PFSS

Generated by Doxygen 1.8.17

Chapter 1

Main Page

This PFSS computation suite computes the PFSS solution using photospheric magnetograms from MDI, HMI, G↩ ONG, and WSO. It can also create maps of the magnetic configuration and the expansion factor at arbitrary heights between photsphere and source surface. Commands are given via command line arguments to the PFSS suite. For an example how to use the command-line interface see below. Several solar observatories are supported and automatically recognized for supplying the photospheric magnetogram.

1.1 Building the documentation (optional)

A PDF containing the documentation of this program is included in the GIT repository. The documentation is implemented via doxygen. In order to obtain the most current documentation, you first need to install doxygen

```
apt install doxygen
```

and then build the documentation:

```
cd pfss
make documentation
```

The documentation will then be produced both as HTML and PDF in the directory PFSS/doc.

1.2 Optimizations (optional)

Several optimizations can be adjusted in the Makefile. If a CUDA-capable device is present, setting the variable CUDA to '1' will build the program to employ the CUDA device, which might decrease computation time substantially. For this optimization the CUDA runtime environment needs to be installed (please consult NVIDIA's webpage for instructions).

The variable NUMTHREADS limits the number of threads to be utilized for multithreaded execution of the program. For optimal performance it should be the same number as CPU cores in the system.

Default parameters for the computation can also be set in the Makefile. The corresponding build variables start with DEFAULT_.

1.3 Building the binary

This program utilizes several third-party libraries which need to be installed in order for the binary to be built. These libraries are very common so there is a good chance that they can be found in your distributions repositories. If you are on a debian system, try installing them via

```
apt install libfreeimage-dev libcfitsio-dev libboost-dev libboost-filesystem-dev libboost-regex-dev
```

If successful the PFSS computation suite can be built via

```
cd pfss
make
```

The binary will be placed in pfss/bin.

1.4 Running the program

If you used make to build the PFSS computation suite, the binary file will be stored in pfss/bin/.

1.4.1 Configuration and model output files

Upon execution the binary reads a configuration file, which specifies the data and configuration directories. The configuration directory contains information about start and stop times of Carrington rotations. The data directory contains all the output from the PFSS computation suite. If you run the binary from the pfss/bin/ directory without specifying a configuration file, the default file pfss/config/config will be used. The default data directoy is then pfss/data. Please consult this default config file to set your own data directory at a location with enough disk space if the default location is not suitable. Absolute paths in your config file allows the binary to be executed from arbitrary shell locations. Manipulation of the configuration file is only necessary if you have special needs for the location of the computed solutions.

1.4.2 Command line options and arguments

The syntax to run the PFSS computation suite is

```
pfss --option0 [argument0] --option1 [argument1] ...
```

The following options and arguments are supported. Default values for not specified options can be adjusted in the Makefile.

-- **config** *filename*

filename is path to configuration file [default: --config ../config/config]

-- **compute** *filename*

Invokes the PFSS solver for given photospheric magnetogram at path *filename*. For additional arguments see below.

-- **map** *filename*

Computes the magnetic configuration at specified height [default: photosphere and source surface]. *filename* points to the configuration file of the computed solution (ending in ∗config.cfg). For additional options see below.

-- **batchcompute** *directory*

Invokes the PFSS solver for all magnetic magnetograms found in *directory* (non-recursive).

-- **batchmap**

Invokes mapper for all solutions computed in data directory.

Additional options for compute

-- **rss** *value* source surface height (multiples of solar radius), *value* is floating point
-- **ell** *value* ellipticity of source surface, *value* is floating point, [default: 1.0 (spheric)]
-- **resCompR** *value* computational grid resolution in radial direction, other directions are determined automatically, *value* is unsigned integer
-- **method** *value* solution method to be used for PFSS computation. *value* is either 'shc' for the classic spherical harmonic coefficient approach or 'numeric' for the finite difference solver
-- **order** *value* maximum principal order to be used with the SHC approach, *value* is unsigned integer

Additional options for map

-- **resMapTheta** *value* resolution of mapping in meridional direction, *value* is unsigned integer
-- **resMapPhi** *value* resolution of mapping in zonal direction, *value* is unsigned integer
-- **height** *value* height between phot. and source surface to be mapped, *value* is multiple of solar radius

1.5 Supported photospheric magnetograms

The following photospheric synoptic magnetogram sources are supported. The PFSS computation suite identifies the source instrument and necessary pre-processing steps by filename, meaning you cannot alter them or the program will not be able to handle the containing data.

observatory name	filename example	resolution	URL
WSO	WSO.2066.F.txt	72 x 30	http://wso.stanford.↩ edu/synopticl.html
SOHO-MDI	synop_MI_0.2066.fits	3600 x 1080	http://sun.stanford.↩ edu/synop/
SDO-HMI	hmi.Synoptic_MI.2100.fits	3600 x 1440	http://hmi.stanford.↩ edu/data/synoptic.html
NSO-GONG	mrmqs080208t0128c2066_000.↩ fits	360 x 180	https://gong.nso.↩ edu/data/magmap/crmap.↩ html

1.6 Example use cases

All examples are executed from the pfss/bin-directory, so change there:

```
cd pfss/bin
```

To perform a PFSS model evaluation with default parameters for the synoptic photospheric magnetogram found at pfss/data/input/synop_Ml_0.2066.fits (not included in the repository, you need to download the magnetogram and place it at that location):

```
./pfss --compute ../data/input/synop_Ml_0.2066.fits
```

To generate magnetic mappings at the photosphere and source surface of the computed solution in the previous example:

```
./pfss --map ../data/2066/2066_MDI_Kiel_PFSS2.50_GridSph_35x87x175_config.cfg
```

To generate a magnetic mappings of the same solution at height r=2.4 r_sol with a resolution of 130 x 200 pixels:

```
./pfss --map ../data/2066/2066_MDI_Kiel_PFSS2.50_GridSph_35x87x175_config.cfg --height 2.4 \
--resMapTheta 130 --resMapPhi 200
```

To evaluate the PFSS model for all photospheric magnetic maps found in directory pfss/data/batchinput with a radial grid resolution of 40 grid points and a source surface radius of r=3.1 r_sol:

```
./pfss --batchcompute ../data/input/ --resCompR 40 --rss 3.1
```

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

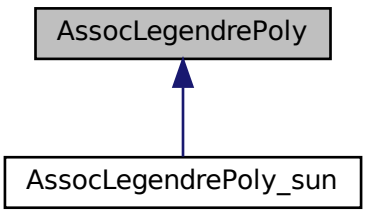
Class Documentation

4.1 AssocLegendrePoly Class Reference

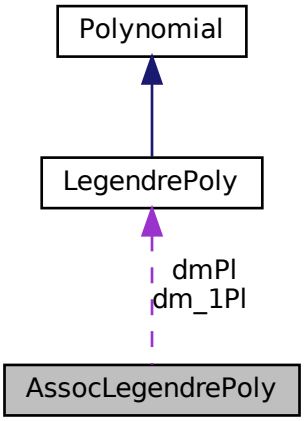
conventional associated Legendre polynomials (as described by, e.g., Wikipedia)

```
#include <hcFunction.h>
```

Inheritance diagram for AssocLegendrePoly:



Collaboration diagram for AssocLegendrePoly:



Public Member Functions

- AssocLegendrePoly (uint l=0, uint m=0)
  std constructor
- AssocLegendrePoly (const AssocLegendrePoly &other)
  cpy constructor
- virtual ~AssocLegendrePoly ()
  destructor
- AssocLegendrePoly & operator= (const AssocLegendrePoly &other)
  assignment op
- virtual hcFloat operator() (hcFloat x)
  evaluator
- void **initNULL** ()
- void **clear** ()
- void **init** (uint l, uint m)
- virtual hcFloat getDeriv (hcFloat theta)
  returns first derivative at cos(theta)
- void **dump** ()

Public Attributes

- uint **l**
- uint **m**
- LegendrePoly dmPl

m'th derivative of the Legendre polynomial

- LegendrePoly dm_1Pl

  the (m+1)st derivative of the Legendre polynomial (necessary for theta component)

4.1.1 Detailed Description

conventional associated Legendre polynomials (as described by, e.g., Wikipedia)

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

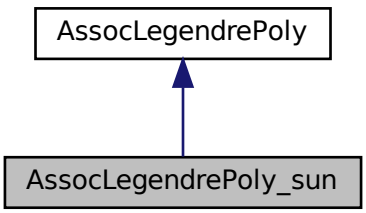
- engine/math/hcFunction.h
- engine/math/hcFunction.cpp

4.2 AssocLegendrePoly_sun Class Reference

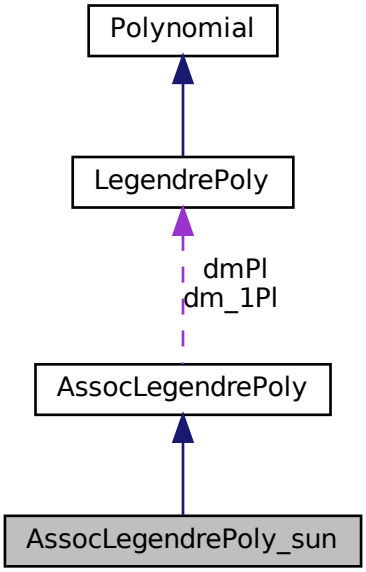
associated Legendre polynomial as used by Xudong Sun for WSO data

```
#include <hcFunction.h>
```

Inheritance diagram for AssocLegendrePoly_sun:



Collaboration diagram for AssocLegendrePoly_sun:



Public Member Functions

- AssocLegendrePoly_sun (uint l=0, uint m=0)
  std constructor
- AssocLegendrePoly_sun (const AssocLegendrePoly_sun &other)
  cpy constructor
- virtual ~AssocLegendrePoly_sun ()
  destructor
- AssocLegendrePoly_sun & operator= (const AssocLegendrePoly_sun &other)
  assignment operator
- virtual hcFloat operator() (hcFloat x)
  evaluator
- virtual hcFloat getDeriv (hcFloat theta)
  returns first derivative at cos(theta)

Additional Inherited Members

4.2.1 Detailed Description

associated Legendre polynomial as used by Xudong Sun for WSO data

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

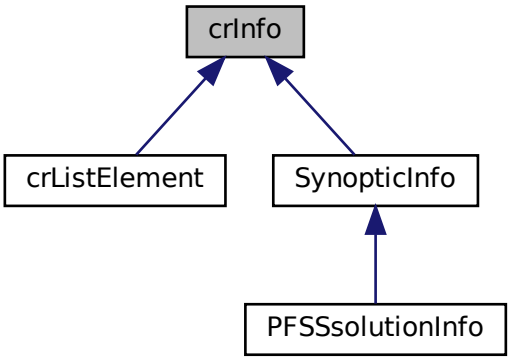
- engine/math/hcFunction.h
- engine/math/hcFunction.cpp

4.3 crInfo Class Reference

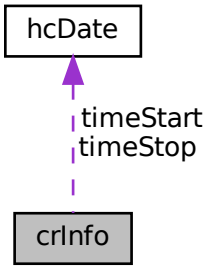
information on specific Carrington rotation

```
#include <carRotInfo.h>
```

Inheritance diagram for crInfo:



Collaboration diagram for crInfo:



Public Member Functions

- crInfo (uint crNum=0)

  std constructor

- crInfo (const crInfo &other)

  cpy constructor

- virtual ~crInfo ()

  destructor

- crInfo & operator= (const crInfo &other)

  assignment operator

- bool operator== (const crInfo &other)

  comparison operator

- bool operator> (const crInfo &other)

  comparison operator

- bool operator< (const crInfo &other)

  comparison operator

- bool operator>= (const crInfo &other)

  comparison operator

- bool operator<= (const crInfo &other)

  comparison operator

- bool **exportBinary** (std::ofstream &stream)
- bool **importBinary** (std::ifstream &stream)
- void **initNULL** ()
- void **clear** ()
- void **init** (uint CRnum)
- void dump (uint indent=0) const

  dumps information on this instance to stdout with optional indendtation

Public Attributes

- uint CRnum

  number of Carrington Rotation

- hcDate timeStart

  start date of Carrington Rotation

- hcDate timeStop

  stop date of Carrington Rotation

4.3.1 Detailed Description

information on specific Carrington rotation

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

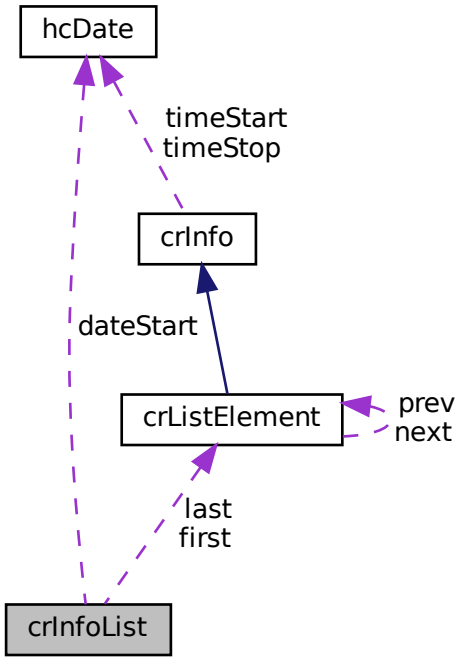
- src/carRotInfo.h
- src/carRotInfo.cpp

4.4 crInfoList Class Reference

list of carrington rotation numbers and corresponding start times

`#include <carRotInfo.h>`

Collaboration diagram for crInfoList:



Public Member Functions

- **crInfoList** (const crInfoList &other)
- crInfoList & **operator=** (const crInfoList &other)
- void **clear** ()
- void **initNULL** ()
- void **init** ()

- crListElement * getcrListElement (uint carRotNum)

  returns the list element representing the specified carRotNum or NULL, if non-existent

- bool **appendObservation** (uint carRotNum, originID origin, int pos)
- void dump () const

  dumps information on this instance to stdout with optional indendtation

Static Public Member Functions

- static bool **initStaticMembers** ()
- static void **clearStaticMembers** ()
- static hcDate getStartDate (int carRotNum)

  computes start date of specific Carrington rotation

- static hcDate getStopDate (int carRotNum)

  computes stop date of specific Carrington rotation

- static int getCRnumber (const hcDate &date)

  computes the Carrington rotation number of a specific date

Public Attributes

- crListElement * first

  first element in list

- crListElement * last

  last element in list

Static Public Attributes

- static uint numCRinList = 0

  number of carrington rotations stored in list

- static int * listCRNum = NULL

  list of carrington rotation numbers

- static long int * listJulianDayNum = NULL

  list of corresponding start times (JD integer part)

- static double * listJulianDayFrac = NULL

  list of corresponding start times (JD fractional part)

- static hcDate * dateStart = NULL

  list of start dates

4.4.1 Detailed Description

list of carrington rotation numbers and corresponding start times

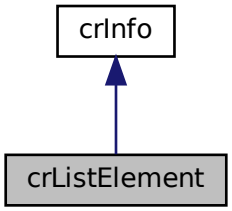
The documentation for this class was generated from the following files:

- src/carRotInfo.h
- src/carRotInfo.cpp

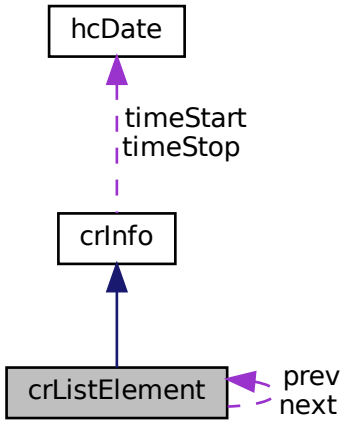
4.5 crListElement Class Reference

`#include <carRotInfo.h>`

Inheritance diagram for crListElement:



Collaboration diagram for crListElement:



Public Member Functions

- crListElement (int carRotNum=0)
  std-constructor
- crListElement (const crListElement &other)
  cpy constructor
- ~crListElement ()
  destructor
- crListElement & **operator=** (const crListElement &other)
- void **clear** ()
- void **initNULL** ()
- void **init** (int carRotNum)
- void addElementToArray (int **array, uint &numEntries, uint newValue)
  expands array by one and appends newValue at the end
- bool appendObservatory (originID origin, int pos)
  appends the position in solutions-set of the new observation to element
- void dump ()
  dumps information on this instance to stdout with optional indendtation

Public Attributes

- crListElement * **prev**
- crListElement * **next**
- int * **WSO_solution**
- int * **KPVT_solution**
- int * **MDI_solution**
- int * **MDIDAILY_solution**
- int * **GONG_solution**
- int * **HMI_solution**
- int * **OWN_solution**
- uint **numWSO**
- uint **numKPVT**
- uint **numMDI**
- uint **numMDIDAILY**
- uint **numGONG**
- uint **numHMI**
- uint **numOWN**

4.5.1 Detailed Description

information on which instrument has been used to compute the magnetic field of Carrington Rotation carRotNum

the integer values XXX_solution are either -1 if no solution has been computed using that instrument or the positional number in the solutions-set of HelioMagfield

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

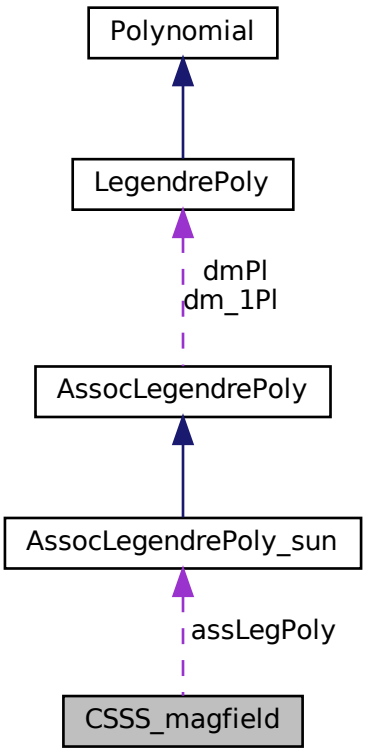
- src/carRotInfo.h
- src/carRotInfo.cpp

4.6 CSSS_magfield Class Reference

solution of the CSSS model (not fully implemented/not working!!)

`#include <pfss.h>`

Collaboration diagram for CSSS_magfield:



Public Member Functions

- CSSS_magfield ()
  std constructor
- CSSS_magfield (const CSSS_magfield &other)
  cpy constructor
- **CSSS_magfield** (const char *filename)
- ~CSSS_magfield ()
  destructor
- CSSS_magfield & operator= (const CSSS_magfield &other)
  assignment operator

- void **initNULL** ()
- void **clear** ()
- void **init** (uint order)
- void **eval** (const Vec3D &pos, Vec3D &result)
- bool **load** (const char *filename)
- void **exportCoefficients** (const char *filename)
- void **dump** ()

Public Attributes

- AssocLegendrePoly_sun ** assLegPoly

  contains all the associated Legendre polynomials required for the solution

- uint order

  highest order of assoc. Legendre Polynomial utilized

- double ** g

  coefficients

- double ** h

  coefficients

- double sourceSurfaceFactor

  location of source surface in multiples of r_sol

4.6.1 Detailed Description

solution of the CSSS model (not fully implemented/not working!!)

