random_generator_bins (const std::vector< float_type > & *bins,* const std::vector< float_type > & *binheights,* bool *splined* = `true`) throw **c2_exception**)

initialize from a grid of points and an std::vector of probability densities (un-normalized) to an interpolator which, when evaluated with a uniform random variate on [0,1] returns random numbers distributed as the input histogram.

See also

Arbitrary random generation inverse_integrated_density starts with a probability density std::vector, generates the integral, and generates an interpolating_function_p of the inverse function which, when evaluated using a uniform random on [0,1] returns values with a density distribution equal to the input distribution If the data are passed in reverse order (large X first), the integral is carried out from the big end.

Parameters

bins	if *bins* .size()==*binheights* .size(), the centers of the bins. if *bins* .size()==*binheights* .size()+1, the edges of the bins
binheights	a vector which describes the density of the random number distribution to be produced. Note density... the numbers in the bins are not counts, but counts/unit bin width.
splined	if true (default), use cubic spline, if false, use linear interpolation. This can often be used to fix ringing if there are very sharp features in the generator.

Returns

an initialized interpolator, which if evaluated randomly with a uniform variate on [0,1] produces numbers distributed according to *binheights*

6.110.3.12 template< typename float_type = double> **interpolating_function_p**<float_type>& **interpolating_function_p**< float_type >::load_random_generator_function (const std::vector< float_type > & *bincenters,* const **c2_function**< float_type > & *binheights*) throw **c2_exception**)

initialize from a grid of points and a c2_function (un-normalized) to an interpolator which, when evaluated with a uniform random variate on [0,1] returns random numbers distributed as the input function.

See also

Arbitrary random generation inverse_integrated_density starts with a probability density std::vector, generates the integral, and generates an interpolating_function_p of the inverse function which, when evaluated using a uniform random on [0,1] returns values with a density distribution equal to the input distribution If the data are passed in reverse order (large X first), the integral is carried out from the big end.

Parameters

bincenters	the positions at which to sample the function *binheights*
binheights	a function which describes the density of the random number distribution to be produced.

Returns

an initialized interpolator, which if evaluated randomly with a uniform variate on [0,1] produces numbers distributed according to *binheights*

6.110.3.13 template<typename float_type = double> interpolating_function_p<float_type>& interpolating_function_p< float_type >::multiply_pointwise (const c2_function< float_type > & *rhs*) const [inline]

produce a newly resampled interpolating_function_p which is the specified product.

Parameters

rhs	the function to multiply, pointwise

Returns

a new interpolating_function_p

6.110.3.14 template<typename float_type = double> interpolating_function_p<float_type>& interpolating_function_p< float_type >::sample_function (const c2_function< float_type > & *func*, float_type *amin*, float_type *amax*, float_type *abs_tol*, float_type *rel_tol*, bool *lowerSlopeNatural*, float_type *lowerSlope*, bool *upperSlopeNatural*, float_type *upperSlope*) throw c2_exception)

do the dirty work of constructing the spline from a function.

Parameters

func	a function without any requirement of valid derivatives to sample into an interpolating function. Very probably a c2_classic_function.
amin	the lower bound of the region to sample
amax	the upper bound of the region to sample
abs_tol	the maximum absolute error permitted when linearly interpolating the points. the real error will be much smaller, since this uses cubic splines at the end.
rel_tol	the maximum relative error permitted when linearly interpolating the points. the real error will be much smaller, since this uses cubic splines at the end.
lowerSlopeNatural	if true, set y'(first point) from 3-point parabola, otherwise compute it from *lowerSope*
lowerSlope	derivative of the function at the lower bound, used only if *lowerSlopeNatural* is false
upperSlopeNatural	if true, set y'(last point) from 3-point parabola, otherwise compute it from *upperSope*
upperSlope	derivative of the function at the upper bound, used only if *upperSlopeNatural* is false

Returns

the same interpolating function, filled

Note

If the interpolator being filled has a log vertical axis, put the desired relative error in *abs_tol*, and 0 in *rel_tol* since the absolute error on the log of a function is the relative error on the function itself.

6.110.3.15 template<typename float_type = double> void interpolating_function_p< float_type >::set_lower_extrapolation (float_type *bound*)

enable extrapolation of the function below the tabulated data.

This allows the interpolator to be extrapolated outside the bounds of the data, using whatever derivatives it already had at the lower bound.

Parameters

bound	the abscissa to which the function should be extended.

6.110.3.16 template<typename float_type = double> void interpolating_function_p< float_type >::set_upper_extrapolation (float_type *bound*)

enable extrapolation of the function above the tabulated data.

This allows the interpolator to be extrapolated outside the bounds of the data, using whatever derivatives it already had at the upper bound.

Parameters

bound	the abscissa to which the function should be extended.

6.110.3.17 template<typename float_type = double> void interpolating_function_p< float_type >::spline (bool *lowerSlopeNatural,* float_type *lowerSlope,* bool *upperSlopeNatural,* float_type *upperSlope*) throw c2_exception) `[protected]`

create the spline coefficients

6.110.3.18 template<typename float_type = double> interpolating_function_p<float_type>& interpolating_function_p< float_type >::subtract_pointwise (const c2_function< float_type > & *rhs*) const `[inline]`

produce a newly resampled interpolating_function_p which is the specified difference.

Parameters

rhs	the function to subtract, pointwise

Returns

a new interpolating_function_p

6.110.3.19 template<typename float_type = double> interpolating_function_p<float_type>& interpolating_function_p< float_type >::unary_operator (const c2_function< float_type > & *source*) const

create a new interpolating_function_p which is the *source* function applied to every point in the interpolating tables

This carefully manages the derivative of the composed function at the two ends.

Parameters

source	the function to apply

Returns

a new interpolating_function_p with the same mappings for x and y

6.110.3.20 template<typename float_type = double> virtual float_type interpolating_function_p< float_type >::value_with_derivatives (float_type *x*, float_type ∗ *yprime*, float_type ∗ *yprime2*) const throw c2_exception) [virtual]

get the value and derivatives.

There is required checking for null pointers on the derivatives, and most implementations should operate faster if derivatives are not needed.

Parameters

in	*x*	the point at which to evaluate the function
out	*yprime*	the first derivative (if pointer is non-null)
out	*yprime2*	the second derivative (if pointer is non-null)

Returns

the value of the function

Implements c2_function< float_type >.

6.110.4 Member Data Documentation

6.110.4.1 template<typename float_type = double> std::vector<float_type> interpolating_function_p< float_type >::F [protected]

6.110.4.2 template<typename float_type = double> const c2_function_transformation<float_type>& interpolating_function_p< float_type >::fTransform

6.110.4.3 template< typename float_type = double > size_t interpolating_function_p< float_type >::lastKLow [mutable], [protected]

6.110.4.4 template< typename float_type = double > c2_const_ptr<float_type> interpolating_function_p< float_type >::sampler_function [protected]

6.110.4.5 template< typename float_type = double > std::vector<float_type> interpolating_function_p< float_type >::X [protected]

6.110.4.6 template< typename float_type = double > bool interpolating_function_p< float_type >::xInverted [protected]

6.110.4.7 template< typename float_type = double > std::vector<float_type> interpolating_function_p< float_type >::Xraw [protected]

6.110.4.8 template< typename float_type = double > std::vector<float_type> interpolating_function_p< float_type >::y2 [protected]

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

- c2_function.hh

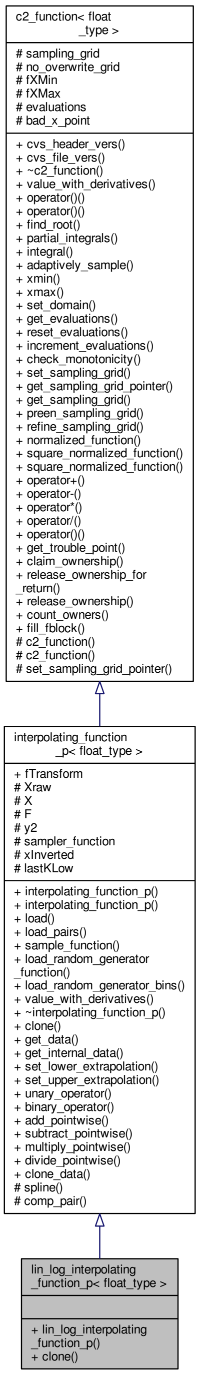
6.111 lin_log_interpolating_function_p< float_type > Class Template Reference

A spline with Y transformed into log space.

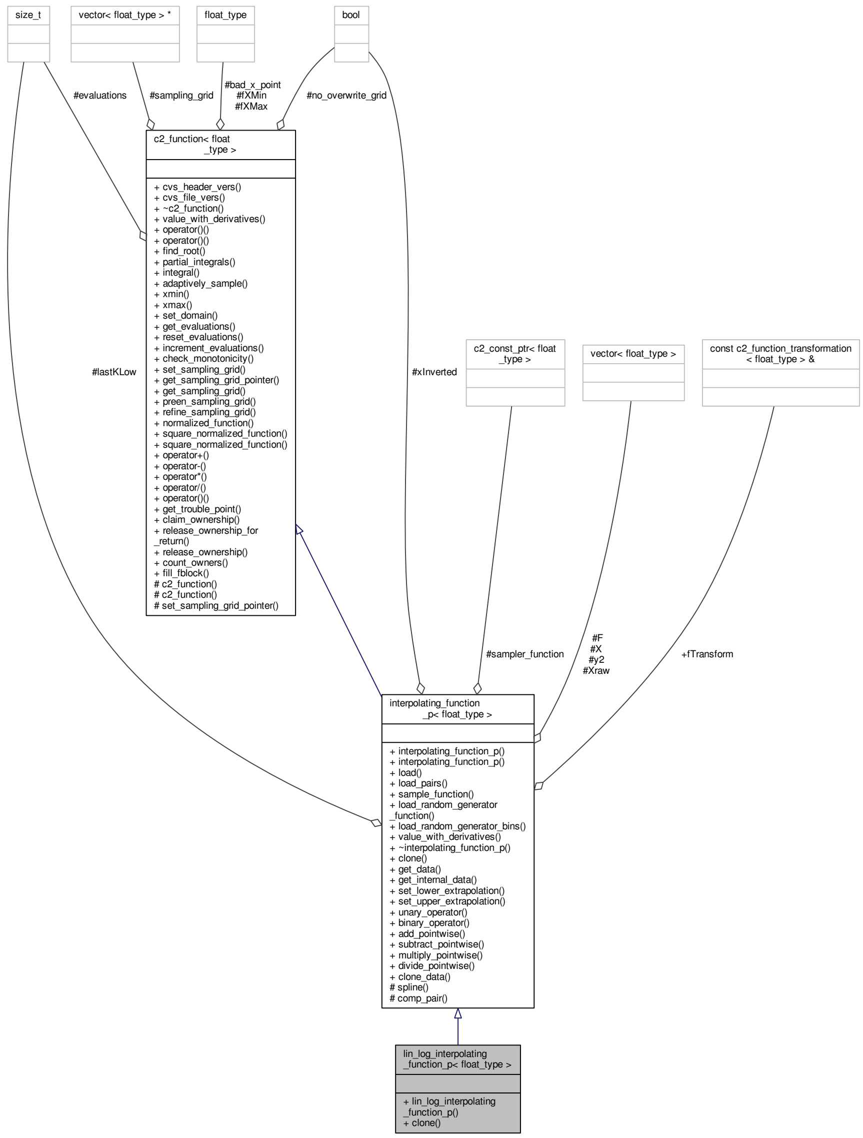
Most useful for functions looking like y=exp(x)

```
#include "c2_function.hh"
```

Inheritance diagram for lin_log_interpolating_function_p< float_type >:



Collaboration diagram for lin_log_interpolating_function_p< float_type >:



Public Member Functions

- lin_log_interpolating_function_p ()

    an empty linear-log cubic-spline interpolating_function_p
- virtual interpolating_function_p< float_type > & clone () const throw (c2_exception)

    create a new, empty interpolating function of this type (virtual constructor)

Additional Inherited Members

6.111.1 Detailed Description

template<typename float_type = double>
class lin_log_interpolating_function_p< float_type >

A spline with Y transformed into log space.

