SphericalSlice

1.0

Generated by Doxygen 1.5.5

Thu Oct 2 20:07:01 2008

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class 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

Namespace Documentation

4.1 SPS Namespace Reference

use Carpet-vectors for fast and easy vector-handling

Classes

- class commstack
- class slices
- class spheredata
- class spheredata_1patch
- class spheredata_2patch
- class spheredata_6patch

Typedefs

- typedef spheredata_1patch< CCTK_REAL >::integrator integrator_1patch
  some shortcuts

Functions

- bool is_1patch (CCTK_INT varno)
  given a variable-number we check if this is a 1-patch slice
- bool is_2patch (CCTK_INT varno)
  given a variable-number we check if this is a 1-patch slice
- bool is_6patch (CCTK_INT varno)
  given a variable-number we check if this is a 1-patch slice
- template<typename T, int D>
  vector< T > & operator<< (vector< T > &a, const vect< T, D > &b)
  conversion to C++ vector

Variables

- slices< spheredata_1patch< CCTK_REAL > > slices_1patch

  all slices for 1patch, 2patch and 6patch systems.

- vector< bool > can_use_Llama_internal

  a flag for each of the slices defining whether it can take advantage of Llama

- vector< CCTK_REAL > radius_internal
- vector< int > ntheta_internal
- vector< void * > radius_pointers

  a vector that stores for each slice-no the pointer to the radius storage.

- bool Llama_activated

  a flag that states whether multipatch is activated or not.

4.1.1 Detailed Description

use Carpet-vectors for fast and easy vector-handling

This file contains integration coefficients for the various methods.

4.1.2 Variable Documentation

4.1.2.1 bool SPS::Llama_activated

a flag that states whether multipatch is activated or not.

if a Multipatch system is present that supports Thornburg04-coordinates, the Llama gets activated

4.1.2.2 vector< int > SPS::ntheta_internal

new angular resolution in case we have Llama activated so that we can directly take integer multiples of the Llama angular resolution

4.1.2.3 vector< CCTK_REAL > SPS::radius_internal

a new radius for the slices that don't exactly lie on Llama radial-gridpoints but not insist of sticking to the given radius so that we can shift the sphere radius to the closest available Llama radial-gridpoint.

Chapter 5

Class Documentation

5.1 SPS::commstack Class Reference

`#include <commstack.hh>`

Public Member Functions

- void push (CCTK_REAL val)

  puts a value to the collective reduction buffer

- void reduce ()

  MPI_Allreduce of the collective buffer.

5.1.1 Detailed Description

This class represents a stack of collective MPI_Allreduce commands. Since MPI_Allreduce is expensive, we collect all reductions and do it in one single call.

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

- src/commstack.hh
- src/commstack.cc

5.2 SPS::slices< SD > Class Template Reference

`#include <slices.hh>`

Public Member Functions

- SD operator() (const int i, const int tl) const
- SD & operator() (const int i, const int tl)
- const vector< deque< SD > > & slice () const

  access to registered slices

- void convert_to_1patch (int const ntheta, int const nphi, CCTK_REAL *const array) const
- int register_slice (const string &varname, int const slice_parameter_no, int const timelevels, const distrib_method_t distrib_method)
- void cycle_timelevels (const int i)

  shifts all timelevels of i-th slice backwards, deletes the last one and creates storage for the first one

5.2.1 Detailed Description

template<class SD> class SPS::slices< SD >

This carries all data of all slices of a given type and assignes groups of processors that can be used for the various slices to get a good load balance.

