

# **NumCosmo Reference Manual**

---

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> NumCosmo Reference Manual		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 19, 2013	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>NumCosmo Overview</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Library Dependencies . . . . .	1
1.3	Compiling and Installing . . . . .	2
1.4	Bibliography . . . . .	4
<b>2</b>	<b>Numcosmo Math</b>	<b>5</b>
2.1	Vector Object . . . . .	5
2.2	Matrix Object . . . . .	16
2.3	Lapack Helper C Functions . . . . .	27
2.4	Model Scalar Parameter Description . . . . .	28
2.5	Model Vector Parameter Description . . . . .	35
2.6	Reparametrization Abstract Class . . . . .	43
2.7	Linear Reparametrization . . . . .	48
2.8	Model Abstract Class . . . . .	49
2.9	Model Update Control Object . . . . .	68
2.10	A Set of NcmModels . . . . .	70
2.11	A Function of NcmMSet . . . . .	88
2.12	Function Evaluator . . . . .	92
2.13	Logarithm Fast Fourier Algorithm . . . . .	93
2.14	Library Configuration . . . . .	96
2.15	Numerical and Physical Constants . . . . .	113
2.16	Splines 1D . . . . .	133
2.16.1	Spline Abstract Class . . . . .	133
2.16.2	GSL Spline . . . . .	140
2.16.3	Cubic Spline Abstract Class . . . . .	143
2.16.4	Notaknot Cubic Spline . . . . .	143
2.16.5	Spline Autoknots . . . . .	145
2.16.6	ODE Spline Interpolation . . . . .	146
2.17	Splines 2D . . . . .	148

2.17.1	Bidimensional Spline Abstract Class . . . . .	148
2.17.2	Bidimensional Spline from Spline . . . . .	155
2.17.3	Bidimensional Bicubic Spline . . . . .	156
2.17.4	Bidimensional Spline from Spline (GSL) . . . . .	160
2.18	Special Functions . . . . .	161
2.18.1	Trigonometric Integrals . . . . .	161
2.18.2	Hypergeometric 0F1 . . . . .	162
2.18.3	Spherical Bessel . . . . .	163
2.18.4	Spherical Bessel Integral . . . . .	166
2.18.5	Spherical Bessel -- Double Precision . . . . .	175
2.18.6	Spherical Bessel Integral -- Double Precision . . . . .	178
2.19	Data Objects . . . . .	186
2.19.1	Data Abstract Class . . . . .	186
2.19.2	Data Set . . . . .	190
2.19.3	Gaussian Data - InvCov . . . . .	197
2.19.4	Gaussian Data - DiagCov . . . . .	198
2.19.5	Gaussian Data - Cov . . . . .	200
2.19.6	Poisson Data . . . . .	202
2.19.7	One Variable Distribution Data . . . . .	204
2.20	Statistical Analysis . . . . .	205
2.20.1	Likelihood . . . . .	205
2.20.2	Statistical Priors . . . . .	210
2.20.3	Fitting State . . . . .	214
2.20.4	Model Fitting Abstract Class . . . . .	218
2.20.5	NLopt Interface Object . . . . .	241
2.20.6	Least Squares -- GSL . . . . .	244
2.20.7	Non-linear Minimization -- GSL . . . . .	245
2.20.8	Non-linear Simplex Minimization -- GSL . . . . .	247
2.20.9	Least Squares -- Levmar . . . . .	250
2.20.10	Monte Carlo Analysis . . . . .	252
2.20.11	Likelihood Ratio 1D . . . . .	255
2.20.12	Likelihood Ratio 2D . . . . .	258
2.21	GObject introspection compatibility . . . . .	262
2.21.1	Gir Scanning Compatibility. . . . .	262
<b>3</b>	<b>Other Utilities Objects</b>	<b>264</b>
3.1	Healpix . . . . .	264
3.2	Spherical Shell Map . . . . .	267

---

<b>4</b>	<b>Models</b>	<b>274</b>
4.1	Cosmological Model Abstract Class . . . . .	274
4.2	Homogeneous and Isotropic Models Priors . . . . .	287
4.3	$\Lambda$ CDM . . . . .	289
4.4	Darkenergy . . . . .	292
4.4.1	Dark Energy Abstract Class . . . . .	292
4.4.2	Dark Energy -- XCDM . . . . .	298
4.4.3	Dark Energy -- Linder . . . . .	300
4.4.4	Dark Energy -- Jassal-Bagla-Padmanabhan . . . . .	302
4.4.5	Dark Energy -- Quintessence (Inspired) . . . . .	304
4.4.6	Quantum Gravity Bouncing Model . . . . .	307
4.5	Kinematical . . . . .	317
4.5.1	Constant Desceleration Parameter Model . . . . .	317
4.5.2	Linear Desceleration Parameter Model . . . . .	321
4.5.3	Linear Spline Desceleration Parameter Model . . . . .	326
4.5.4	Spline Desceleration Parameter Model . . . . .	332
<b>5</b>	<b>Cosmological Functions</b>	<b>340</b>
5.1	Cosmological Distances and Times . . . . .	340
5.2	Recombination . . . . .	351
5.3	Recombination Seager 1999. . . . .	363
5.4	Supernovae Distance Covariance . . . . .	364
5.5	Scale Factor . . . . .	370
<b>6</b>	<b>Perturbations</b>	<b>374</b>
6.1	Linear Perturbations . . . . .	374
6.2	Perturbation Covariance . . . . .	383
6.3	Hydrodynamic Perturbations . . . . .	383
<b>7</b>	<b>Large Scale Structure</b>	<b>384</b>
7.1	Window Function . . . . .	384
7.1.1	Window Function Abstract Class . . . . .	384
7.1.2	Top-hat Window Function . . . . .	386
7.1.3	Gaussian window function . . . . .	388
7.2	Transfer Function . . . . .	389
7.2.1	Transfer Function Abstract Class . . . . .	389
7.2.2	BBKS Transfer Function . . . . .	391
7.2.3	EH Transfer Function . . . . .	392
7.2.4	CAMB Transfer Function . . . . .	392
7.2.5	Pert Transfer Function . . . . .	393

7.3	Perturbations Growth Function . . . . .	394
7.4	Matter Fluctuation Variance . . . . .	397
7.5	Multiplicity Function . . . . .	403
7.5.1	Multiplicity Function . . . . .	403
7.5.2	Press-Schechter Multiplicity Function . . . . .	405
7.5.3	Sheth-Tormen Multiplicity Function . . . . .	406
7.5.4	Jenkins Multiplicity Function . . . . .	410
7.5.5	Warren Multiplicity Function . . . . .	415
7.5.6	Tinker Multiplicity Function . . . . .	418
7.5.7	Mean Tinker Multiplicity Function . . . . .	423
7.5.8	Critical Tinker Multiplicity Function . . . . .	424
7.6	Mass Function . . . . .	426
7.7	Halo Bias Function Type . . . . .	434
7.7.1	Halo Bias Function Type . . . . .	434
7.7.2	PS Halo Bias Function Type . . . . .	435
7.7.3	ST Halo Bias Function Type . . . . .	437
7.7.4	ST Halo Bias Function Type . . . . .	440
7.7.5	Tinker Halo Bias Function Type . . . . .	443
7.8	Mean Halo Bias Function . . . . .	447
7.9	Galaxy Angular Corelation Function . . . . .	450
7.10	Cluster Redshift . . . . .	451
7.10.1	Abstract Cluster Redshift Object . . . . .	451
7.10.2	Cluster Abundance Redshift No Distribution . . . . .	455
7.10.3	Individual Gaussian Photoz Cluster . . . . .	456
7.10.4	Global Gaussian Photoz Cluster . . . . .	458
7.11	Cluster Mass . . . . .	460
7.11.1	Cluster Mass Distribution . . . . .	460
7.11.2	Cluster Mass No Distribution . . . . .	465
7.11.3	Cluster Mass Ln Normal Distribution . . . . .	467
7.11.4	SZ Cluster Mass Distribution . . . . .	468
7.11.5	SZ Cluster Mass Distribution . . . . .	472
7.11.6	SZ and X ray Cluster Abundance Mass Distributions . . . . .	476
7.12	Cluster Abundance Distribution . . . . .	479