Most useful for functions looking like y=exp(x)

The factory function c2_factory::lin_log_interpolating_function() creates *new lin_log_interpolating_function_p()

6.111.2 Constructor & Destructor Documentation

6.111.2.1 template<typename float_type = double> lin_log_interpolating_function_p< float_type >::lin_log_interpolating_function_p ( ) [inline]

an empty linear-log cubic-spline interpolating_function_p

6.111.3 Member Function Documentation

6.111.3.1 template<typename float_type = double> virtual interpolating_function_p<float_type>& lin_log_interpolating_function_p< float_type >::clone ( ) const throw c2_exception) [inline], [virtual]

create a new, empty interpolating function of this type (virtual constructor)

Reimplemented from interpolating_function_p< float_type >.

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

- c2_function.hh

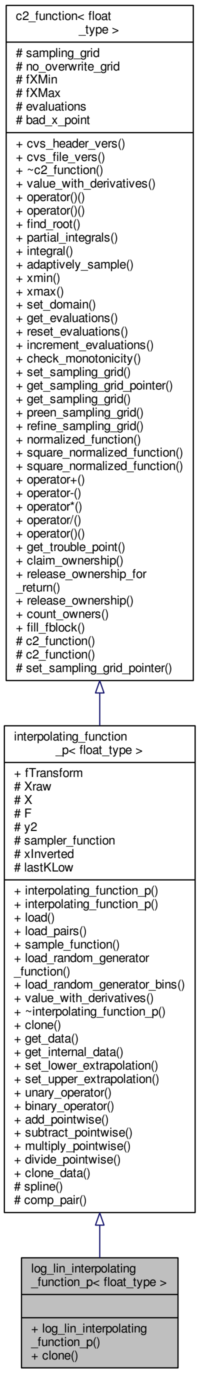
6.112 log_lin_interpolating_function_p< float_type > Class Template Reference

A spline with X transformed into log space.

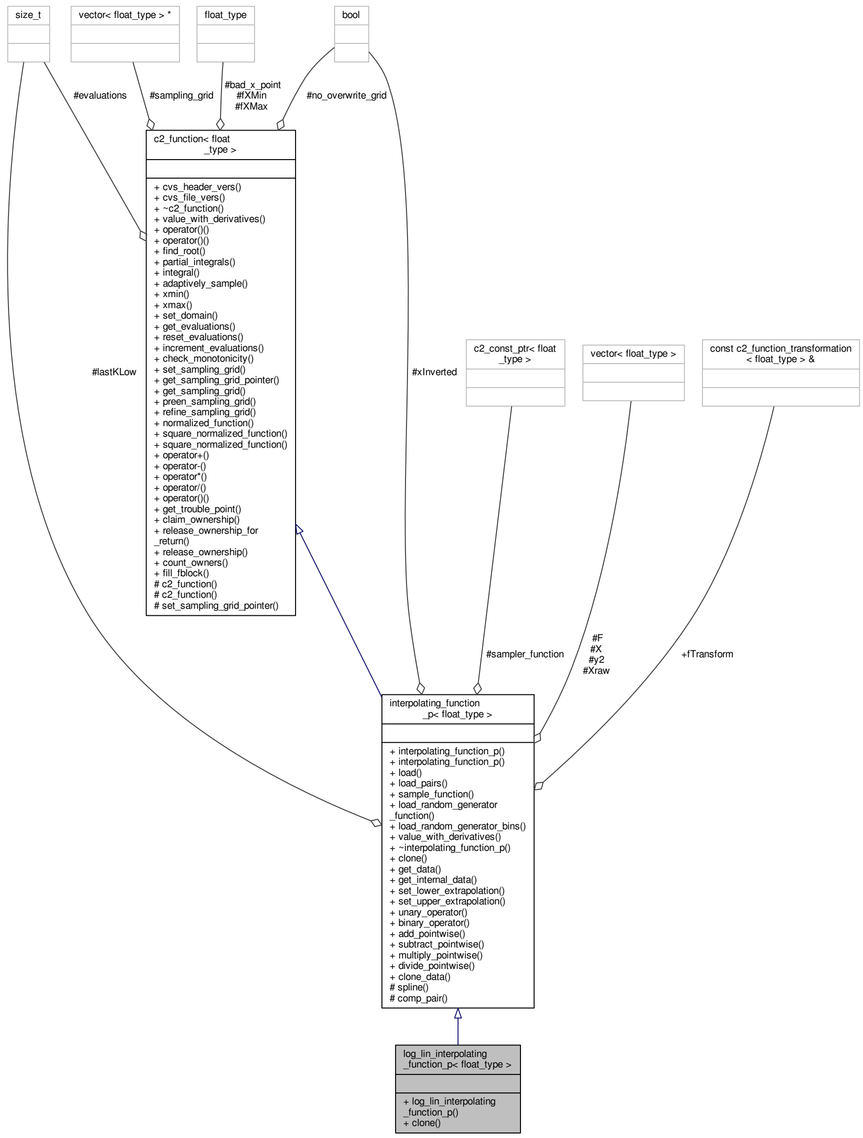
Most useful for functions looking like y=log(x) or any other function with a huge X dynamic range, and a slowly varying Y.

```
#include "c2_function.hh"
```

Inheritance diagram for log_lin_interpolating_function_p< float_type >:



Collaboration diagram for log_lin_interpolating_function_p< float_type >:



Public Member Functions

- log_lin_interpolating_function_p ()

  an empty log-linear cubic-spline interpolating_function_p
- virtual interpolating_function_p< float_type > & clone () const throw (c2_exception)

  create a new, empty interpolating function of this type (virtual constructor)

Additional Inherited Members

6.112.1 Detailed Description

template< typename float_type = double>
class log_lin_interpolating_function_p< float_type >

A spline with X transformed into log space.

Most useful for functions looking like y=log(x) or any other function with a huge X dynamic range, and a slowly varying Y.

The factory function c2_factory::log_lin_interpolating_function() creates *new log_lin_interpolating_function_p()

6.112.2 Constructor & Destructor Documentation

6.112.2.1 template<typename float_type = double> log_lin_interpolating_function_p< float_type >::log_lin_interpolating_function_p ( ) [inline]

an empty log-linear cubic-spline interpolating_function_p

6.112.3 Member Function Documentation

6.112.3.1 template<typename float_type = double> virtual interpolating_function_p<float_type>& log_lin_interpolating_function_p< float_type >::clone ( ) const throw c2_exception) [inline], [virtual]

create a new, empty interpolating function of this type (virtual constructor)

Reimplemented from interpolating_function_p< float_type >.

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

- c2_function.hh

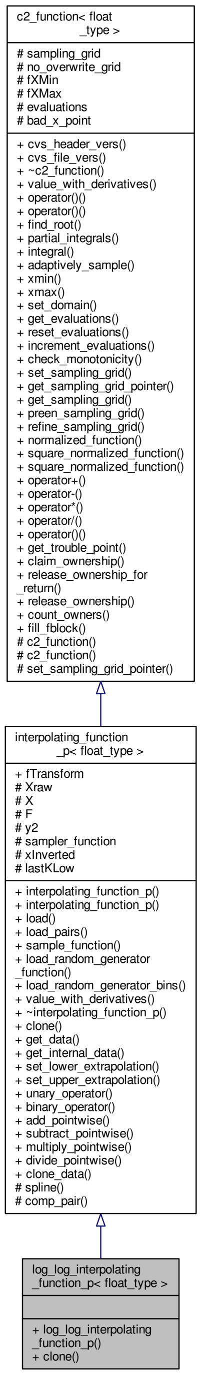
6.113 log_log_interpolating_function_p< float_type > Class Template Reference

A spline with X and Y transformed into log space.

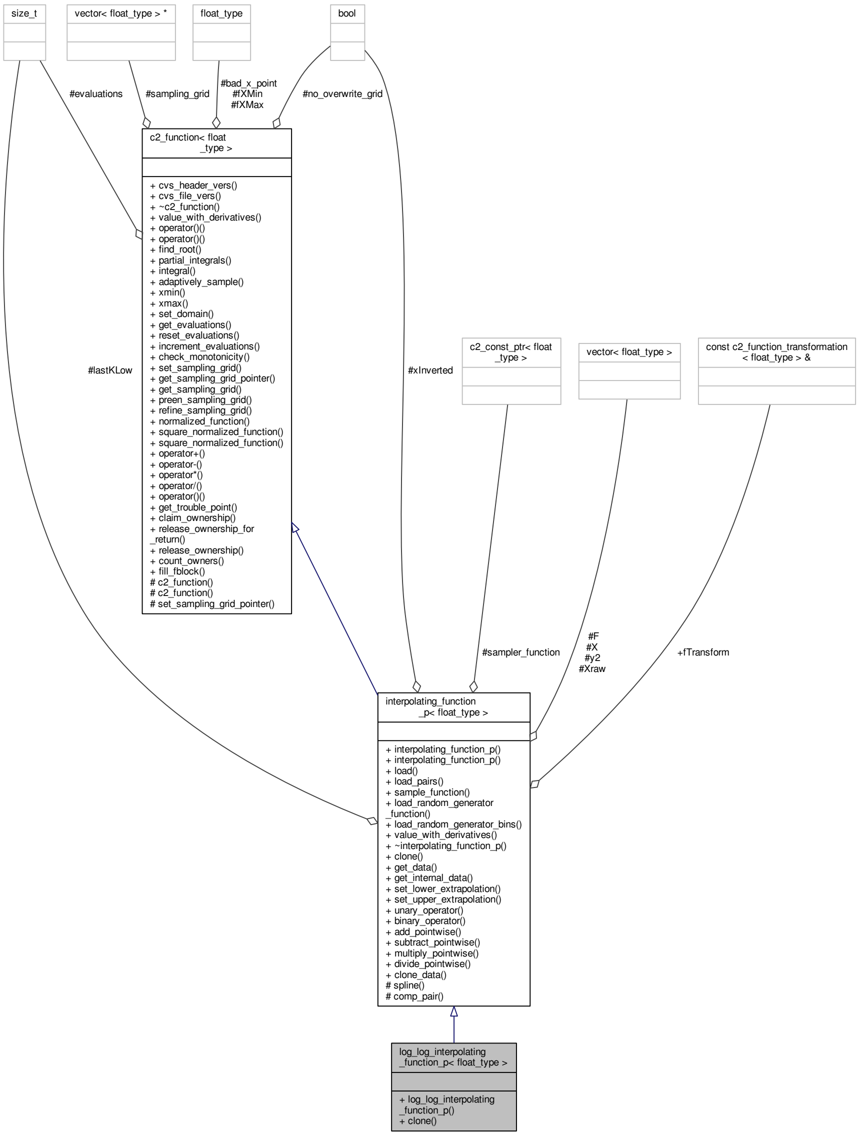
Most useful for functions looking like y=x^n or any other function with a huge X and Y dynamic range.

```
#include "c2_function.hh"
```

Inheritance diagram for log_log_interpolating_function_p< float_type >:



Collaboration diagram for log_log_interpolating_function_p< float_type >:



Public Member Functions

- log_log_interpolating_function_p ()

  an empty log-log cubic-spline interpolating_function_p
- virtual interpolating_function_p< float_type > & clone () const throw (c2_exception)

  create a new, empty interpolating function of this type (virtual constructor)

Additional Inherited Members

6.113.1 Detailed Description

template< typename float_type = double>
class log_log_interpolating_function_p< float_type >

A spline with X and Y transformed into log space.

Most useful for functions looking like y=x^n or any other function with a huge X and Y dynamic range.