5.2.2 Member Function Documentation

5.2.2.1 template<class SD> SD SPS::slices< SD >::operator() (const int *i*, const int *tl*) const [inline]

access "i-th" slices at timelevel "tl" stored in this class. This function is used to access slices (and its functions) from other thorns (if needed)

5.2.2.2 template<class SD> SD& SPS::slices< SD >::operator() (const int *i*, const int *tl*) [inline]

modify "i-th" slices at timelevel "tl" stored in this class This function is used to modify slices from other thorns (if needed)

5.2.2.3 template<class SD> void SPS::slices< SD >::convert_to_1patch (int const *ntheta*, int const *nphi*, CCTK_REAL *const *array*) const [inline]

convert a slice to a standard spherical surface with 1 patch by using the given number of gridpoints

5.2.2.4 template<class SD> int SPS::slices< SD >::register_slice (const string & *varname*, int const *slice_parameter_no*, int const *timelevels*, const distrib_method_t *distrib_method*) [inline]

create storage for a new slice with some timelevels as decribed by the n-th parameter in the parfile and return the slice-id

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

- src/slices.hh
- src/slices.cc

5.3 SPS::spheredata< T > Class Template Reference

`#include <spheredata.hh>`

Inheritance diagram for SPS::spheredata< T >:

Public Member Functions

- T operator() (const int p, const int i, const int j) const
  access local surface data on patch "p", index i,j
- T & operator() (const int p, const int i, const int j)
  modify local surface data on patch "p", index i,j
- void * data_pointer () const
  return pointer to surface data
- CCTK_REAL radius (const int p, const int i, const int j) const
  access local surface radius on patch "p", index i,j
- CCTK_REAL & radius (const int p, const int i, const int j)
  modify local surface radius on patch "p", index i,j
- void * radius_pointer () const
  return pointer to surface radius data
- CCTK_REAL cart_x (const int p, const int i, const int j) const
  returns x-coordinate value of local point i,j on patch p
- CCTK_REAL cart_y (const int p, const int i, const int j) const
  returns y-coordinate value of local point i,j on patch p
- CCTK_REAL cart_z (const int p, const int i, const int j) const
  returns z-coordinate value of local point i,j on patch p
- vect< CCTK_REAL, 2 > delta () const
  access delta-spacing
- vect< CCTK_REAL, 2 > coord (const int p, const int i, const int j) const
  access local angular coordinates (e.g. six-patch coordinate system)
- vect< CCTK_REAL, 2 > coord_spherical (const int p, const int i, const int j) const
  access global angular coordinates (standard theta,phi spherical coordinate system)
- vect< CCTK_REAL, 3 > origin () const
  access origin data
- vect< CCTK_REAL, 3 > & origin ()
  modify origin data

- vect< int, 2 > gsh (const int p) const
  global surface size on patch "p" (== npoints == vect<int, 2>(ntheta, nphi))
- vect< int, 2 > lsh (const int p) const
  local size on patch "p"
- vect< int, 2 > ubnd (const int p) const
  upper local bound on patch "p"
- vect< int, 2 > lbnd (const int p) const
  upper local bound on patch "p"
- vect< int, 2 > npoints () const
  the number of global gridpoints on one patch (is supposed to be the same on all patches)
- int ntheta () const
- int nghosts () const
  number of ghostpoints (interpatch and interprocess for all directions)
- int SINDEX2D (const int p, const int i, const int j) const
  given two surface indeices this will return the linear index
- int proc_id () const
  returns the MPI-proc-id of the process that owns the local data
- CCTK_REAL interpolate (const CCTK_REAL theta, const CCTK_REAL phi) const
  access any theta/phi coordinate via interpolation
- void interpolate (const cGH *const cctkGH)
  interpolate from Cactus gridfunctions onto sphere
- CCTK_REAL integrate () const
- CCTK_REAL dx (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j in theta direction
- CCTK_REAL dy (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j in phi direction
- CCTK_REAL dxdx (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j
- CCTK_REAL dxdy (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j
- CCTK_REAL dydy (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j
- CCTK_COMPLEX contract_with_sYlm (const int s, const int l, const int m) const
- CCTK_REAL det (const_iter &i) const
  returns the surface determinant at current point

- bool has_constant_radius () const
- bool uses_Llama () const
- vect< bool, 3 > symmetry () const

  the symmetry of the sphere (symmetric_x, symmetric_y, symmetric_z)

- string name () const

  returns the name of the slice

- string varname () const

  returns the original name of the Cactus variable that is sliced

- int ID () const

  returns the ID of the slice == n-th spherical slice in parfile

- void decompose ()

  return decomposed local piece of data for this MPI process

- distrib_method_t distrib_method () const

  returns the distribution method

- vector< int > processors () const

  a group of processors that carry parts of the slice

- bool can_use_Llama () const

  this returns whether this slice can in principle take advantage of Llama.