---

<b>8</b>	<b>Cosmological Data</b>	<b>491</b>
8.1	CMB Data . . . . .	491
8.2	Cosmic Microwave Background Data -- Shift Parameter . . . . .	492
8.3	Cosmic Microwave Background Data -- Distance priors . . . . .	494
8.4	Hubble Function Data . . . . .	495
8.5	Hubble Function Data from BAO . . . . .	498
8.6	BAO Data . . . . .	500
8.7	Baryonic Oscillation Data -- Acoustic Scale . . . . .	501
8.8	Baryonic Oscillation Data -- Volume Mean . . . . .	503
8.9	Baryonic Oscillation Data -- rDv . . . . .	505
8.10	Baryonic Oscillation Data -- DVDV . . . . .	508
8.11	SN Ia Data . . . . .	509
8.12	Distance Modulus Data . . . . .	512
8.13	Supernovae Ia Data -- Covariance . . . . .	514
8.14	Cluster number count data . . . . .	519
8.15	Cluster number count data . . . . .	524
<b>9</b>	<b>Mathematical Utilities</b>	<b>526</b>
9.1	Divided Difference . . . . .	526
9.2	Binnary Splitting . . . . .	527
9.3	Polynomials . . . . .	530
9.4	Quaternions . . . . .	532
9.5	Miscellaneous Utilities . . . . .	538
9.6	Matrix Exponential . . . . .	546
9.7	MPQ Tree . . . . .	547
9.8	Sundials CVODE interface . . . . .	547
9.9	Magnus Iserles Ode Method . . . . .	548
9.10	Unidimensional Grid . . . . .	550
9.11	Quadrature Algorithms . . . . .	555
9.12	Function Cache . . . . .	557
9.13	Memory Pool . . . . .	560
9.14	Numerical Integration . . . . .	562
<b>10</b>	<b>Object Hierarchy</b>	<b>569</b>
<b>11</b>	<b>Annotation Glossary</b>	<b>572</b>
<b>12</b>	<b>Index</b>	<b>574</b>

# Chapter 1

## NumCosmo Overview

### 1.1 Introduction

Introduction — A general description of the library functionality

#### Overview

The NumCosmo is a free software C library whose main purposes are to test cosmological models using observational data and to provide a set of tools to perform cosmological calculations. Particularly, the current version has implemented three different probes: cosmic microwave background (CMB),supernovae type Ia (SNeIa) and large scale structure (LSS) information, such as baryonic acoustic oscillations (BAO) and galaxy cluster abundance. The code supports a joint analysis of these data and the parameter space can include cosmological and phenomenological parameters. It is worth emphasizing that NumCosmo matter power spectrum and CMB codes were written independently of other implementations such as CMBFAST, CAMB, etc.

The library is structured in such way to simplify the inclusion of non-standard cosmological models. Besides the functions related to cosmological quantities, this library also implements mathematical and statistical tools. The former was developed to enable the inclusion of other probes and/or theoretical models and to optimize the codes. The statistical framework comprises algorithms which define likelihood functions, minimization, Monte Carlo, Fisher Matrix and profile likelihood methods.

### 1.2 Library Dependencies

Dependencies — The dependency set of NumCosmo

#### Required Dependencies

- GLib/GObject
  - <http://www.gtk.org/>, <http://developer.gnome.org/glib/stable/>
  - Object oriented infrastructure, automatic bindings through GObject introspection.
  - Portability issues: types, filesystem, threads, etc.
  - Several generic data structures: hash tables, binary trees, etc.
  - Utilities: timers for benchmark, logging facilities, etc.
- GSL - GNU Scientific Library
  - <http://www.gnu.org/software/gsl/>
  - Special functions.



- Multidimensional minimization, including both algorithms with and without derivatives.
- Interpolation.
- Several other mathematical algorithms.
- GMP/MPFR - GNU Multiple Precision Arithmetic Library / Multiple-precision Floating-point computations with correct Rounding
  - <http://gmplib.org/> / <http://www.mpfr.org/>
  - Special function with arbitrary precision.
  - Binary splitting algorithms.
  - One dimensional grid with rationals.
- FFTW3 - The Fastest Fourier Transform in the West
  - <http://www.fftw.org/>
  - Self optimized discrete Fourier transform used in spherical harmonics decomposition.
- CFitsio
  - <http://heasarc.gsfc.nasa.gov/fitsio/>
  - Manipulate several observational data files.
- Sundials - SUite of Nonlinear and DIfferential/ALgebraic equation Solvers
  - <https://computation.llnl.gov/casc/sundials/main.html>
  - Ode system solver with support to several algorithms including stiff ode systems.

## Optional Dependencies

- ATLAS - Automatically Tuned Linear Algebra Software
  - <http://math-atlas.sourceforge.net/>
  - Optimized basic linear algebra package (used transparently by several algorithms in NumCosmo).
- Levmar - Levenberg-Marquardt nonlinear least squares algorithms in C/C++
  - <http://www.ics.forth.gr/~lourakis/levmar/>
  - Least-Squares algorithms.
- NLOpt - NonLinear Optimization
  - <http://ab-initio.mit.edu/wiki/index.php/NLOpt>
  - Common interface to several optimization (minimization) algorithms including global and local, constrained and unconstrained optimization.
- Cuba Library - library for multidimensional numerical integration
  - <http://www.feynarts.de/cuba/>
  - Several algorithms (deterministic and non-deterministic) for numerical integration of multidimensional functions.

## 1.3 Compiling and Installing

Installation — Instructions for compilation and installation

---

## Prerequisites

The NumCosmo library depends on several other libraries, most of them are commonly found on most modern linux distributions. Their description are here: [Library Dependencies](#).

For Debian like systems (including Ubuntu) it is necessary to install the following prerequisite packages:

- For most systems the following packages can be found in the main repositories: *gobject-introspection*, *gir1.2-glib-2.0*, *libgirepository1.0-dev*, *gcc*, *pkg-config*, *libglib2.0-dev*, *libgmp3-dev*, *libmpfr-dev*, *libgsl0-dev*, *libsqlite3-dev*, *libfftw3-dev*.  
If you want to build from the repository you also need: *autotools-dev*, *libtool*, *gtk-doc-tools*.
- The other packages are usually not found on the official repositories.
  - For atlas support on Debian see: <http://wiki.debian.org/DebianScience/LinearAlgebraLibraries>.
  - For atlas support on Ubuntu you need something similar to: *libatlas-base-dev*, *liblapack-dev*.  
They also have versions optimized to some processors, however, ideally you should compile these packages locally to take full advantage of you system.
  - The other packages sometimes can be found with the following package names: *libnlopt-dev*, *libsundials-dev*, *libcfitsio3-dev*, *libcuba3-dev*. The packages above also support the usual: `configure`; `make`; `make install` (however, see [NumCosmo compilation](#))

For RPM based distributions (including Fedora, OpenSuse, etc) it is necessary to install the following prerequisite packages:

- For most systems the following packages can be found in the main repositories: *pkg-config*, *gobject-introspection-devel*, *glib2-devel*, *gsl-devel*, *gmp-devel*, *mpfr-devel*, *fftw3-devel*, *sqlite3-devel*.  
If you want to build from the repository you also need: *autoconf*, *automake*, *libtool*, *gtk-doc*.
- The other packages are usually not found on the official repositories.
  - The other packages sometimes can be found with the following package names: *nlopt-devel*, *sundials-devel*, *libcfitsio-devel*, *lapack-devel*, *atlas-devel*, *lapack-devel*. Most of the packages above also support the usual: `configure`; `make`; `make install` (however, see [NumCosmo compilation](#))

## NumCosmo compilation

The sources can be downloaded from <http://download.savannah.gnu.org/releases/numcosmo/>.

You can also obtain the sources directly from the git repository, for example the command:

```
git clone git://git.sv.gnu.org/numcosmo.git
```

creates a local clone of the project repository. See also <http://git.savannah.gnu.org/cgit/numcosmo.git/>.

For the version cloned from the repository it is necessary to first build the configure script, to do that, execute:

```
NOCONFIGURE=yes ./autogen.sh
```

this command requires additional dependencies cited above. Note that if you don't want to develop the library itself you probably want to get a release from the link above where the configure script is already built and, therefore, there is no need of additional dependencies.

To prepare the project for compilation one can run from the sources directory

```
./configure --prefix=/usr --libdir=/usr/lib
```

in the command above the installation prefix and library directories are explicit set, this is necessary to install the gobject typelibs in the correct path and allow its usage.

Note also that for some 64 bits system the library directory is */usr/lib64* and, therefore, in this case the command above must be properly modified. In the default directory organization the bindings typelibs stay in */usr/lib/girepository-1.0/* and the bindings xml in */usr/share/gir-1.0/*. You can look for these directories to check where the system bindings are located to determine the correct directories to pass to the configure script.

For additional options run

```
./configure --help
```

After running successfully the configure script the library can be compile by running

```
make -j4
```

where the *-j4* tells make to run four parallel compilation. It can be used to take advantage of system with multiple cores.

To check if everything went ok, run

```
make check
```

and to install in your system

```
make install
```

note that, for the last command, the user must have root privileges to install the library in the system.