The factory function c2_factory::log_log_interpolating_function() creates *new log_log_interpolating_function_p()

6.113.2 Constructor & Destructor Documentation

6.113.2.1 template< typename float_type = double> log_log_interpolating_function_p< float_type >::log_log_interpolating_function_p ( ) [inline]

an empty log-log cubic-spline interpolating_function_p

6.113.3 Member Function Documentation

6.113.3.1 template< typename float_type = double> virtual interpolating_function_p<float_type>& log_log_interpolating_function_p< float_type >::clone ( ) const throw c2_exception) [inline], [virtual]

create a new, empty interpolating function of this type (virtual constructor)

Reimplemented from interpolating_function_p< float_type >.

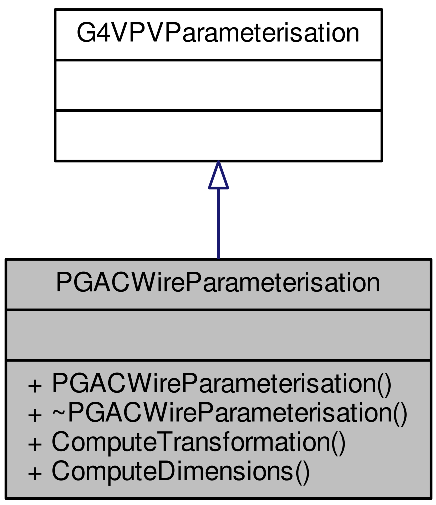
The documentation for this class was generated from the following file:

- c2_function.hh

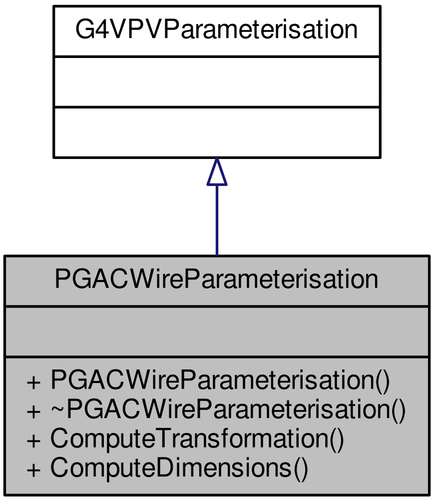
6.114 PGACWireParameterisation Class Reference

```
#include "PGACWireParameterisation.hh"
```

Inheritance diagram for PGACWireParameterisation:



Collaboration diagram for PGACWireParameterisation:



Public Member Functions

- PGACWireParameterisation (G4int noChambers, G4double startZ, G4double spacing, G4double widthChamber, G4double lengthInitial, G4double lengthFinal)
- virtual ∼PGACWireParameterisation ()
- void ComputeTransformation (const G4int copyNo, G4VPhysicalVolume ∗physVol) const
- void ComputeDimensions (G4Tubs &trackerLayer, const G4int copyNo, const G4VPhysicalVolume ∗physVol) const

6.114.1 Detailed Description

A parameterisation that describes a series of cylinders along Z.

The cylinders have equal width, & their lengths are a linear equation. They are spaced an equal distance apart, starting from given location.

6.114.2 Constructor & Destructor Documentation

6.114.2.1 PGACWireParameterisation::PGACWireParameterisation (G4int *noChambers,* G4double *startZ,* G4double *spacing,* G4double *widthChamber,* G4double *lengthInitial,* G4double *lengthFinal*)

6.114.2.2 virtual PGACWireParameterisation::∼PGACWireParameterisation ( ) [virtual]

6.114.3 Member Function Documentation

6.114.3.1 void PGACWireParameterisation::ComputeDimensions (G4Tubs & *trackerLayer,* const G4int *copyNo,* const G4VPhysicalVolume ∗ *physVol*) const

6.114.3.2 void PGACWireParameterisation::ComputeTransformation (const G4int *copyNo,* G4VPhysicalVolume ∗ *physVol*) const

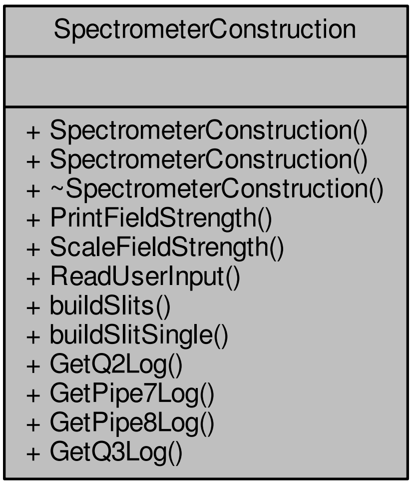
The documentation for this class was generated from the following file:

- PGACWireParameterisation.hh

6.115 SpectrometerConstruction Class Reference

`#include "SpectrometerConstruction.hh"`

Collaboration diagram for SpectrometerConstruction:



Public Member Functions

- SpectrometerConstruction ()
- SpectrometerConstruction (G4Material ∗, G4Material ∗, G4LogicalVolume ∗, G4double Pipe1length)
- virtual ∼SpectrometerConstruction ()
- void PrintFieldStrength ()
- void ScaleFieldStrength (G4double msf, G4double esf)
- void ReadUserInput ()
- void buildSlits (G4LogicalVolume ∗SpecWorldLogical, G4String nameSolid, G4String nameLogical, G4String namePhys, G4double slitsThick, G4double angle, G4ThreeVector pos, G4bool insert_hSlits, G4double h←Aper)
- void buildSlitSingle (G4LogicalVolume ∗SpecWorldLogical, G4String nameSolid, G4String nameLogical, G4String namePhys, G4double slitsThick, G4ThreeVector pos, G4bool insert_rSlits, G4double rAper, G4bool insert_lSlits, G4double lAper)
- G4LogicalVolume ∗ GetQ2Log ()
- G4LogicalVolume ∗ GetPipe7Log ()
- G4LogicalVolume ∗ GetPipe8Log ()
- G4LogicalVolume ∗ GetQ3Log ()

6.115.1 Constructor & Destructor Documentation

6.115.1.1 SpectrometerConstruction::SpectrometerConstruction ( )

6.115.1.2 SpectrometerConstruction::SpectrometerConstruction (G4Material ∗ , G4Material ∗ , G4LogicalVolume ∗ , G4double *Pipe1length*)

6.115.1.3 virtual SpectrometerConstruction::∼SpectrometerConstruction ( ) [virtual]

6.115.2 Member Function Documentation

6.115.2.1 void SpectrometerConstruction::buildSlits (G4LogicalVolume ∗ *SpecWorldLogical,* G4String *nameSolid,* G4String *nameLogical,* G4String *namePhys,* G4double *slitsThick,* G4double *angle,* G4ThreeVector *pos,* G4bool *insert_hSlits,* G4double *hAper*)

6.115.2.2 void SpectrometerConstruction::buildSlitSingle (G4LogicalVolume ∗ *SpecWorldLogical,* G4String *nameSolid,* G4String *nameLogical,* G4String *namePhys,* G4double *slitsThick,* G4ThreeVector *pos,* G4bool *insert_rSlits,* G4double *rAper,* G4bool *insert_lSlits,* G4double *lAper*)

6.115.2.3 G4LogicalVolume∗ SpectrometerConstruction::GetPipe7Log ( ) [inline]

6.115.2.4 G4LogicalVolume∗ SpectrometerConstruction::GetPipe8Log ( ) [inline]

6.115.2.5 G4LogicalVolume∗ SpectrometerConstruction::GetQ2Log ( ) [inline]

6.115.2.6 G4LogicalVolume∗ SpectrometerConstruction::GetQ3Log ( ) [inline]

6.115.2.7 void SpectrometerConstruction::PrintFieldStrength ( )

6.115.2.8 void SpectrometerConstruction::ReadUserInput ( )

6.115.2.9 void SpectrometerConstruction::ScaleFieldStrength (G4double *msf,* G4double *esf*)

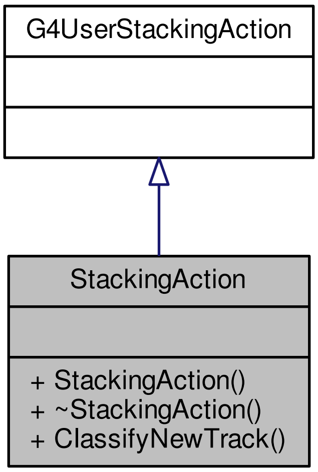
The documentation for this class was generated from the following file:

- SpectrometerConstruction.hh

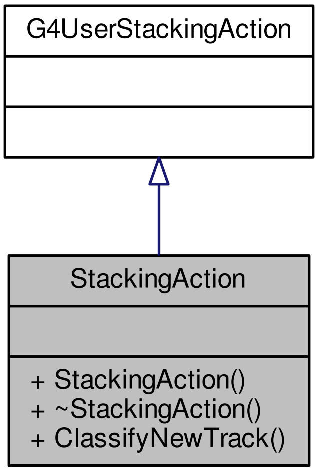
6.116 StackingAction Class Reference

```
#include "StackingAction.hh"
```

Inheritance diagram for StackingAction:



Collaboration diagram for StackingAction:



Public Member Functions

- StackingAction ()
- ∼StackingAction ()
- G4ClassificationOfNewTrack ClassifyNewTrack (const G4Track ∗aTrack)

6.116.1 Constructor & Destructor Documentation

6.116.1.1 StackingAction::StackingAction ( )

6.116.1.2 StackingAction::∼StackingAction ( )

6.116.2 Member Function Documentation

6.116.2.1 G4ClassificationOfNewTrack StackingAction::ClassifyNewTrack (const G4Track ∗ *aTrack*)

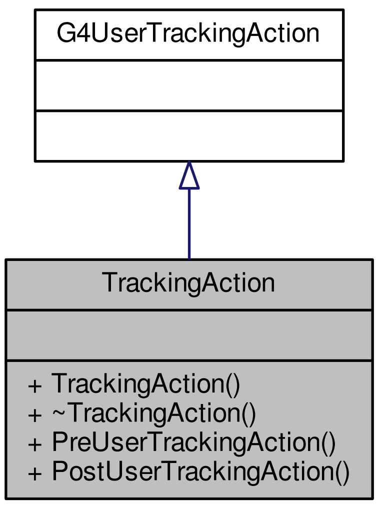
The documentation for this class was generated from the following file:

- StackingAction.hh

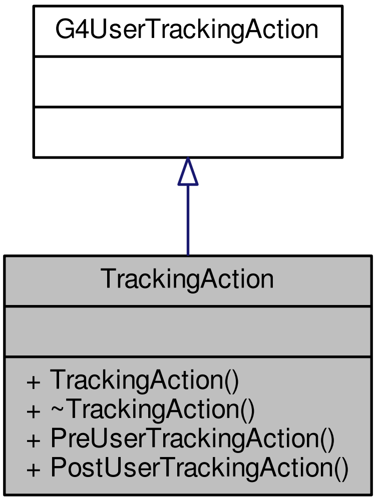
6.117 TrackingAction Class Reference

```
#include "TrackingAction.hh"
```

Inheritance diagram for TrackingAction:



Collaboration diagram for TrackingAction:



Public Member Functions

- TrackingAction ()
- ∼TrackingAction ()
- void PreUserTrackingAction (const G4Track ∗PreTrack)
- void PostUserTrackingAction (const G4Track ∗PostTrack)

6.117.1 Constructor & Destructor Documentation

6.117.1.1 TrackingAction::TrackingAction ( )

6.117.1.2 TrackingAction::∼TrackingAction ( ) [inline]

6.117.2 Member Function Documentation

6.117.2.1 void TrackingAction::PostUserTrackingAction (const G4Track ∗ *PostTrack*)

6.117.2.2 void TrackingAction::PreUserTrackingAction (const G4Track ∗ *PreTrack*)