- void output (io_base &io) const

Protected Attributes

- vect< bool, 3 > _symmetry

  for now we will not work with symmetries....

- distrib_method_t _distrib_method

  the distribution method

- int _proc_id

  ...and the corresponding process that carries the data

- string _name

  name of the slice (sliced variable-name)

- int _id

  the id of this slice

- string _varname

  the name of the Cactus gridfunction which we want to slice

- CCTK_REAL _radius

  a constant radius

- vect< CCTK_REAL, 3 > _origin

  the slice's origin

- int _ntheta

  the resolution of one patch

- int _nghosts

  the number of ghostpoints for each patch and direction

- bool _can_use_Llama
- vector< int > _processors

  the processor-ids among which to distribute the slice

- bool _valid

  a flag specifying whether this surface is valid or not.

Classes

- class const_iter

  iterator class to traverse the grid of the slice

5.3.1 Detailed Description

template<typename T> class SPS::spheredata< T >

Abstract base class that defines the interface for a container that carries spherical surface data.

5.3.2 Member Function Documentation

5.3.2.1 template<typename T> int SPS::spheredata< T >::ntheta () const [inline]

returns the global number of points (ghostpoints inclusive) on one patch (which is always the same for all patches)

5.3.2.2 template<typename T> CCTK_REAL SPS::spheredata< T >::integrate () const [inline]

surface integral over surface with optional function pointer that is supposed to be multiplied to each value on the sphere

Reimplemented in SPS::spheredata_1patch< T >, and SPS::spheredata_6patch< T >.

5.3.2.3 template<typename T> CCTK_COMPLEX SPS::spheredata< T >::contract_with_sYlm (const int *s*, const int *l*, const int *m*) const [inline]

project onto spin-weighted spherical harmonic with spin s, and usual l, m

Reimplemented in SPS::spheredata_1patch< T >, and SPS::spheredata_6patch< T >.

5.3.2.4 template<typename T> bool SPS::spheredata< T >::has_constant_radius () const [inline]

this flag is used to distinguish between constant spheres that don't need to store a pointwise radius and spheres that need to.

5.3.2.5 template<typename T> bool SPS::spheredata< T >::uses_Llama () const [inline]

determines whether this slice takes advantage of Llama (only the 6patch slices can potentially take advantage)

Reimplemented in SPS::spheredata_6patch< T >.

5.3.2.6 template<typename T> void SPS::spheredata< T >::output (io_base & *io*) const [inline]

write the SphereData including all attributes to the output stream defined by io_base (or its inheritants)

Reimplemented in SPS::spheredata_1patch< T >, and SPS::spheredata_6patch< T >.

5.3.3 Member Data Documentation

5.3.3.1 template<typename T> bool SPS::spheredata< T >::_can_use_Llama [protected]

this flag is set initialy according to whether Llama is activated, the origin is 0, radius=const, and whether this slice lies on a Llama radial gridpoint.

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

- src/spheredata.hh

5.4 SPS::spheredata< T >::const_iter Class Reference

iterator class to traverse the grid of the slice

`#include <spheredata.hh>`

Inherited by SPS::spheredata< T >::iter.