## 1.4 Bibliography

Bibliography — The set of books and papers used in the library

### References

- [1] D. Pequignot, P. Petitjean, and C. Boisson, *Astron. Astrophys.* *251*, 680 (1991), URL <http://adsabs.harvard.edu/abs/1991A%26A>.
- [2] W. Hu and N. Sugiyama, *Astrophys. J.* *471*, 542 (1996), [astro-ph/9510117](#).
- [3] D. J. Eisenstein and W. Hu, *Astrophys. J.* *496*, 605 (1998), [astro-ph/9709112](#).
- [4] D. G. Hummer and P. J. Storey, *Mon. Not. R. Astron. Soc.* *297*, 1073 (1998).
- [5] S. Seager, D. D. Sasselov, and D. Scott, *Astrophys. J.* *523*, L1 (1999).
- [6] S. Seager, D. D. Sasselov, and D. Scott, *Astrophys. J. Suppl. Ser.* *128*, 407 (2000), [arXiv:astro-ph/9912182](#).
- [7] E. V. Linder, *Phys. Rev. Lett.* *90*, 091301 (2003), [astro-ph/0208512](#).
- [8] H. K. Jassal, J. S. Bagla, and T. Padmanabhan, *Mon. Not. R. Astron. Soc.* *356*, L11 (2005), [astro-ph/0404378](#).
- [9] D. J. Eisenstein, I. Zehavi, D. W. Hogg, R. Scoccimarro, M. R. Blanton, R. C. Nichol, R. Scranton, H. Seo, M. Tegmark, Z. Zheng,
- [10] W. J. Percival, S. Cole, D. J. Eisenstein, R. C. Nichol, J. A. Peacock, A. C. Pope, and A. S. Szalay, *Mon. Not. R. Astron. Soc.* *33*,
- [11] S. Weinberg, *Cosmology* (Oxford University Press, 2008).
- [12] A. Conley, J. Guy, M. Sullivan, N. Regnault, P. Astier, C. Balland, S. Basa, R. G. Carlberg, D. Fouchez, D. Hardin, et al., *Astrophys. J.* *686*, 59 (2008), [arXiv:0804.3438](#).
- [13] M. Sullivan, J. Guy, A. Conley, N. Regnault, P. Astier, C. Balland, S. Basa, R. G. Carlberg, D. Fouchez, D. Hardin, et al., *Astrophys. J.* *686*, 86 (2008), [arXiv:0804.3439](#).

## Chapter 2

# Numcosmo Math

## 2.1 Vector Object

Vector Object — Vector object representing arrays of doubles

### Synopsis

```
enum                NcmVectorInternal;
struct              NcmVectorClass;
struct              NcmVector;
#define              NCM_N2VECTOR                (v)
#define              NCM_VECTOR_DATA              (cv)
NcmVector *         ncm_vector_new                (gsize n);
NcmVector *         ncm_vector_new_gsl            (gsl_vector *gv);
NcmVector *         ncm_vector_new_gsl_static     (gsl_vector *gv);
NcmVector *         ncm_vector_new_array         (GArray *a);
NcmVector *         ncm_vector_new_data_slice     (gdouble *d,
                                                    const gsize size,
                                                    const gsize stride);
NcmVector *         ncm_vector_new_data_malloc    (gdouble *d,
                                                    const gsize size,
                                                    const gsize stride);
NcmVector *         ncm_vector_new_data_static    (gdouble *d,
                                                    const gsize size,
                                                    const gsize stride);
NcmVector *         ncm_vector_new_variant        (GVariant *var);
NcmVector *         ncm_vector_ref               (NcmVector *cv);
const NcmVector *    ncm_vector_new_data_const    (const gdouble *d,
                                                    const gsize size,
                                                    const gsize stride);
NcmVector *         ncm_vector_get_subvector      (NcmVector *cv,
                                                    const gsize k,
                                                    const gsize size);
GVariant *          ncm_vector_get_variant        (NcmVector *v);
const NcmVector *    ncm_vector_new_gsl_const     (const gsl_vector *v);
gdouble             ncm_vector_get               (const NcmVector *cv,
                                                    const guint i);
gdouble             ncm_vector_fast_get          (const NcmVector *cv,
                                                    const guint i);
gdouble *           ncm_vector_ptr               (NcmVector *cv,
```

void	ncm_vector_set	const guint i); (NcmVector *cv, const guint i, const gdouble val);
void	ncm_vector_fast_set	(NcmVector *cv, const guint i, const gdouble val);
void	ncm_vector_addto	(NcmVector *cv, const guint i, const gdouble val);
void	ncm_vector_subfrom	(NcmVector *cv, const guint i, const gdouble val);
void	ncm_vector_fast_subfrom	(NcmVector *cv, const guint i, const gdouble val);
void	ncm_vector_set_all	(NcmVector *cv, const gdouble val);
void	ncm_vector_scale	(NcmVector *cv, const gdouble val);
void	ncm_vector_div	(NcmVector *cv1, const NcmVector *cv2);
void	ncm_vector_add	(NcmVector *cv1, const NcmVector *cv2);
void	ncm_vector_sub	(NcmVector *cv1, const NcmVector *cv2);
void	ncm_vector_set_zero	(NcmVector *cv);
void	ncm_vector_memcpy	(NcmVector *cv1, const NcmVector *cv2);
void	ncm_vector_memcpy2	(NcmVector *cv1, const NcmVector *cv2, const guint cv1_start, const guint cv2_start, const guint size);
GArray *	ncm_vector_get_array	(NcmVector *cv);
GArray *	ncm_vector_dup_array	(NcmVector *cv);
gsl_vector *	ncm_vector_gsl	(NcmVector *cv);
const gsl_vector *	ncm_vector_const_gsl	(const NcmVector *cv);
guint	ncm_vector_len	(const NcmVector *cv);
guint	ncm_vector_stride	(const NcmVector *cv);
NcmVector *	ncm_vector_dup	(const NcmVector *cv);
void	ncm_vector_free	(NcmVector *cv);
void	ncm_vector_clear	(NcmVector **cv);
void	ncm_vector_const_free	(const NcmVector *cv);
N_Vector	ncm_vector_nvector	(NcmVector *cv);

## Object Hierarchy

```
GObject
+----NcmVector
```

## Properties

"values"                      GVariant\*                      : Read / Write

## Description

This object defines the functions for allocating and accessing vectors. Also includes several vector operations.

## Details

### enum NcmVectorInternal

```
typedef enum {
    NCM_VECTOR_SLICE = 0,
    NCM_VECTOR_GSL_VECTOR,
    NCM_VECTOR_MALLOC,
    NCM_VECTOR_ARRAY,
    NCM_VECTOR_DERIVED,
} NcmVectorInternal;
```

FIXME

**NCM\_VECTOR\_SLICE** FIXME

**NCM\_VECTOR\_GSL\_VECTOR** FIXME

**NCM\_VECTOR\_MALLOC** FIXME

**NCM\_VECTOR\_ARRAY** FIXME

**NCM\_VECTOR\_DERIVED** FIXME

### struct NcmVectorClass

```
struct NcmVectorClass {
};
```

### struct NcmVector

```
struct NcmVector;
```

### NCM\_N2VECTOR()

```
#define NCM_N2VECTOR(v) ((NcmVector *) ((v)->content))
```

### NCM\_VECTOR\_DATA()

```
#define NCM_VECTOR_DATA(cv) ((cv)->vv.vector.data)
```

### ncm\_vector\_new ()

```
NcmVector *          ncm_vector_new          (gsize n);
```

This function allocates memory for a new **NcmVector** of double with  $n$  components.

**$n$**  : defines the size of the vector.

**Returns** : A new **NcmVector**.

**ncm\_vector\_new\_gsl ()**

```
NcmVector *      ncm_vector_new_gsl      (gsl_vector *gv);
```

This function saves *gv* internally and frees it when it is no longer necessary. The *gv* vector must not be freed.

**gv** : vector from GNU Scientific Library (GSL) to be converted into a **NcmVector**.

**Returns** : A new **NcmVector**.

**ncm\_vector\_new\_gsl\_static ()**

```
NcmVector *      ncm_vector_new_gsl_static      (gsl_vector *gv);
```

This function saves *gv* internally and does not frees. The *gv* vector must be valid during the life of the created **NcmVector**.

**gv** : vector from GNU Scientific Library (GSL) to be converted into a **NcmVector**.

**Returns** : A new **NcmVector**.

**ncm\_vector\_new\_array ()**

```
NcmVector *      ncm_vector_new_array      (GArray *a);
```

This function saves *a* internally and frees it when it is no longer necessary. The *a* array must not be freed.