Gets as parameter a file with coefficients for associated Legendre polynomials. Returns on request the solution (magnetic field strength B) at a specific location in spherical (preferred) or cartesian coordinates

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

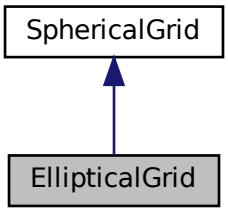
- src/pfss.h
- src/pfss.cpp

4.7 EllipticalGrid Class Reference

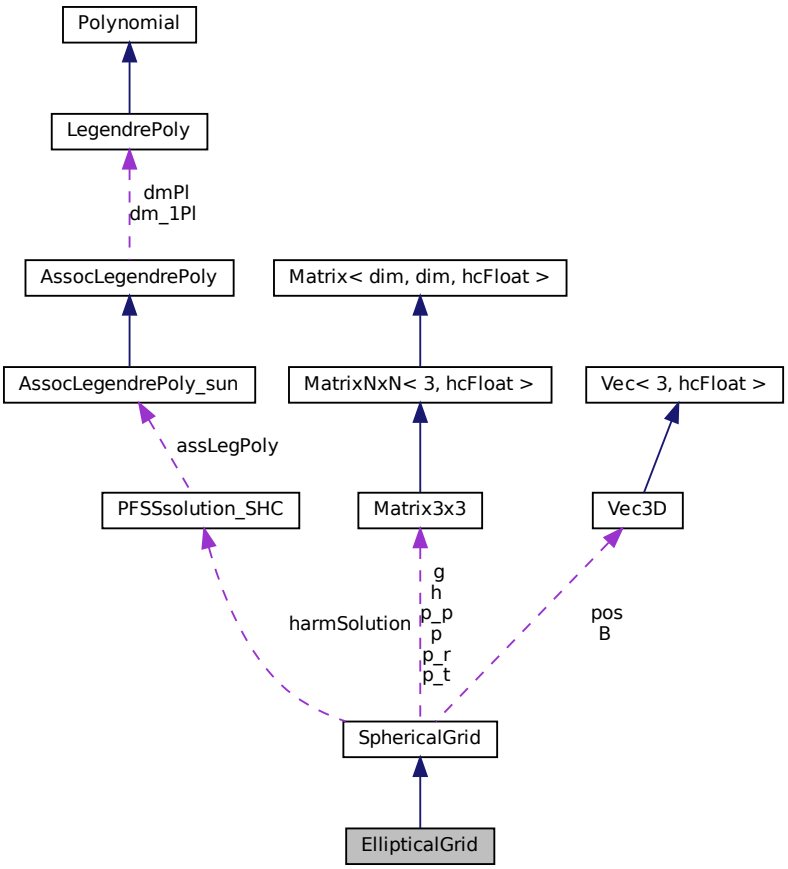
grid structure for numerical computation in 3D space

`#include <ellipticalGrid.h>`

Inheritance diagram for EllipticalGrid:



Collaboration diagram for EllipticalGrid:



Public Member Functions

- virtual bool isElliptical () const

determine whether the grid is spherical or elliptical

- virtual Vec3D **getPos** (uint index, bool ellipticCoords=false) const
- virtual Vec3D **getB** (uint index, bool ellipticCoords=false) const
- virtual void **setB** (uint index, Vec3D value, bool ellipticCoords=false)
- Vec3D getSphericalPos (uint index)

  returns position in spherical coordinates, same as getPos, but not virtual

- hcFloat * getEllAArray () const

  parameter a of ellipsis

- hcFloat **getEllA** (uint index) const
- EllipticalGrid ()

  std constructor

- EllipticalGrid (const EllipticalGrid &grid)

  cpy constructor

- virtual ~EllipticalGrid ()

  destructor

- EllipticalGrid & operator= (const EllipticalGrid &other)

  assignment operator

- EllipticalGrid & operator= (const SphericalGrid &other)

  assignment operator

- void **initNULL_CPU** ()
- void **initNULL** ()
- void **clear_CPU** ()
- void **clear** ()
- void **dump** () const
- hcFloat getEllParamsFromPos (Vec3D pos, bool posElliptic) const

  computes elliptic parameters a for arbitrary position pos

- void **convertMagMapping** (MagMapping &map)
- virtual void init (bool sinLatGrid, hcFloat maxSinLat, hcFloat minSinLat, hcFloat lowerR, hcFloat upperR, uint numR, bool clearGPU=true, hcFloat a=1.0)
- bool getGradientVectors (Vec3D cartPos, Vec3D &er, Vec3D &et, Vec3D &ep, bool prolate) const

  returns cartesian gradient (contravariant) basis vectors in physical domain

- bool getTangentVectors (Vec3D cartPos, Vec3D &er, Vec3D &et, Vec3D &ep, bool prolate) const

  returns cartesian tangent (covariant) basis vectors in physical domain

Public Attributes

- bool **prolate**

Protected Attributes

- hcFloat * a

  parameter a of ellipsis

Additional Inherited Members

4.7.1 Detailed Description

grid structure for numerical computation in 3D space

4.7.2 Member Function Documentation

4.7.2.1 init()

```
void EllipticalGrid::init (
            bool sinLatGrid,
            hcFloat maxSinLat,
            hcFloat minSinLat,
            hcFloat lowerR,
            hcFloat upperR,
            uint numR,
            bool clearGPU = true,
            hcFloat ell = 1.0 )  [virtual]
```

additional information like different domains (block regular) and distribution of grid points have to be supplied

Parameters

rDistribution	(0 - equally spaced, 1 - geometric series)

Reimplemented from SphericalGrid.

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

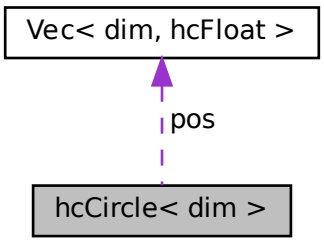
- src/ellipticalGrid.h
- src/ellipticalGrid.cpp

4.8 hcCircle< dim > Class Template Reference

mathematical model and functions for circles in nD-space

```
#include <hcCircle.h>
```

Collaboration diagram for hcCircle< dim >:



Public Member Functions

- hcCircle ()
  std constructor
- hcCircle (const hcCircle &other)
  cpy constructor
- hcCircle (const Vec< dim, hcFloat > &pos, hcFloat radius)
  constructor
- ~hcCircle ()
  destructor
- hcCircle< dim > & operator= (const hcCircle< dim > &other)
  assignment operator
- void **init** (const Vec< dim, hcFloat > &pos, hcFloat radius)
- void **dump** () const

Public Attributes

- Vec< dim, hcFloat > **pos**
- float **radius**

4.8.1 Detailed Description

template<uint dim>
class hcCircle< dim >

mathematical model and functions for circles in nD-space

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

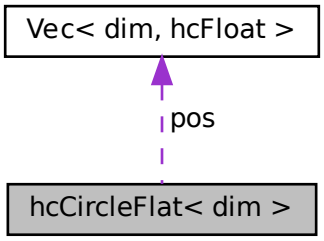
- engine/math/hcCircle.h

4.9 hcCircleFlat< dim > Class Template Reference

mathematical model and functions for circles in 2D-space

`#include <hcCircle.h>`

Collaboration diagram for hcCircleFlat< dim >:



Public Member Functions

- hcCircleFlat (const Vec2D &pos=Vec2D(0.0, 0.0), float radius=1.0)
- hcCircleFlat (const hcCircleFlat &other)

  cpy constructor
- hcCircleFlat & **operator=** (const hcCircleFlat &other)
- void **init** (const Vec2D &pos, float radius)
- void **set** (const Vec2D &pos, float radius)
- int intersectsL2D (const hcLine2D &line, Vec2D &result0, Vec2D &result1) const

  computes the intersections point(s) with a line for the std-norm
- int intersectsC2D (const hcCircleFlat &circle, Vec2D &result1, Vec2D &result2, hcLine2D &resultline) const
- int getTangentVecForced (const Vec2D &in, Vec2D &result) const

  computes the tangent vector to a point on the circle

Public Attributes

- Vec< dim, hcFloat > **pos**
- float **radius**

4.9.1 Detailed Description

template<uint dim>
class hcCircleFlat< dim >

mathematical model and functions for circles in 2D-space

this class is not fit for circles oriented in nD-space

4.9.2 Constructor & Destructor Documentation

4.9.2.1 hcCircleFlat()

```
template<uint dim>
hcCircleFlat< dim >::hcCircleFlat (
            const Vec2D & pos = Vec2D(0.0, 0.0),
            float radius = 1.0 )
```

Parameters

pos	std constructor

4.9.3 Member Function Documentation

4.9.3.1 getTangentVecForced()

```
template<uint dim>
int hcCircleFlat< dim >::getTangentVecForced (
            const Vec2D & in,
            Vec2D & result ) const
```

computes the tangent vector to a point on the circle

No checks are made during computation. This speeds up the process and gets rid of numerical artifacts, though it has to be assured beforehand that the point really lies on the circle. Otherwise following computations might yield rubbish

4.9.3.2 intersectsC2D()

```
template<uint dim>
int hcCircleFlat< dim >::intersectsC2D (
            const hcCircleFlat< dim > & circle,
            Vec2D & result0,
            Vec2D & result1,
            hcLine2D & resultline ) const
```

computes the intersection of 2 circles in 2D-space (std-norm)

Return values

0	no intersection / areas disjunct or one circle in the other
1	two intersection points, stored in result1 and result2, the line produced by these two points is stored in resultline
2	tangential point, stored in result1

4.9.3.3 intersectsL2D()

```
template<uint dim>
int hcCircleFlat< dim >::intersectsL2D (
            const hcLine2D & line,
            Vec2D & result0,
            Vec2D & result1 ) const
```

computes the intersections point(s) with a line for the std-norm

Return values

0	no intersection of line with circle
1	two intersections, stored in result1 and result2
2	line tangential, tangential point stored in result1

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

- engine/math/hcCircle.h

4.10 hcDate Class Reference

Public Member Functions

- hcDate ()
  std constructor
- hcDate (const hcDate &other)
  cpy constructor
- **hcDate** (long int year, uint month, uint day, uint hour, uint minute, uint second, hcTimeStandard timeID=H↩C_TT, hcCalendarID calendarID=HC_GREGORIAN)
- **hcDate** (long int year, uint doy, uint hour, uint minute, uint second, hcTimeStandard timeID=HC_TT, hc↩CalendarID calendarID=HC_GREGORIAN)
- **hcDate** (double year, hcTimeStandard timeID=HC_TT, hcCalendarID calendarID=HC_GREGORIAN)
- **hcDate** (string timestamp)
- void **initNULL** ()
- hcDate & **operator=** (const hcDate &other)
- hcDate & **operator+=** (int128 timeDiff)
- hcDate & **operator-=** (int128 timeDiff)
- int128 **operator-** (const hcDate &other) const
- bool operator> (const hcDate &other) const
  determines if this comes after other
- bool operator>= (const hcDate &other) const
  determines if this comes after other
- bool operator< (const hcDate &other) const

determines if other comes after this

- bool operator<= (const hcDate &other) const
  determines if other comes after this
- bool operator== (const hcDate &other) const
  determines equality
- void setFromSystemTime ()
  reads the time from the system and converts it to hcDate
- bool setFromTimeStamp (const string &timestamp)
  reads time from string produced by toString
- void set (long int year, uint doy, uint hour, uint minute, uint second, hcTimeStandard timeID=HC_TT, hcCalendarID calendarID=HC_GREGORIAN)
  set to a specific point in time
- bool set (long int year, uint month, uint day, uint hour, uint minute, uint second, hcTimeStandard timeID=HC_TT, hcCalendarID calendarID=HC_GREGORIAN)
  set to a specific point in time (Relative to me. Eat that, Einstein!!)
- void setFromTT (int128 absoluteTime)
  set date from absolute TT time since J2000
- void setFromJD (long int julianDay, double frac)
  convert julienDay.frac to hcDate
- void setFromUnix (int128 unixTime)
  set date from UNIX time (seconds since 1970-01-01, 00:00:00 UTC)
- void setFromCarringtonTime (const double &crTime)
  set date from Carrington time
- bool isLeapYear ()
  tells if this year is a leap year
- uint monthLength (uint numMonth)
  returns the length of the given month in days
- void **computeInternalTT** ()
- void **getJulianDate** (long &julianDayNum, double &frac) const
- void **getModifiedJulianDate** (long &mjd, double &mjd_frac) const
- int128 getUnixTime () const
  *get UNIX time seconds * facsec since Jan. 01, 1970 00:00:00 UTC*
- double getCarringtonLongitude () const
  get Carrington Longitude in degrees (sub-earth point, measured from 0° at TODO ???)
- double getCarringtonTime () const
  get Carrington Longitude in degrees (sub-earth point, measured from 0° at TODO ???)
- string getTOD () const
  returns human readable time of day

- uint getCarringtonRotationNum () const
- hcDate getCarringtonRotStartDate ()

  find start time of Carrington rotation number crNum

- hcDate getCarringtonRotEndDate ()

  find end time of Carrington rotation number crNum

- uint computeDOY ()

  computes day of year from values set in the fields year, month and day

- void computeDayAndMonthFromDOY ()

  computes day and month from the fields doy and year

- void **setFromInternalTT** ()
- void **convert2** (hcTimeStandard std)
- uint **numLSsinceBeginning** () const
- bool **isLeapSec** () const
- uint **getWeekDay** () const
- string **toString** () const
- string **toSpiceString** () const
- bool **exportBinary** (std::ofstream &stream)
- bool **importBinary** (std::ifstream &Sstream)
- void **str** (char *out)
- void **dump** () const

Public Attributes

- hcCalendarID calendarID

  type of calendar used (Gregorian, Julian,...)

- hcTimeStandard timeStandard

  time standard used to express the date (TT, UTC,...)

- int128 absoluteTime

  Absolute time in seconds from epoch J2000.0, Jan. 01, 2000 : 12:00:00 given in Terrestrial Time.
- int128 equTTtime

  what the TT clock would show at this date

- int128 timeSinceJ2000

  *seconds * facSec from epoch in timeStandard*

- long int year

  christian year

- uint doy

  day of year

- uint month

  month of year

- uint dom

  day of month

- uint hour

  hour of day

- uint minute

  minute of hour

- uint second

  second of minute

- uint millisec

  ms of second

- uint microsec

  us of ms

- uint nanosec

  ns of us

Static Public Attributes

- static const int128 facSec = 1000000000

  time is stored with nano-second accuracy

4.10.1 Member Function Documentation

4.10.1.1 getCarringtonRotationNum()

```
uint hcDate::getCarringtonRotationNum ( ) const
```

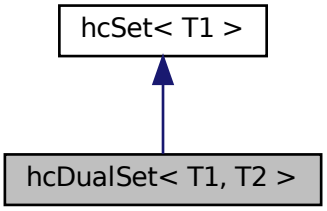
brief get Carrington rotation number corresponding to this date

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

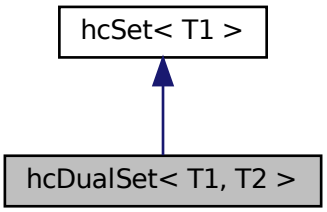
- engine/hcTime.h
- engine/hcTime.cpp

4.11 hcDualSet< T1, T2 > Class Template Reference

Inheritance diagram for hcDualSet< T1, T2 >:



Collaboration diagram for hcDualSet< T1, T2 >:



Public Member Functions

- **hcDualSet** (uint numMaxSlots=1)
- void **clear** ()
- void **initNULL** ()
- void **init** (uint numMaxSlots)
- T2 * **getOtherObject** (const T1 &object)
- int **appendElement** (T1 &object, T2 &otherObject)
- int removeElement (T1 &object)

  remove given object from this set

Public Attributes

- T2 ** **otherElements**

4.11.1 Member Function Documentation

4.11.1.1 removeElement()

```
template<class T1 , class T2 >
int hcDualSet< T1, T2 >::removeElement (
            T1 & object )  [virtual]
```

remove given object from this set

Returns

position where object has been found and removed or -1 on failure

Reimplemented from hcSet< T1 >.

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

- engine/hcSet.h

4.12 hcImage< T > Class Template Reference

Public Member Functions

- hcImage ()
  std constructor
- hcImage (const hcImage< T > &other)
  cpy constructor
- **hcImage** (uint width, uint height)
- hcImage & operator= (const hcImage &other)
  assignment operator
- T & operator() (uint x, uint y)
  returns manipulatable reference to contents
- T content (uint x, uint y) const
  returns copy of contents
- void **initNULL** ()
- void **clear** ()
- void init (uint width, uint height)
  allocates memory for an image of the according dimenstions
- hcImage extractSubimage (int extractPosX, int extractPosY, uint width, uint height)

extracts subimage of (width ∗ height) from the specified position

- bool extractSubimage (int extractPosX, int extractPosY, hcImage< T > &retval)

  see above

- void shiftXdirection (int numPixels)

  shifts image numPixels in positive x-direction, pixels leaving right boundary enter from the left

- void pushToArray (T **array, uint &width, uint &height)

  saves data to array and reports back width and height

- T ∗ **getData** ()
- void **dump** () const

Public Attributes

- uint **width**
- uint **height**

Protected Attributes

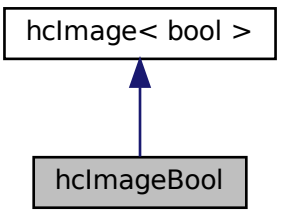
- T ∗ **data**

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

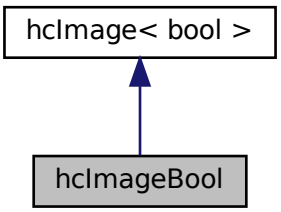
- engine/hcImage.h

4.13 hcImageBool Class Reference

Inheritance diagram for hcImageBool:



Collaboration diagram for hcImageBool:



Public Member Functions

- hcImageBool ()

  std constructor

- hcImageBool (const hcImageBool &other)

  cpy constructor

- **hcImageBool** (uint width, uint height)
- hcImageBool & **operator=** (const hcImageBool &other)
- void **init** (uint width, uint height)
- void **dump** () const

Additional Inherited Members

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

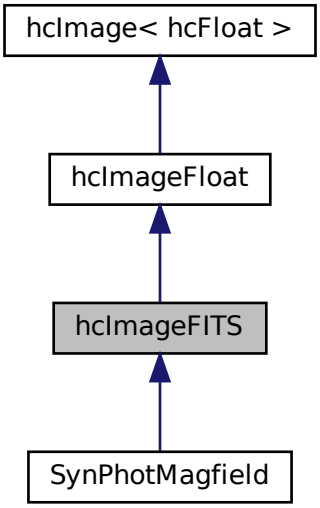
- engine/hcImage.h
- engine/hcImage.cpp

4.14 hcImageFITS Class Reference

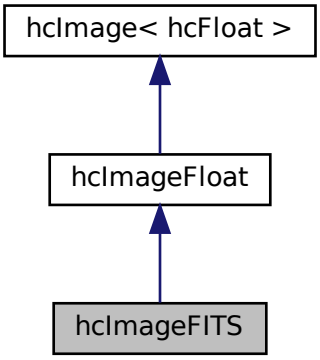
handler for 2D FITS image files

```
#include <hcImageFITS.h>
```

Inheritance diagram for hcImageFITS:



Collaboration diagram for hcImageFITS:



Public Member Functions

- hcImageFITS ()
- hcImageFITS (const hcImageFITS &other)
- **hcImageFITS** (uint width, uint height)
- hcImageFITS (const string &filename)
- virtual ~hcImageFITS ()
- hcImageFITS & operator= (const hcImageFITS &other)

- void **clear** ()
- void **initNULL** ()
- virtual bool load (const string &filename)

  load FITS image from file

- virtual bool save (const string &filename)

  write FITS image to file

- bool dumpAllKeys ()

  print all stored keys to screen

- bool readKeyString (const string &keyname, string &retval)

  read key keyname and give back as string

- bool readKeyFloat (const string &keyname, hcFloat &value)

  read key keyname and convert to floating point value before returning

- bool writeKeyFloat (const string &keyname, const string &comment, hcFloat value)

  write key keyname with content value

- bool writeKeyString (const string &keyname, const string &comment, const string &value)

  write key keyname with content value

- bool **writeKeyComment** (const string &comment)
- hcFloat **getHistogramMaximum** (uint numBins)
- ImageStatistics **getImageStatistics** () const
- percentileDataStruct **getPercentiles** () const
- void **normalize** ()
- void **rescale** (uint newWidth, uint newHeight)
- hcFloat **crosscor** (const hcImageFITS &other, uint i, uint j)

Static Public Member Functions

- static void **initStaticMembers** ()

Public Attributes

- string **filename**
- fitsfile * filePtr

Static Public Attributes

- static pthread_mutex_t **mutexFits**

Additional Inherited Members

4.14.1 Detailed Description

handler for 2D FITS image files

4.14.2 Constructor & Destructor Documentation

4.14.2.1 hcImageFITS() [1/3]

```
hcImageFITS::hcImageFITS ( )
```

std constructor

4.14.2.2 hcImageFITS() [2/3]

```
hcImageFITS::hcImageFITS (
            const hcImageFITS & other )
```

cpy constructor

4.14.2.3 hcImageFITS() [3/3]

```
hcImageFITS::hcImageFITS (
            const string & filename )
```

creates handler by opening a file

4.14.2.4 ~hcImageFITS()

```
hcImageFITS::~hcImageFITS ( )  [virtual]
```

destructor

4.14.3 Member Function Documentation

4.14.3.1 operator=()

```
hcImageFITS & hcImageFITS::operator= (
            const hcImageFITS & other )
```

assignment operator

4.14.4 Member Data Documentation

4.14.4.1 filePtr

```
fitsfile* hcImageFITS::filePtr
```

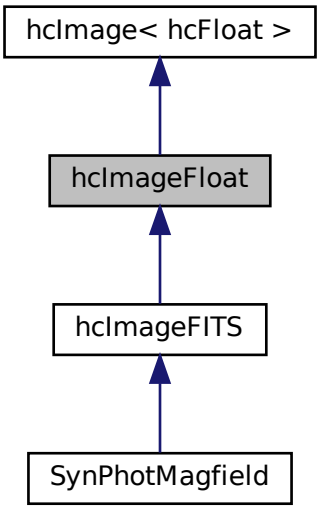
pointer to input FITS file

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

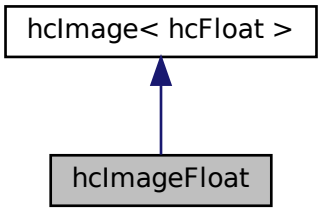
- engine/hcImageFITS.h
- engine/hcImageFITS.cpp
- main.cpp

4.15 hcImageFloat Class Reference

Inheritance diagram for hcImageFloat:



Collaboration diagram for hcImageFloat:



Public Member Functions

- hcImageFloat ()
  std constructor
- hcImageFloat (const hcImageFloat &other)
  cpy constructor
- **hcImageFloat** (uint width, uint height)
- **hcImageFloat** (uint width, uint height, float color)
- hcImageFloat & **operator=** (const hcImageFloat &other)
- void **init** (uint width, uint height)
- void init (uint width, uint height, float color)
  allocates memory for an image of the specified dimensions and paints it according to color
- virtual bool load (const string &filename)
  loads an image from filename
- virtual bool save (const string &filename)
  saves to filename
- void reldiffImage (const hcImageFloat &img0, const hcImageFloat &img1)
  computes pixelwise relative difference between img0 and img1
- void diffImage (const hcImageFloat &img0, const hcImageFloat &img1)
  compute pixelwise difference between img0 and img1
- float absMean ()
  sums up all absolute pixel values and divides by number of pixels
- void setBackgroundColor (float bgColor, int insertPosX=0, int insertPosY=0, uint subWidth=0, uint subHeight=0)
  paints the entire image or parts of it in bgColor
- bool insertSubimage (const hcImageFloat &src, int insertPosX, int insertPosY)
  inserts a image into this one at the specified location