Collaboration diagram for SPS::spheredata< T >::const_iter:

Public Member Functions

- T operator* () const
  dereferencing operator as data-point accessor
- bool done () const
  query whether iterator is done

Classes

- struct idx_t
  index-struct

5.4.1 Detailed Description

template<typename T> class SPS::spheredata< T >::const_iter

iterator class to traverse the grid of the slice

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

- src/spheredata.hh

5.5 SPS::spheredata< T >::const_iter::idx_t Struct Reference

index-struct

`#include <spheredata.hh>`

Public Attributes

- int p
  current patch
- int i
  current grid indices
- int ij
  curent lienar index
- CCTK_REAL theta
  current local coordinates

5.5.1 Detailed Description

template<typename T> struct SPS::spheredata< T >::const_iter::idx_t

index-struct

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

- src/spheredata.hh

5.6 SPS::spheredata_1patch< T > Class Template Reference

`#include <spheredata_1patch.hh>`

Inheritance diagram for SPS::spheredata_1patch< T >:Collaboration diagram for SPS::spheredata_1patch< T >:

Public Member Functions

- int SINDEX2D (const int p, const int i, const int j) const
  given two surface indices this will return the linear index
- T operator() (const int p, const int i, const int j) const
  access local surface data on patch "p", index i,j
- T operator() (const const_iter &i) const
  same as above but using iterator
- T & operator() (const int p, const int i, const int j)
  modify local surface data on patch "p", index i,j
- T & operator() (const const_iter &i)
  same as above but using iterator
- void * data_pointer ()
  return pointer to data
- CCTK_REAL radius (const int p, const int i, const int j) const
  access local surface radius on patch "p", index i,j
- CCTK_REAL radius (const const_iter &i) const
  same as above: access local surface radius using iterator
- CCTK_REAL & radius (const int p, const int i, const int j)
  modify local surface radius on patch "p", index i,j
- CCTK_REAL & radius (const const_iter &i)
  same as above: modify local surface radius using iterator
- void * radius_pointer () const
  return pointer to surface radius data
- vect< CCTK_REAL, 2 > delta () const
  access delta-spacing
- vect< CCTK_REAL, 2 > coord (const int p, const int i, const int j) const
  access local angular coordinates on patch p
- vect< CCTK_REAL, 2 > coord (const const_iter &i) const
  same as above but using iterator

- vect< CCTK_REAL, 2 > coord_spherical (const int p, const int i, const int j) const
- vect< CCTK_REAL, 2 > coord_spherical (const const_iter &i) const

  same as above but using iterator

- CCTK_REAL cart_x (const int p, const int i, const int j) const

  returns x-coordinate value of local point i,j on patch p

- CCTK_REAL cart_x (const const_iter &i) const

  same as above but using iterator

- CCTK_REAL cart_y (const int p, const int i, const int j) const

  returns y-coordinate value of local point i,j on patch p

- CCTK_REAL cart_y (const const_iter &i) const

  same as above but using iterator

- CCTK_REAL cart_z (const int p, const int i, const int j) const

  returns z-coordinate value of local point i,j on patch p

- CCTK_REAL cart_z (const const_iter &i) const

  same as above but using iterator

- vect< int, 2 > gsh (const int p) const

  global surface size on patch "p"

- vect< int, 2 > lsh (const int p) const

  local size on patch "p"

- vect< int, 2 > ubnd (const int p) const

  upper local bound on patch "p"

- vect< int, 2 > lbnd (const int p) const

  upper local bound on patch "p"

- CCTK_REAL interpolate (const CCTK_REAL theta, const CCTK_REAL phi) const

  access any theta/phi coordinate via interpolation

- void interpolate (const cGH *const cctkGH)