**a** : array of doubles to be converted into a **NcmVector**. [*array*][*element-type double*][*transfer full*]

**Returns** : A new **NcmVector**.

**ncm\_vector\_new\_data\_slice ()**

```
NcmVector *      ncm_vector_new_data_slice      (gdouble *d,
                                                  const gsize size,
                                                  const gsize stride);
```

This function returns a **NcmVector** of the array *d* allocated using *g\_slice* function. This function saves *a* internally and frees it when it is no longer necessary. The *a* vector must not be freed.

**d** : pointer to the first double allocated.

**size** : number of doubles allocated.

**stride** : the step-size from one element to the next in physical memory, measured in units of double.

**Returns** : A new **NcmVector**.

**ncm\_vector\_new\_data\_malloc ()**

```
NcmVector *      ncm_vector_new_data_malloc      (gdouble *d,
                                                  const gsize size,
                                                  const gsize stride);
```

This function returns a **NcmVector** of the array *d* allocated using malloc. It saves *d* internally and frees it when it is no longer necessary.

**d** : pointer to the first double allocated.

**size** : number of doubles allocated.

**stride** : the step-size from one element to the next in physical memory, measured in units of double.

**Returns** : A new **NcmVector**.

**ncm\_vector\_new\_data\_static ()**

```
NcmVector *      ncm_vector_new_data_static      (gdouble *d,
                                                  const gsize size,
                                                  const gsize stride);
```

This function returns a **NcmVector** of the array *d*. The memory allocated is kept during all time life of the object and must not be freed during this period.

**d** : pointer to the first double allocated.

**size** : number of doubles allocated.

**stride** : the step-size from one element to the next in physical memory, measured in units of double.

**Returns** : A new **NcmVector**.

**ncm\_vector\_new\_variant ()**

```
NcmVector *      ncm_vector_new_variant          (GVariant *var);
```

This function convert a **GVariant** array to a **NcmVector**.

**var** : a **GVariant** of the type "ad".

**Returns** : A new **NcmVector**.

**ncm\_vector\_ref ()**

```
NcmVector *      ncm_vector_ref                  (NcmVector *cv);
```

This function increses the reference count of *cv* the object.

**cv** : a **NcmVector**.

**Returns** : *cv*. *[transfer full]*



**ncm\_vector\_new\_data\_const ()**

```
const NcmVector *    ncm_vector_new_data_const      (const gdouble *d,
                                                    const gsize size,
                                                    const gsize stride);
```

This function returns a constant **NcmVector** of the array *d*. The memory allocated is kept during all time life of the object and must not be freed during this period.

**d** : pointer to the first double allocated.

**size** : number of doubles allocated.

**stride** : the step-size from one element to the next in physical memory, measured in units of double.

**Returns** : A new constant **NcmVector**.

**ncm\_vector\_get\_subvector ()**

```
NcmVector *          ncm_vector_get_subvector      (NcmVector *cv,
                                                    const gsize k,
                                                    const gsize size);
```

This function returns a **NcmVector** which is a subvector of the vector *cv*. The start of the new vector is the *k*-th component from the original vector *cv*. The new vector has *size* elements.

**cv** : a **NcmVector**.

**k** : component index of the original vector.

**size** : number of components of the subvector.

**Returns** : A **NcmVector**. *[transfer full]*

**ncm\_vector\_get\_variant ()**

```
GVariant *           ncm_vector_get_variant      (NcmVector *v);
```

Convert *v* to a GVariant of the type "ad" without destroying the original vector *v*;

**v** : a **NcmVector**.

**Returns** : A **GVariant** of the type "ad". *[transfer full]*

**ncm\_vector\_new\_gsl\_const ()**

```
const NcmVector *    ncm_vector_new_gsl_const      (const gsl_vector *v);
```

This function converts *v* into a constant **NcmVector**.

**v** : vector from GNU Scientific Library (GSL).

**Returns** : A new constant **NcmVector**.

**ncm\_vector\_get ()**

gdouble	ncm_vector_get	(const NcmVector *cv, const guint i);
---------	----------------	--

**cv** : a constant **NcmVector**.**i** : component index.**Returns** : The *i*-th component of the vector *cv*.**ncm\_vector\_fast\_get ()**

gdouble	ncm_vector_fast_get	(const NcmVector *cv, const guint i);
---------	---------------------	--

**cv** : a constant **NcmVector**.**i** : component index.**Returns** : The *i*-th component of the vector *cv* assuming stride == 1.**ncm\_vector\_ptr ()**

gdouble *	ncm_vector_ptr	(NcmVector *cv, const guint i);
-----------	----------------	------------------------------------

**cv** : a **NcmVector**.**i** : component index.**Returns** : A pointer to the *i*-th component of the vector *cv*.**ncm\_vector\_set ()**

void	ncm_vector_set	(NcmVector *cv, const guint i, const gdouble val);
------	----------------	--

This function sets the value of the *i*-th component of the vector *cv* to *val*.**cv** : a **NcmVector**.**i** : component index.**val** : a constant double.**ncm\_vector\_fast\_set ()**

void	ncm_vector_fast_set	(NcmVector *cv, const guint i, const gdouble val);
------	---------------------	--

This function sets the value of the *i*-th component of the vector *cv* to *val* assuming stride == 1.**cv** : a **NcmVector**.**i** : component index.**val** : a constant double.

**ncm\_vector\_addto ()**

```
void                ncm_vector_addto                (NcmVector *cv,  
                                                    const guint i,  
                                                    const gdouble val);
```

This function adds *val* to the value of the *i*-th component of *cv*.

**cv** : a **NcmVector**.

**i** : component index.

**val** : a constant double.

**ncm\_vector\_subfrom ()**

```
void                ncm_vector_subfrom              (NcmVector *cv,  
                                                    const guint i,  
                                                    const gdouble val);
```

This function subtracts *val* from the value of the *i*-th component of *cv*.

**cv** : a **NcmVector**.

**i** : component index.

**val** : a constant double.

**ncm\_vector\_fast\_subfrom ()**

```
void                ncm_vector_fast_subfrom         (NcmVector *cv,  
                                                    const guint i,  
                                                    const gdouble val);
```

This function subtracts *val* from the value of the *i*-th component of *cv* assuming `stride == 1`.

**cv** : a **NcmVector**.

**i** : component index.

**val** : a constant double.

**ncm\_vector\_set\_all ()**

```
void                ncm_vector_set_all              (NcmVector *cv,  
                                                    const gdouble val);
```

This function sets all the components of the vector *cv* to the value *val*.

**cv** : a **NcmVector**.

**val** : a constant double.

---

**ncm\_vector\_scale ()**

```
void                ncm_vector_scale                (NcmVector *cv,  
                                                    const gdouble val);
```

This function multiplies the components of the vector *cv* by the constant factor *val*.

**cv** : a **NcmVector**.

**val** : a constant double.

**ncm\_vector\_div ()**

```
void                ncm_vector_div                (NcmVector *cv1,  
                                                    const NcmVector *cv2);
```

This function divides the components of the vector *cv1* by the components of the vector *cv2*. The two vectors must have the same length.

**cv1** : a **NcmVector**, numerator.

**cv2** : a **NcmVector**, denominator.

**ncm\_vector\_add ()**

```
void                ncm_vector_add                (NcmVector *cv1,  
                                                    const NcmVector *cv2);
```

This function adds the components of the vector *cv2* to the components of the vector *cv1*. The two vectors must have the same length.

**cv1** : a **NcmVector**.

**cv2** : a **NcmVector**.

**ncm\_vector\_sub ()**

```
void                ncm_vector_sub                (NcmVector *cv1,  
                                                    const NcmVector *cv2);
```

This function subtracts the components of the vector *cv2* to the components of the vector *cv1*. The two vectors must have the same length.

**cv1** : a **NcmVector**.

**cv2** : a **NcmVector**.

**ncm\_vector\_set\_zero ()**

```
void                ncm_vector_set_zero            (NcmVector *cv);
```

This function sets all the components of the vector *cv* to zero.

**cv** : a **NcmVector**.

---

**ncm\_vector\_memcpy ()**

```
void          ncm_vector_memcpy          (NcmVector *cv1,
                                         const NcmVector *cv2);
```

This function copies the components of the vector *cv2* into the vector *cv1*. The two vectors must have the same length.

**cv1** : a **NcmVector**.

**cv2** : a **NcmVector**.

**ncm\_vector\_memcpy2 ()**

```
void          ncm_vector_memcpy2        (NcmVector *cv1,
                                         const NcmVector *cv2,
                                         const guint cv1_start,
                                         const guint cv2_start,
                                         const guint size);
```

This function copies *size* components of *cv2*, counting from *cv2\_start*, to the vector *cv1*, starting from the *cv1\_start* component. It is useful for vectors with different sizes.