- bool loadFromArray (float *array, uint width, uint height)

  loads image data from array

- hcFloat **meanSquaredDiff** (hcImage &other)
- void meanFilter (uint windowWidth, bool circular)

  applies mean filter to picture, x-coordinate may be circular

- void medianFilter (uint windowWidth, bool circular)

  applies median filter to picture, x-coordinate may be circular

- void dump () const

  rescales image to new dimensions

Additional Inherited Members

4.15.1 Member Function Documentation

4.15.1.1 dump()

```
void hcImageFloat::dump ( ) const
```

rescales image to new dimensions

<

4.15.1.2 insertSubimage()

```
bool hcImageFloat::insertSubimage (
            const hcImageFloat & src,
            int insertPosX,
            int insertPosY )
```

inserts a image into this one at the specified location

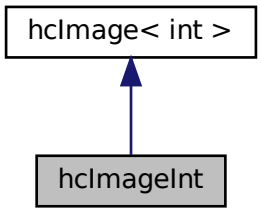
inserts image supplied by src at the supplied positions by replacing the data stored at these positions in this (opaque, no transperency)

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

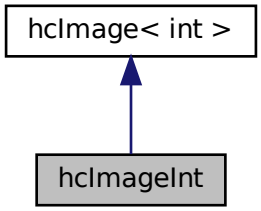
- engine/hcImage.h
- engine/hcImage.cpp

4.16 hclmageInt Class Reference

Inheritance diagram for hclmageInt:



Collaboration diagram for hclmageInt:



Public Member Functions

- hclmageInt ()
  std constructor
- hclmageInt (const hclmageInt &other)
  cpy constructor
- **hclmageInt** (uint width, uint height)
- hclmageInt & **operator=** (const hclmageInt &other)
- void **init** (uint width, uint height)
- void **dump** () const

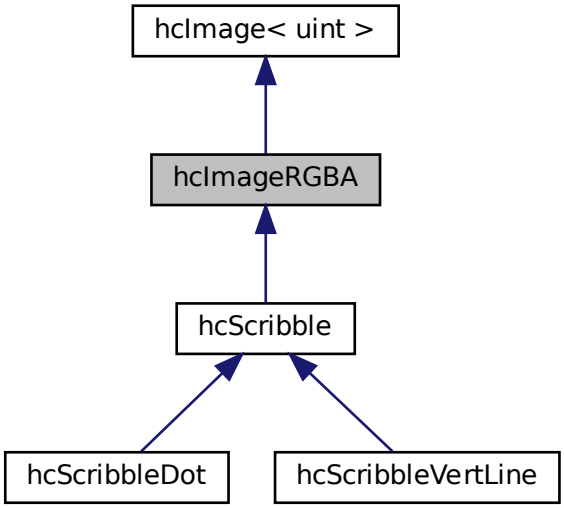
Additional Inherited Members

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

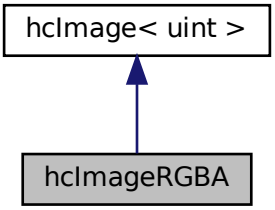
- engine/hclmage.h
- engine/hclmage.cpp

4.17 hcImageRGBA Class Reference

Inheritance diagram for hcImageRGBA:



Collaboration diagram for hcImageRGBA:



Public Member Functions

- hcImageRGBA ()
  std constructor
- hcImageRGBA (const hcImageRGBA &other)
  cpy constructor
- **hcImageRGBA** (uint width, uint height)

- **hcImageRGBA** (uint width, uint height, uint color)
- hcImageRGBA & **operator=** (const hcImageRGBA &other)
- void **init** (uint width, uint height)
- void init (uint width, uint height, uint color)

  allocates memory for an image of the specified dimensions and paints it according to color

- virtual bool save (const string &filename)

  saves to filename

- uint **numNotBlackPixels** () const
- void **magDiff** (hcImage &img1, hcImage &img2)
- void setBackgroundColor (uint bgColor, int insertPosX=0, int insertPosY=0, uint subWidth=0, uint subHeight=0)

  paints the entire image or parts of it in bgColor

- bool insertSubimage (hcImage &src, int insertPosX, int insertPosY)

  inserts a image into this one at the specified location

- bool loadFromArray (uint *array, uint width, uint height)

  loads image data from array

- bool interpolateRectangularImage (uint width_out, uint height_out)

  interpolates image to the requested dimension

- void **dump** () const

Additional Inherited Members

4.17.1 Member Function Documentation

4.17.1.1 interpolateRectangularImage()

```
bool hcImageRGBA::interpolateRectangularImage (
            uint width_out,
            uint height_out )
```

interpolates image to the requested dimension

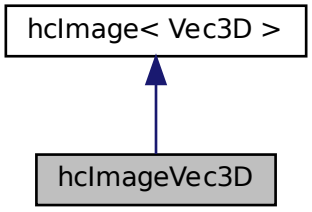
resizes a rectangular image via bilinear interpolation TODO: redo this function!

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

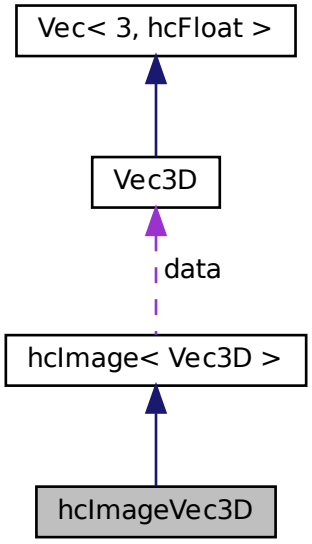
- engine/hcImage.h
- engine/hcImage.cpp

4.18 hcImageVec3D Class Reference

Inheritance diagram for hcImageVec3D:



Collaboration diagram for hcImageVec3D:



Public Member Functions

- hcImageVec3D ()
  std constructor
- hcImageVec3D (const hcImageVec3D &other)
  cpy constructor

- **hcImageVec3D** (uint width, uint height)
- hcImageVec3D & **operator=** (const hcImageVec3D &other)
- void **init** (uint width, uint height)
- void **dump** () const

Additional Inherited Members

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

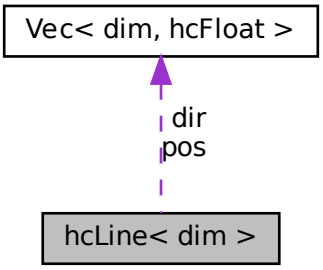
- engine/hcImage.h
- engine/hcImage.cpp

4.19 hcLine< dim > Class Template Reference

contains mathematical model and functions for nD lines

`#include <hcLine.h>`

Collaboration diagram for hcLine< dim >:



Public Member Functions

- hcLine ()
  std-constructor
- hcLine (const hcLine< dim > &other)
  cpy-constructor
- hcLine (const Vec< dim, hcFloat > &pos, const Vec< dim, hcFloat > &dir)
  direction constructor
- ~hcLine ()
  destructor

- hcLine< dim > & operator= (const hcLine< dim > &other)
  assignment operator
- int intersectsL (const hcLine< dim > &l, Vec< dim, hcFloat > &result) const
  tests the intersectio of this line with another line
- int intersectsP (const Vec< dim, hcFloat > &vec, float &result) const
  tests if a point lies on this line
- int createLineThroughPoints (const Vec< dim, hcFloat > &point1, const Vec< dim, hcFloat > &point2)
  creates a line through point1 and point2
- float **pointProjection** (const Vec< dim, hcFloat > &point) const
- void **set** (const Vec< dim, hcFloat > &vec1, const Vec< dim, hcFloat > &vec2)
- int **setV1** (const Vec< dim, hcFloat > &vec)
- int **setV2** (const Vec< dim, hcFloat > &vec)
- Vec< dim, hcFloat > getPos (hcFloat lambda) const
  computes the nD-Vetor obtained by inserting lambda in this Line equation
- void **dump** () const

Public Attributes

- Vec< dim, hcFloat > **pos**
- Vec< dim, hcFloat > **dir**

4.19.1 Detailed Description

template<uint dim>
class hcLine< dim >

contains mathematical model and functions for nD lines

The nD-Line is parametrized by by two nD-vectors (position and direction vectors): line = pos + lambda ∗ dir lambda is element of [0,1] if line is initialized by giving two points to be connected by the line

4.19.2 Member Function Documentation

4.19.2.1 createLineThroughPoints()

```
template<uint dim>
int hcLine< dim >::createLineThroughPoints (
            const Vec< dim, hcFloat > & point1,
            const Vec< dim, hcFloat > & point2 )
```

creates a line through point1 and point2

creates ND-line through two ND-points, if they are not too close to each other

Return values

0	points too close or dimension mismatch
1	line successfully created

4.19.2.2 intersectsL()

```
template<uint dim>
int hcLine< dim >::intersectsL (
            const hcLine< dim > & l,
            Vec< dim, hcFloat > & result ) const
```

tests the intersectio of this line with another line

computes the intersection with another nD-line

Return values

0	no intersection found
1	exactly one intersection found
2	lines are identical

the Vector result contains the intersection point in case 1 and the position vector of this Line in the case 2

4.19.2.3 intersectsP()

```
template<uint dim>
int hcLine< dim >::intersectsP (
            const Vec< dim, hcFloat > & vec,
            float & result ) const
```

tests if a point lies on this line

checks if the nD-point denoted by vec is on this Line

Return values

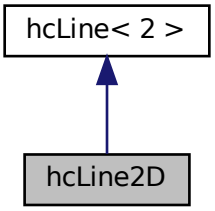
0	point not on Line
1	point on Line, line parameter is returned in result

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

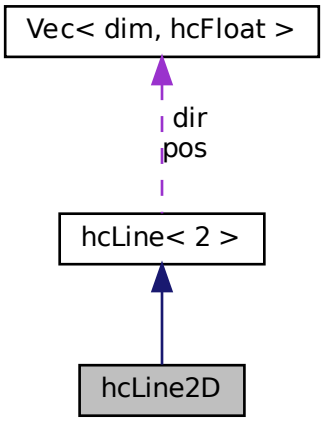
- engine/math/hcLine.h

4.20 hcLine2D Class Reference

Inheritance diagram for hcLine2D:



Collaboration diagram for hcLine2D:



Public Member Functions

- hcLine2D (const Vec2D &pos=Vec2D(0.0, 0.0), const Vec2D &dir=Vec2D(1.0, 1.0))

  std constructor

- hcLine2D (const hcLine2D &other)

  cpy constructor

- **hcLine2D** (float x1, float y1, float x2, float y2)
- hcLine2D & **operator=** (const hcLine2D &other)

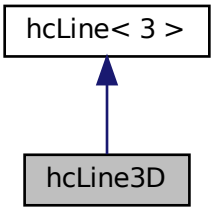
Additional Inherited Members

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

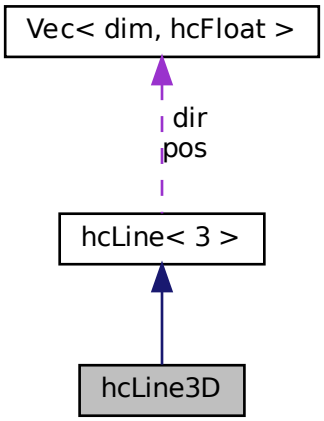
- engine/math/hcLine.h
- engine/math/hcLine.cpp

4.21 hcLine3D Class Reference

Inheritance diagram for hcLine3D:



Collaboration diagram for hcLine3D:



Public Member Functions

- hcLine3D (const Vec3D &pos0=Vec3D(0.0, 0.0, 0.0), const Vec3D &pos1=Vec3D(1.0, 1.0, 1.0))
- hcLine3D (const hcLine3D &other)

  cpy constructor

- ~hcLine3D ()

  destructor

- hcLine3D & operator= (const hcLine3D &other)

  assignment operator

- void **init** (const Vec3D &pos0, const Vec3D &pos1)
- int intersectsSphere3D (const hcSphere< 3 > &sphere, hcFloat &result0, hcFloat &result1, bool getInside↩Pos)

  returns the parameter of the line where it intersects the sphere
- Vec3D intersectsPlane3D (const hcPlane3D &plane, uint &result)

  returns the point of intersection
- int getIntersectionsWithSphere (const hcSphere< 3 > &sphere, Vec3D &result0, Vec3D &result1, bool get↩InsidePos=false)

  gives the two points where the line pireces the sphere (or the point where it touches the sphere)

Additional Inherited Members

4.21.1 Constructor & Destructor Documentation

4.21.1.1 hcLine3D()

```
hcLine3D::hcLine3D (
            const Vec3D & pos0 = Vec3D(0.0, 0.0, 0.0),
            const Vec3D & pos1 = Vec3D(1.0, 1.0, 1.0) )
```

Parameters

pos0	std constructor

4.21.2 Member Function Documentation

4.21.2.1 getIntersectionsWithSphere()

```
int hcLine3D::getIntersectionsWithSphere (
            const hcSphere< 3 > & sphere,
```

```
            Vec3D & result0,
            Vec3D & result1,
            bool getInsidePos = false )
```

gives the two points where the line pireces the sphere (or the point where it touches the sphere)

returns the positions where the line pierces the sphere

Parameters

sphere	the sphere to be pierced
result0	first intersection or tangent point of sphere and line
result1	second intersection point of sphere and line
getInsidePos	numerical parameter, determines if result0/1 shall be slightly above or below sphere surface

Return values

0	no intersection of line with sphere
1	line tangential, tangential point stored in result0
2	two intersections, stored in result0 and result1

4.21.2.2 intersectsPlane3D()

```
Vec3D hcLine3D::intersectsPlane3D (
            const hcPlane3D & plane,
            uint & result )
```

returns the point of intersection

returns the point of intersection (if existing, zero-vec otherwise)

Parameters

plane	the plane to be pierced
result	0 -> no intersection, 1 -> one intersection, 2 -> line lies in plane

4.21.2.3 intersectsSphere3D()

```
int hcLine3D::intersectsSphere3D (
            const hcSphere< 3 > & sphere,
            hcFloat & result0,
            hcFloat & result1,
            bool getInsidePos )
```

returns the parameter of the line where it intersects the sphere

returns the line parameter lambda0 and lambda1 where the line pierces the sphere

Parameters

sphere	the sphere to be pierced
result0	first intersection or tangent parameter of sphere and line
result1	second intersection parameter of sphere and line
getInsidePos	numerical parameter, determines if result0/1 shall be slightly above or below sphere surface this parameter affects only the pirecing (retval = 1) and not not the tangent (retval = 2) case

Return values

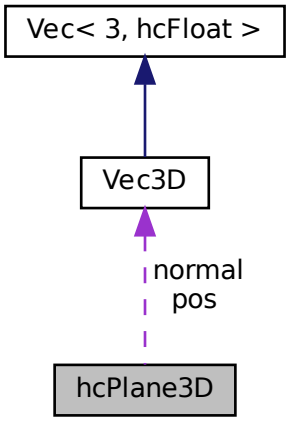
0	no intersection of line with sphere
1	line tangential, tangential point stored in result0
2	two intersections, stored in result0 and result1

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

- engine/math/hcLine.h
- engine/math/hcLine.cpp

4.22 hcPlane3D Class Reference

Collaboration diagram for hcPlane3D:



Public Member Functions

- hcPlane3D (const Vec3D &pos=Vec3D(0.0, 0.0, 0.0), const Vec3D &normal=Vec3D(0.0, 0.0, 1.0))

  std constructor

- hcPlane3D (const hcPlane3D &other)

  cpy constructor

- **hcPlane3D** (const Vec3D &pos0, const Vec3D &pos1, const Vec3D &pos2)
- hcPlane3D & **operator=** (const hcPlane3D &other)
- bool operator== (const hcPlane3D &other)

  tests identity of this plane with another

- void **clear** ()
- void **initNULL** ()
- bool init (const Vec3D &pos, const Vec3D &normal)

  initializes plane by giving position and normal vectors

- bool init (const Vec3D &pos0, const Vec3D &pos1, const Vec3D &pos2)

  initializes plane by giving three points on the plane

- void **dump** () const

Public Attributes

- Vec3D **pos**
- Vec3D **normal**

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

- engine/math/hcPlane.h
- engine/math/hcPlane.cpp

4.23 hcPlaneND Class Reference

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

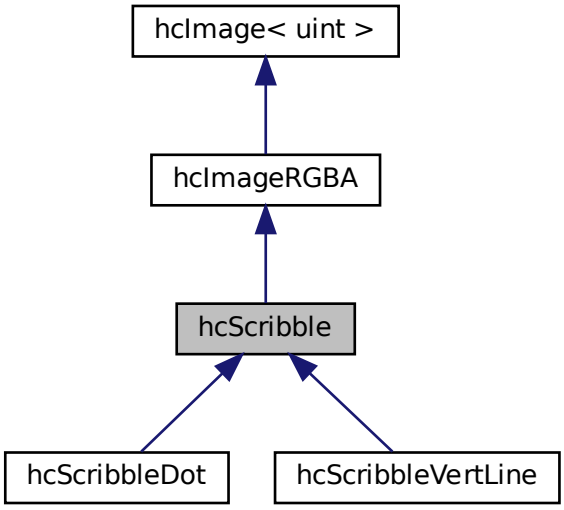
- engine/math/hcPlane.h

4.24 hcScribble Class Reference

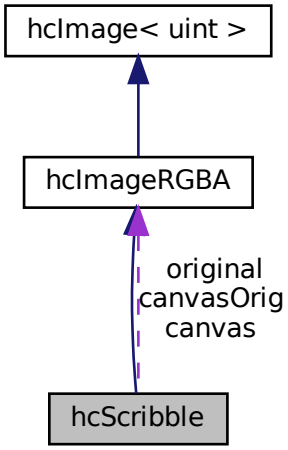
implements objects (lines, points, crosses, whatever) for scribbeling in hcImages

```
#include <hcImage.h>
```

Inheritance diagram for hcScribble:



Collaboration diagram for hcScribble:



Public Member Functions

- **hcScribble** (const hcScribble &other)
- hcScribble & **operator=** (const hcScribble &other)
- void **initNULL** ()

- virtual void **clear** ()
- void **init** (double relWidth, double relPosX, double relPosY, hcImageRGBA *canvas)
- void setCanvas (hcImageRGBA *canvas)

  sets canvas to be painted upon

- void reinit ()

  this function empties canvasOrig so that a new canvas can be loaded without parts of the old one be dumped in it
- bool updateScaledScribble ()

  adjusts this hcImageRGBA according to relative position / width

- bool updatePosition ()

  recomputes the insertPos of the scribble according to relPos

- bool draw ()

  draws this scribble at the specified position

- bool redrawOrig ()

  redraws the original part of the image where the scribble was drawn

- void **dump** () const

Public Attributes

- hcImageRGBA * canvas

  not-owning pointer to the image to be scribbled in

- hcImageRGBA canvasOrig

  canvas content that has been overdrawn by this scribble (TODO: this allows scribbles to think other scribbles are the original image)
- hcImageRGBA original

  not scaled image to be scribbled in canvas

- int insertPosX

  x-pos of this in canvas

- int insertPosY

  y-pos of this in canvas

- double relPosX

  relative (-1 < x < 1) x-pos of this in canvas

- double relPosY

  relative (-1 < x < 1) y-pos of this in canvas

- double newRelPosX

  where to draw when draw-function is called next time

- double newRelPosY

  where to draw when draw-function is called next time

- double relWidth

  relative width of this scribble in canvas

Additional Inherited Members

4.24.1 Detailed Description

implements objects (lines, points, crosses, whatever) for scribbeling in hcImages

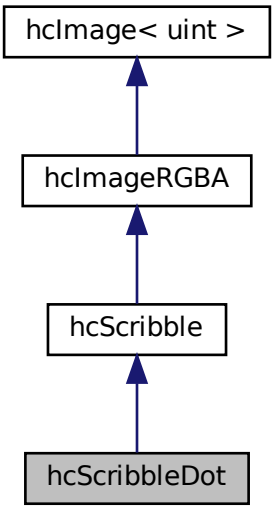
<

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

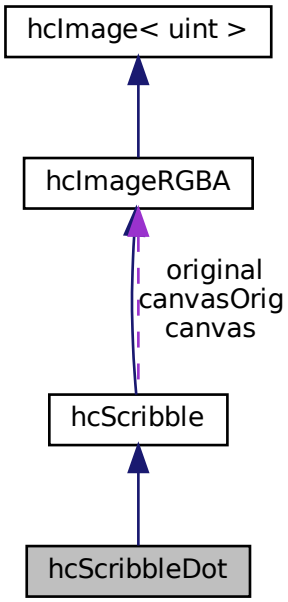
- engine/hcImage.h
- engine/hcImage.cpp

4.25 hcScribbleDot Class Reference

Inheritance diagram for hcScribbleDot:



Collaboration diagram for hcScribbleDot:



Public Member Functions

- hcScribbleDot ()

  std constructor

- hcScribbleDot (const hcScribbleDot &other)

  cpy constructor

- **hcScribbleDot** (uint color, double relWidth, double relPosX, double relPosY, hcImageRGBA *canvas)
- hcScribbleDot & **operator=** (const hcScribbleDot &other)
- void **initNULL** ()
- virtual void **clear** ()
- void **init** (uint color, double relWidth, double relPosX, double relPosY, hcImageRGBA *canvas)
- void **dump** () const

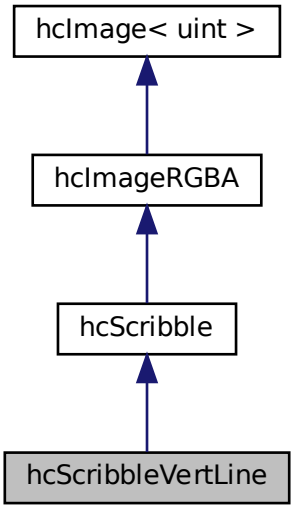
Additional Inherited Members

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

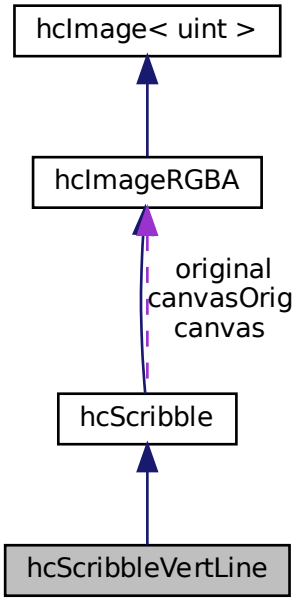
- engine/hcImage.h
- engine/hcImage.cpp

4.26 hcScribbleVertLine Class Reference

Inheritance diagram for hcScribbleVertLine:



Collaboration diagram for hcScribbleVertLine:



Public Member Functions

- hcScribbleVertLine ()
  std constructor
- hcScribbleVertLine (const hcScribbleVertLine &other)
  cpy constructor
- **hcScribbleVertLine** (uint color, double relWidth, double relPosX, double relPosY, hcImageRGBA *canvas)
- hcScribbleVertLine & **operator=** (const hcScribbleVertLine &other)
- void **initNULL** ()
- virtual void **clear** ()
- void **init** (uint color, double relWidth, double relPosX, double relPosY, hcImageRGBA *canvas)
- void **dump** () const

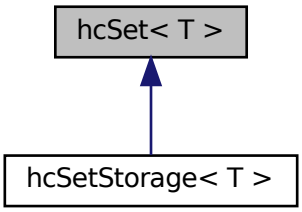
Additional Inherited Members

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

- engine/hcImage.h
- engine/hcImage.cpp

4.27 hcSet< T > Class Template Reference

Inheritance diagram for hcSet< T >:



Public Member Functions

- **hcSet** (uint numMaxSlots=4)
- **hcSet** (const hcSet< T > &other)
- hcSet< T > & **operator=** (const hcSet< T > &other)
- T & **operator[ ]** (uint i) const
- void **clear** ()
- void **initNULL** ()
- void **init** (uint numMaxSlots)

- virtual void **expand** (uint numNewElements)
- virtual int **findElement** (const T &object)
- virtual bool **hasElement** (const T &object)
- virtual bool isSlotOccupied (uint n) const

  is slot n occupied?