  interpolate from Cactus gridfunctions onto sphere

- CCTK_REAL integrate () const

  surface integral over slice

- CCTK_REAL dx (const int p, const int i, const int j) const

  take pointwise derivative on patch p and point i,j in theta direction

- CCTK_REAL dx (const const_iter &i) const

  same as above but using iterator

- CCTK_REAL dy (const int p, const int i, const int j) const

  take pointwise derivative on patch p and point i,j in phi direction

- CCTK_REAL dy (const const_iter &i) const

  same as above but using iterator

- CCTK_REAL dxdx (const int p, const int i, const int j) const

  take pointwise derivative on patch p and point i,j

- CCTK_REAL dxdx (const const_iter &i) const

  same as above but using iterator

- CCTK_REAL dxdy (const int p, const int i, const int j) const

  take pointwise derivative on patch p and point i,j

- CCTK_REAL dxdy (const const_iter &i) const

  same as above but using iterator

- CCTK_REAL dydy (const int p, const int i, const int j) const

  take pointwise derivative on patch p and point i,j

- CCTK_REAL dydy (const const_iter &i) const

  same as above but using iterator

- CCTK_COMPLEX contract_with_sYlm (const int s, const int l, const int m) const
- CCTK_REAL det (const_iter &i) const

  calculate surface determinant for given point

- void output (io_base &io) const

Classes

- class const_iter

  iterator class to traverse the grid of the slice

- class integrator

  integrator class

5.6.1 Detailed Description

template<typename T> class SPS::spheredata_1patch< T >

The standard "old-school" SphericalSurface-style spherical slice description. This will be used for simple user output (if requested) or other thorns that want to work on such simpler slices.

5.6.2 Member Function Documentation

5.6.2.1 template<typename T> vect<CCTK_REAL, 2> SPS::spheredata_1patch< T >::coord_spherical (const int *p*, const int *i*, const int *j*) const [inline]

access global angular coordinates (standard theta,phi spherical coordinate system which for this system is the same as the local coordinate system since we are already in theta,phi-coordinates)

Reimplemented from SPS::spheredata< T >.

5.6.2.2 template<typename T> CCTK_COMPLEX SPS::spheredata_1patch< T >::contract_with_sYlm (const int *s*, const int *l*, const int *m*) const [inline]

project onto spin-weighted spherical harmonic with spin s, and usual l, m

Reimplemented from SPS::spheredata< T >.

5.6.2.3 template<typename T> void SPS::spheredata_1patch< T >::output (io_base & *io*) const [inline]

write the SphereData including all attributes to the output stream defined by io_base (or its inheritants)

Reimplemented from SPS::spheredata< T >.

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

- src/spheredata_1patch.hh

5.7 SPS::spheredata_1patch< T >::const_iter Class Reference

iterator class to traverse the grid of the slice

`#include <spheredata_1patch.hh>`

Inherited by SPS::spheredata_1patch< T >::iter.

Collaboration diagram for SPS::spheredata_1patch< T >::const_iter:

Public Member Functions

- T operator* () const
  dereferencing operator as data-point accessor
- void operator++ (int)
  increment iterator
- bool done () const
  query whether iterator is done
- bool ghostzone () const
  query whether iterator is in ghostzone
- idx_t idx () const
  access current index-struct

Classes

- struct idx_t
  index-struct

5.7.1 Detailed Description

template<typename T> class SPS::spheredata_1patch< T >::const_iter

iterator class to traverse the grid of the slice

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

- src/spheredata_1patch.hh

5.8 SPS::spheredata_1patch< T >::const_iter::idx_t Struct Reference

index-struct

`#include <spheredata_1patch.hh>`

Public Attributes

- int p
  current patch
- int i
  current grid indices
- int ij
  curent lienar index
- CCTK_REAL theta
  current local coordinates

5.8.1 Detailed Description

template<typename T> struct SPS::spheredata_1patch< T >::const_iter::idx_t

index-struct

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

- src/spheredata_1patch.hh

5.9 SPS::spheredata_1patch< T >::integrator Class Reference

integrator class

`#include <spheredata_1patch.hh>`

Public Member Functions

- void init ()
  initialize
- CCTK_REAL finalize (commstack *const cs=NULL)
  finalize integration and return result
- CCTK_REAL result () const
  return the result of the integration (integration must already be finalized)
- void sum (const_iter &it, CCTK_REAL f=1.0, CCTK_REAL det=1.0)
  sum over function on the sphere (and multiply by given function and determinant)