**cv1** : a **NcmVector**.

**cv2** : a **NcmVector**.

**cv1\_start** : component of *cv1*.

**cv2\_start** : component of *cv2*.

**size** : number of components.

**ncm\_vector\_get\_array ()**

```
GArray *      ncm_vector_get_array      (NcmVector *cv);
```

FIXME

**cv** : a **NcmVector**.

**Returns** : FIXME. *[transfer container][element-type double]*

**ncm\_vector\_dup\_array ()**

```
GArray *      ncm_vector_dup_array      (NcmVector *cv);
```

FIXME

**cv** : a **NcmVector**.

**Returns** : FIXME. *[transfer full][element-type double]*

**ncm\_vector\_gsl ()**

```
gsl_vector *      ncm_vector_gsl      (NcmVector *cv);
```

FIXME

**cv** : a **NcmVector**.

**Returns** : FIXME

**ncm\_vector\_const\_gsl ()**

```
const gsl_vector * ncm_vector_const_gsl (const NcmVector *cv);
```

FIXME

**cv** : a **NcmVector**.

**Returns** : FIXME

**ncm\_vector\_len ()**

```
quint      ncm_vector_len      (const NcmVector *cv);
```

FIXME

**cv** : a **NcmVector**.

**Returns** : FIXME

**ncm\_vector\_stride ()**

```
quint      ncm_vector_stride      (const NcmVector *cv);
```

FIXME

**cv** : a **NcmVector**.

**Returns** : FIXME

**ncm\_vector\_dup ()**

```
NcmVector *      ncm_vector_dup      (const NcmVector *cv);
```

This function copies the elements of the constant vector **cv** into a new **NcmVector**.

**cv** : a constant **NcmVector**.

**Returns** : A **NcmVector**. *[transfer full]*

---

**ncm\_vector\_free ()**

```
void                ncm_vector_free                (NcmVector *cv);
```

Atomically decrements the reference count of *cv* by one. If the reference count drops to 0, all memory allocated by *cv* is released.

*cv* : a **NcmVector**.

**ncm\_vector\_clear ()**

```
void                ncm_vector_clear                (NcmVector **cv);
```

Atomically decrements the reference count of *cv* by one. If the reference count drops to 0, all memory allocated by *cv* is released. The pointer is set to NULL.

*cv* : a **NcmVector**.

**ncm\_vector\_const\_free ()**

```
void                ncm_vector_const_free            (const NcmVector *cv);
```

Atomically decrements the reference count of *cv* by one. If the reference count drops to 0, all memory allocated by *cv* is released.

*cv* : a constant **NcmVector**.

**ncm\_vector\_nvector ()**

```
N_Vector            ncm_vector_nvector                (NcmVector *cv);
```

FIXME

*cv* : a **NcmVector**.

**Returns** : FIXME

**Property Details****The "values" property**

"values"	GVariant*	: Read / Write
----------	-----------	----------------

values.

Allowed values: GVariant<a\*>

Default value: NULL

**2.2 Matrix Object**

Matrix Object — Matrix object representing an array of doubles.





```

void                ncm_matrix_memcpy                (NcmMatrix *cm1,
                                                    const NcmMatrix *cm2);
void                ncm_matrix_set_col               (NcmMatrix *cm,
                                                    const guint n,
                                                    const NcmVector *cv);
gdouble            ncm_matrix_fast_get              (NcmMatrix *cm,
                                                    const guint ij);
void                ncm_matrix_fast_set              (NcmMatrix *cm,
                                                    const guint ij,
                                                    const gdouble val);
NcmMatrix *         ncm_matrix_dup                  (const NcmMatrix *cm);
void                ncm_matrix_add_mul               (NcmMatrix *cm,
                                                    const gdouble alpha,
                                                    NcmMatrix *b);
void                ncm_matrix_free                  (NcmMatrix *cm);
void                ncm_matrix_clear                 (NcmMatrix **cm);
void                ncm_matrix_cholesky_decomp       (NcmMatrix *cm);

```

## Object Hierarchy

```

GObject
+----NcmMatrix

```

## Properties

```

"values"                GVariant*                : Read / Write

```

## Description

This object defines the functions for allocating and accessing matrices. Also includes serveral matrix operations.

## Details

### enum NcmMatrixInternal

```

typedef enum {
    NCM_MATRIX_SLICE = 0,
    NCM_MATRIX_GSL_MATRIX,
    NCM_MATRIX_MALLOC,
    NCM_MATRIX_GARRAY,
    NCM_MATRIX_DERIVED,
} NcmMatrixInternal;

```

FIXME

**NCM\_MATRIX\_SLICE** FIXME

**NCM\_MATRIX\_GSL\_MATRIX** FIXME

**NCM\_MATRIX\_MALLOC** FIXME

**NCM\_MATRIX\_GARRAY** FIXME

**NCM\_MATRIX\_DERIVED** FIXME

**struct NcmMatrixClass**

```
struct NcmMatrixClass {  
};
```

**struct NcmMatrix**

```
struct NcmMatrix;
```

FIXME

**NCM\_MATRIX\_GSL()**

```
#define NCM_MATRIX_GSL(cm) (&(cm)->mv.matrix)
```

**NCM\_MATRIX\_COL\_LEN()**

```
#define NCM_MATRIX_COL_LEN(cm) ((cm)->mv.matrix.size1)
```

**NCM\_MATRIX\_ROW\_LEN()**

```
#define NCM_MATRIX_ROW_LEN(cm) ((cm)->mv.matrix.size2)
```

**NCM\_MATRIX\_DATA()**

```
#define NCM_MATRIX_DATA(cm) ((cm)->mv.matrix.data)
```

**NCM\_MATRIX\_NROWS()**

```
#define NCM_MATRIX_NROWS(cm) ((cm)->mv.matrix.size1)
```

**NCM\_MATRIX\_NCOLS()**

```
#define NCM_MATRIX_NCOLS(cm) ((cm)->mv.matrix.size2)
```

**ncm\_matrix\_new ()**

```
NcmMatrix * ncm_matrix_new (const gsize nrows,  
                             const gsize ncols);
```

This function allocates memory for a new **NcmMatrix** of doubles with *nrows* rows and *ncols* columns.

***nrows*** : number of rows.

***ncols*** : number of columns.

**Returns** : A new **NcmMatrix**.

---

**ncm\_matrix\_new\_gsl ()**

```
NcmMatrix *          ncm_matrix_new_gsl          (gsl_matrix *gm);
```

This function saves *gm* internally and frees it when it is no longer necessary. The *gm* matrix must not be freed.

**gm** : matrix from GNU Scientific Library (GSL) to be converted into a **NcmMatrix**.

**Returns** : A new **NcmMatrix**.

**ncm\_matrix\_new\_gsl\_static ()**

```
NcmMatrix *          ncm_matrix_new_gsl_static   (gsl_matrix *gm);
```

This function saves *gm* internally and does not frees it. The *gm* matrix must be valid during the life of the created **NcmMatrix**.

**gm** : matrix from GNU Scientific Library (GSL) to be converted into a **NcmMatrix**.

**Returns** : A new **NcmMatrix**.

**ncm\_matrix\_new\_array ()**

```
NcmMatrix *          ncm_matrix_new_array        (GArray *a,
                                                  const gsize ncols);
```

The number of rows is defined dividing the lenght of *a* by *ncols*. This function saves *a* internally and frees it when it is no longer necessary. The GArray *a* must not be freed.

**a** : GArray of doubles to be converted into a **NcmMatrix**. [*element-type double*]

**ncols** : number of columns.

**Returns** : A new **NcmMatrix**.

**ncm\_matrix\_new\_data\_slice ()**

```
NcmMatrix *          ncm_matrix_new_data_slice   (gdouble *d,
                                                  const gsize nrows,
                                                  const gsize ncols);
```

This function returns a **NcmMatrix** of the array *d* allocated using *g\_slice* function. It saves *d* internally and frees it when it is no longer necessary. The matrix has *nrows* rows and *ncols* columns. The physical number of columns in memory is also given by *ncols*.

**d** : pointer to the first double allocated.

**nrows** : number of rows.

**ncols** : number of columns.

**Returns** : A new **NcmMatrix**.

**ncm\_matrix\_new\_data\_malloc ()**

```
NcmMatrix *          ncm_matrix_new_data_malloc          (gdouble *d,
                                                         const gsize nrows,
                                                         const gsize ncols);
```

This function returns a **NcmMatrix** of the array *d* allocated using malloc. It saves *d* internally and frees it when it is no longer necessary.