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

- TrackingAction.hh

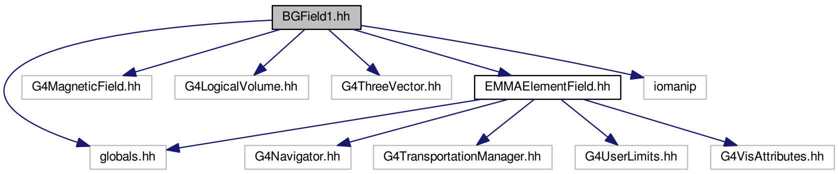
Chapter 7

File Documentation

7.1 BGField1.hh File Reference

```
#include "globals.hh"
#include "G4MagneticField.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "EMMAElementField.hh"
#include <iomanip>
```

Include dependency graph for BGField1.hh:



Classes

- class BGField1

Variables

- G4double currentCharge
- G4double userCharge

7.1.1 Variable Documentation

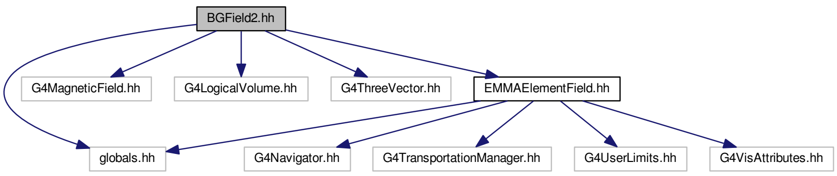
7.1.1.1 G4double currentCharge

7.1.1.2 G4double userCharge

7.2 BGField2.hh File Reference

```
#include "globals.hh"
#include "G4MagneticField.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "EMMAElementField.hh"
```

Include dependency graph for BGField2.hh:



Classes

- class BGField2

Variables

- G4double currentCharge
- G4double userCharge

7.2.1 Variable Documentation

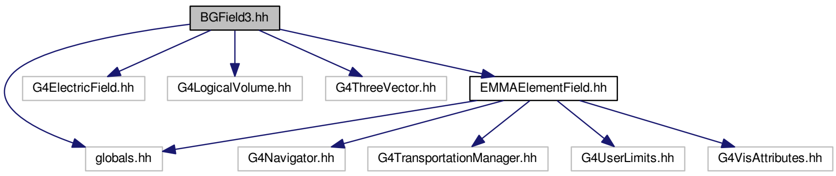
7.2.1.1 G4double currentCharge

7.2.1.2 G4double userCharge

7.3 BGField3.hh File Reference

```
#include "globals.hh"
#include "G4ElectricField.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "EMMAElementField.hh"
```

Include dependency graph for BGField3.hh:



Classes

- class BGField3

Variables

- G4double currentCharge
- G4double userCharge

7.3.1 Variable Documentation

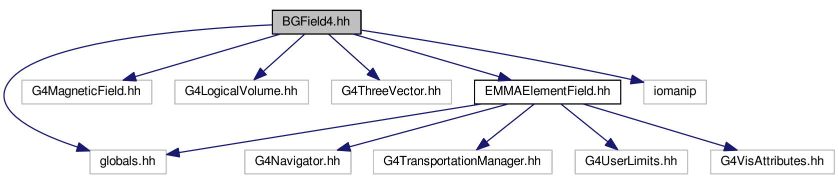
7.3.1.1 G4double currentCharge

7.3.1.2 G4double userCharge

7.4 BGField4.hh File Reference

```
#include "globals.hh"
#include "G4MagneticField.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "EMMAElementField.hh"
#include <iomanip>
```

Include dependency graph for BGField4.hh:



Classes

- class BGField4

Variables

- G4double currentCharge
- G4double userCharge

7.4.1 Variable Documentation

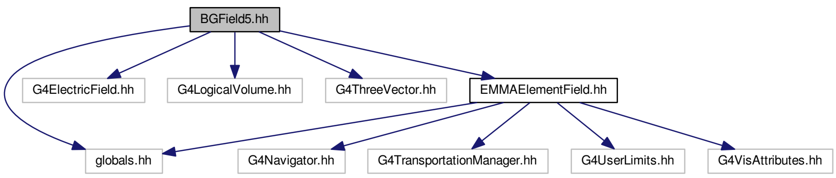
7.4.1.1 G4double currentCharge

7.4.1.2 G4double userCharge

7.5 BGField5.hh File Reference

```
#include "globals.hh"
#include "G4ElectricField.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "EMMAElementField.hh"
```

Include dependency graph for BGField5.hh:



Classes

- class BGField5

Variables

- G4double currentCharge
- G4double userCharge

7.5.1 Variable Documentation

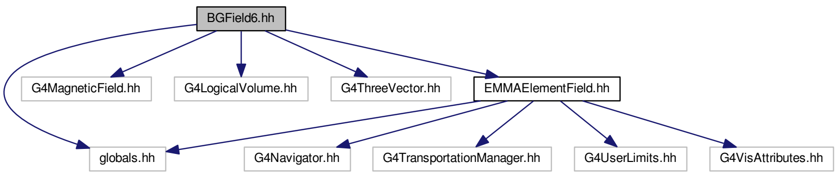
7.5.1.1 G4double currentCharge

7.5.1.2 G4double userCharge

7.6 BGField6.hh File Reference

```
#include "globals.hh"
#include "G4MagneticField.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "EMMAElementField.hh"
```

Include dependency graph for BGField6.hh:



Classes

- class BGField6

Variables

- G4double currentCharge
- G4double userCharge

7.6.1 Variable Documentation

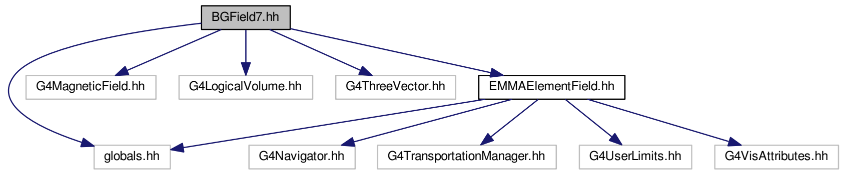
7.6.1.1 G4double currentCharge

7.6.1.2 G4double userCharge

7.7 BGField7.hh File Reference

```
#include "globals.hh"
#include "G4MagneticField.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "EMMAElementField.hh"
```

Include dependency graph for BGField7.hh:



Classes

- class BGField7

Variables

- G4double currentCharge
- G4double userCharge

7.7.1 Variable Documentation

7.7.1.1 G4double currentCharge

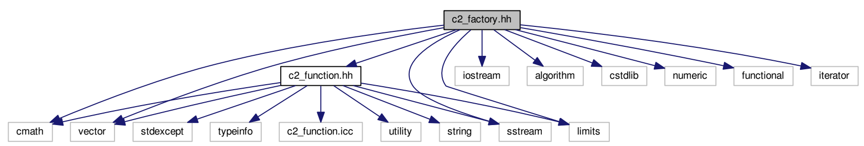
7.7.1.2 G4double userCharge

7.8 c2_factory.hh File Reference

Provides a factory class to avoid an infinite number of template declarations.

```
#include "c2_function.hh"
```

Include dependency graph for c2_factory.hh:



Classes

- class c2_factory< float_type >
  a factory of pre-templated c2_function generators

7.8.1 Detailed Description

Provides a factory class to avoid an infinite number of template declarations.

Author

Created by R. A. Weller and Marcus H. Mendenhall on 7/9/05.
2005 Vanderbilt University.

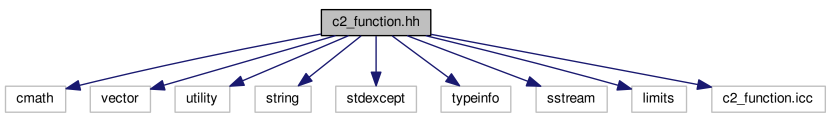
```
\version c2_factory.hh,v 1.13 2008/05/22 12:45:19 marcus Exp
```

7.9 c2_function.hh File Reference

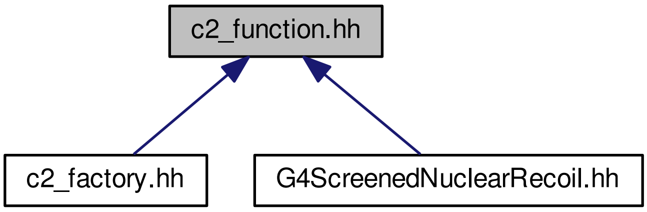
Provides the headers for the general c2_function algebra which supports fast, flexible operations on piecewise-twice-differentiable functions.

```
#include <cmath>
#include <vector>
#include <utility>
#include <string>
#include <stdexcept>
#include <typeinfo>
#include <sstream>
#include <limits>
#include "c2_function.icc"
```

Include dependency graph for c2_function.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class c2_exception

    the exception class for c2_function operations.
- class c2_composed_function_p< float_type >

    Provides function composition (nesting)
    This allows evaluation of f(g(x)) where f and g are c2_function objects.
- class c2_sum_p< float_type >

    create a c2_function which is the sum of two other c2_function objects.
    This should always be constructed using c2_function::operator+()
- class c2_diff_p< float_type >

    create a c2_function which is the difference of two other c2_functions.
    This should always be constructed using c2_function::operator-()
- class c2_product_p< float_type >

    create a c2_function which is the product of two other c2_functions.
    This should always be constructed using c2_function::operator()*
- class c2_ratio_p< float_type >

    create a c2_function which is the ratio of two other c2_functions.
    This should always be constructed using c2_function::operator/()
- class c2_piecewise_function_p< float_type >

    create a c2_function which is a piecewise assembly of other c2_functions.
    The functions must have increasing, non-overlapping domains. Any empty space between functions will be filled with a linear interpolation.
- class c2_quadratic_p< float_type >

    create a quadratic mapping of another function
    for example, given a c2_function f
- class c2_ptr< float_type >

    create a container for a c2_function which handles the reference counting.
- class c2_fblock< float_type >

    structure used to hold evaluated function data at a point.
- class c2_function< float_type >

    the parent class for all c2_functions.
    c2_functions know their value, first, and second derivative at almost every point. They can be efficiently combined with binary operators, via c2_binary_function, composed via c2_composed_function_, have their roots found via find_root(), and be adaptively integrated via partial_integrals() or integral(). They also can carry information with them about how to find 'interesting' points on the function. This information is set with set_sampling_grid() and extracted with get_sampling_grid().
- class c2_classic_function_p< float_type >

    a container into which any conventional c-style function can be dropped, to create a degenerate c2_function without derivatives. Mostly useful for sampling into interpolating functions. construct a reference to this with c2_classic_function()
    *The factory function c2_factory::classic_function() creates *new c2_classic_function_p()*
- class c2_const_ptr< float_type >

    create a container for a c2_function which handles the reference counting.
    It is useful as a smart container to hold a c2_function and keep the reference count correct. The recommended way for a class to store a c2_function which is handed in from the outside is for it to have a c2_ptr member into which the passed-in function is stored. This way, when the class instance is deleted, it will automatically dereference any function which it was handed.
- class c2_ptr< float_type >

    create a container for a c2_function which handles the reference counting.
- class c2_typed_ptr< float_type, c2_class >

    create a non-generic container for a c2_function which handles the reference counting.
- class c2_plugin_function_p< float_type >

    a container into which any other c2_function can be dropped, to allow expressions with replacable components.
    It is useful for plugging different InterpolatingFunctions into a c2_function expression. It saves a lot of effort in other places with casting away const declarations.