5.9.1 Detailed Description

template<typename T> class SPS::spheredata_1patch< T >::integrator

integrator class

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

- src/spheredata_1patch.hh

5.10 SPS::spheredata_2patch< T > Class Template Reference

`#include <spheredata_2patch.hh>`

Inheritance diagram for SPS::spheredata_2patch< T >:Collaboration diagram for SPS::spheredata_2patch< T >:

5.10.1 Detailed Description

template<typename T> class SPS::spheredata_2patch< T >

A stereographic 2-patch system covering the sphere. This is probably not what we want for now...it's just here for later convenience if someone feels like it...

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

- src/spheredata_2patch.hh

5.11 SPS::spheredata_6patch< T > Class Template Reference

`#include <spheredata_6patch.hh>`

Inheritance diagram for SPS::spheredata_6patch< T >:Collaboration diagram for SPS::spheredata_6patch< T >:

Public Member Functions

- bool uses_Llama () const
- int SINDEX2D (const int p, const int i, const int j) const
  given two surface indices this will return the linear index
- T operator() (const int p, const int i, const int j) const
  access local surface data on patch "p", index i,j
- T operator() (const const_iter &i) const
  same as above but using iterator
- T & operator() (const int p, const int i, const int j)
  modify local surface data on patch "p", index i,j
- T & operator() (const const_iter &i)
  same as above but using iterator
- void * data_pointer (const int m)
  return pointer to data
- CCTK_REAL radius (const int p, const int i, const int j) const
  access local surface radius on patch "p", index i,j
- CCTK_REAL radius (const const_iter &i) const
  same as above: access local surface radius using iterator
- CCTK_REAL & radius (const int p, const int i, const int j)
  modify local surface radius on patch "p", index i,j
- CCTK_REAL & radius (const const_iter &i)
  same as above: modify local surface radius using iterator
- void * radius_pointer () const
  return pointer to surface radius data
- vect< CCTK_REAL, 2 > delta () const
  access delta-spacing
- vect< CCTK_REAL, 2 > coord (const int p, const int i, const int j) const
  access local angular coordinates on patch p (local 6-patch ("inflated cube") coordinates)
- vect< CCTK_REAL, 2 > coord (const const_iter &i) const

same as above but using iterator

- vect< CCTK_REAL, 2 > coord_spherical (const int p, const int i, const int j) const
  access global angular coordinates (standard theta,phi spherical coordinate system)
- vect< CCTK_REAL, 2 > coord_spherical (const const_iter &i) const
  same as above but using iterator
- CCTK_REAL cart_x (const int p, const int i, const int j) const
  returns x-coordinate value of local point i,j on patch p
- CCTK_REAL cart_x (const const_iter &i) const
  same as above but using iterator
- CCTK_REAL cart_y (const int p, const int i, const int j) const
  returns y-coordinate value of local point i,j on patch p
- CCTK_REAL cart_y (const const_iter &i) const
  same as above but using iterator
- CCTK_REAL cart_z (const int p, const int i, const int j) const
  returns z-coordinate value of local point i,j on patch p
- CCTK_REAL cart_z (const const_iter &i) const
  same as above but using iterator
- vect< int, 2 > gsh (const int p) const
  global surface size on patch "p"
- vect< int, 2 > lsh (const int p) const
  local size on patch "p"
- vect< int, 2 > ubnd (const int p) const
  upper local bound on patch "p"
- vect< int, 2 > lbnd (const int p) const
  upper local bound on patch "p"
- CCTK_REAL interpolate (const CCTK_REAL theta, const CCTK_REAL phi) const
  access any theta/phi coordinate via interpolation
- void interpolate (const cGH *const cctkGH)
  interpolate from Cactus gridfunctions onto sphere
- CCTK_REAL integrate () const
  surface integral over slice
- CCTK_REAL dx (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j in theta direction