**d** : pointer to the first double allocated.

**nrows** : number of rows.

**ncols** : number of columns.

**Returns** : A new **NcmMatrix**.

**ncm\_matrix\_new\_data\_static ()**

```
NcmMatrix *          ncm_matrix_new_data_static          (gdouble *d,
                                                         const gsize nrows,
                                                         const gsize ncols);
```

This function returns a **NcmMatrix** of the array *d*. The memory allocated is kept during all time life of the object and must not be freed during this period.

**d** : pointer to the first double allocated.

**nrows** : number of rows.

**ncols** : number of columns.

**Returns** : A new **NcmMatrix**.

**ncm\_matrix\_new\_data\_static\_tda ()**

```
NcmMatrix *          ncm_matrix_new_data_static_tda      (gdouble *d,
                                                         const gsize nrows,
                                                         const gsize ncols,
                                                         const gsize tda);
```

This function returns a **NcmMatrix** of the array *d* with a physical number of columns *tda* which may differ from the corresponding dimension of the matrix. The matrix has *nrows* rows and *ncols* columns, and the physical number of columns in memory is given by *tda*.

**d** : pointer to the first double allocated.

**nrows** : number of rows.

**ncols** : number of columns.

**tda** : physical number of columns which may differ from the corresponding dimension of the matrix.

**Returns** : A new **NcmMatrix**.

**ncm\_matrix\_get\_submatrix ()**

```
NcmMatrix *      ncm_matrix_get_submatrix      (NcmMatrix *cm,
                                                const gsize k1,
                                                const gsize k2,
                                                const gsize nrows,
                                                const gsize ncols);
```

This function returns a submatrix **NcmMatrix** of the matrix *cm*. The upper-left element of the submatrix is the element  $(k1, k2)$  of the original matrix. The submatrix has *nrows* rows and *ncols* columns.

**cm** : a **NcmMatrix**.

**k1** : row index of the original matrix *cm*.

**k2** : column index of the original matrix *cm*.

**nrows** : number of rows of the submatrix.

**ncols** : number of columns of the submatrix.

**Returns** : A **NcmMatrix**. *[transfer full]*

**ncm\_matrix\_get\_col ()**

```
NcmVector *      ncm_matrix_get_col            (NcmMatrix *cm,
                                                const gsize col);
```

This function returns the elements of the *col* column of the matrix *cm* into a **NcmVector**.

**cm** : a **NcmMatrix**.

**col** : column index.

**Returns** : A **NcmVector**. *[transfer full]*

**ncm\_matrix\_get\_row ()**

```
NcmVector *      ncm_matrix_get_row           (NcmMatrix *cm,
                                                const gsize row);
```

This function returns the elements of the *row* row of the matrix *cm* into a **NcmVector**.

**cm** : a **NcmMatrix**.

**row** : row index.

**Returns** : A **NcmVector**. *[transfer full]*

**ncm\_matrix\_new\_gsl\_const ()**

```
const NcmMatrix * ncm_matrix_new_gsl_const    (gsl_matrix *m);
```

This function converts *m* into a constant **NcmMatrix**.

**m** : matrix from GNU Scientific Library (GSL).

**Returns** : A new constant **NcmMatrix**.

**ncm\_matrix\_get ()**

```
gdouble          ncm_matrix_get          (const NcmMatrix *cm,
                                           const guint i,
                                           const guint j);
```

**cm** : a constant **NcmMatrix**.

**i** : row index.

**j** : column index.

**Returns** : The  $(i,j)$ -th element of the matrix *cm*.

**ncm\_matrix\_ptr ()**

```
gdouble *        ncm_matrix_ptr         (NcmMatrix *cm,
                                           const guint i,
                                           const guint j);
```

**cm** : a **NcmMatrix**.

**i** : row index.

**j** : column index.

**Returns** : A pointer to the  $(i,j)$ -th element of the matrix *cm*.

**ncm\_matrix\_ref ()**

```
NcmMatrix *      ncm_matrix_ref         (NcmMatrix *cm);
```

Increase the reference count of *cm* by one.

**cm** : a **NcmMatrix**.

**Returns** : *cm*. *[transfer full]*

**ncm\_matrix\_get\_array ()**

```
GArray *         ncm_matrix_get_array   (NcmMatrix *cm);
```

FIXME

**cm** : a **NcmMatrix**.

**Returns** : FIXME. *[transfer container][element-type double]*

**ncm\_matrix\_set ()**

```
void                ncm_matrix_set                (NcmMatrix *cm,  
                                                  const guint i,  
                                                  const guint j,  
                                                  const gdouble val);
```

This function sets the value of the  $(i,j)$ -th element of the matrix  $cm$  to  $val$ .

**$cm$**  : a **NcmMatrix**.

**$i$**  : row index.

**$j$**  : column index.

**$val$**  : a double.

**ncm\_matrix\_transpose ()**

```
void                ncm_matrix_transpose         (NcmMatrix *cm);
```

This function replaces the matrix  $cm$  by its transpose by copying the elements of the matrix in-place. The matrix must be square for this operation to be possible.

**$cm$**  : a **NcmMatrix**.

**ncm\_matrix\_set\_identity ()**

```
void                ncm_matrix_set_identity      (NcmMatrix *cm);
```

This function sets the elements of the matrix  $cm$  to the corresponding elements of the identity matrix, i.e. a unit diagonal with all off-diagonal elements zero. This applies to both square and rectangular matrices.

**$cm$**  : a **NcmMatrix**.

**ncm\_matrix\_set\_zero ()**

```
void                ncm_matrix_set_zero         (NcmMatrix *cm);
```

This function sets all the elements of the matrix  $cm$  to zero.

**$cm$**  : a **NcmMatrix**.

**ncm\_matrix\_scale ()**

```
void                ncm_matrix_scale            (NcmMatrix *cm,  
                                                  const gdouble val);
```

This function multiplies the elements of the matrix  $cm$  by the constant factor  $val$ . The result is stored in  $cm$ .

**$cm$**  : a **NcmMatrix**.

**$val$**  : a double.

---

**ncm\_matrix\_memcpy ()**

```
void                ncm_matrix_memcpy                (NcmMatrix *cm1,
                                                    const NcmMatrix *cm2);
```

This function copies the elements of the matrix *cm1* into the matrix *cm2*. The two matrices must have the same size.

**cm1** : a **NcmMatrix**.

**cm2** : a **NcmMatrix**.

**ncm\_matrix\_set\_col ()**

```
void                ncm_matrix_set_col                (NcmMatrix *cm,
                                                    const guint n,
                                                    const NcmVector *cv);
```

This function copies the elements of the vector *cv* into the *n*-th column of the matrix *cm*. The length of the vector must be the same as the length of the column.

**cm** : a **NcmMatrix**.

**n** : column index.

**cv** : a constant **NcmVector**.

**ncm\_matrix\_fast\_get ()**

```
gdouble            ncm_matrix_fast_get                (NcmMatrix *cm,
                                                    const guint ij);
```

FIXME

**cm** : a **NcmMatrix**.

**ij** : FIXME

**Returns** : FIXME

**ncm\_matrix\_fast\_set ()**

```
void                ncm_matrix_fast_set                (NcmMatrix *cm,
                                                    const guint ij,
                                                    const gdouble val);
```

FIXME

**cm** : a **NcmMatrix**.

**ij** : FIXME

**val** : FIXME



**ncm\_matrix\_dup ()**

```
NcmMatrix *      ncm_matrix_dup      (const NcmMatrix *cm);
```

Duplicates *cm* setting the same values of the original properties.

**cm** : a constant **NcmMatrix**

**Returns** : A **NcmMatrix**. *[transfer full]*

**ncm\_matrix\_add\_mul ()**

```
void      ncm_matrix_add_mul      (NcmMatrix *cm,
                                   const gdouble alpha,
                                   NcmMatrix *b);
```

FIXME

**cm** : FIXME

**alpha** : FIXME

**b** : FIXME

**ncm\_matrix\_free ()**

```
void      ncm_matrix_free      (NcmMatrix *cm);
```

Atomically decrements the reference count of *cm* by one. If the reference count drops to 0, all memory allocated by *cm* is released.

**cm** : a **NcmMatrix**.

**ncm\_matrix\_clear ()**

```
void      ncm_matrix_clear      (NcmMatrix **cm);
```

Atomically decrements the reference count of *cm* by one. If the reference count drops to 0, all memory allocated by *cm* is released. The pointer is set to NULL.

**cm** : a **NcmMatrix**.

**ncm\_matrix\_cholesky\_decomp ()**

```
void      ncm_matrix_cholesky_decomp      (NcmMatrix *cm);
```

Calculates inplace the Cholesky decomposition for a symmetric positive definite matrix.

**cm** : a **NcmMatrix**.

### Property Details

#### The "values" property

"values"	GVariant*	: Read / Write
----------	-----------	----------------

values.