- class c2_const_plugin_function_p< float_type >

  a c2_plugin_function_p which promises not to fiddle with the plugged function.
  The factory function c2_factory::const_plugin_function() creates ∗new c2_const_plugin_function_p()
- class c2_binary_function< float_type >

  Provides support for c2_function objects which are constructed from two other c2_function objects.
- class c2_scaled_function_p< float_type >

  *Create a very lightweight method to return a scalar multiple of another function. \ *
  The factory function c2_factory::scaled_function() creates ∗new c2_scaled_function_p.
- class c2_cached_function_p< float_type >

  A container into which any other c2_function can be dropped.
  It allows a function to be pre-evaluated at a point, and used at multiple places in an expression efficiently. If it is re-evaluated at the previous point, it returns the remembered values; otherwise, it re-evauates the function at the new point.
- class c2_composed_function_p< float_type >

  Provides function composition (nesting)
  This allows evaluation of f(g(x)) where f and g are c2_function objects.
- class c2_sum_p< float_type >

  create a c2_function which is the sum of two other c2_function objects.
  This should always be constructed using c2_function::operator+()
- class c2_diff_p< float_type >

  create a c2_function which is the difference of two other c2_functions.
  This should always be constructed using c2_function::operator-()
- class c2_product_p< float_type >

  create a c2_function which is the product of two other c2_functions.
  This should always be constructed using c2_function::operator∗()
- class c2_ratio_p< float_type >

  create a c2_function which is the ratio of two other c2_functions.
  This should always be constructed using c2_function::operator/()
- class c2_constant_p< float_type >

  a c2_function which is constant
  The factory function c2_factory::constant() creates ∗new c2_constant_p()
- class c2_transformation< float_type >

  a transformation of a coordinate, including an inverse
- class c2_transformation_linear< float_type >

  the identity transform
- class c2_transformation_log< float_type >

  log axis transform
- class c2_transformation_recip< float_type >

  reciprocal axis transform
- class c2_function_transformation< float_type >

  a transformation of a function in and out of a coordinate space, using 2 c2_transformations
- class c2_lin_lin_function_transformation< float_type >

  a transformation of a function in and out of lin-lin space
- class c2_log_log_function_transformation< float_type >

  a transformation of a function in and out of log-log space
- class c2_lin_log_function_transformation< float_type >

  a transformation of a function in and out of lin-log space
- class c2_log_lin_function_transformation< float_type >

  a transformation of a function in and out of log-lin space
- class c2_arrhenius_function_transformation< float_type >

  a transformation of a function in and out of Arrhenius (1/x vs. log(y)) space
- class interpolating_function_p< float_type >

create a cubic spline interpolation of a set of (x,y) pairs
This is one of the main reasons for c2_function objects to exist.

- class log_lin_interpolating_function_p< float_type >

  A spline with X transformed into log space.
  Most useful for functions looking like y=log(x) or any other function with a huge X dynamic range, and a slowly varying Y.

- class lin_log_interpolating_function_p< float_type >

  A spline with Y transformed into log space.
  Most useful for functions looking like y=exp(x)

- class log_log_interpolating_function_p< float_type >

  A spline with X and Y transformed into log space.
  Most useful for functions looking like y=x^n or any other function with a huge X and Y dynamic range.

- class arrhenius_interpolating_function_p< float_type >

  A spline with X in reciprocal space and Y transformed in log space.
  *Most useful for thermodynamic types of data where Y is roughly A*exp(-B/x). Typical examples are reaction rate data, and thermistor calibration data.*

- class c2_sin_p< float_type >

  compute sin(x) with its derivatives.
  *The factory function c2_factory::sin() creates *new c2_sin_p*

- class c2_cos_p< float_type >

  compute cos(x) with its derivatives.
  *The factory function c2_factory::cos() creates *new c2_cos_p*

- class c2_tan_p< float_type >

  compute tan(x) with its derivatives.
  *The factory function c2_factory::tan() creates *new c2_tan_p*

- class c2_log_p< float_type >

  compute log(x) with its derivatives.
  *The factory function c2_factory::log() creates *new c2_log_p*

- class c2_exp_p< float_type >

  compute exp(x) with its derivatives.
  *The factory function c2_factory::exp() creates *new c2_exp_p*

- class c2_sqrt_p< float_type >

  compute sqrt(x) with its derivatives.
  *The factory function c2_factory::sqrt() creates *new c2_sqrt_p()*

- class c2_recip_p< float_type >

  compute scale/x with its derivatives.
  *The factory function c2_factory::recip() creates *new c2_recip_p*

- class c2_identity_p< float_type >

  compute x with its derivatives.
  *The factory function c2_factory::identity() creates *new c2_identity_p*

- class c2_linear_p< float_type >

  create a linear mapping of another function
  for example, given a c2_function f

- class c2_quadratic_p< float_type >

  create a quadratic mapping of another function
  for example, given a c2_function f

- class c2_power_law_p< float_type >

  create a power law mapping of another function
  for example, given a c2_function f

- class c2_inverse_function_p< float_type >

  create the formal inverse function of another function
  for example, given a c2_function f

- class accumulated_histogram< float_type >

An interpolating_function_p which is the cumulative integral of a histogram.
Note than binedges should be one element longer than binheights, since the lower & upper edges are specified. Note that this is a malformed spline, since the second derivatives are all zero, so it has less continuity. Also, note that the bin edges can be given in backwards order to generate the reversed accumulation (starting at the high end)

- class c2_connector_function_p< float_type >

  create a c2_function which smoothly connects two other c2_functions.
  This takes two points and generates a polynomial which matches two c2_function arguments at those two points, with two derivatives at each point, and an arbitrary value at the center of the region. It is useful for splicing together functions over rough spots (0/0, for example).

- class c2_piecewise_function_p< float_type >

  create a c2_function which is a piecewise assembly of other c2_functions.
  The functions must have increasing, non-overlapping domains. Any empty space between functions will be filled with a linear interpolation.

Macros

- #define c2_isnan std::isnan
- #define c2_isfinite std::isfinite

7.9.1 Detailed Description

Provides the headers for the general c2_function algebra which supports fast, flexible operations on piecewise-twice-differentiable functions.

Author

Created by R. A. Weller and Marcus H. Mendenhall on 7/9/05.


```
\version c2_function.hh 490 2012-04-10 19:05:40Z marcus
```

See also

Factory Functions for information on constructing things in here

7.9.2 Macro Definition Documentation

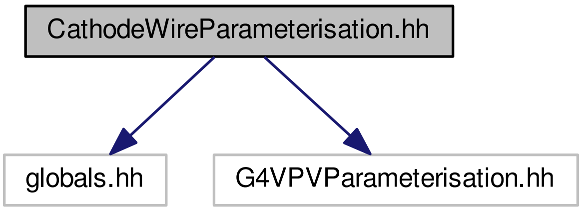
7.9.2.1 #define c2_isfinite std::isfinite

7.9.2.2 #define c2_isnan std::isnan

7.10 CathodeWireParameterisation.hh File Reference

```
#include "globals.hh"
#include "G4VPVParameterisation.hh"
```

Include dependency graph for CathodeWireParameterisation.hh:



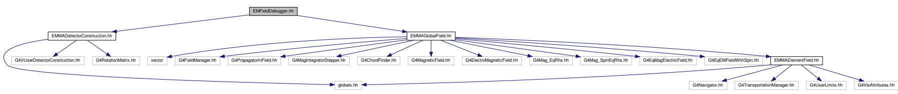
Classes

- class CathodeWireParameterisation

7.11 EMFieldDebugger.hh File Reference

```
#include "EMMADetectorConstruction.hh"
#include "EMMAGlobalField.hh"
```

Include dependency graph for EMFieldDebugger.hh:



Classes

- class EMFieldDebugger

Variables

- G4double zQ1ends
- G4double zQ2ends
- G4double zQ3ends
- G4double zQ4ends
- G4double zQ1begins
- G4double zQ2begins
- G4double zQ3begins

- G4double zQ4begins
- G4double zQ1fieldbegins
- G4double zQ2fieldbegins
- G4double zQ2fieldends
- G4double zQ3fieldbegins
- G4double zQ4fieldbegins
- G4double zQ4fieldends
- G4double zED1fieldbegins
- G4double xED1fieldends
- G4double zED1fieldends
- G4double xMDfieldbegins
- G4double zMDfieldbegins
- G4double xMDfieldends
- G4double zMDfieldends
- G4double xED2fieldbegins
- G4double zED2fieldbegins
- G4double zED2fieldends
- G4double xED1center
- G4double zED1center
- G4double xED2center
- G4double zED2center
- G4double rED
- G4double xMDcenter
- G4double zMDcenter
- G4double rMD
- G4double Q1before
- G4double Q2before
- G4double ED1before
- G4double MDbefore
- G4double ED2before
- G4double Q3before
- G4double Q4before
- G4double Q1after
- G4double Q2after
- G4double ED1after
- G4double MDafter
- G4double ED2after
- G4double Q3after
- G4double Q4after
- G4double Pipe4z
- G4double Pipe4HL
- G4double Pipe7HL
- G4double magneticScaling
- G4double electricScaling
- G4String fieldFileName

7.11.1 Variable Documentation

7.11.1.1 G4double ED1after

7.11.1.2 G4double ED1before

7.11.1.3 G4double ED2after

7.11.1.4 G4double ED2before

7.11.1.5 G4double electricScaling

7.11.1.6 G4String fieldFileName

7.11.1.7 G4double magneticScaling

7.11.1.8 G4double MDafter

7.11.1.9 G4double MDbefore

7.11.1.10 G4double Pipe4HL

7.11.1.11 G4double Pipe4z

7.11.1.12 G4double Pipe7HL

7.11.1.13 G4double Q1after

7.11.1.14 G4double Q1before

7.11.1.15 G4double Q2after

7.11.1.16 G4double Q2before

7.11.1.17 G4double Q3after

7.11.1.18 G4double Q3before

7.11.1.19 G4double Q4after

7.11.1.20 G4double Q4before

7.11.1.21 G4double rED

7.11.1.22 G4double rMD

7.11.1.23 G4double xED1center

7.11.1.24 G4double xED1fieldends

7.11.1.25 G4double xED2center

7.11.1.26 G4double xED2fieldbegins

7.11.1.27 G4double xMDcenter

7.11.1.28 G4double xMDfieldbegins

7.11.1.29 G4double xMDfieldends

7.11.1.30 G4double zED1center

7.11.1.31 G4double zED1fieldbegins

7.11.1.32 G4double zED1fieldends

7.11.1.33 G4double zED2center

7.11.1.34 G4double zED2fieldbegins

7.11.1.35 G4double zED2fieldends

7.11.1.36 G4double zMDcenter

7.11.1.37 G4double zMDfieldbegins

7.11.1.38 G4double zMDfieldends

7.11.1.39 G4double zQ1begins

7.11.1.40 G4double zQ1ends

7.11.1.41 G4double zQ1fieldbegins

7.11.1.42 G4double zQ2begins

7.11.1.43 G4double zQ2ends

7.11.1.44 G4double zQ2fieldbegins

7.11.1.45 G4double zQ2fieldends

7.11.1.46 G4double zQ3begins

7.11.1.47 G4double zQ3ends

7.11.1.48 G4double zQ3fieldbegins

7.11.1.49 G4double zQ4begins

7.11.1.50 G4double zQ4ends

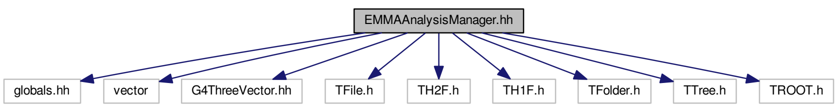
7.11.1.51 G4double zQ4fieldbegins

7.11.1.52 G4double zQ4fieldends

7.12 EMMAAnalysisManager.hh File Reference

```
#include "globals.hh"
#include <vector>
#include "G4ThreeVector.hh"
#include <TFile.h>
#include <TH2F.h>
#include <TH1F.h>
#include <TFolder.h>
#include <TTree.h>
#include <TROOT.h>
```

Include dependency graph for EMMAAnalysisManager.hh:



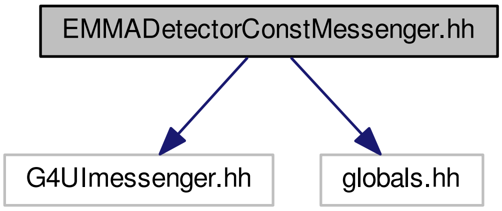
Classes

- class EMMAAnalysisManager

7.13 EMMADetectorConstMessenger.hh File Reference

```
#include "G4UImessenger.hh"
#include "globals.hh"
```

Include dependency graph for EMMADetectorConstMessenger.hh:



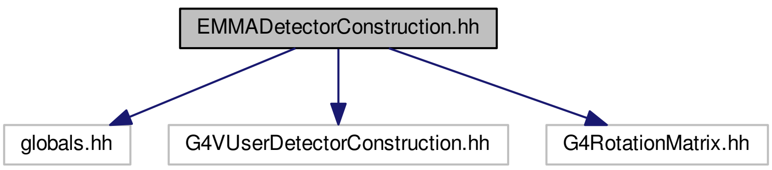
Classes

- class EMMADetectorConstMessenger

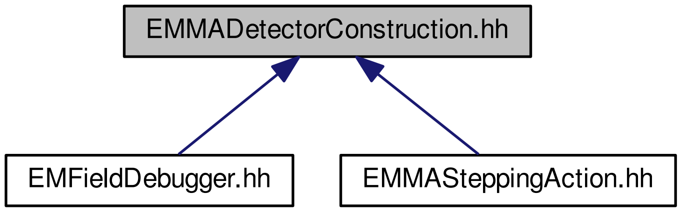
7.14 EMMADetectorConstruction.hh File Reference

```
#include "globals.hh"
#include "G4VUserDetectorConstruction.hh"
#include "G4RotationMatrix.hh"
```

Include dependency graph for EMMADetectorConstruction.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class EMMADetectorConstruction

Variables

- G4double zQ1begins
- G4double zQ4ends
- G4double zAnode
- G4double zFocalPlane
- G4String MotherDir
- G4String UserDir

7.14.1 Variable Documentation

7.14.1.1 G4String MotherDir

7.14.1.2 G4String UserDir

7.14.1.3 G4double zAnode

7.14.1.4 G4double zFocalPlane

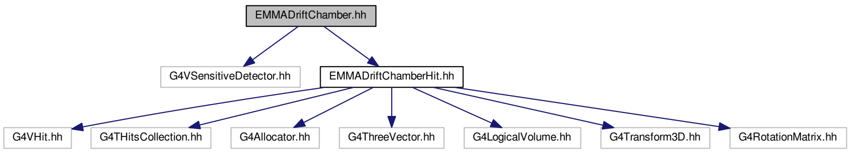
7.14.1.5 G4double zQ1begins

7.14.1.6 G4double zQ4ends

7.15 EMMADriftChamber.hh File Reference

```
#include "G4VSensitiveDetector.hh"
#include "EMMADriftChamberHit.hh"
```

Include dependency graph for EMMADriftChamber.hh:



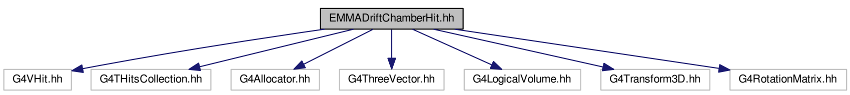
Classes

- class EMMADriftChamber

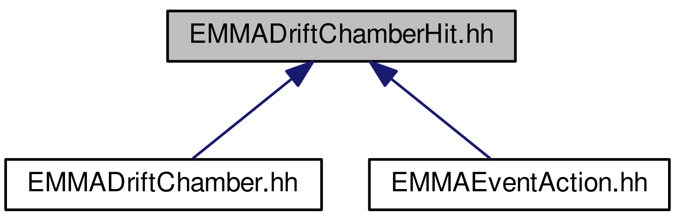
7.16 EMMADriftChamberHit.hh File Reference

```
#include "G4VHit.hh"
#include "G4THitsCollection.hh"
#include "G4Allocator.hh"
#include "G4ThreeVector.hh"
#include "G4LogicalVolume.hh"
#include "G4Transform3D.hh"
#include "G4RotationMatrix.hh"
```

Include dependency graph for EMMADriftChamberHit.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class EMMADriftChamberHit

Typedefs

- typedef G4THitsCollection< EMMADriftChamberHit > EMMADriftChamberHitsCollection

Variables

- G4String focalPlaneFileName
- G4String MotherDir
- G4Allocator< EMMADriftChamberHit > EMMADriftChamberHitAllocator

7.16.1 Typedef Documentation

7.16.1.1 typedef G4THitsCollection<**EMMADriftChamberHit**> **EMMADriftChamberHitsCollection**

7.16.2 Variable Documentation

7.16.2.1 G4Allocator<**EMMADriftChamberHit**> EMMADriftChamberHitAllocator

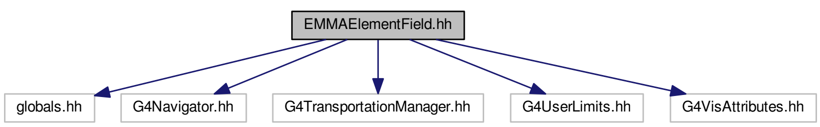
7.16.2.2 G4String focalPlaneFileName

7.16.2.3 G4String MotherDir

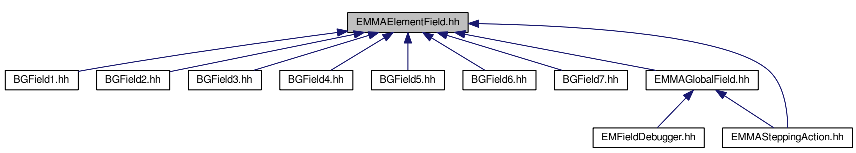
7.17 EMMAElementField.hh File Reference

```
#include "globals.hh"
#include "G4Navigator.hh"
#include "G4TransportationManager.hh"
#include "G4UserLimits.hh"
#include "G4VisAttributes.hh"
```

Include dependency graph for EMMAElementField.hh:



This graph shows which files directly or indirectly include this file:



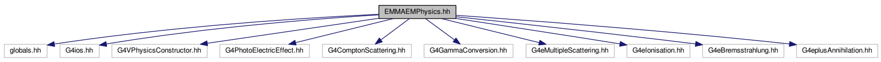
Classes

- class EMMAElementField

7.18 EMMAEMPhysics.hh File Reference

```
#include "globals.hh"
#include "G4ios.hh"
#include "G4VPhysicsConstructor.hh"
#include "G4PhotoElectricEffect.hh"
#include "G4ComptonScattering.hh"
#include "G4GammaConversion.hh"
#include "G4eMultipleScattering.hh"
#include "G4eIonisation.hh"
#include "G4eBremsstrahlung.hh"
#include "G4eplusAnnihilation.hh"
```

Include dependency graph for EMMAEMPhysics.hh:



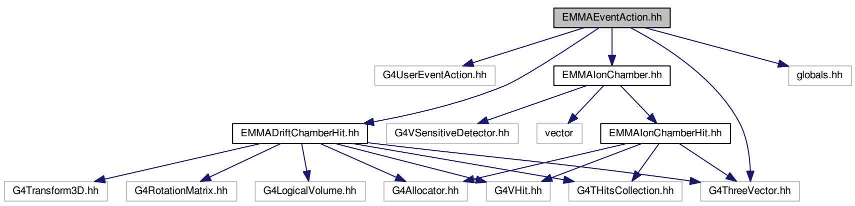
Classes

- class EMMAEMPhysics

7.19 EMMAEventAction.hh File Reference

```
#include "G4UserEventAction.hh"
#include "G4ThreeVector.hh"
#include "globals.hh"
#include "EMMADriftChamberHit.hh"
#include "EMMAIonChamber.hh"
```

Include dependency graph for EMMAEventAction.hh:



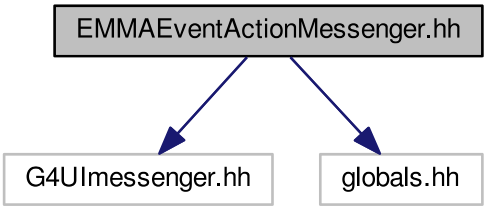
Classes

- class EMMAEventAction

7.20 EMMAEventActionMessenger.hh File Reference

```
#include "G4UImessenger.hh"
#include "globals.hh"
```

Include dependency graph for EMMAEventActionMessenger.hh:



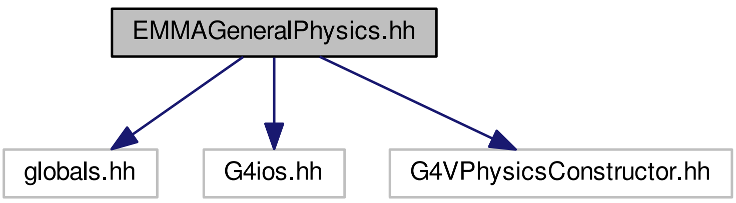
Classes

- class EMMAEventActionMessenger

7.21 EMMAGeneralPhysics.hh File Reference

```
#include "globals.hh"
#include "G4ios.hh"
#include "G4VPhysicsConstructor.hh"
```

Include dependency graph for EMMAGeneralPhysics.hh:



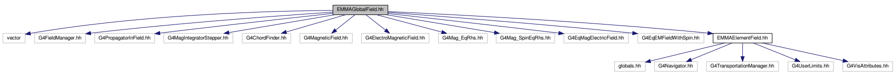
Classes

- class EMMAGeneralPhysics

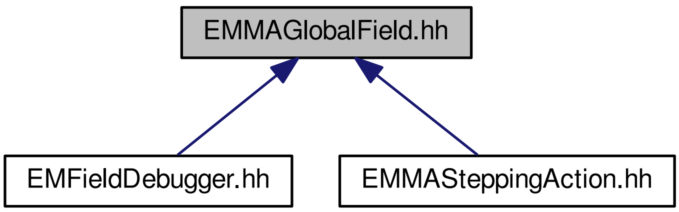
7.22 EMMAGlobalField.hh File Reference

```
#include <vector>
#include "G4FieldManager.hh"
#include "G4PropagatorInField.hh"
#include "G4MagIntegratorStepper.hh"
#include "G4ChordFinder.hh"
#include "G4MagneticField.hh"
#include "G4ElectroMagneticField.hh"
#include "G4Mag_EqRhs.hh"
#include "G4Mag_SpinEqRhs.hh"
#include "G4EqMagElectricField.hh"
#include "G4EqEMFieldWithSpin.hh"
#include "EMMAElementField.hh"
```

Include dependency graph for EMMAGlobalField.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class EMMAGlobalField