- int appendElement (T &object)

  appends given object to this set

- bool appendElementAtPos (T &object, uint pos)

  appends element at a specific position if possible

- virtual int removeElement (T &object)

  remove given object from this set

- T & elementAt (uint num)

  returns reference to element at position num

- void **dump** () const

Public Attributes

- uint **numMaxSlots**
- uint **numOccupiedSlots**
- uint **numFreeSlots**
- uint **firstFreeSlot**
- uint **lastOccSlot**
- T ∗∗ **elements**
- bool ∗ **occupied**

4.27.1 Member Function Documentation

4.27.1.1 appendElement()

```
template<class T >
int hcSet< T >::appendElement (
            T & object )
```

appends given object to this set

Returns

position where object is stored

4.27.1.2 removeElement()

```
template<class T >
int hcSet< T >::removeElement (
            T & object )  [virtual]
```

remove given object from this set

Returns

position where object has been found and removed or -1 on failure

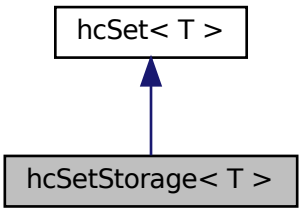
Reimplemented in hcDualSet< T1, T2 >, and hcSetStorage< T >.

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

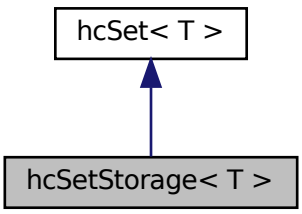
- engine/hcSet.h

4.28 hcSetStorage< T > Class Template Reference

Inheritance diagram for hcSetStorage< T >:



Collaboration diagram for hcSetStorage< T >:



Public Member Functions

- hcSetStorage (uint numMaxSlots=1)
  std constructor
- hcSetStorage (const hcSetStorage &other)
  cpy constructor
- hcSetStorage< T > & **operator=** (const hcSetStorage< T > &other)
- virtual int removeElement (T &object)
  removes and destroys an element
- bool extractElement (T &object)
  removes an element and transfers ownership

Additional Inherited Members

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

- engine/hcSet.h

4.29 hcSortedList< T > Class Template Reference

Public Member Functions

- **hcSortedList** (uint numMaxSlots=128)
- **hcSortedList** (const hcSortedList< T > &other)
- hcSortedList< T > & **operator=** (const hcSortedList< T > &other)
- hcSortedList< T > & **operator=** (const hcSortedListStorage< T > &other)
- T & **operator[ ]** (uint i) const
- void **clear** ()
- void **initNULL** ()
- void **init** (uint numMaxSlots=1024)
- virtual void **expand** (uint numNewElements)
- virtual int **findElement** (const T &object)
- virtual bool **hasElement** (const T &object)
- uint insertElement (T &object)
  appends given object to this list
- virtual int removeElement (T &object)
  remove given object from this set
- bool **sort** ()
- void **dump** () const

Public Attributes

- uint **numElements**
- uint **numMaxSlots**
- T ** **elements**

4.29.1 Member Function Documentation

4.29.1.1 insertElement()

```
template<class T >
uint hcSortedList< T >::insertElement (
            T & object )
```

appends given object to this list

Returns

position where object is stored

4.29.1.2 removeElement()

```
template<class T >
int hcSortedList< T >::removeElement (
            T & object )  [virtual]
```

remove given object from this set

Returns

position where object has been found and removed or -1 on failure

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

- engine/hcSortedList.h

4.30 hcSortedListStorage< T > Class Template Reference

Public Member Functions

- hcSortedListStorage (uint numMaxSlots=128)
  std constructor
- hcSortedListStorage (const hcSortedListStorage &other)
  cpy constructor
- hcSortedListStorage< T > & **operator=** (const hcSortedListStorage< T > &other)
- virtual int removeElement (T &object)
  removes and destroys an element
- bool extractElement (T &object)
  removes an element and transfers ownership

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

- engine/hcSortedList.h

4.31 hcSphere< dim > Class Template Reference

mathematical model and functions for nD-spheres

`#include <hcSphere.h>`

Public Member Functions

- hcSphere ()
  std constructor
- hcSphere (const hcSphere< dim > &sphere)
  cpy constructor
- hcSphere (const Vec< dim, hcFloat > &pos, float radius)
  constructor
- ~hcSphere ()
  destructor
- hcSphere< dim > & operator= (const hcSphere< dim > &other)
  assignment operator
- void **init** (const Vec< dim, hcFloat > &pos, float radius)
- void **dump** () const

4.31.1 Detailed Description

template<uint dim>
class hcSphere< dim >

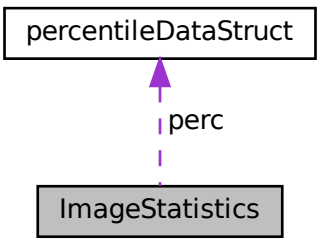
mathematical model and functions for nD-spheres

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

- engine/math/hcLine.h
- engine/math/hcSphere.h

4.32 ImageStatistics Class Reference

Collaboration diagram for ImageStatistics:



Public Member Functions

- **ImageStatistics** (const ImageStatistics &other)
- **ImageStatistics** (percentiles perc, uint numPixels, hcFloat mean, hcFloat stddev)
- ImageStatistics & operator= (const ImageStatistics &other)

  assignment operator

- void operator+= (const ImageStatistics &other)

  operator for computing "average" statistics

- void operator/= (hcFloat factor)

  operator for computing "average" statistics

- void **initNULL** ()
- string **toString** () const
- void **dump** () const

Public Attributes

- uint numPixels

  number of non-zero pixels

- percentiles perc

  percentiles of content

- hcFloat mean

  mean value of image content

- hcFloat stddev

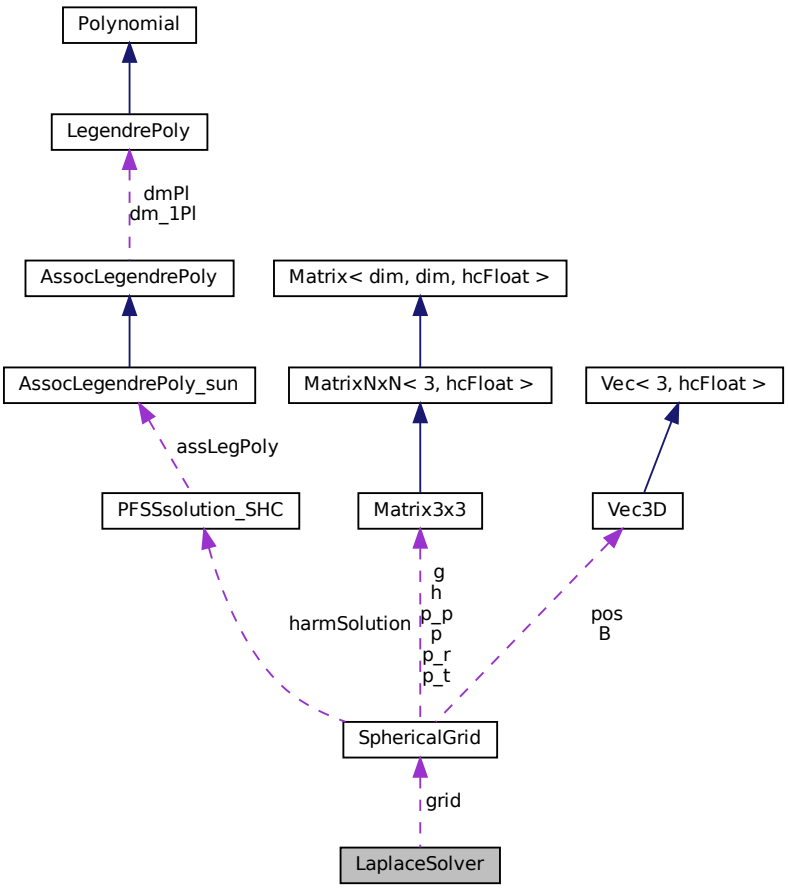
  standard deviation of mean of image content

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

- src/imageStatistics.h
- src/imageStatistics.cpp

4.33 LaplaceSolver Class Reference

Collaboration diagram for LaplaceSolver:



Public Member Functions

- LaplaceSolver ()
  std constructor
- LaplaceSolver (const LaplaceSolver &solver)
  cpy constructor
- LaplaceSolver & **operator=** (const LaplaceSolver &other)
- bool **init** (bool sinLatGrid, hcFloat maxSinLat, hcFloat r_ss, uint numR, hcFloat ell)
- void iterateSpheric ()
- void iterateElliptic_MT ()
- void iterateElliptic_ST ()
- void iterateElliptic ()
- void iterate_CPU ()
- int computeSolution (hcImageFITS &photImage)
  computes solution of Laplace's equation via finite differences
- void **dump** () const

Static Public Member Functions

- static void * iterateElliptic_threadEntry (void *parameter)

  thread entry point forelliptic iteration

Public Attributes

- SphericalGrid * **grid**
- bool solutionComputed

  has some computation been done on this?

4.33.1 Member Function Documentation

4.33.1.1 iterate_CPU()

```
void LaplaceSolver::iterate_CPU ( )
```

iteration switcher for spheric/elliptic grid with unit/non-unit base vectors

4.33.1.2 iterateElliptic()

```
void LaplaceSolver::iterateElliptic ( )
```

iteration procedure for general spheric/elliptic grid with general (non-unit) base vectors

4.33.1.3 iterateElliptic_MT()

```
void LaplaceSolver::iterateElliptic_MT ( )
```

iteration procedure for general spheric/elliptic grid with general (non-unit) base vectors, multi-threaded

4.33.1.4 iterateElliptic_ST()

```
void LaplaceSolver::iterateElliptic_ST ( )
```

iteration procedure for general spheric/elliptic grid with general (non-unit) base vectors, single-threaded

4.33.1.5 iterateElliptic_threadEntry()

```
void * LaplaceSolver::iterateElliptic_threadEntry (
            void * parameter )  [static]
```

thread entry point forelliptic iteration

entry function for multicore PFSS iteration function

4.33.1.6 iterateSpheric()

```
void LaplaceSolver::iterateSpheric ( )
```

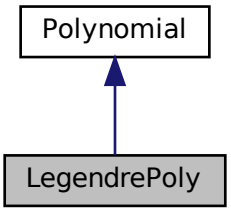
iteration procedure only for spherical grid with unit base vectors

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

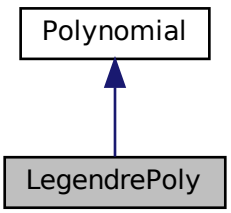
- src/laplaceSolver.h
- src/laplaceSolver.cpp

4.34 LegendrePoly Class Reference

Inheritance diagram for LegendrePoly:



Collaboration diagram for LegendrePoly:



Public Member Functions

- LegendrePoly (uint order=0)
  std constructor
- LegendrePoly (const LegendrePoly &other)
  cpy constructor
- ~LegendrePoly ()
  destructor
- LegendrePoly & operator= (const LegendrePoly &other)
  assignment operator
- void **initNULL** ()
- void **clear** ()
- void **init** (uint n)

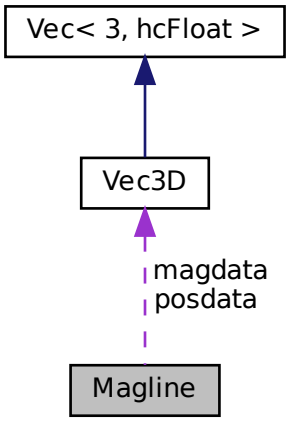
Additional Inherited Members

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

- engine/math/hcFunction.h
- engine/math/hcFunction.cpp

4.35 Magline Class Reference

Collaboration diagram for Magline:



Public Member Functions

- Magline ()
  std constructor
- Magline (const Magline &other)
  cpy constructor
- virtual ~Magline ()
  destructor
- Magline & operator= (const Magline &other)
  assignment operator
- void **init** ()
- void **initNULL** ()
- void **clear** ()
- Vec3D getMagVec (uint num)
  returns magnetic vector at given numerical position in array
- Vec3D getPosVec (uint num)
  returns positional vector (spherical coordinates) at given numerical position in array
- bool getValuesAtHeight (hcFloat height, Vec3D &posData, Vec3D &magData)
  (linearily) interpolates values at sample points to obtain intermediate values at a specific height
- int getAllValuesAtHeight (hcFloat height, Vec3D *posData, Vec3D *magData)
  returns number of positions where this maglines pierces through a specific height
- unsigned char createMaglineThroughPos (SphericalGrid &grid, const Vec3D &posParam, bool debug, hcFloat epsMin=1E1, hcFloat epsMax=1E2)
  tracks a magnetic field line through a specific position
- bool lowerDistLTupperDist (const Magline &other)
  checks if the distance between the lower endings of two lines is less than their upper distance
- bool isInSameFluxtubeAs (const Magline &other)
  calls lowerDistLTupperDist, this criterion is just sufficient
- bool exportBinary (std::ofstream &stream)
  dumps instance into stream
- bool importBinary (std::ifstream &stream)
  imports instance from stream
- void dump () const
  dump data structure to screen

Static Public Member Functions

- static void initStaticMembers ()
  initialize static member variables

Public Attributes

- uint numSamples

  number of points where the magline has been sampled

- Vec3D * magdata

  array of magnetic field strengths where magline has been sampled (spherical)

- Vec3D * posdata

  array of positions where magline has been sampled (spherical)

- bool closed

  does the magline connect two points on the phtotosphere?

- bool valid

  are there valid values in magdata and posdata?

- bool polarity

  true if coming out of surface, false if going in, don't care if valid=false

Static Public Attributes

- static uint **colorInvalid**
- static uint **colorClosed**
- static uint **colorPositive**
- static uint **colorNegative**

4.35.1 Member Function Documentation

4.35.1.1 createMaglineThroughPos()

```
unsigned char Magline::createMaglineThroughPos (
            SphericalGrid & grid,
            const Vec3D & posParam,
            bool debug,
            hcFloat epsMin = 1E1,
            hcFloat epsMax = 1E2 )
```

tracks a magnetic field line through a specific position

produces the magnetic and positional values of a magnetic field line.

Parameters

grid	utilized for tracking field line
posParam	position through which to draw the magline (spherical/elliptical coordinates)
debug	gives debugging information throughout computation
stepSize	distance between two data samples in the visualization

Returns

error value (see below)

(1 << 0) = 1 maglineDown empty (1 << 1) = 2 maglineDown has too many samples

(1 << 2) = 4 maglineUp empty (1 << 3) = 8 maglineUp has too many samples

(1 << 4) = 16 both directions have no samples (except first point) (1 << 5) = 32 both directions combined have too many samples

(1 << 6) = 64 magline connects two points on the source surface (1 << 7) = 128 start position not valid

4.35.1.2 getAllValuesAtHeight()

```
int Magline::getAllValuesAtHeight (
            hcFloat height,
            Vec3D * posData,
            Vec3D * magData )
```

returns number of positions where this maglines pierces through a specific height

Parameters

height	heliocentric height where this magline is to be sampled
posData	positions of sampled magline (spherical coordinates)
magData	magnetic direction vectors at height (spherical coordinates)

Return values

number	of positions where magline pierces through height if more than MAGLINE_NUM_POSITIONS positions exist, only the first MAGLINE_NUM_POSITIONS will be reported

posData and magData are pre-allocated arrays of size MAGLINE_NUM_POSITIONS

4.35.1.3 getValuesAtHeight()

```
bool Magline::getValuesAtHeight (
            hcFloat height,
            Vec3D & posData,
            Vec3D & magData )
```

(linearily) interpolates values at sample points to obtain intermediate values at a specific height

Parameters

height	heliocentric height where this magline is to be sampled
posData	position of sampled magline (spherical coordinates)
magData	magnetic direction vector at height (spherical coordinates)

4.35.1.4 lowerDistLTupperDist()

```
bool Magline::lowerDistLTupperDist (
            const Magline & other )
```

checks if the distance between the lower endings of two lines is less than their upper distance

lower distance between maglines is less than (LT) the upper distance of the same maglines

Parameters

other	Magline to be compared to

Returns

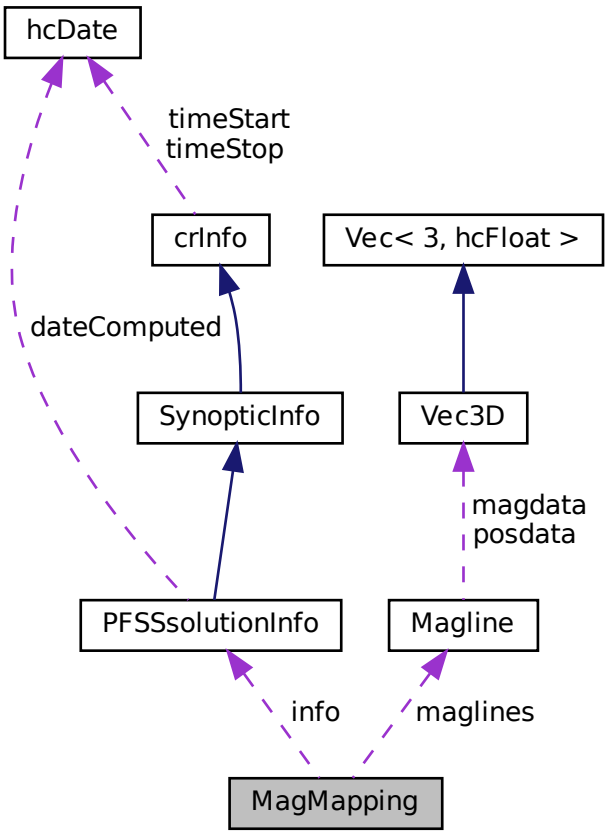
true, if distance at the lower boundary is less than distance at upper boundary, false otherwise

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

- src/magline.h
- main.cpp
- src/magline.cpp

4.36 MagMapping Class Reference

Collaboration diagram for MagMapping:



Public Member Functions

- MagMapping ()
  std constructor
- MagMapping (const MagMapping &other)
  cpy constructor
- MagMapping (const PFSSsolutionInfo &info, bool sinLatFormat, bool compCoords, hcFloat maxSinLat, uint numTheta, uint numPhi, hcFloat height)
  constructor
- virtual ~MagMapping ()
  destructor
- MagMapping & operator= (const MagMapping &other)

assignment operator

- Magline & operator() (uint indTheta, uint indPhi)

  returns magline at position

- void init (const PFSSsolutionInfo &info, bool sinLatFormat, bool compCoords, hcFloat maxSinLat, uint numTheta, uint numPhi, hcFloat height)

  initializer

- bool isComputed () const

  true if mapping has been computed

- uint **getNumTheta** ()
- uint **getNumPhi** ()
- Vec3D getCoords (uint j, uint k)

  returns the coordinates for pixel (j,k)