Allowed values: GVariant<a\*>

Default value: NULL

### 2.3 Lapack Helper C Functions

Lapack Helper C Functions — Encapsulated fortran lapack functions

#### Synopsis

```
void                dptsv_(
                    (gint *N,
                     gint *NRHS,
                     gdouble *d,
                     gdouble *e,
                     gdouble *b,
                     gint *ldb,
                     gint *info);
gint                ncm_lapack_dptsv
                    (gdouble *d,
                     gdouble *e,
                     gdouble *b,
                     guint size);
#define              NCM_LAPACK_CHECK_INFO
                    (func,
                     info)
```

#### Description

FIXME

#### Details

##### dptsv\_()

void	dptsv_	(gint *N, gint *NRHS, gdouble *d, gdouble *e, gdouble *b, gint *ldb, gint *info);
------	--------	---

FIXME

***N*** : FIXME

***NRHS*** : FIXME

***d*** : FIXME***e*** : FIXME***b*** : FIXME***ldb*** : FIXME***info*** : FIXME**ncm\_lapack\_dptsv ()**

```
gint          ncm_lapack_dptsv          (gdouble *d,
                                         gdouble *e,
                                         gdouble *b,
                                         guint size);
```

FIXME

***d*** : FIXME***e*** : FIXME***b*** : FIXME***size*** : FIXME***Returns*** : FIXME**NCM\_LAPACK\_CHECK\_INFO()**

```
#define NCM_LAPACK_CHECK_INFO(func,info) if ((info) != 0) g_error ("Lapack[%s] error %d", ↵
func, (info))
```

## 2.4 Model Scalar Parameter Description

Model Scalar Parameter Description — Describes the properties of a scalar parameter

### Synopsis

```
struct          NcmSParamClass;
enum            NcmParamType;
struct          NcmSParam;
NcmSParam *     ncm_sparam_new          (gchar *name,
                                         gchar *symbol,
                                         gdouble lower_bound,
                                         gdouble upper_bound,
                                         gdouble scale,
                                         gdouble abstol,
                                         gdouble default_val,
                                         NcmParamType ftype);

NcmSParam *     ncm_sparam_copy         (NcmSParam *sparam);
NcmSParam *     ncm_sparam_ref          (NcmSParam *sparam);
void            ncm_sparam_free         (NcmSParam *sparam);
void            ncm_sparam_clear        (NcmSParam **sparam);
```

void	ncm_sparam_set_lower_bound	(NcmSParam *sparam, const gdouble lb);
void	ncm_sparam_set_upper_bound	(NcmSParam *sparam, const gdouble ub);
void	ncm_sparam_set_scale	(NcmSParam *sparam, const gdouble scale);
void	ncm_sparam_set_absolute_tolerance	(NcmSParam *sparam, const gdouble abstol);
void	ncm_sparam_set_default_value	(NcmSParam *sparam, const gdouble default_val);
void	ncm_sparam_set_fit_type	(NcmSParam *sparam, const NcmParamType ftype);
void	ncm_sparam_take_name	(NcmSParam *sparam, gchar *name);
void	ncm_sparam_take_symbol	(NcmSParam *sparam, gchar *symbol);
const gchar *	ncm_sparam_name	(const NcmSParam *sparam);
const gchar *	ncm_sparam_symbol	(const NcmSParam *sparam);
gdouble	ncm_sparam_get_lower_bound	(const NcmSParam *sparam);
gdouble	ncm_sparam_get_upper_bound	(const NcmSParam *sparam);
gdouble	ncm_sparam_get_scale	(const NcmSParam *sparam);
gdouble	ncm_sparam_get_absolute_tolerance	(const NcmSParam *sparam);
gdouble	ncm_sparam_get_default_value	(const NcmSParam *sparam);
NcmParamType	ncm_sparam_get_fit_type	(const NcmSParam *sparam);

## Object Hierarchy

```
GObject
+----NcmSParam
```

## Properties

"absolute-tolerance"	gdouble	: Read / Write
"default-value"	gdouble	: Read / Write
"fit-type"	NcmParamType	: Read / Write
"lower-bound"	gdouble	: Read / Write
"name"	gchar*	: Read / Write / Construct Only
"scale"	gdouble	: Read / Write
"symbol"	gchar*	: Read / Write / Construct Only
"upper-bound"	gdouble	: Read / Write

## Description

This object comprises the necessary properties to define a scalar parameter. It is used by **NcmModel** to store the description of the scalar model parameters.

## Details

### struct NcmSParamClass

```
struct NcmSParamClass {
};
```

**enum NcmParamType**

```
typedef enum {
    NCM_PARAM_TYPE_FREE = 0,
    NCM_PARAM_TYPE_FIXED,
} NcmParamType;
```

FIXME

**NCM\_PARAM\_TYPE\_FREE** FIXME

**NCM\_PARAM\_TYPE\_FIXED** FIXME

**struct NcmSParam**

```
struct NcmSParam;
```

**ncm\_sparam\_new ()**

```
NcmSParam *          ncm_sparam_new          (gchar *name,
                                              gchar *symbol,
                                              gdouble lower_bound,
                                              gdouble upper_bound,
                                              gdouble scale,
                                              gdouble abstol,
                                              gdouble default_val,
                                              NcmParamType ftype);
```

This function allocates memory for a new **NcmSParam** object and sets its properties to the values from the input arguments.

The *name* parameter is restricted to the interval [*lower\_bound*, *upper\_bound*]. *scale* is an initial step for the statistical algorithms. *abstol* is the absolute error tolerance of the parameter. *ftype* indicates if the parameter will be fitted or not.

**name** : "name".

**symbol** : "symbol".

**lower\_bound** : value of "lower-bound".

**upper\_bound** : value of "upper-bound".

**scale** : value of "scale".

**abstol** : value of "absolute-tolerance".

**default\_val** : value of "default-value".

**ftype** : a **NcmParamType**.

**Returns** : A new **NcmSParam**.

**ncm\_sparam\_copy ()**

```
NcmSParam *          ncm_sparam_copy          (NcmSParam *sparam);
```

Duplicates the **NcmSParam** object setting the same values of the original properties.

**sparam** : a **NcmSParam**.

**Returns** : A new **NcmSParam**. [*transfer full*]

**ncm\_sparam\_ref ()**

```
NcmSParam *          ncm_sparam_ref          (NcmSParam *sparam);
```

Atomically increase the reference count of *sparam* by one.

***sparam*** : a **NcmSParam**.

**Returns** : *sparam*. *[transfer full]*

**ncm\_sparam\_free ()**

```
void                ncm_sparam_free          (NcmSParam *sparam);
```

Atomically decrements the reference count of *sparam* by one. If the reference count drops to 0, all memory allocated by *sparam* is released.

***sparam*** : a **NcmSParam**.

**ncm\_sparam\_clear ()**

```
void                ncm_sparam_clear        (NcmSParam **sparam);
```

Atomically decrements the reference count of *sparam* by one. If the reference count drops to 0, all memory allocated by *sparam* is released. Set the pointer to NULL.

***sparam*** : a **NcmSParam**.

**ncm\_sparam\_set\_lower\_bound ()**

```
void                ncm_sparam_set_lower_bound (NcmSParam *sparam,  
                                                const gdouble lb);
```

Sets the value *lb* to the **"lower-bound"** property.

***sparam*** : a **NcmSParam**.

***lb*** : value of **"lower-bound"**.

**ncm\_sparam\_set\_upper\_bound ()**

```
void                ncm_sparam_set_upper_bound (NcmSParam *sparam,  
                                                const gdouble ub);
```

Sets the value *ub* to the **"upper-bound"** property.

***sparam*** : a **NcmSParam**.

***ub*** : value of **"upper-bound"**.

---

**ncm\_sparam\_set\_scale ()**

```
void                ncm_sparam_set_scale      (NcmSParam *sparam,
                                              const gdouble scale);
```

Sets the value *scale* to the "scale" property.

**sparam** : a **NcmSParam**.

**scale** : value of "scale".

**ncm\_sparam\_set\_absolute\_tolerance ()**

```
void                ncm_sparam_set_absolute_tolerance  (NcmSParam *sparam,
                                              const gdouble abstol);
```

Sets the value *abstol* to the "absolute-tolerance" property.

**sparam** : a **NcmSParam**.

**abstol** : value of "absolute-tolerance".

**ncm\_sparam\_set\_default\_value ()**

```
void                ncm_sparam_set_default_value      (NcmSParam *sparam,
                                              const gdouble default_val);
```

Sets the value *default\_val* to the "default-value" property.

**sparam** : a **NcmSParam**.

**default\_val** : value of "default-value".

**ncm\_sparam\_set\_fit\_type ()**

```
void                ncm_sparam_set_fit_type          (NcmSParam *sparam,
                                              const NcmParamType ftype);
```

Sets the value *ftype* to the "fit-type" property.