Typedefs

- typedef std::vector< EMMAElementField * > FieldList

7.22.1 Typedef Documentation

7.22.1.1 typedef std::vector<EMMAElementField*> FieldList

7.23 EMMAHadronPhysics.hh File Reference

```
#include "globals.hh"
```

```
#include "G4ios.hh"
#include "G4VPhysicsConstructor.hh"
#include "G4hMultipleScattering.hh"
#include "G4hIonisation.hh"
#include "G4HadronElasticProcess.hh"
#include "G4HadronFissionProcess.hh"
#include "G4HadronCaptureProcess.hh"
#include "G4PionPlusInelasticProcess.hh"
#include "G4PionMinusInelasticProcess.hh"
#include "G4KaonPlusInelasticProcess.hh"
#include "G4KaonZeroSInelasticProcess.hh"
#include "G4KaonZeroLInelasticProcess.hh"
#include "G4KaonMinusInelasticProcess.hh"
#include "G4ProtonInelasticProcess.hh"
#include "G4AntiProtonInelasticProcess.hh"
#include "G4NeutronInelasticProcess.hh"
#include "G4AntiNeutronInelasticProcess.hh"
#include "G4LambdaInelasticProcess.hh"
#include "G4AntiLambdaInelasticProcess.hh"
#include "G4SigmaPlusInelasticProcess.hh"
#include "G4SigmaMinusInelasticProcess.hh"
#include "G4AntiSigmaPlusInelasticProcess.hh"
#include "G4AntiSigmaMinusInelasticProcess.hh"
#include "G4XiZeroInelasticProcess.hh"
#include "G4XiMinusInelasticProcess.hh"
#include "G4AntiXiZeroInelasticProcess.hh"
#include "G4AntiXiMinusInelasticProcess.hh"
#include "G4DeuteronInelasticProcess.hh"
#include "G4TritonInelasticProcess.hh"
#include "G4AlphaInelasticProcess.hh"
#include "G4OmegaMinusInelasticProcess.hh"
#include "G4AntiOmegaMinusInelasticProcess.hh"
#include "G4LElastic.hh"
#include "G4LFission.hh"
#include "G4LCapture.hh"
#include "G4LEPionPlusInelastic.hh"
#include "G4LEPionMinusInelastic.hh"
#include "G4LEKaonPlusInelastic.hh"
#include "G4LEKaonZeroSInelastic.hh"
#include "G4LEKaonZeroLInelastic.hh"
#include "G4LEKaonMinusInelastic.hh"
#include "G4LEProtonInelastic.hh"
#include "G4LEAntiProtonInelastic.hh"
#include "G4LENeutronInelastic.hh"
#include "G4LEAntiNeutronInelastic.hh"
#include "G4LELambdaInelastic.hh"
#include "G4LEAntiLambdaInelastic.hh"
#include "G4LESigmaPlusInelastic.hh"
#include "G4LESigmaMinusInelastic.hh"
#include "G4LEAntiSigmaPlusInelastic.hh"
#include "G4LEAntiSigmaMinusInelastic.hh"
#include "G4LEXiZeroInelastic.hh"
#include "G4LEXiMinusInelastic.hh"
#include "G4LEAntiXiZeroInelastic.hh"
#include "G4LEAntiXiMinusInelastic.hh"
#include "G4LEDeuteronInelastic.hh"
#include "G4LETritonInelastic.hh"
#include "G4LEAlphaInelastic.hh"
#include "G4LEOmegaMinusInelastic.hh"
#include "G4LEAntiOmegaMinusInelastic.hh"
#include "G4HEPionPlusInelastic.hh"
#include "G4HEPionMinusInelastic.hh"
#include "G4HEKaonPlusInelastic.hh"
#include "G4HEKaonZeroInelastic.hh"
#include "G4HEKaonMinusInelastic.hh"
```

Include dependency graph for EMMAHadronPhysics.hh:

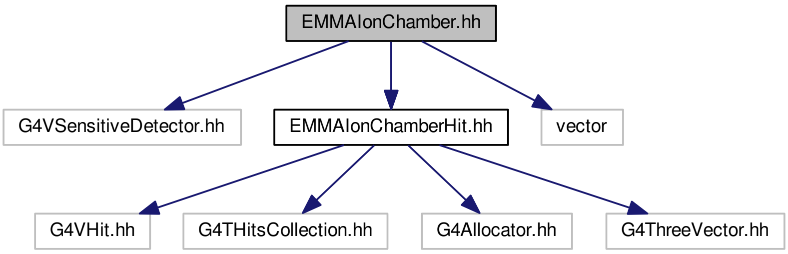
Classes

- class EMMAHadronPhysics

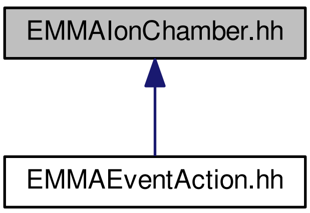
7.24 EMMAIonChamber.hh File Reference

```
#include "G4VSensitiveDetector.hh"
#include "EMMAIonChamberHit.hh"
#include <vector>
```

Include dependency graph for EMMAIonChamber.hh:



This graph shows which files directly or indirectly include this file:



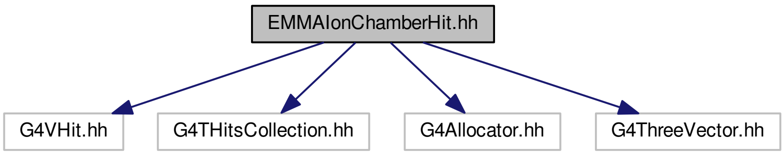
Classes

- class EMMAIonChamber

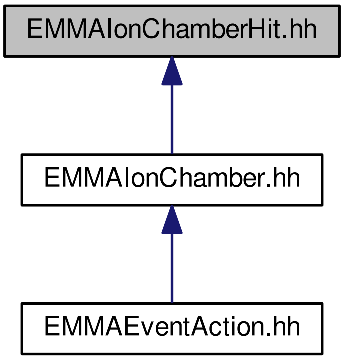
7.25 EMMAIonChamberHit.hh File Reference

```
#include "G4VHit.hh"
#include "G4THitsCollection.hh"
#include "G4Allocator.hh"
#include "G4ThreeVector.hh"
```

Include dependency graph for EMMAIonChamberHit.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class EMMAIonChamberHit

Typedefs

- typedef G4THitsCollection< EMMAIonChamberHit > EMMAIonChamberHitsCollection

Variables

- G4Allocator< EMMAIonChamberHit > EMMAIonChamberHitAllocator

7.25.1 Typedef Documentation

7.25.1.1 typedef G4THitsCollection<EMMAIonChamberHit> EMMAIonChamberHitsCollection

7.25.2 Variable Documentation

7.25.2.1 G4Allocator<EMMAIonChamberHit> EMMAIonChamberHitAllocator

7.26 EMMAIonPhysics.hh File Reference

```
#include "globals.hh"
#include "G4ios.hh"
#include "G4VPhysicsConstructor.hh"
#include "G4HadronElasticProcess.hh"
#include "G4LElastic.hh"
#include "G4DeuteronInelasticProcess.hh"
#include "G4LEDeuteronInelastic.hh"
#include "G4TritonInelasticProcess.hh"
#include "G4LETritonInelastic.hh"
#include "G4AlphaInelasticProcess.hh"
#include "G4LEAlphaInelastic.hh"
#include "G4hIonisation.hh"
#include "G4ionIonisation.hh"
#include "G4hMultipleScattering.hh"
#include "EMMANuclearReactionProcess.hh"
#include "EMMANuclearReactionTwoBody.hh"
```

Include dependency graph for EMMAIonPhysics.hh:

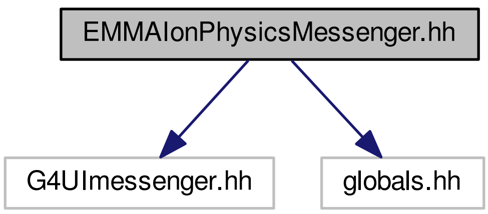
Classes

- class EMMAIonPhysics

7.27 EMMAIonPhysicsMessenger.hh File Reference

```
#include "G4UImessenger.hh"
#include "globals.hh"
```

Include dependency graph for EMMAIonPhysicsMessenger.hh:



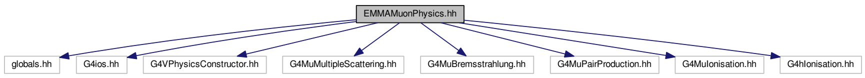
Classes

- class EMMAIonPhysicsMessenger

7.28 EMMAMuonPhysics.hh File Reference

```
#include "globals.hh"
#include "G4ios.hh"
#include "G4VPhysicsConstructor.hh"
#include "G4MuMultipleScattering.hh"
#include "G4MuBremsstrahlung.hh"
#include "G4MuPairProduction.hh"
#include "G4MuIonisation.hh"
#include "G4hIonisation.hh"
```

Include dependency graph for EMMAMuonPhysics.hh:



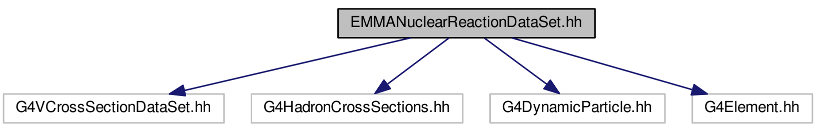
Classes

- class EMMAMuonPhysics

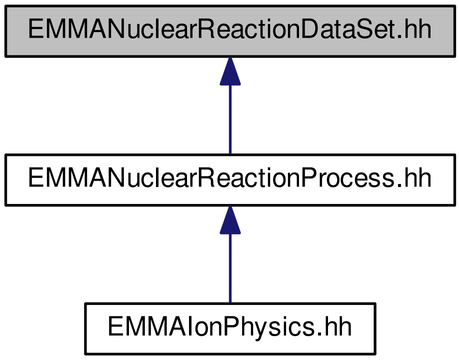
7.29 EMMANuclearReactionDataSet.hh File Reference

```
#include "G4VCrossSectionDataSet.hh"
#include "G4HadronCrossSections.hh"
#include "G4DynamicParticle.hh"
#include "G4Element.hh"
```

Include dependency graph for EMMANuclearReactionDataSet.hh:



This graph shows which files directly or indirectly include this file:



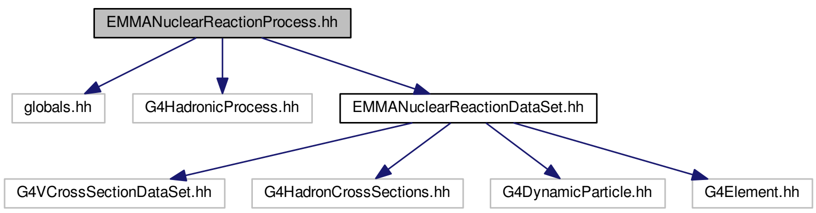
Classes

- class EMMANuclearReactionDataSet

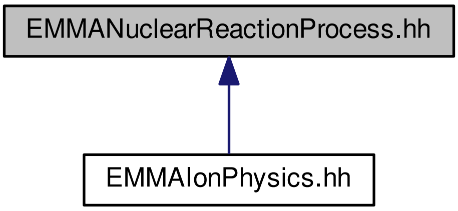
7.30 EMMANuclearReactionProcess.hh File Reference

```
#include "globals.hh"
#include "G4HadronicProcess.hh"
#include "EMMANuclearReactionDataSet.hh"
```

Include dependency graph for EMMANuclearReactionProcess.hh:



This graph shows which files directly or indirectly include this file:



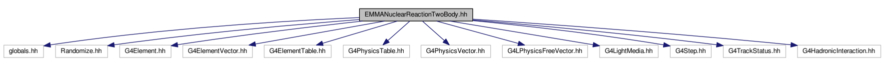
Classes

- class EMMANuclearReactionProcess

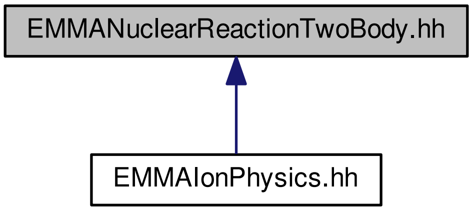
7.31 EMMANuclearReactionTwoBody.hh File Reference

```
#include "globals.hh"
#include "Randomize.hh"
#include "G4Element.hh"
#include "G4ElementVector.hh"
#include "G4ElementTable.hh"
#include "G4PhysicsTable.hh"
#include "G4PhysicsVector.hh"
#include "G4LPhysicsFreeVector.hh"
#include "G4LightMedia.hh"
#include "G4Step.hh"
#include "G4TrackStatus.hh"
#include "G4HadronicInteraction.hh"
```

Include dependency graph for EMMANuclearReactionTwoBody.hh:



This graph shows which files directly or indirectly include this file:



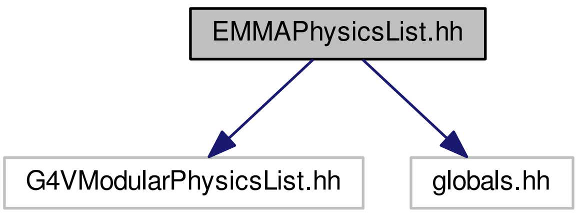
Classes

- class EMMANuclearReactionTwoBody

7.32 EMMAPhysicsList.hh File Reference

```
#include "G4VModularPhysicsList.hh"
#include "globals.hh"
```

Include dependency graph for EMMAPhysicsList.hh:



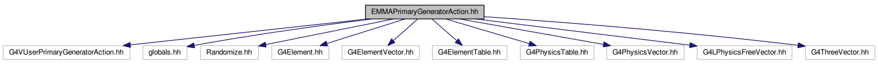
Classes

- class EMMAPhysicsList

7.33 EMMAPrimaryGeneratorAction.hh File Reference

```
#include "G4VUserPrimaryGeneratorAction.hh"
#include "globals.hh"
#include "Randomize.hh"
#include "G4Element.hh"
#include "G4ElementVector.hh"
#include "G4ElementTable.hh"
#include "G4PhysicsTable.hh"
#include "G4PhysicsVector.hh"
#include "G4LPhysicsFreeVector.hh"
#include "G4ThreeVector.hh"
```

Include dependency graph for EMMAPrimaryGeneratorAction.hh:



Classes

- class EMMAPrimaryGeneratorAction

Variables

- G4double targetThickness
- G4double targetZoffset
- G4String MotherDir
- G4String UserDir

7.33.1 Variable Documentation

7.33.1.1 G4String MotherDir

7.33.1.2 G4double targetThickness

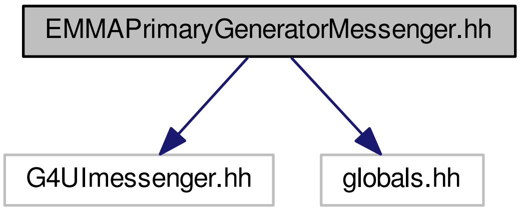
7.33.1.3 G4double targetZoffset

7.33.1.4 G4String UserDir

7.34 EMMAPrimaryGeneratorMessenger.hh File Reference

```
#include "G4UImessenger.hh"
#include "globals.hh"
```

Include dependency graph for EMMAPrimaryGeneratorMessenger.hh:



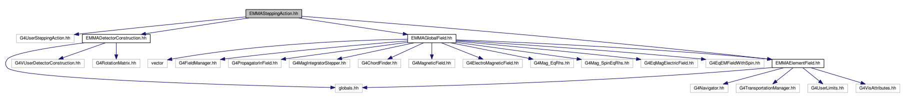
Classes

- class EMMAPrimaryGeneratorMessenger

7.35 EMMASteppingAction.hh File Reference

```
#include "G4UserSteppingAction.hh"
#include "EMMADetectorConstruction.hh"
#include "EMMAGlobalField.hh"
#include "EMMAElementField.hh"
```

Include dependency graph for EMMASteppingAction.hh:



Classes

- class EMMASteppingAction

Variables

- G4double targetThickness
- G4bool prepareBeam
- G4String inTargetFileName
- G4String postTargetFileName
- G4String postDegrader1FileName
- G4double depth

- G4int NOHslits1
- G4int NOHslits2
- G4int NOHslits3
- G4int NOHslits4
- G4double zQ1ends
- G4double zQ2ends
- G4double zQ3ends
- G4double zQ4ends
- G4double magneticScaling
- G4double electricScaling

7.35.1 Variable Documentation

7.35.1.1 G4double depth

7.35.1.2 G4double electricScaling

7.35.1.3 G4String inTargetFileName

7.35.1.4 G4double magneticScaling

7.35.1.5 G4int NOHslits1

7.35.1.6 G4int NOHslits2

7.35.1.7 G4int NOHslits3

7.35.1.8 G4int NOHslits4

7.35.1.9 G4String postDegrader1FileName

7.35.1.10 G4String postTargetFileName

7.35.1.11 G4bool prepareBeam

7.35.1.12 G4double targetThickness

7.35.1.13 G4double zQ1ends

7.35.1.14 G4double zQ2ends

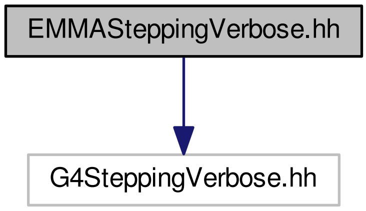
7.35.1.15 G4double zQ3ends

7.35.1.16 G4double zQ4ends

7.36 EMMASteppingVerbose.hh File Reference

```
#include "G4SteppingVerbose.hh"
```

Include dependency graph for EMMASteppingVerbose.hh:



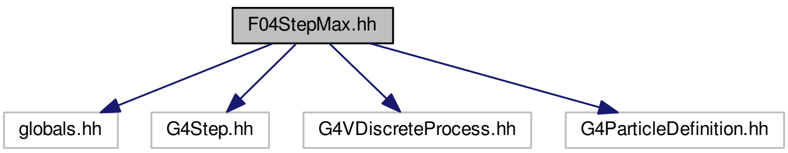
Classes

- class EMMASteppingVerbose

7.37 F04StepMax.hh File Reference

```
#include "globals.hh"
#include "G4Step.hh"
#include "G4VDiscreteProcess.hh"
#include "G4ParticleDefinition.hh"
```

Include dependency graph for F04StepMax.hh:



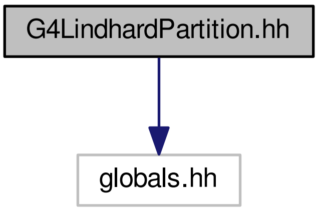
Classes

- class F04StepMax

7.38 Fortran_subs.inc File Reference

7.39 G4LindhardPartition.hh File Reference

```
#include "globals.hh"
```
Include dependency graph for G4LindhardPartition.hh:

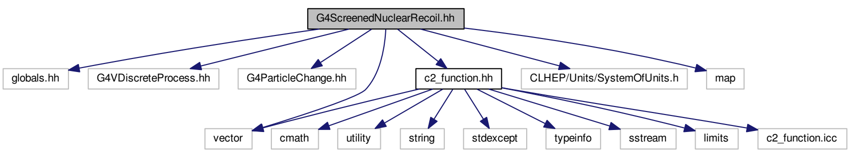


Classes

- class G4VNIELPartition
- class G4LindhardRobinsonPartition

7.40 G4ScreenedNuclearRecoil.hh File Reference

```
#include "globals.hh"
#include "G4VDiscreteProcess.hh"
#include "G4ParticleChange.hh"
#include "c2_function.hh"
#include "CLHEP/Units/SystemOfUnits.h"
#include <map>
#include <vector>
```
Include dependency graph for G4ScreenedNuclearRecoil.hh:



Classes

- struct G4ScreeningTables
- class G4ScreenedCoulombCrossSectionInfo
- class G4ScreenedCoulombCrossSection
- struct G4CoulombKinematicsInfo
- class G4ScreenedCollisionStage
- class G4ScreenedCoulombClassicalKinematics
- class G4SingleScatter
- class G4ScreenedNuclearRecoil

  A process which handles screened Coulomb collisions between nuclei.
- class G4NativeScreenedCoulombCrossSection

Typedefs

- typedef c2_const_ptr< G4double > G4_c2_const_ptr
- typedef c2_ptr< G4double > G4_c2_ptr
- typedef c2_function< G4double > G4_c2_function
- typedef struct G4ScreeningTables G4ScreeningTables
- typedef struct G4CoulombKinematicsInfo G4CoulombKinematicsInfo

Functions

- G4_c2_function & ZBLScreening (G4int z1, G4int z2, size_t npoints, G4double rMax, G4double ∗auval)
- G4_c2_function & MoliereScreening (G4int z1, G4int z2, size_t npoints, G4double rMax, G4double ∗auval)
- G4_c2_function & LJScreening (G4int z1, G4int z2, size_t npoints, G4double rMax, G4double ∗auval)
- G4_c2_function & LJZBLScreening (G4int z1, G4int z2, size_t npoints, G4double rMax, G4double ∗auval)

7.40.1 Typedef Documentation

7.40.1.1 typedef **c2_const_ptr**<G4double> **G4_c2_const_ptr**

7.40.1.2 typedef **c2_function**<G4double> **G4_c2_function**

7.40.1.3 typedef **c2_ptr**<G4double> **G4_c2_ptr**

7.40.1.4 typedef struct **G4CoulombKinematicsInfo G4CoulombKinematicsInfo**

7.40.1.5 typedef struct **G4ScreeningTables G4ScreeningTables**

7.40.2 Function Documentation

7.40.2.1 **G4_c2_function&** LJScreening (G4int *z1*, G4int *z2*, size_t *npoints*, G4double *rMax*, G4double ∗ *auval*)

7.40.2.2 **G4_c2_function&** LJZBLScreening (G4int *z1*, G4int *z2*, size_t *npoints*, G4double *rMax*, G4double ∗ *auval*)

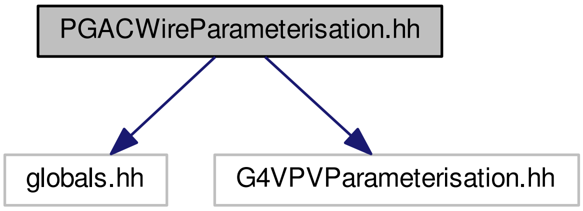
7.40.2.3 **G4_c2_function&** MoliereScreening (G4int *z1*, G4int *z2*, size_t *npoints*, G4double *rMax*, G4double ∗ *auval*)

7.40.2.4 G4_c2_function& ZBLScreening (G4int *z1*, G4int *z2*, size_t *npoints*, G4double *rMax*, G4double * *auval*)

7.41 PGACWireParameterisation.hh File Reference

```
#include "globals.hh"
#include "G4VPVParameterisation.hh"
```

Include dependency graph for PGACWireParameterisation.hh:



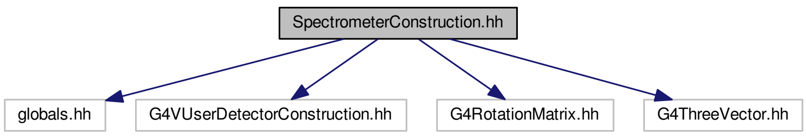
Classes

- class PGACWireParameterisation

7.42 SpectrometerConstruction.hh File Reference

```
#include "globals.hh"
#include "G4VUserDetectorConstruction.hh"
#include "G4RotationMatrix.hh"
#include "G4ThreeVector.hh"
```

Include dependency graph for SpectrometerConstruction.hh:



Classes

- class SpectrometerConstruction

Variables

- G4String MotherDir
- G4String UserDir

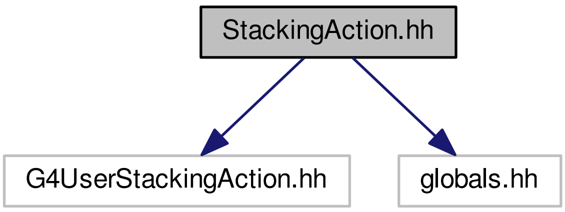
7.42.1 Variable Documentation

7.42.1.1 G4String MotherDir

7.42.1.2 G4String UserDir

7.43 StackingAction.hh File Reference

```
#include "G4UserStackingAction.hh"
#include "globals.hh"
```
Include dependency graph for StackingAction.hh:

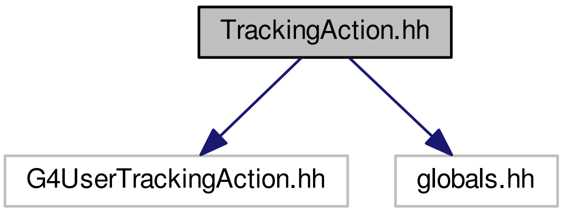


Classes

- class StackingAction

7.44 TrackingAction.hh File Reference

```
#include "G4UserTrackingAction.hh"
#include "globals.hh"
```
Include dependency graph for TrackingAction.hh:



Classes

- class TrackingAction