- bool setCoords (uint j, uint k, const Vec3D &coord)

  sets the coordinates for pixel (j,k)

- hcFloat getHeight () const
- bool exportBinary (const string &filename)

  exports mapping to binary file

- bool importBinary (const string &filename)

  imports mapping from binary file

- hcFloat diffFootpoints (MagMapping &other)

  computes averaged squared distance of photospheric footpoints

- hcFloat getOpenFlux ()

  computes the open magnetic flux through mapping

- virtual bool createAtHeight (const PFSSsolutionInfo &info, LaplaceSolver &solver, hcFloat height, uint numThetaIn, uint numPhiIn, hcFloat maxSinLatIn, bool sinLatFormat, bool compCoords)

  create Mapping at desired height level

- bool createAtHeight_SP (const PFSSsolutionInfo &info, LaplaceSolver &solver, hcFloat height, uint numThetaIn, uint numPhiIn, hcFloat maxSinLatIn, bool sinLatFormat, bool compCoords)

  single core version of magnetic field line creation

- bool createAtHeight_MP (const PFSSsolutionInfo &info, LaplaceSolver &solver, hcFloat height, uint numThetaIn, uint numPhiIn, hcFloat maxSinLatIn, bool sinLatFormat, bool compCoords)

  multi core version of magnetic field line creation

- void createMappingHeader (char *header, hcFloat height, hcFloat *otherHeights, uint numHeights, hcFloat lowerR, hcFloat upperR)

  gives human readable explanation of what is in exported ASCII file

- string getExpansionFactorComment ()
- bool exportImagePolarity ()

  creates polarity image of this mapping

- bool exportImageFootpoint (string filename)

creates polarity image of photospheric footpoints for each magline in this mapping

- bool exportASCII (string filename, hcFloat *heights, uint numHeights, hcFloat lowerR, hcFloat sourceSurfaceR)

  exports mag mapping with connection to several height levels to txt- and image-file

- bool exportImageExpansionFactor ()

  export expansion factor of magnetic field lines in this map

- bool exportImageExpansionFactorMax ()

  export maximum expansion factor of magnetic field lines in this map

- bool exportImageMagfield ()

  export magnetic field at this height

- void dump (uint indent=0)

  dumps information on instance to stdout with optional indentation

Static Public Member Functions

- static void * createAtHeight_threadEntryPoint (void *parameter)

  thread entry point for createAtHeight_MP

Public Attributes

- PFSSsolutionInfo **info**
- bool sinLatFormat

  azimuthal spacing linear in latitude (false, TODO not working) or sine(latitude) (true)

- hcFloat maxSinLat

  sinLat-Boundary in case of sinLatFormat==true

- bool compCoords

  height level in computationalCoords

- uint numTheta

  number of "pixels" in meridional direction

- uint numPhi

  number of "pixels" in zonal direction

- Magline ** maglines

  array of magnetic field lines this mapping consists of

4.36.1 Member Function Documentation

4.36.1.1 createAtHeight()

```
bool MagMapping::createAtHeight (
            const PFSSsolutionInfo & info,
            LaplaceSolver & solver,
            hcFloat height,
            uint numThetaIn,
            uint numPhiIn,
            hcFloat maxSinLatIn,
            bool sinLatFormatIn,
            bool compCoords )  [virtual]
```

create Mapping at desired height level

computed and stored entirely in computatinal coordinates(vectors, transformation to physical coordinates/vectors is done afterwards when re-importing data

Parameters

solver	gives the environment in which maglines shall be created
height	the height above photosphere where equidistant "pixels" are evaluated
numTheta	the number of pixels in theta direction (if 0, the entire mapping is computed in the native grid of solver)
numPhi	the number of pixels in phi direction
maxSinLat	the highest point in mapping
minSinLat	the lowest point in mapping
sinLatGrid	evenly spaced in sinLat or in lat?
compCoords	height given in computational coordinates (true) or world coordinates (false)

Returns

success

4.36.1.2 createAtHeight_MP()

```
bool MagMapping::createAtHeight_MP (
            const PFSSsolutionInfo & info,
            LaplaceSolver & solver,
            hcFloat height,
            uint numTheta,
            uint numPhi,
            hcFloat maxSinLat,
            bool sinLatFormat,
            bool compCoords )
```

multi core version of magnetic field line creation

multi-threaded version of magnetic field line creation

4.36.1.3 createAtHeight_threadEntryPoint()

```
void * MagMapping::createAtHeight_threadEntryPoint (
            void * parameter )  [static]
```

thread entry point for createAtHeight_MP

entry function for multicore creation of magnetic field lines

4.36.1.4 diffFootpoints()

```
hcFloat MagMapping::diffFootpoints (
            MagMapping & other )
```

computes averaged squared distance of photospheric footpoints

compares photospheric footpoints of back mapped magnetic field lines

This function only makes sense if it is a backmapping from source surface down to the photosphere

4.36.1.5 exportASCII()

```
bool MagMapping::exportASCII (
            string filename,
            hcFloat * heights,
            uint numHeights,
            hcFloat lowerR,
            hcFloat upperR )
```

exports mag mapping with connection to several height levels to txt- and image-file

outputs textfiles where every line corresponds to one pixel position of the map

the output format is described in the header of the output file

4.36.1.6 getExpansionFactorComment()

```
string MagMapping::getExpansionFactorComment ( )
```

create comment to add to FITS files

4.36.1.7 getHeight()

```
hcFloat MagMapping::getHeight ( ) const
```

returns base height of this map

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

- src/magMapping.h
- src/magMapping.cpp

4.37 Matrix< rows, cols, T > Class Template Reference

implementation of the mathematical matrix construct, only float values supported so far

`#include <hcMatrix.h>`

Public Member Functions

- Matrix ()
  std constructor
- template<class S >
  Matrix (const Matrix< rows, cols, S > &other)
  copy constructor
- Matrix (const Vec< rows, T > &v1, const Vec< rows, T > &v2)
  create column-matrix
- Matrix (const Vec< rows, T > &v1, const Vec< rows, T > &v2, const Vec< rows, T > &v3)
  create column-matrix
- ~Matrix ()
  destructor
- template<class S >
  Matrix & **operator=** (const Matrix< rows, cols, S > &other)
- Matrix & operator*= (T scale)
  scales the matrix
- Matrix & operator/= (T scale)
  scales the matrix
- Matrix & operator+= (const Matrix< rows, cols, T > &other)
  adds another matrix to this
- Matrix & operator-= (const Matrix< rows, cols, T > &other)
  adds another matrix to this
- Vec< rows, T > operator* (const Vec< cols, T > &vec)
  *this * vec*
- T & operator() (uint i, uint j)
  access to element in matrix at row j, column j
- T operator[ ] (uint i) const
  access to content in row-major ordering
- T getElement (uint i, uint j) const
  access to element in matrix at row j, column j
- void loadIdentity ()
  load identity matrix

- void loadZeroes ()
  initialize matrix with zeros
- void loadTransMat (const Vec< rows-1, T > &vec)
  creates a translation matrix in homogeneous coordinates
- void switchRows (uint i, uint j)
  changes rows i and j
- void **addRow** (uint dest, uint source, T scale)
- void scale (T scale)
  add scale ∗ row(source) to row(dest)
- void scaleRow (uint row, T scale)
  scales a specific row by scale
- uint solveSLE (Vec< rows, hcFloat > &result)
  solves a system of linear equations using gauss' method
- int translate (const Vec< rows-1, T > &vec)
  assumes homogeneous TF-matrix, translates scene by vec
- Matrix< cols, rows, T > transpose ()
  transposes matrix
- void **dump** () const
- void **subdump** ()
- template<class S >
  Matrix< rows, cols, T > & **operator=** (const Matrix< rows, cols, S > &other)

Public Attributes

- T content [rows ∗cols]
  contains the entries of the matrix in row-major ordering

4.37.1 Detailed Description

template<uint rows, uint cols, class T>
class Matrix< rows, cols, T >

implementation of the mathematical matrix construct, only float values supported so far

Stores entries of a matrix in row-major ordering in array content

4.37.2 Member Function Documentation

4.37.2.1 scale()

```
template<uint rows, uint cols, class T >
void Matrix< rows, cols, T >::scale (
            T scale )
```

add scale ∗ row(source) to row(dest)

scales every entry

4.37.2.2 solveSLE()

```
template<uint rows, uint cols, class T >
uint Matrix< rows, cols, T >::solveSLE (
            Vec< rows, hcFloat > & result )
```

solves a system of linear equations using gauss' method

Solves a System of Linear Equations (Ax=y) posed by the matrix "Ay" for x, where the rightmost column is interpreted as the solution vector y. The Gaussian elimination algorithm is used. So far, only unique solutions are stored in the vector result. The matrix Ay is not altered but copied to a local object

Return values

0	there is no solution or ill posed problem
1	unique solution, stored now in result
2	there is an infinte number of solution

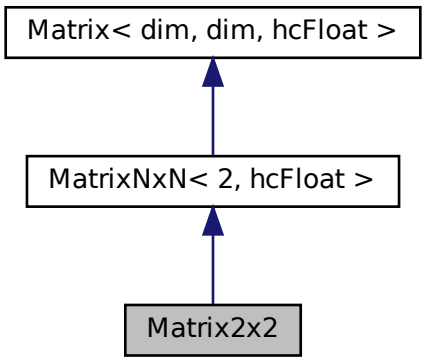
only in case 1 does the result vector contain useful data

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

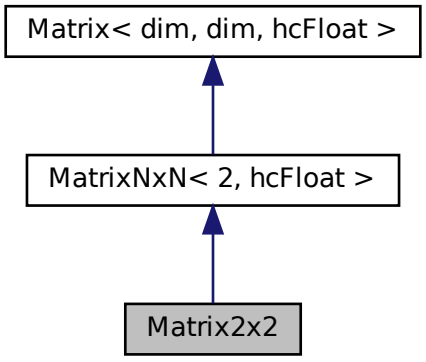
- engine/math/hcMatrix.h

4.38 Matrix2x2 Class Reference

Inheritance diagram for Matrix2x2:



Collaboration diagram for Matrix2x2:



Public Member Functions

- Matrix2x2 ()

  std constructor

- Matrix2x2 (const Matrix2x2 &other)

  cpy constructor

- template<class S >
  Matrix2x2 (const MatrixNxN< 2, S > &other)

pseudo cpy constructor

- Matrix2x2 & **operator=** (const Matrix2x2 &other)
- template<class S >
  Matrix2x2 & **operator=** (const MatrixNxN< 2, S > &other)
- template<class S >
  Matrix2x2 & **operator=** (const Matrix< 2, 2, S > &other)
- template<class S >
  void **scalex** (S scale)
- template<class S >
  void **scaley** (S scale)
- template<class S >
  void **scale** (S scalex, S scaley)

Additional Inherited Members

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

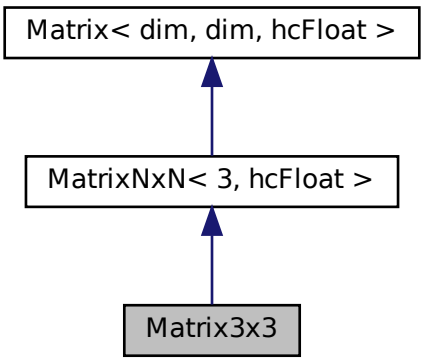
- engine/math/hcMatrix.h
- engine/math/hcMatrix.cpp

4.39 Matrix3x3 Class Reference

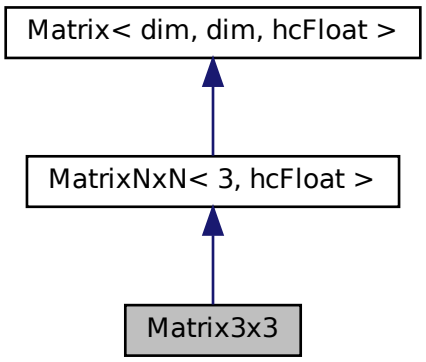
implements a 3x3 matrix for use, e.g., with homogeneus 2D space

```
#include <hcMatrix.h>
```

Inheritance diagram for Matrix3x3:



Collaboration diagram for Matrix3x3:



Public Member Functions

- **Matrix3x3** (const Matrix3x3 &other)
- template<class S >
  Matrix3x3 (const MatrixNxN< 3, S > &other)
- template<class S >
  Matrix3x3 (const Vec< 3, S > &vec1, const Vec< 3, S > &vec2, const Vec< 3, S > &vec3)
- Matrix3x3 & **operator=** (const Matrix3x3 &other)
- template<class S >
  Matrix3x3 & **operator=** (const MatrixNxN< 3, S > &other)
- void **convertSphericalToCartesian** (const Vec3D &cartPos)
- void **convertCartesianToSpherical** (const Vec3D &cartPos)
- template<class S >
  void scalex (S scale)

  loads the conversion matrix to convert from cartesian coords to spherical coords
- template<class S >
  void **scaley** (S scale)
- template<class S >
  void **scalez** (S scale)
- template<class S >
  void **scale** (S scalex, S scaley, S scalez)
- template<class S >
  void **loadRotationX** (S theta)
- template<class S >
  void **loadRotationY** (S theta)
- template<class S >
  void **loadRotationZ** (S theta)
- template<class S >
  void **rotateAroundAxis** (Vec3D axis, S angle)
- template<class S >
  void **loadEulerTransform** (S omega, S theta, S phi)

Additional Inherited Members

4.39.1 Detailed Description

implements a 3x3 matrix for use, e.g., with homogeneus 2D space

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

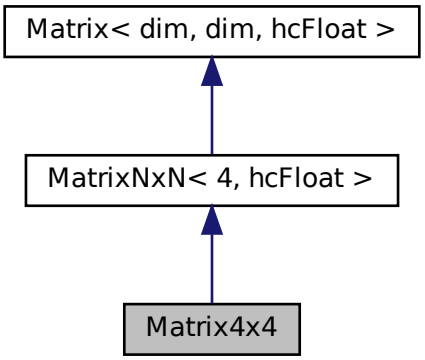
- engine/math/hcMatrix.h
- engine/math/hcMatrix.cpp

4.40 Matrix4x4 Class Reference

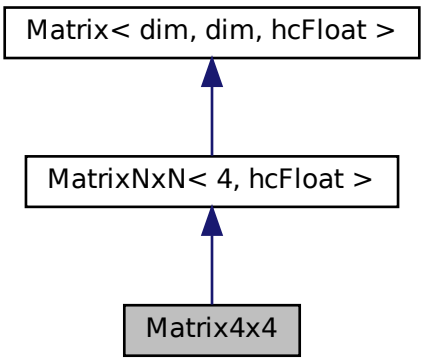
implements a 4x4 matrix for use, e.g., with homogeneous 3D space

`#include <hcMatrix.h>`

Inheritance diagram for Matrix4x4:



Collaboration diagram for Matrix4x4:



Public Member Functions

- Matrix4x4 ()
  std constructor
- Matrix4x4 (const Matrix4x4 &other)
  cpy constructor
- template<class S >
  Matrix4x4 (const MatrixNxN< 4, S > &other)
  pseudo-cpy constructor
- ~Matrix4x4 ()
  std constructor
- Matrix4x4 & **operator=** (const Matrix4x4 &other)
- template<class S >
  Matrix4x4 & **operator=** (const MatrixNxN< 4, S > &other)
- void **loadTFMatrix** (const Vec3D &right, const Vec3D &up, const Vec3D &back, const Vec3D &pos)
- template<class S >
  void **scalex** (S scale)
- template<class S >
  void **scaley** (S scale)
- template<class S >
  void **scalez** (S scale)
- template<class S >
  void **scalew** (S scale)
- template<class S >
  void **scale** (S scalex, S scaley, S scalez, S scalew)
- template<class S >
  void **rotatex** (S phi)
- template<class S >
  void **rotatey** (S phi)
- template<class S >
  void **rotatez** (S phi)
- void rotateAtoB (const Vec3D a, const Vec3D b)
  creates a rotation matrix that rotates a onto b
- template<class S >
  void **translate** (S x, S y, S z)
- void translate (const Vec< dim-1, T > &v)
  assumes homogeneous TF-matrix, translates along vec v in N-1 dimensions

Additional Inherited Members

4.40.1 Detailed Description

implements a 4x4 matrix for use, e.g., with homogeneous 3D space

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

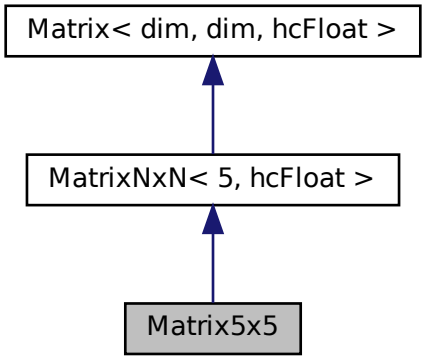
- engine/math/hcMatrix.h
- engine/math/hcMatrix.cpp

4.41 Matrix5x5 Class Reference

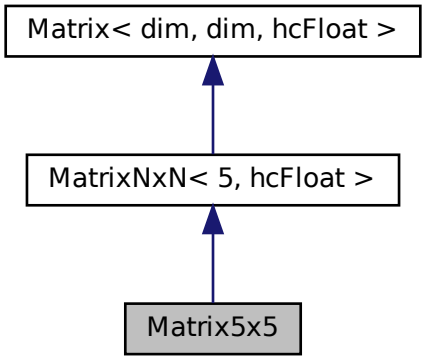
implements a 5x5 matrix for use, e.g., with homogeneous 4D space

`#include <hcMatrix.h>`

Inheritance diagram for Matrix5x5:



Collaboration diagram for Matrix5x5:



Public Member Functions

- **Matrix5x5** (const Matrix5x5 &other)
- template<class S >
  Matrix5x5 (const MatrixNxN< 5, S > &other)
- Matrix5x5 & **operator=** (const Matrix5x5 &other)

- template<class S >
  Matrix5x5 & **operator=** (const MatrixNxN< 5, S > &other)
- void **loadTFMatrix** (const Vec4D &right, const Vec4D &up, const Vec4D &back, const Vec4D &over, const Vec4D &pos)
- template<class S >
  void **rotateXY** (S phi)
- template<class S >
  void **rotateXZ** (S phi)
- template<class S >
  void **rotateYZ** (S phi)
- template<class S >
  void **rotateXW** (S phi)
- template<class S >
  void **rotateYW** (S phi)
- template<class S >
  void **rotateZW** (S phi)
- template<class S >
  void **translate** (S x, S y, S z)

Additional Inherited Members

4.41.1 Detailed Description

implements a 5x5 matrix for use, e.g., with homogeneous 4D space

TODO: not tested

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

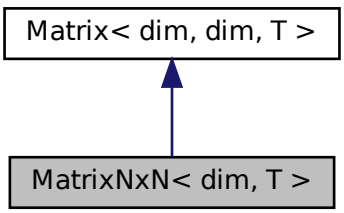
- engine/math/hcMatrix.h
- engine/math/hcMatrix.cpp

4.42 MatrixNxN< dim, T > Class Template Reference

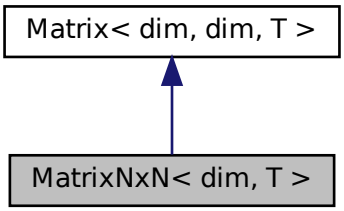
implementation of a square matrix

`#include <hcMatrix.h>`

Inheritance diagram for MatrixNxN< dim, T >:



Collaboration diagram for MatrixNxN< dim, T >:



Public Member Functions

- MatrixNxN ()
  std constructor
- template<class S >
  MatrixNxN (const MatrixNxN< dim, S > &other)
  cpy constructor
- ~MatrixNxN ()
  destructor
- MatrixNxN< dim, T > & operator= (const MatrixNxN< dim, T > &other)
  assignment operator
- double det ()
  computes determinant of matrix (double precision)
- int invert ()
  computes inverse of matrix
- void loadIdentity ()
  initializes matrix with ones on main diagonal
- void scaleAllDim (T scale)
  applies a scale operation on all dimensions
- void scaleHom (T scale)
  assumes homogeneous TF-matrix, scales the scene isotropically
- void scaleAxis (uint n, T scale)
  assumes homogeneous TF-matrix, scales along a specific axis
- void translate (const Vec< dim-1, T > &v)
  assumes homogeneous TF-matrix, translates along vec v in N-1 dimensions
- void **dump** () const

Additional Inherited Members

4.42.1 Detailed Description

template<uint dim, class T>
class MatrixNxN< dim, T >

implementation of a square matrix

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

- engine/math/hcMatrix.h

4.43 percentileDataStruct Struct Reference

Public Member Functions

- **percentileDataStruct** (const string &perc00, const string &perc01, const string &perc02, const string &perc05, const string &perc10, const string &perc20, const string &perc25, const string &perc30, const string &perc40, const string &perc50, const string &perc60, const string &perc70, const string &perc75, const string &perc80, const string &perc90, const string &perc95, const string &perc98, const string &perc99, const string &perc100)
- percentileDataStruct & **operator=** (const percentileDataStruct &other)
- void **operator+=** (const percentileDataStruct &other)
- void **operator/=** (hcFloat factor)
- string **toString** () const
- void **dump** ()