- CCTK_REAL dx (const const_iter &i) const
  same as above but using iterator
- CCTK_REAL dy (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j in phi direction
- CCTK_REAL dy (const const_iter &i) const
  same as above but using iterator
- CCTK_REAL dxdx (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j
- CCTK_REAL dxdx (const const_iter &i) const
  same as above but using iterator
- CCTK_REAL dxdy (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j
- CCTK_REAL dxdy (const const_iter &i) const
  same as above but using iterator
- CCTK_REAL dydy (const int p, const int i, const int j) const
  take pointwise derivative on patch p and point i,j
- CCTK_REAL dydy (const const_iter &i) const
  same as above but using iterator
- CCTK_COMPLEX contract_with_sYlm (const int s, const int l, const int m) const
- CCTK_REAL det (const_iter &i) const
  calculates surface determinant for given point
- void output (io_base &io) const

Classes

- class const_iter
  iterator class to traverse the grid of the slice
- class integrator
  integrator class

5.11.1 Detailed Description

template<typename T> class SPS::spheredata_6patch< T >

A 6-patch system covering the sphere. It offers a uniform spatial sampling of the sphere. In addition it will work hand in hand with Llama (Thornburg04 coordinates).

5.11.2 Member Function Documentation

5.11.2.1 template<typename T> bool SPS::spheredata_6patch< T >::uses_Llama () const [inline]

determines whether this slice takes advantage of Llama (only the 6patch slices can potentially take advantage)

Reimplemented from SPS::spheredata< T >.

5.11.2.2 template<typename T> CCTK_COMPLEX SPS::spheredata_6patch< T >::contract_with_sYlm (const int *s*, const int *l*, const int *m*) const [inline]

project onto spin-weighted spherical harmonic with spin s, and usual l, m

Reimplemented from SPS::spheredata< T >.

5.11.2.3 template<typename T> void SPS::spheredata_6patch< T >::output (io_base & *io*) const [inline]

write the SphereData including all attributes to the output stream defined by io_base (or its inheritants)

Reimplemented from SPS::spheredata< T >.

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

- src/spheredata_6patch.hh

5.12 SPS::spheredata_6patch< T >::const_iter Class Reference

iterator class to traverse the grid of the slice

`#include <spheredata_6patch.hh>`

Inherited by SPS::spheredata_6patch< T >::iter.

Collaboration diagram for SPS::spheredata_6patch< T >::const_iter:

Public Member Functions

- T operator* () const
  dereferencing operator as data-point accessor
- void operator++ (int)
  increment iterator
- bool done () const
  query whether iterator is done
- bool ghostzone () const
  query whether iterator is in ghostzone
- idx_t idx () const
  access current index-struct

Classes

- struct idx_t
  index-struct

5.12.1 Detailed Description

template<typename T> class SPS::spheredata_6patch< T >::const_iter

iterator class to traverse the grid of the slice

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

- src/spheredata_6patch.hh

5.13 SPS::spheredata_6patch< T >::const_iter::idx_t Struct Reference

index-struct

`#include <spheredata_6patch.hh>`

Public Attributes

- int p
  current patch
- int i
  current grid indices
- int ij
  curent lienar index
- CCTK_REAL theta
  current local coordinates

5.13.1 Detailed Description

template<typename T> struct SPS::spheredata_6patch< T >::const_iter::idx_t

index-struct

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

- src/spheredata_6patch.hh

5.14 SPS::spheredata_6patch< T >::integrator Class Reference

integrator class

`#include <spheredata_6patch.hh>`

Public Member Functions

- void init ()
  initialize
- CCTK_REAL finalize (commstack *const cs=NULL)
  finalize integration and return result
- CCTK_REAL result () const
  return the result of the integration (integration must already be finalized)
- void sum (const_iter &it, CCTK_REAL f=1.0, CCTK_REAL det=1.0)
  sum over function on the sphere (and multiply by given function and determinant)

5.14.1 Detailed Description

template<typename T> class SPS::spheredata_6patch< T >::integrator

integrator class

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

- src/spheredata_6patch.hh

Index