Public Attributes

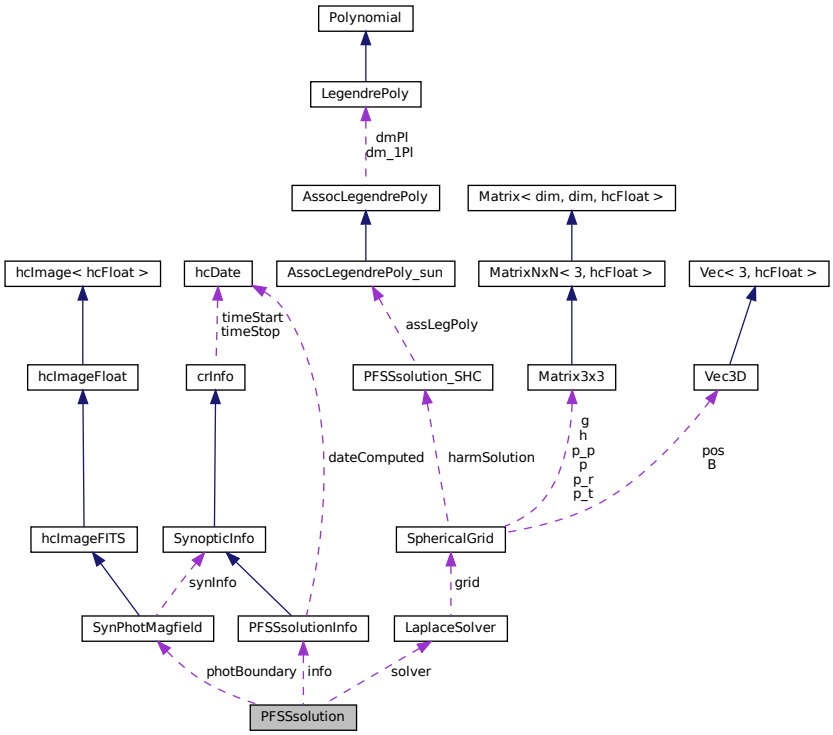
- hcFloat **perc00**
- hcFloat **perc01**
- hcFloat **perc02**
- hcFloat **perc05**
- hcFloat **perc10**
- hcFloat **perc20**
- hcFloat **perc25**
- hcFloat **perc30**
- hcFloat **perc40**
- hcFloat **perc50**
- hcFloat **perc60**
- hcFloat **perc70**
- hcFloat **perc75**
- hcFloat **perc80**
- hcFloat **perc90**
- hcFloat **perc95**
- hcFloat **perc98**
- hcFloat **perc99**
- hcFloat **perc100**

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

- engine/hcTools.h

4.44 PFSSsolution Class Reference

Collaboration diagram for PFSSsolution:



Public Member Functions

- PFSSsolution ()

  std constructor

- PFSSsolution (const PFSSsolution &other)

  cpy constructor

- void **initNULL** ()
- void **clear** ()
- void **init** ()
- bool isDirSet ()

  checks if output directory is set

- string getFilenamePhotMagfield ()

  get filename for photospheric synoptic magfield file

- string getFilenameGrid ()

  get filename for binary grid data

- string getFilenameConfig ()

  get filename for configuration file

- PFSSsolution & operator= (const PFSSsolution &other)

  assignment operator

- LaplaceSolver & getSolver ()

  returns reference to solver member variable

- bool loadPhotBoundary (const string &filename, uint scaleMethod=7)

  loads magnetic map of photosphere and initializes this object

- bool save ()

  stores solution computed by solver into file located at path

- bool load (const string &cfgFN)

  loads precomputed solution into this object

- bool computeKielSHC (const string &filename, uint order, hcFloat r_ss, uint resCompR)

  compute solution according to Spherical Harmonic Coefficient approach

- bool computeKielGrid (const string &filename, const string &optionalID, hcFloat r_ss, uint compResR, hcFloat surfShapeA, uint scaleMethod=7)

  computes finite difference laplace equation via GPU or CPU

- bool mapHeightLevel (hcFloat height, hcFloat maxSinLat, uint numTheta, uint numPhi, bool sinLatFormat=true, bool computationalCoords=true)

  routine for computing magnetic height maps in memory

- string getMagneticMappingFilename (const MagMapping &magmap)

  getting filename for export/import of magnetic mappings

- bool multiMapSolution (uint numTheta, uint numPhi, bool computeIntermediateHeightLevels, bool sinLatFormat, bool computaionalCoords)

  computes magnetic field line maps to several height levels

- bool computeAndMapKielSHC (const string &filename, uint order, hcFloat r_ss, uint compResR, uint mapResTheta, uint mapResPhi, bool mapIntermediateHeights)

  computes spherical harmonic coefficients for given photospheric magfield

- bool computeAndMapKielGrid (const string &filename, const char *optionalID, hcFloat r_ss, uint compResR, uint mapResTheta, uint mapResPhi, bool mapIntermediateHeights, hcFloat ellipticity, uint scaleMethod=7)

  computes and maps Kiel grid approach

- bool loadAndMapKielGrid (const string &filename, uint mapResTheta, uint mapResPhi, hcFloat height=0.0)

  loads Kiel grid solution and computes mapping

- bool loadAndMapStanfordSHC (const char *photFilename, const char *StanfordCoeffFilename, uint compResR, uint imgThetaRes, uint imgPhiRes, bool mapIntermediateHeights)

  loads Stanford SHC file and computes mapping

- bool batchKielSHC (const string &inDir, hcFloat r_ss, uint resCompR, uint order)

  computes and maps Kiel SHC approach for all files in inDir

- bool batchKielGrid (const string &inDir, hcFloat r_ss, uint resCompR, hcFloat ellipticity=1.0)

  computes and maps Kiel grid approach for all files in inDir

- bool batchMap (uint resMapTheta, uint resMapPhi, hcFloat height=0.0)

compute maps for all computes solutions in dirData

- void paramStudyThresh (const char *inDir, uint compRadialRes, uint compThetaRes, uint compPhiRes, uint imgThetaRes, uint imgPhiRes)

  analyses impact of solver threshold level on results

- void paramStudyRss (const char *inDir, uint compRadialRes, uint compThetaRes, uint compPhiRes, uint imgThetaRes, uint imgPhiRes)

  analyses impact of source surface height on results

- void paramStudyRes ()

  analyzes impact of grid point numbers on results

- void paramStudyScaleMethod (const char *filename)

  analyses impact on image scaling algorithm on results

- void paramStudyRadialRes (const char *filename)

  parameter study regarding radial resolution of computational grid

- void compareAnaInter (const char *inDir)

  comparison study regarding analytical and interpolated magnetic field line tracking

- void **compareSHCorders** (const string &inDir, hcFloat r_ss, uint compRadialRes, uint imgThetaRes, uint imgPhiRes, bool mapIntermediateHeights=false)
- void **compareRdist** (const char *inDir, uint compRadialRes, uint compThetaRes, uint compPhiRes, uint imgThetaRes, uint imgPhiRes)
- void compare2Stanford (const char *photFilename, const char *StanfordCoeffFilename, uint compRadial↩ Res, uint compThetaRes, uint compPhiRes, bool rDistribution, hcFloat geometricFactor, uint imgThetaRes, uint imgPhiRes, bool mapIntermediateHeights)

  computes grid method, shc method and loads Stanford file for comparison

- bool **footpointAnalysis** (hcFloat latThresh=INFINITY)
- bool batchFootpointAnalysis (const string &inDir, hcFloat r_ss, uint compResR, methodID method, hcFloat ellipticity, uint orderSHC, hcFloat latThresh=INFINITY)

  computes PFSS according to SHC approach and performs footpoint analysis on EIT maps

- bool batchFootpointAnalysisAllComputed (hcFloat latThresh=INFINITY)

  (re-)computes footpoint analysis for all solutions stored in outDir

- void **dump** () const

Public Attributes

- methodID method

  TODO: info? SHC, numeric spheric or numeric elliptic

- LaplaceSolver solver

  the actual solver

- PFSSsolutionInfo info

  information about solution

- SynPhotMagfield photBoundary

  boundary condition for solver

4.44.1 Member Function Documentation

4.44.1.1 batchKielGrid()

```
bool PFSSsolution::batchKielGrid (
            const string & inDir,
            hcFloat r_ss,
            uint resCompR,
            hcFloat ellipticity = 1.0 )
```

computes and maps Kiel grid approach for all files in inDir

opens inDir and computes PFSS model for each photospheric map as well as magnetic field line maps. This function only does the computational work, for the GUI version please see HelioMagfield::batchRotationComputing

Parameters

inDir	direcory to be worked upon
r_ss	heliocenric position of source surface (m)
resCompR	radial resolution of computational grid
ellipticity	ellipticity of source surface (default: 1.0 - spherical)

4.44.1.2 batchKielSHC()

```
bool PFSSsolution::batchKielSHC (
            const string & inDir,
            hcFloat r_ss,
            uint resCompR,
            uint order )
```

computes and maps Kiel SHC approach for all files in inDir

Parameters

inDir	direcory to be worked upon
r_ss	heliocenric position of source surface (m)
resCompR	radial resolution of computational grid
order	maximum principal order of SHC approach

4.44.1.3 computeAndMapKielGrid()

```
bool PFSSsolution::computeAndMapKielGrid (
```

```
            const string & filename,
            const char * optionalID,
            hcFloat r_ss,
            uint compResR,
            uint mapResTheta,
            uint mapResPhi,
            bool mapIntermediateHeights,
            hcFloat ellipticity,
            uint scaleMethod = 7 )
```

computes and maps Kiel grid approach

Parameters

filename	filename of the photospheric magnetogram to be analyzed
optionalID	optional identifier to be used in output filenames
r_ss	heliocentric position of surce surface
compResR	radial resolution of computational grid
compResTheta	meridional resolutions of computational grid
compResPhi	zonal resolution of computational grid
rDist	distribution of radial grid shells (use 0 for equidistant spacing in r-direction)
geomIncFactor	geometric increment factor for non-equidistant r-spacing
mapResTheta	meridional resolution of magnetic mapping
mapResPhi	zonal resolution of magnetic mapping
mapIntermediateHeights	true, if for each r-shell of the comp grid a magnetic mapping is to be computed
surfaceShape	0 - spherical grid, 1 - elliptical grid
surfShapeA	stretching parameter for elliptical grid
scaleMethod	rescaling algorithm for photospheric data

4.44.1.4 computeAndMapKielSHC()

```
bool PFSSsolution::computeAndMapKielSHC (
            const string & filename,
            uint order,
            hcFloat r_ss,
            uint compResR,
            uint mapResTheta,
            uint mapResPhi,
            bool mapIntermediateHeights )
```

computes spherical harmonic coefficients for given photospheric magfield

Even though the magnetic field is to be computed analytically employing the spherical harmonic coefficient approach, further processing (like mapping of magnetic field lines from source surface down to the photosphere) is done on a 3D grid.

Parameters

filename	synoptic photospheric magnetogram to be loaded

Parameters

order	maximum order of coefficients to be computed
r_ss	heliocentric distance of source surface (in m)
compResR	radial resolution of computational grid
compResTheta	meridional resolutions of computational grid
compResPhi	zonal resolution of computational grid
rDist	distribution of radial grid shells (use 0 for equidistant spacing in r-direction)
geomIncFactor	geometric increment factor for non-equidistant r-spacing
mapResTheta	meridional resolution of magnetic mapping
mapResPhi	zonal resolution of magnetic mapping
mapIntermediateHeights	true, if for each r-shell of the comp grid a magnetic mapping is to be computed

Returns

true, if computation of PFSS and mapping successful

4.44.1.5 load()

```
bool PFSSsolution::load (
            const string & cfgFN )
```

loads precomputed solution into this object

given a rotation/instrument-cfg filename and a path where the solution is stored, this function imports a previously computed solution from disk ∗

4.44.1.6 loadAndMapKielGrid()

```
bool PFSSsolution::loadAndMapKielGrid (
            const string & filename,
            uint mapResTheta,
            uint mapResPhi,
            hcFloat height = 0.0 )
```

loads Kiel grid solution and computes mapping

Parameters

filename	configuration file to be loaded
mapResTheta	meridional resolution of magnetic mapping
mapResPhi	zonal resolution of magnetic mapping
mapIntermediateHeights	true, if for each r-shell of the comp grid a magnetic mapping is to be computed ∗
height	height to be mapped or 0.0 for source surface and photosphere

4.44.1.7 loadAndMapStanfordSHC()

```
bool PFSSsolution::loadAndMapStanfordSHC (
            const char * photFilename,
            const char * StanfordCoeffFilename,
            uint compResR,
            uint mapResTheta,
            uint mapResPhi,
            bool mapIntermediateHeights )
```

loads Stanford SHC file and computes mapping

A computational grid is necessary for mapping magnetic field lines

Parameters

photFilename	Photospheric synoptic magnetogram
StanfordCoeffFilename	Spherical harmonic coefficients to be evaluated
compResR	radial resolution of computational grid
compResTheta	meridional resolutions of computational grid
compResPhi	zonal resolution of computational grid
rDist	distribution of radial grid shells (use 0 for equidistant spacing in r-direction)
geomIncFactor	geometric increment factor for non-equidistant r-spacing
mapResTheta	meridional resolution of magnetic mapping
mapResPhi	zonal resolution of magnetic mapping
mapIntermediateHeights	true, if for each r-shell of the comp grid a magnetic mapping is to be computed

4.44.1.8 mapHeightLevel()

```
bool PFSSsolution::mapHeightLevel (
            hcFloat height,
            hcFloat maxSinLat,
            uint numTheta,
            uint numPhi,
            bool sinLatFormat = true,
            bool compCoords = true )
```

routine for computing magnetic height maps in memory

Magnetic field lines are invoked at a given height in a quasi equidistant manner (sine-latitude grid). The field lines are then traced through the heliosphere and sampled at other given heights. This way we can track the "evolution" of the magnetic field from different starting points.

Parameters

height	at which quasi-equidistant field lines are to be invoked
maxSinLat	highest sine(latitude)
numTheta	number of grid points in latitudinal direction
numPhi	number of grid points in azimuthal direction

4.44.1.9 multiMapSolution()

```
bool PFSSsolution::multiMapSolution (
            uint numTheta,
            uint numPhi,
            bool computeIntermediateHeightLevels,
            bool sinLatFormat,
            bool compCoords )
```

computes magnetic field line maps to several height levels

creates magnetic field maps with footpoints at several heights and mapped to all other heights in this list.

4.44.1.10 paramStudyRes()

```
void PFSSsolution::paramStudyRes ( )
```

analyzes impact of grid point numbers on results

conducts a parameter study of the resolution of the numerical grid

4.44.1.11 paramStudyRss()

```
void PFSSsolution::paramStudyRss (
            const char * inDir,
            uint compRadialRes,
            uint compThetaRes,
            uint compPhiRes,
            uint imgThetaRes,
            uint imgPhiRes )
```

analyses impact of source surface height on results

conducts a parameter study of the height of the source surface on files in directory inDir

4.44.1.12 paramStudyThresh()

```
void PFSSsolution::paramStudyThresh (
            const char * filename,
            uint compRadialRes,
            uint compThetaRes,
            uint compPhiRes,
            uint imgThetaRes,
            uint imgPhiRes )
```

analyses impact of solver threshold level on results

conducts a parameter study of the solver threshold level on input file named filename

4.44.1.13 save()

```
bool PFSSsolution::save ( )
```

stores solution computed by solver into file located at path

stores computed PFSS solution to disk and stores meta-information into cfg-files (cfg file for information on this specific solution and one entry in super-cfg so that the program knows that there is a solution for this specific Carrington rotation and instrument)

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

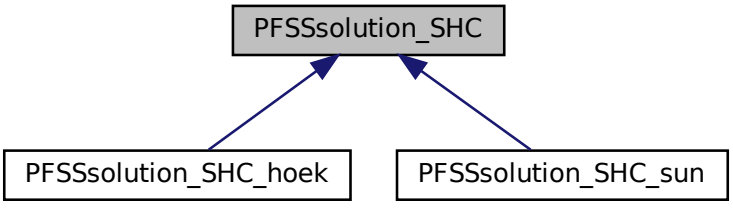
- src/pfssSolution.h
- src/pfssSolution.cpp
- src/pfssSolution_batches.cpp

4.45 PFSSsolution_SHC Class Reference

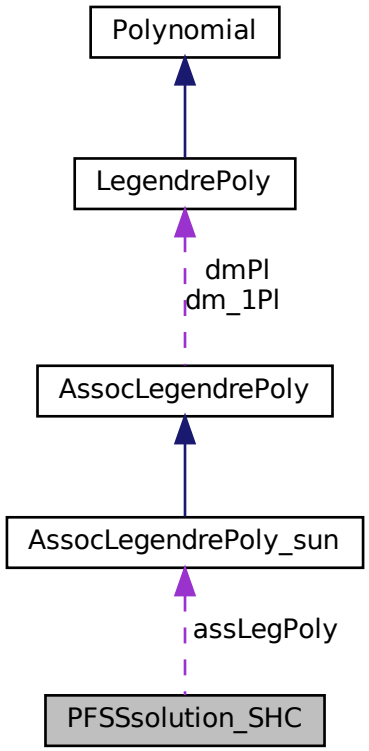
solution of the PFSS model via spherical harmonic coefficients (SHC)

```
#include <pfss.h>
```

Inheritance diagram for PFSSsolution_SHC:



Collaboration diagram for PFSSsolution_SHC:



Public Member Functions

- PFSSsolution_SHC (const char *filename, bool WSOfile, hcFloat r_ss)
  constructor
- PFSSsolution_SHC ()
  std constructor
- PFSSsolution_SHC (const PFSSsolution_SHC &other)
  cpy constructor
- virtual ~PFSSsolution_SHC ()
  destructor
- PFSSsolution_SHC & operator= (const PFSSsolution_SHC &other)
  assignment operator
- void **initNULL** ()
- void **clear** ()
- virtual void **init** (uint order, hcFloat r_ss)=0
- virtual void eval (const Vec3D &pos, Vec3D &result)=0

evaluates the magnetic field at spherical position pos

- bool importCoefficients (const char *filename, bool WSOfile, hcFloat r_ss)
  reads coefficients from human readable file
- void exportCoefficients (const char *filename)
  writes coefficients to human readable file
- void determineCoefficientsFromPhotMagfield (uint order, hcFloat r_ss, SynPhotMagfield &photMagfield)
  computes spherical harmonic coefficients from synoptic magnetogram

Public Attributes

- AssocLegendrePoly_sun ** assLegPoly
  contains all the associated Legendre polynomials required for the solution
- uint coeffOrder
  highest order of assoc. Legendre Polynomial utilized
- hcFloat ** g
  g coefficients
- hcFloat ** h
  h coefficients
- hcFloat sourceSurfaceFactor
  location of source surface in multiples of r_sol

4.45.1 Detailed Description

solution of the PFSS model via spherical harmonic coefficients (SHC)

4.45.2 Member Function Documentation

4.45.2.1 determineCoefficientsFromPhotMagfield()

```
void PFSSsolution_SHC::determineCoefficientsFromPhotMagfield (
            uint order,
            hcFloat r_ss,
            SynPhotMagfield & photMagfield )
```

computes spherical harmonic coefficients from synoptic magnetogram

computes spherical harmonic coefficients from synoptic magnetogram

Parameters

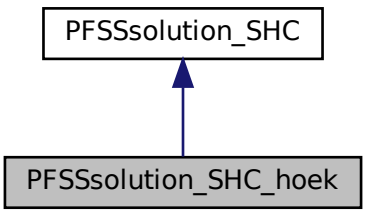
order	maximum order of coefficients to be computed
r_ss	heliocentric position of source surface (in m)
photMagfield	data structure containing synoptic photospheric (LOS) magnetic field strength

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

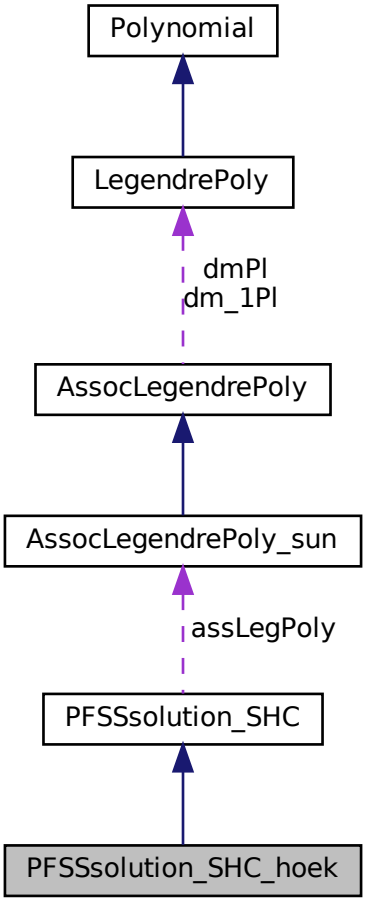
- src/pfss.h
- src/pfss.cpp

4.46 PFSSsolution_SHC_hoek Class Reference

Inheritance diagram for PFSSsolution_SHC_hoek:



Collaboration diagram for PFSSsolution_SHC_hoek:



Public Member Functions

- **PFSSsolution_SHC_hoek** (const PFSSsolution_SHC_hoek &other)
- PFSSsolution_SHC_hoek & **operator=** (const PFSSsolution_SHC_hoek &other)
- PFSSsolution_SHC_hoek operator- (const PFSSsolution_SHC_hoek &other)

  difference between two solutions (difference of coefficients)

- virtual void **init** (uint order, hcFloat r_ss)
- virtual void eval (const Vec3D &pos, Vec3D &result)

  evaluates the magnetic field at spherical position pos

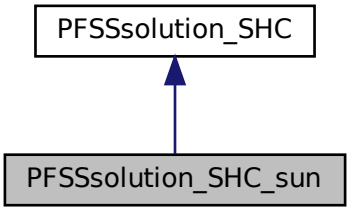
Additional Inherited Members

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

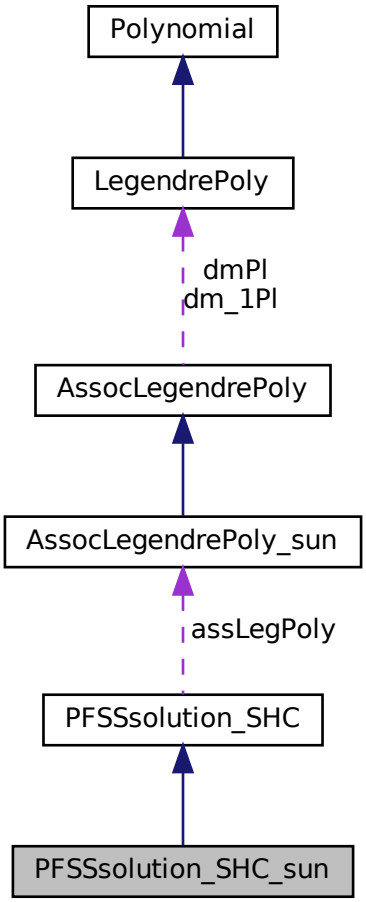
- src/pfss.h
- src/pfss.cpp

4.47 PFSSsolution_SHC_sun Class Reference

Inheritance diagram for PFSSsolution_SHC_sun:



Collaboration diagram for PFSSsolution_SHC_sun:



Public Member Functions

- PFSSsolution_SHC_sun ()
  std constructor
- PFSSsolution_SHC_sun (const PFSSsolution_SHC_sun &other)
  cpy constructor
- ~PFSSsolution_SHC_sun ()
  destructor
- PFSSsolution_SHC_sun & operator= (const PFSSsolution_SHC_sun &other)
  assignment operator
- PFSSsolution_SHC_sun operator- (const PFSSsolution_SHC_sun &other)
  difference between two solutions (difference of coefficients)
- virtual void **init** (uint order, hcFloat r_ss)
- virtual void eval (const Vec3D &pos, Vec3D &result)
  evaluates the magnetic field at spherical position pos

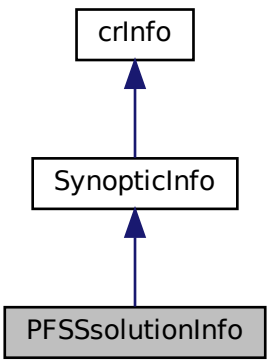
Additional Inherited Members

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

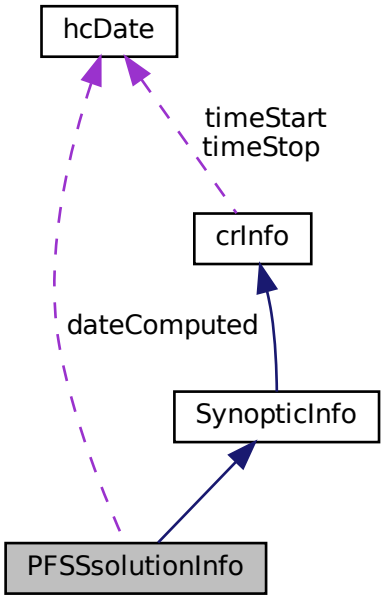
- src/pfss.h
- src/pfss.cpp

4.48 PFSSsolutionInfo Class Reference

Inheritance diagram for PFSSsolutionInfo:



Collaboration diagram for PFSSsolutionInfo:



Public Member Functions

- PFSSsolutionInfo ()
  std constructor
- PFSSsolutionInfo (const PFSSsolutionInfo &other)
  cpy constructor
- PFSSsolutionInfo (SynopticInfo synInf, uint sizeofFloat, modelID model, methodID method, groupID group, hcFloat rss, hcFloat ell, uint orderSHC, uint numR, uint numTheta, uint numPhi)
  constructor
- PFSSsolutionInfo & operator= (const PFSSsolutionInfo &other)
  assignment operator
- bool operator== (const PFSSsolutionInfo &other)
  comparison operator
- bool operator> (const PFSSsolutionInfo &other)
  comparison operator
- bool operator< (const PFSSsolutionInfo &other)
  comparison operator
- void **initNULL** ()
- void **clear** ()

- void **init** (SynopticInfo synInf, uint sizeofFloat, modelID model, methodID method, groupID group, hcFloat rss, hcFloat ell, uint orderSHC, uint numR, uint numTheta, uint numPhi)
- bool exportBinary (ofstream &stream)

  export instance to binary stream

- bool importBinary (ifstream &stream)

  imports instance from binary stream

- string toString () const

  returns information on this instance in string

- void dump (uint indent=0) const

  dumps information on this instance to stdout with optional indendtation

Public Attributes

- uint sizeofFloat

  floating point type used for computation 4=float, 8=double

- modelID model

  model employed for magnetic computation

- methodID method

  method applied for (PFSS) computation

- groupID group

  working group which computed this solution

- hcFloat rss

  source surface radius (m)
- hcFloat ell

  ellipticity of source surface (if method==METH_ELLIPTICAL)

- uint orderSHC

  maximum principal order of SHC solution (if method==METH_SHC)

- uint numR

  number of grid points in radial direction

- uint numTheta

  number of grid points in meridional direction

- uint numPhi

  number of grid points in zonal direction

- hcDate dateComputed

  date when the solution was computed

- uint computationTime

  time in seconds it took to compute

- uint solutionSteps

number of solution steps to fall below accuracy threshold

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

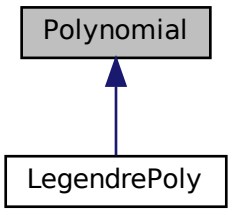
- src/pfssSolutionInfo.h
- src/pfssSolutionInfo.cpp

4.49 Polynomial Class Reference

loads coefficients for Spherical functions computed by Bala via CSSS

`#include <hcFunction.h>`

Inheritance diagram for Polynomial:



Public Member Functions

- Polynomial (uint order=0)
  std constructor
- Polynomial (const Polynomial &other)
  cpy constructor
- **Polynomial** (double *factor, uint n)
- ~Polynomial ()
  destructor
- Polynomial & operator= (const Polynomial &other)
  assignment operator
- void **initNULL** ()
- void **clear** ()
- void **init** (uint n)
- double **operator()** (double x)
- void derivative (uint m)
  transforms this to its m'th derivative
- void **scale** (double scale)
- void **dump** ()

Public Attributes

- double * **factor**
- uint **n**

4.49.1 Detailed Description

loads coefficients for Spherical functions computed by Bala via CSSS

<

polynomial of order n

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

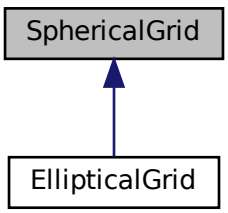
- engine/math/hcFunction.h
- engine/math/hcFunction.cpp

4.50 SphericalGrid Class Reference

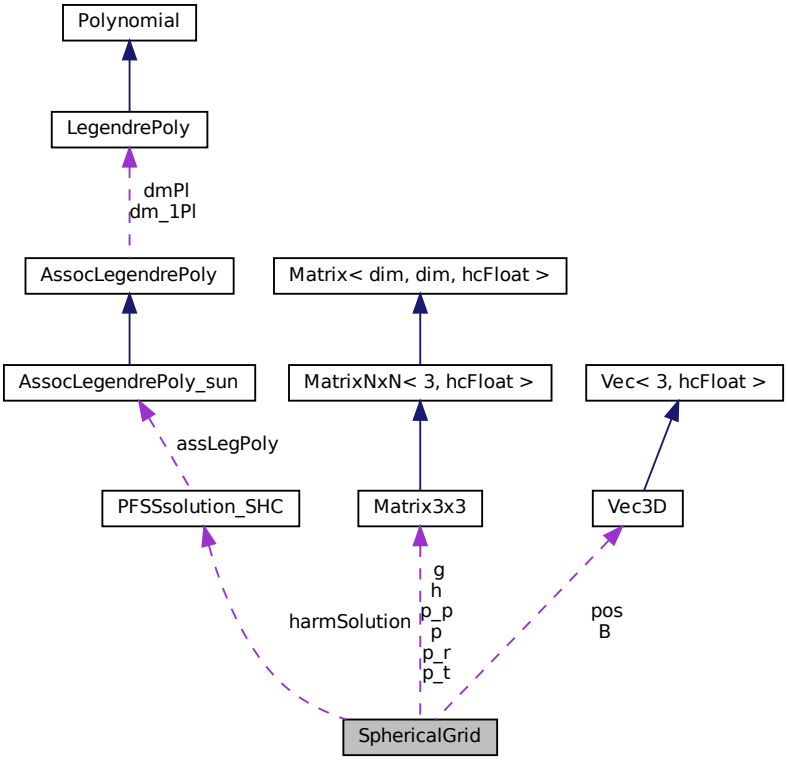
grid structure for numerical computation in 3D space

```
#include <grids.h>
```

Inheritance diagram for SphericalGrid:



Collaboration diagram for SphericalGrid:



Public Member Functions

- SphericalGrid ()
  std constructor
- SphericalGrid (const SphericalGrid &grid)
  cpy constructor
- virtual ~SphericalGrid ()
  destructor
- virtual SphericalGrid & operator= (const SphericalGrid &other)
  assignment operator
- virtual void **initNULL** ()
- virtual void **initNULL_CPU** ()
- virtual void **clear** ()
- virtual void **clear_CPU** ()
- virtual void init (bool sinLatGrid, hcFloat maxSinLat, hcFloat minSinLat, hcFloat lowerR, hcFloat upperR, uint numR, bool clearGPU=true, hcFloat a=1.0)
  initializes the grid
- void initMembers (uint numR, uint numT, uint numP, bool sinLatGrid, hcFloat maxSinLat, hcFloat minSinLat, hcFloat lowerR, hcFloat upperR)

initializes member variables, usable by elliptical and spherical grid

- void getScalingFactors ()

  computes scaling factors to be used by the Laplace solver

- void clearValues ()

  clears the values (B_r, relError, temp, etc.) not the positions (pos)

- void diff (const SphericalGrid &other)

  computes difference in psi array

- hcFloat maxSpacingR (uint &index)

  returns index of grid point where spacing in r-direction is tallest

- hcFloat maxSpacingTheta (uint &index)

  returns index of grid point where spacing in theta-direction is tallest

- hcFloat maxSpacingPhi (uint &index)

  returns index of grid point where spacing in phi-direction is tallest

- virtual bool isElliptical () const

  determine whether the grid is spherical or elliptical

- virtual Vec3D **getPos** (uint index, bool ellipticCoords) const
- virtual Vec3D **getB** (uint index, bool ellipticCoords=false) const
- virtual void **setB** (uint index, Vec3D value, bool ellipticCoords=false)
- virtual void **dumpCoords** (uint fixed=0, bool ellipticCoords=false) const
- void iterateElliptic_gridPoint (uint i, uint j, uint k)

  solve Laplace equation on a single grid point according to the elliptic computation scheme

- hcFloat **getPsi** (uint index) const
- void **setPsi** (uint index, hcFloat value)
- void **setRelError** (uint index, hcFloat value)
- hcFloat **getTemp** (uint index) const
- void **setTemp** (uint index, hcFloat value)
- hcFloat **getRelError** (uint index) const
- hcFloat * getPsiArray () const

  scalar potential

- hcFloat * getRelErrorArray () const

  relative error compered to preciding step

- Vec3D * getPosArray () const

  position in world coordinates (spherical)

- Vec3D * getBArray () const

  magnetic field components (spherical)

- hcFloat * getTempArray () const

  temp value for computations on spec. grid point

- float getStepSize () const

  returns the optimal step size for following values (e.g., Magnetic field lines) throughout the grid

- uint **getIndex** (uint i, uint j, uint k) const
- uint **getIndexPhiPlus** (uint ind)
- void getIndicesFromIndex (uint ind, uint &i, uint &j, uint &k) const

  if ind = k ∗ numR ∗ numTheta + j ∗ numR + i is given, computes i, j, k

- void **printGridIndicesFromIndex** (uint ind)
- void **computeLowerBoundaryPsi** (float ∗image)
- virtual unsigned char getNearestNeighbors (Vec3D &pos, Vec< 8, hcFloat > &indices, bool debug=false) const

  computes the nearest grid cells in the grid
- virtual bool getInterpolatedB (Vec3D &B, Vec3D &pos, unsigned char &error, bool debug=false) const

  computes an interpolated value for B from the surrounding grid points via trilinear interpolation (r->theta->phi)
- Vec3D **getBFromPsi** (uint r, uint t, uint p)
- void compDerivR (Matrix3x3 ∗arr, Matrix3x3 ∗deriv)

  computes derivative w.r.t. r of quantity stored at arr

- void compDerivT (Matrix3x3 ∗arr, Matrix3x3 ∗deriv)

  computes derivative w.r.t. theta of quantity stored at arr

- void compDerivP (Matrix3x3 ∗arr, Matrix3x3 ∗deriv)

  computes derivative w.r.t. phi of quantity stored at arr

- bool exportEntireGrid (const char ∗filename)

  export grid to binary file

- bool importEntireGrid (const char ∗filename)

  import grid from binary file

- bool **evaluateCSSS** (const char ∗filename)
- virtual void **dump** () const

Static Public Member Functions

- static void getOptimalGridParameters (uint numR, hcFloat lowerR, hcFloat upperR, hcFloat &geometricFactor, uint &numT, uint &numP, bool high=false)

  computes parameters so that distances in all directions between grid points are more or less equal

Public Attributes

- uint sizeofFloat

  4-single, 8-double, 16-long double(not supported in CUDA yet)

- uint numR

  number of r-steps

- uint numTheta

  number of theta-steps

- uint numPhi

  number of phi-steps

- uint numGridPoints
  overall grid points
- bool sinLatGrid
  is the grid equally spaced in latitude (=false) or sin(lat) (=true)
- hcFloat maxSinLat
  sin(latitude) of northernmost pixel border
- hcFloat minSinLat
  sin(latitude) of southernmost pixel border
- hcFloat lowerR
  lowest radial coordinate (photosphere)
- hcFloat upperR
  uppermost radial coordinate (source surface)
- hcFloat geometricFactor
  factor by which radial spacing increases from shell to shell
- PFSSsolution_SHC * harmSolution
  PFSS solution via spherical harmonic function approach, temporary testing variable
- Vec3D * pos
  position in computational coordinates (spherical)
- Vec3D * B
  magnetic field components (spherical)
- hcFloat * psi
  scalar potential
- hcFloat * relError
  relative error compered to preciding step
- hcFloat * temp
  temp value for computations on spec. grid point
- Matrix3x3 * g
  metric coefficients
- Matrix3x3 * h
  dual metric coefficients
- Matrix3x3 * p
  *metric helper variable (sqrt(g)*g)*
- Matrix3x3 * p_r
  derivative of p w.r.t. r
- Matrix3x3 * p_t
  derivative of p w.r.t. theta

- Matrix3x3 * p_p
  derivative of p w.r.t. phi

- hcFloat * **s_ijk**
- hcFloat * **s_imjk**
- hcFloat * **s_ipjk**
- hcFloat * **s_ijmk**
- hcFloat * **s_ijpk**
- hcFloat * **s_ijkm**
- hcFloat * **s_ijkp**
- hcFloat * **s_imjmk**
- hcFloat * **s_imjpk**
- hcFloat * **s_ipjmk**
- hcFloat * **s_ipjpk**
- hcFloat * **s_imjkm**
- hcFloat * **s_imjkp**
- hcFloat * **s_ipjkm**
- hcFloat * **s_ipjkp**
- hcFloat * **s_ijmkm**
- hcFloat * **s_ijmkp**
- hcFloat * **s_ijpkm**
- hcFloat * **s_ijpkp**
- hcFloat * **s_ijmmk**
- hcFloat * **s_imjmmk**
- hcFloat * **s_ipjmmk**
- hcFloat * **s_ijmmkm**
- hcFloat * **s_ijmmkp**
- hcFloat * **s_ijppk**
- hcFloat * **s_imjppk**
- hcFloat * **s_ipjppk**
- hcFloat * **s_ijppkm**
- hcFloat * **s_ijppkp**

4.50.1 Detailed Description

grid structure for numerical computation in 3D space

4.50.2 Member Function Documentation

4.50.2.1 getInterpolatedB()

```
bool SphericalGrid::getInterpolatedB (
            Vec3D & B_out,
            Vec3D & position,
            unsigned char & error,
            bool debug = false ) const  [inline], [virtual]
```

computes an interpolated value for B from the surrounding grid points via trilinear interpolation (r->theta->phi)

Parameters

B	(out) the magnetic field at pos (spherical coordinate system)
position	(in/out) the position (spherical/computational coordinates) where the magnetic field is requested
error	(out) error code (see below)

returns true only if a valid magnetic field vector has been found

The error codes returned in error are:

(1 << 0) = 1 pos.r below lower boundary (1 << 1) = 2 pos.r above upper boundary

(1 << 7) = 128 different error, should not be possible to reach

4.50.2.2 getNearestNeighbors()

```
unsigned char SphericalGrid::getNearestNeighbors (
            Vec3D & position,
            Vec< 8, hcFloat > & indices,
            bool debug = false ) const  [inline], [virtual]
```

computes the nearest grid cells in the grid

Parameters

position	for which nearest neighbors shall be found (spherical/computational coordinates)
indices	vector containing grid indices for nearest neighbors

returns bit-coded clipping id if pos is outside grid:

(1 << 0) = 1 pos.r below lower boundary (1 << 1) = 2 pos.r above upper boundary

(1 << 2) = 4 pos.theta above highest latitude (north pole) in these two cases, nearest neighbors are being computed in accordance with the (1 << 3) = 8 pos.theta below lowest latitude (south pole) polar boundary conditions

(1 << 7) = 128 different error, should not be possible to reach

indices has to be 8-dim vector, numbering is done just like with the grid:

indices[0] -> rm, tm, pm indices[1] -> rp, tm, pm indices[2] -> rm, tp, pm indices[3] -> rp, tp, pm

indices[4] -> rm, tm, pp indices[5] -> rp, tm, pp indices[6] -> rm, tp, pp indices[7] -> rp, tp, pp

4.50.2.3 getStepSize()

```
float SphericalGrid::getStepSize ( ) const  [inline]
```

returns the optimal step size for following values (e.g., Magnetic field lines) throughout the grid

for now assumes equidistant grid! this should probably be amended sometime

4.50.2.4 init()

```
void SphericalGrid::init (
            bool sinLatGrid,
            hcFloat maxSinLat,
            hcFloat minSinLat,
            hcFloat lowerR,
            hcFloat upperR,
            uint numR,
            bool clearGPU = true,
            hcFloat a = 1.0 )  [virtual]
```

initializes the grid

additional information like different domains (block regular) and distribution of grid points have to be supplied

Parameters

rDistribution	(false - equally spaced, true - geometric series)

Reimplemented in EllipticalGrid.

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

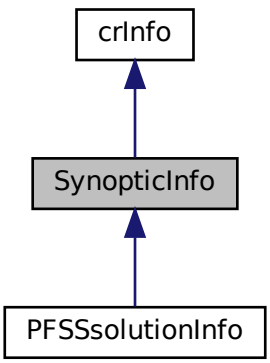
- src/grids.h
- src/ellipticalGrid.cpp
- src/grids.cpp
- src/laplaceSolver.cpp

4.51 SynopticInfo Class Reference

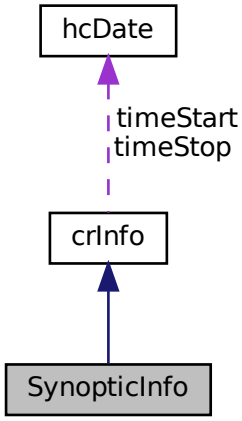
information on photospheric magnetic field data such as instrument which was used, sin(latitude)-format, ...

`#include <synPhotMagfield.h>`

Inheritance diagram for SynopticInfo:



Collaboration diagram for SynopticInfo:



Public Member Functions

- SynopticInfo ()
- SynopticInfo (const SynopticInfo &other)
- **SynopticInfo** (originID id, hcFloat maxSinLat, uint CRnum, uint dailyID=0)
- virtual ~SynopticInfo ()
- SynopticInfo & operator= (const SynopticInfo &other)
- bool operator== (const SynopticInfo &other)

  comparison operator

- bool operator> (const SynopticInfo &other)

  comparison operator

- bool operator< (const SynopticInfo &other)

  comparison operator

- bool exportBinary (ofstream &stream)

  export instance to stream

- bool importBinary (ifstream &stream)

  import instance from stream

- void **initNULL** ()
- void **clear** ()
- void **init** (originID id, hcFloat maxSinLat, uint CRnum, uint dailyID)
- void dump (uint indent=0) const

  dumps information on this instance to stdout

Public Attributes

- originID instrument
- uint dailyID
- bool sinLatFormat
- hcFloat maxSinLat

4.51.1 Detailed Description

information on photospheric magnetic field data such as instrument which was used, sin(latitude)-format, ...

4.51.2 Constructor & Destructor Documentation

4.51.2.1 SynopticInfo() [1/2]

```
SynopticInfo::SynopticInfo ( )
```

std constructor

4.51.2.2 SynopticInfo() [2/2]

```
SynopticInfo::SynopticInfo (
            const SynopticInfo & other )
```

cpy constructor

4.51.2.3 ~SynopticInfo()

```
SynopticInfo::~SynopticInfo ( )  [virtual]
```

destructor

4.51.3 Member Function Documentation

4.51.3.1 operator=()

```
SynopticInfo & SynopticInfo::operator= (
            const SynopticInfo & other )
```

assignment operator

4.51.4 Member Data Documentation

4.51.4.1 dailyID

```
uint SynopticInfo::dailyID
```

identification for daily synoptic maps

4.51.4.2 instrument

```
originID SynopticInfo::instrument
```

instrument used for synoptic map

4.51.4.3 maxSinLat

```
hcFloat SynopticInfo::maxSinLat
```

sin(latitude) of northern grid boundary

4.51.4.4 sinLatFormat

```
bool SynopticInfo::sinLatFormat
```

y-Axis in synoptic map given in sin(latitude)?

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

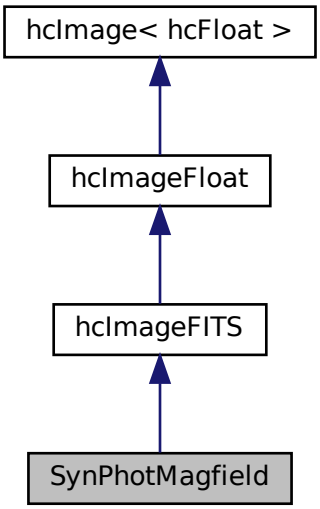
- src/synPhotMagfield.h
- src/synPhotMagfield.cpp

4.52 SynPhotMagfield Class Reference

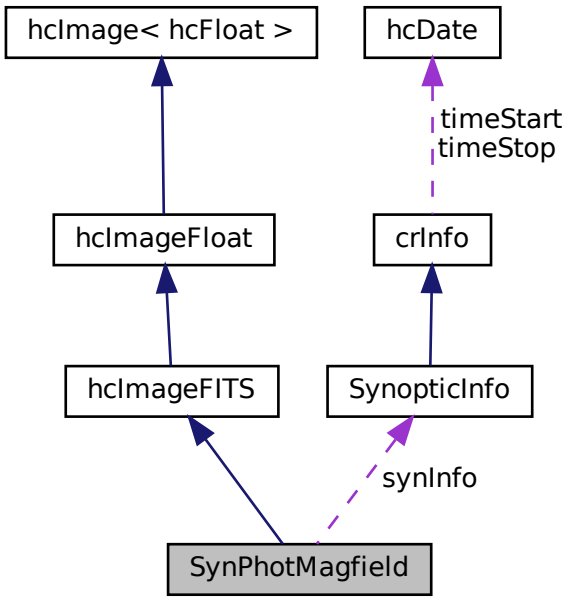
handles synoptic photospheric magnetograms stored in GAUSS

```
#include <synPhotMagfield.h>
```

Inheritance diagram for SynPhotMagfield:



Collaboration diagram for SynPhotMagfield:



Public Member Functions

- SynPhotMagfield ()
  std constructor
- SynPhotMagfield (const SynPhotMagfield &other)
  cpy constructor
- SynPhotMagfield & operator= (const SynPhotMagfield &other)
  assignment operator
- void **initNULL** ()
- virtual bool load (const string &filename)
  load FITS image from file
- bool **openKPVT** (const string &filename)
- bool **openWSO** (const string &filename)
- bool **openWSOconverted** (const string &filename)
- bool **openSOHOMDI** (const string &filename)
- bool **openSOHOMDI_DAILY** (const string &filename)
- bool **openNSOGONG** (const string &filename)
- bool **openSDOHMI** (const string &filename)
- bool **openVerDAILY** (const string &filename)
- bool **openOwnFormat** (const string &filename)
- bool createLOSfromGrid (SphericalGrid &grid)

computes line-of-sight magnetogram from magnetic field values of grid

- bool loadDipole (hcFloat dipole, uint numTheta, uint numPhi, hcFloat maxSinLat=0.0, hcFloat minSinLat=-0.0)

  loads artificial dipole to line-of-sight magnetogram

- void removeMonopole ()

  removes monopole component from magnetogram

- bool remeshImage (uint newWidth, uint newHeight, uint scaleMethod)

  rescales magnetogram

- void cropPolesFromImage (uint numPixelsToCrop)

  remove north-most and south-most lines from magnetogram

- bool convertWSOtxtToWSOfits (const string &infile, const string &outfile)

  converts Stanford ASCII files to FITS files

- virtual bool save (const string &filename)

  stores photospheric magfield in FITS format to be re-imported later on

- void dump () const

  dumps information on this instance to stdout

Public Attributes

- SynopticInfo synInfo

  information on the synoptic data

Additional Inherited Members

4.52.1 Detailed Description

handles synoptic photospheric magnetograms stored in GAUSS

1G = 1E-4T = 100uT

4.52.2 Member Function Documentation

4.52.2.1 remeshImage()

```
bool SynPhotMagfield::remeshImage (
            uint newWidth,
            uint newHeight,
            uint scaleMethod )
```

rescales magnetogram

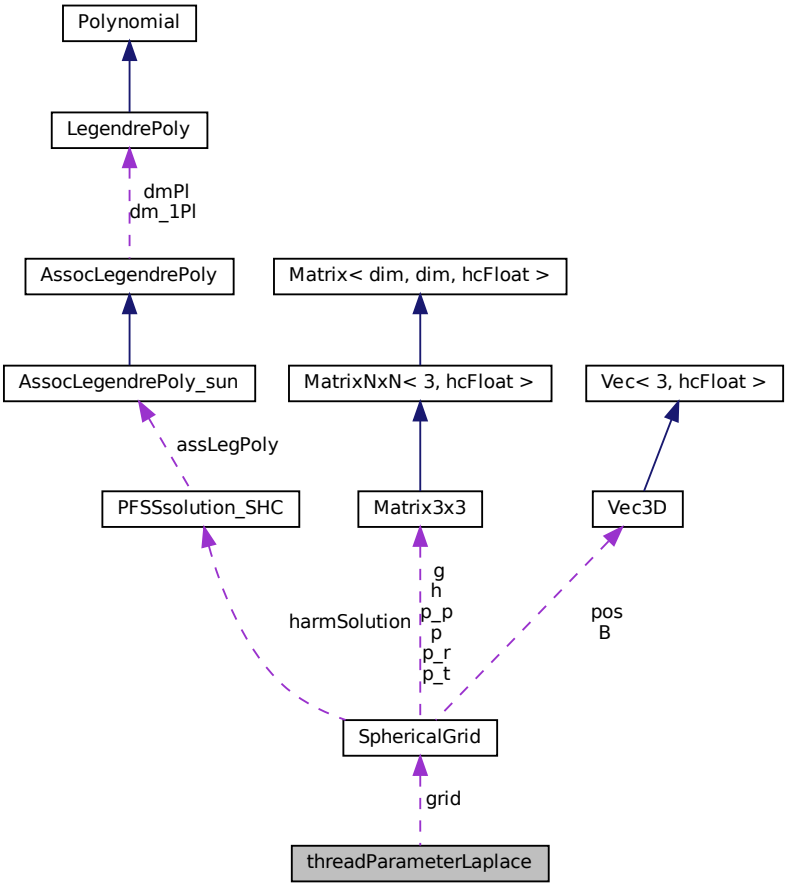
Note: maxSinLat and minSinLat only relevant if scaleMethod=0 or scaleMethod=1

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

- src/synPhotMagfield.h
- src/synPhotMagfield.cpp

4.53 threadParameterLaplace Struct Reference

Collaboration diagram for threadParameterLaplace:



Public Member Functions

- void **init** (uint threadID=0, volatile _Atomic_word *numRunningThreads=NULL, pthread_mutex_t *running↩ Mutex=NULL, volatile _Atomic_word *threadRunning=NULL, SphericalGrid *grid=NULL)
- void **set** (const uint &idx, const uint &gppt)

Public Attributes

- uint **threadID**
- pthread_mutex_t * **runningMutex**
- SphericalGrid * **grid**
- volatile _Atomic_word * **numRunningThreads**
- volatile _Atomic_word * **threadRunning**
- uint idx

  id of grid point to be computed, as in CUDA version

- uint gppt

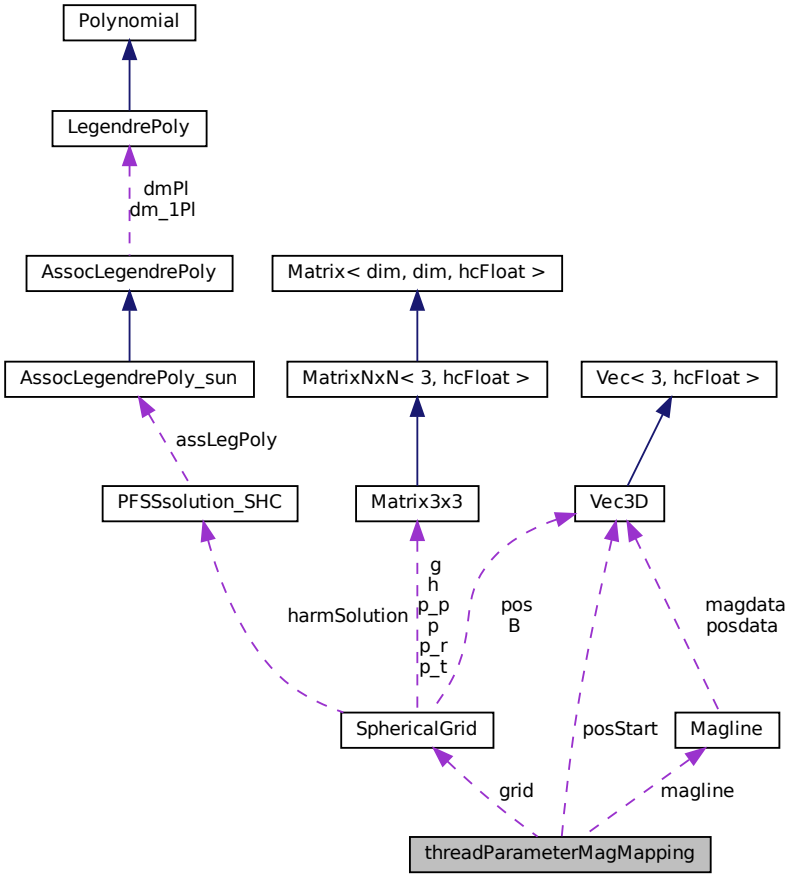
  grid points per thread

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

- src/laplaceSolver.h

4.54 threadParameterMagMapping Struct Reference

Collaboration diagram for threadParameterMagMapping:



Public Member Functions

- void **init** (uint threadID, volatile _Atomic_word *numRunningThreads, pthread_mutex_t *runningMutex, volatile _Atomic_word *threadRunning, SphericalGrid *grid)
- void **set** (Magline *magline, Vec3D *posStart)

Public Attributes

- uint **threadID**
- pthread_mutex_t * **runningMutex**
- SphericalGrid * **grid**
- volatile _Atomic_word * **numRunningThreads**
- volatile _Atomic_word * **threadRunning**
- Magline * magline

  pointer to magline to be worked upon

- Vec3D * posStart

  pointer to position from which to start tracking field line, in computational coordinates

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

- src/magMapping.h

4.55 Vec< dim, T > Class Template Reference

implementation of the mathematical (finite dimensionality) vector concept

`#include <hcVec.h>`

Public Member Functions

- Vec ()

  std constructor

- template<class S >
  Vec (const Vec< dim, S > &other)

  cpy constructor

- ~Vec ()

  destructor

- T & **operator()** (uint n)
- T **operator[ ]** (uint n) const
- template<class S >
  Vec< dim, T > & operator= (const Vec< dim, S > &other)

  assignment operator

- Vec & **operator*=** (T scale)
- Vec & **operator/=** (T scale)
- Vec & **operator+=** (const Vec &other)
- Vec & **operator-=** (const Vec &other)
- Vec **operator-** ()
- bool **operator==** (const Vec &vec) const
- bool isAlmostEqual (const Vec< dim, T > &other, T eps=1E-6)

  checks if vectors are the same within numerical uncertainty bounds

- Vec< dim, T > & **normalize** ()
- void **loadZeroes** ()
- void **loadHom** (const Vec< dim-1, T > &vec)
- T sp (const Vec &v) const

  std-scalar product with other vector

- double **sp_double** (const Vec &other) const
- T dist (const Vec &other)

  distance to point in std-norm

- T length () const

  distance to origin in std-norm / length of vector

- void scale (T factor)

  scales the vector by factor

- void zero ()

  sets all components to 0

- bool isNullVector () const
- bool isValid () const

  checks if some element is INF or NAN
- bool **exportBinary** (std::ofstream &stream)
- bool **importBinary** (std::ifstream &stream, uint sizeofFloat=0)
- void **dump** () const

Public Attributes

- T content [dim]

  elements of the vector

4.55.1 Detailed Description

template<uint dim, class T>
class Vec< dim, T >

implementation of the mathematical (finite dimensionality) vector concept

4.55.2 Member Function Documentation

4.55.2.1 isNullVector()

```
template<uint dim, class T >
bool Vec< dim, T >::isNullVector
```

< copies components from Vector and overwrites the dimensionality of this

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

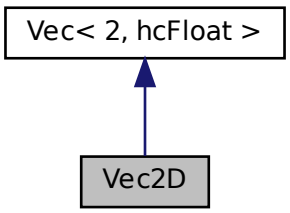
- engine/math/hcVec.h

4.56 Vec2D Class Reference

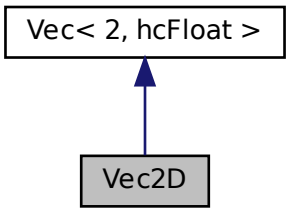
3D Vectors

`#include <hcVec.h>`

Inheritance diagram for Vec2D:



Collaboration diagram for Vec2D:



Public Member Functions

- Vec2D ()
  std constructor
- template<class S >
  Vec2D (S x, S y)
- Vec2D (const Vec2D &other)
  cpy constructor
- template<class S >
  Vec2D (const Vec< 2, S > &other)
  pseudo-cpy constructor

- ~Vec2D ()

  destructor

- Vec2D & **operator=** (const Vec2D &other)
- template<class S >
  Vec2D & **operator=** (const Vec< 2, S > &other)
- template<class S >
  void **rotate** (S phi)
- template<class S >
  void **set** (S x, S y)

Additional Inherited Members

4.56.1 Detailed Description

3D Vectors

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

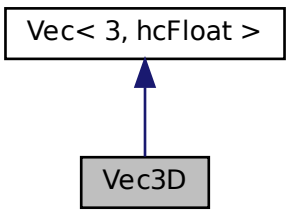
- engine/math/hcVec.h
- engine/math/hcVec.cpp

4.57 Vec3D Class Reference

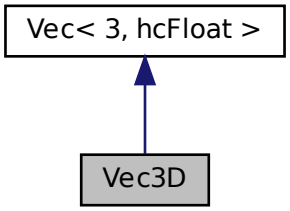
3D Vectors

`#include <hcVec.h>`

Inheritance diagram for Vec3D:



Collaboration diagram for Vec3D:



Public Member Functions

- Vec3D ()
  std constructor
- Vec3D (const Vec3D &other)
  cpy constructor
- template<class S >
  Vec3D (const Vec< 3, S > &other)
  pseudo-cpy constructor
- template<class S >
  Vec3D (S x, S y, S z)
- ~Vec3D ()
  destructor
- Vec3D & **operator=** (const Vec3D &other)
- template<class S >
  Vec3D & **operator=** (const Vec< 3, S > &other)
- template<class S >
  void rotateX (S angle)
  rotate around x-axis in global cartesian coordinate system
- template<class S >
  void rotateY (S angle)
  rotate around y-axis in global cartesian coordinate system
- template<class S >
  void rotateZ (S angle)
  rotate around z-axis in global cartesian coordinate system
- void stereoProjUnitSphere (Vec2D tangent)
  stereographic projection of this (in cartCoords) onto plane tangent at unit sphere in spherical coordinates (Vec2D tangent)
- void convertEllipticalCoordsToCartesian (hcFloat a, bool prolate)
  transforms elliptical coordinates to cartesian coordinates
- void convertCartesianCoordsToElliptical (hcFloat a, bool prolate)
  transforms cartesian coordinates to elliptical coordinates

- Vec3D **convCoordSpher2Cart** () const
- Vec3D **convCoordCart2Spher** () const
- Vec3D **convCoordCart2Ell** (const EllipticalGrid &grid) const
- Vec3D **convCoordEll2Cart** (const EllipticalGrid &grid) const
- Vec3D **convCoordSpher2Ell** (EllipticalGrid &grid) const
- Vec3D **convCoordEll2Spher** (EllipticalGrid &grid) const
- Vec3D **convVecSpher2Cart** (const Vec3D &cartPos) const
- Vec3D **convVecCart2Spher** (const Vec3D &cartPos)
- Vec3D convCoordCart2Cart (Vec3D e1x, Vec3D e1y, Vec3D e1z, Vec3D e2x, Vec3D e2y, Vec3D e2z) const

  convert position vector from cartesian coordinate system 1 to cartesian coordinate system 2

- Vec3D **convVecSpher2Ell** (const Vec3D &posSpher, const EllipticalGrid &grid) const
- Vec3D **convVecEll2Spher** (const Vec3D &posEll, const EllipticalGrid &grid) const
- Vec3D **convVecEll2Cart** (const Vec3D &cartPos, const EllipticalGrid &grid) const
- Vec3D **convVecCart2Ell** (const Vec3D &cartPos, const EllipticalGrid &grid) const
- Vec3D **convVecCart2Ell2** (const Vec3D &cartPos, EllipticalGrid &grid) const
- Vec3D convVecHAE2GSE (const hcDate &date)

  convert vecter in Heliocentric Aries Ecliptic cartesian coordinates to Geocentric Solar Ecliptic cartesian coordsinates

- Vec3D convVecGSE2HAE (const hcDate &date)

  convert vecter in Geocentric Solar Ecliptic cartesian coordsinates to Heliocentric Aries Ecliptic cartesian coordinates

- Vec3D convVecHAE2SW (const Vec3D &pos_HAE_c, hcFloat swSpeed, hcFloat lowerR, const hcDate &date) const

  convert vector from Heliocentric Aries Ecliptic cartesian coordinates to solar wind frame

- template<class S >
  void transformSphericalCoordSystem (S theta, S phi)

  transforms point given in spherical coordinates into another spherical system

- template<class S >
  void transformSphericalCoordSystemBack (S theta, S phi)

  transform back point given in spherical coordinates (see transformSphericalCoordSystem)

- void cp (const Vec3D &v, Vec3D *res) const

  cross product of two Vec3D

- Vec3D cp (const Vec3D &other) const

  cross product of two Vec3D

- hcFloat **getAngle** (const Vec3D &other) const
- hcFloat **getAngle2** (const Vec3D &other) const
- void **dumpSphericalCoords** ()

Additional Inherited Members

4.57.1 Detailed Description

3D Vectors

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

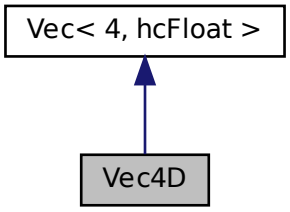
- engine/math/hcVec.h
- engine/math/hcVec.cpp
- src/ellipticalGrid.cpp

4.58 Vec4D Class Reference

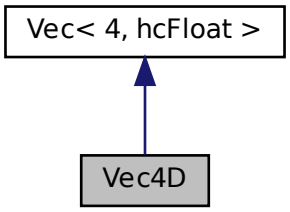
4D Vectors

`#include <hcVec.h>`

Inheritance diagram for Vec4D:



Collaboration diagram for Vec4D:



Public Member Functions

- Vec4D ()
  std constructor
- Vec4D (const Vec4D &other)
  cpy constructor
- Vec4D (const Vec3D &other)
  homogen-cpy constructor
- template<class S >
  Vec4D (const Vec< 4, S > &other)

pseudo-cpy constructor

- template<class S >
  Vec4D (S x, S y, S z, S w)
- ~Vec4D ()

  destructor

- Vec4D & **operator=** (const Vec4D &other)
- template<class S >
  Vec4D & **operator=** (const Vec< 4, S > &other)
- template<class S >
  void **set** (S x, S y, S z, S w)
- template<class S >
  void **setPos** (S x, S y)
- template<class S >
  void **setTex** (S u, S v)

Additional Inherited Members

4.58.1 Detailed Description

4D Vectors

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

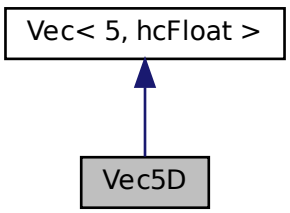
- engine/math/hcVec.h
- engine/math/hcVec.cpp

4.59 Vec5D Class Reference

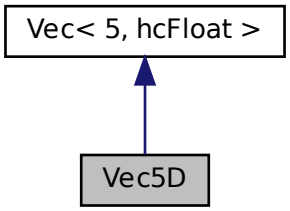
5D Vectors

`#include <hcVec.h>`

Inheritance diagram for Vec5D:



Collaboration diagram for Vec5D:



Public Member Functions

- template<class S >
  Vec5D (S x, S y, S z, S u, S v)
- template<class S >
  void **set** (S x, S y, S z, S u, S v)

Additional Inherited Members

4.59.1 Detailed Description

5D Vectors

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

- engine/math/hcVec.h
- engine/math/hcVec.cpp

Index