**sparam** : a **NcmSParam**.

**ftype** : a **NcmParamType**.

**ncm\_sparam\_take\_name ()**

```
void                ncm_sparam_take_name            (NcmSParam *sparam,
                                              gchar *name);
```

Take *name* as the name string. The caller doesn't have to free it any more.

**sparam** : a **NcmSParam**.

**name** : a string

**ncm\_sparam\_take\_symbol ()**

```
void                ncm_sparam_take_symbol      (NcmSParam *sparam,  
                                                gchar *symbol);
```

Take *symbol* as the symbol string. The caller doesn't have to free it any more.

***sparam*** : a **NcmSParam**.

***symbol*** : a string

**ncm\_sparam\_name ()**

```
const gchar *       ncm_sparam_name            (const NcmSParam *sparam);
```

***sparam*** : a **NcmSParam**.

**Returns** : the internal name string. The caller must not free it.

**ncm\_sparam\_symbol ()**

```
const gchar *       ncm_sparam_symbol         (const NcmSParam *sparam);
```

***sparam*** : a **NcmSParam**.

**Returns** : the internal symbol string. The caller must not free it.

**ncm\_sparam\_get\_lower\_bound ()**

```
gdouble             ncm_sparam_get_lower_bound (const NcmSParam *sparam);
```

***sparam*** : a **NcmSParam**.

**Returns** : the value of "lower-bound" property.

**ncm\_sparam\_get\_upper\_bound ()**

```
gdouble             ncm_sparam_get_upper_bound (const NcmSParam *sparam);
```

***sparam*** : a **NcmSParam**.

**Returns** : The value of "upper-bound" property.

**ncm\_sparam\_get\_scale ()**

```
gdouble             ncm_sparam_get_scale      (const NcmSParam *sparam);
```

***sparam*** : a **NcmSParam**.

**Returns** : The value of "scale" property.

---



**ncm\_sparam\_get\_absolute\_tolerance ()**

```
gdouble          ncm_sparam_get_absolute_tolerance    (const NcmSParam *sparam);
```

**sparam** : a **NcmSParam**.

**Returns** : the value of "**absolute\_tolerance**" property.

**ncm\_sparam\_get\_default\_value ()**

```
gdouble          ncm_sparam_get_default_value        (const NcmSParam *sparam);
```

**sparam** : a **NcmSParam**.

**Returns** : the value of "**default-value**" property.

**ncm\_sparam\_get\_fit\_type ()**

```
NcmParamType     ncm_sparam_get_fit_type            (const NcmSParam *sparam);
```

**sparam** : a **NcmSParam**.

**Returns** : the **NcmParamType** value of "**fit-type**" property.

**Property Details****The "**absolute-tolerance**" property**

"absolute-tolerance"	gdouble	: Read / Write
----------------------	---------	----------------

Absolute tolerance, whose value is restricted to [0, G\_MAXDOUBLE], is the size of the error used by **NcmFit**.

Allowed values:  $\geq 0$

Default value: 0

**The "**default-value**" property**

"default-value"	gdouble	: Read / Write
-----------------	---------	----------------

Parameter's default value.

Default value: 0

**The "**fit-type**" property**

"fit-type"	NcmParamType	: Read / Write
------------	--------------	----------------

Parameter's fit type: FIXED or FREE.

Default value: NCM\_PARAM\_TYPE\_FREE

**The "lower-bound" property**

"lower-bound"	gdouble	: Read / Write
---------------	---------	----------------

Lower parameter threshold whose value is restricted to  $[-G\_MAXDOUBLE, G\_MAXDOUBLE]$ .

Default value: 0

**The "name" property**

"name"	gchar*	: Read / Write / Construct Only
--------	--------	---------------------------------

The parameter's name must be a string written using only ASCII and `-`.

Default value: NULL

**The "scale" property**

"scale"	gdouble	: Read / Write
---------	---------	----------------

Scale, whose value is restricted to  $[0, G\_MAXDOUBLE]$ , is the step used by **NcmFit** to increment the value of the parameter.

Allowed values:  $\geq 0$

Default value: 0

**The "symbol" property**

"symbol"	gchar*	: Read / Write / Construct Only
----------	--------	---------------------------------

Parameter's name written in a usual form (including latex).

Default value: NULL

**The "upper-bound" property**

"upper-bound"	gdouble	: Read / Write
---------------	---------	----------------

Upper parameter threshold whose value is restricted to  $[-G\_MAXDOUBLE, G\_MAXDOUBLE]$ .

Default value: 0

## 2.5 Model Vector Parameter Description

Model Vector Parameter Description — Describes the properties of a vector parameter

## Synopsis

```

struct          NcmVParamClass;
struct          NcmVParam;
NcmVParam *     ncm_vparam_new                                (guint len,
                                                             NcmSParam *default_param);
NcmVParam *     ncm_vparam_full_new                          (guint len,
                                                             gchar *name,
                                                             gchar *symbol,
                                                             gdouble lower_bound,
                                                             gdouble upper_bound,
                                                             gdouble scale,
                                                             gdouble abstol,
                                                             gdouble default_val,
                                                             NcmParamType ftype);
NcmVParam *     ncm_vparam_copy                              (NcmVParam *vparam);
void            ncm_vparam_free                              (NcmVParam *vparam);
void            ncm_vparam_clear                             (NcmVParam **vparam);
void            ncm_vparam_set_len                           (NcmVParam *vparam,
                                                             guint len);
guint           ncm_vparam_get_len                           (NcmVParam *vparam);
void            ncm_vparam_set_sparam                        (NcmVParam *vparam,
                                                             guint n,
                                                             NcmSParam *spn);
void            ncm_vparam_set_sparam_full                  (NcmVParam *vparam,
                                                             guint n,
                                                             gchar *name,
                                                             gchar *symbol,
                                                             gdouble lower_bound,
                                                             gdouble upper_bound,
                                                             gdouble scale,
                                                             gdouble abstol,
                                                             gdouble default_val,
                                                             NcmParamType ftype);
NcmSParam *     ncm_vparam_peek_sparam                      (const NcmVParam *vparam,
                                                             guint n);
NcmSParam *     ncm_vparam_get_sparam                       (NcmVParam *vparam,
                                                             guint n);
void            ncm_vparam_set_lower_bound                  (NcmVParam *vparam,
                                                             guint n,
                                                             const gdouble lb);
void            ncm_vparam_set_upper_bound                  (NcmVParam *vparam,
                                                             guint n,
                                                             const gdouble ub);
void            ncm_vparam_set_scale                         (NcmVParam *vparam,
                                                             guint n,
                                                             const gdouble scale);
void            ncm_vparam_set_absolute_tolerance            (NcmVParam *vparam,
                                                             guint n,
                                                             const gdouble abstol);
void            ncm_vparam_set_default_value                 (NcmVParam *vparam,
                                                             guint n,
                                                             const gdouble default_val);
void            ncm_vparam_set_fit_type                      (NcmVParam *vparam,
                                                             guint n,
                                                             const NcmParamType ftype);
gdouble         ncm_vparam_get_lower_bound                  (const NcmVParam *vparam,

```

---

		guint n);
gdouble	ncm_vparam_get_upper_bound	(const NcmVParam *vparam,
		guint n);
gdouble	ncm_vparam_get_scale	(const NcmVParam *vparam,
		guint n);
gdouble	ncm_vparam_get_absolute_tolerance	(const NcmVParam *vparam,
		guint n);
gdouble	ncm_vparam_get_default_value	(const NcmVParam *vparam,
		guint n);
NcmParamType	ncm_vparam_get_fit_type	(const NcmVParam *vparam,
		guint n);

## Object Hierarchy

```

GObject
+----NcmVParam

```

## Properties

"default-sparam"	NcmSParam*	: Read / Write / Construct Only
------------------	------------	---------------------------------

## Description

This object comprises the necessary properties to define a vector parameter. It is used by **NcmModel** to store the description of the vector model parameters.

## Details

### struct NcmVParamClass

```

struct NcmVParamClass {
};

```

### struct NcmVParam

```

struct NcmVParam;

```

### ncm\_vparam\_new ()

NcmVParam *	ncm_vparam_new	(guint len,
		NcmSParam *default_param);

This function allocates memory for a new **NcmVParam** object and sets its properties to the values from the input arguments. *len* provides the number of components.

**len** : vector length.

**default\_param** : a **NcmSParam**.

**Returns** : A new **NcmVParam**.

**ncm\_vparam\_full\_new ()**

```
NcmVParam *          ncm_vparam_full_new          (guint len,
                                                    gchar *name,
                                                    gchar *symbol,
                                                    gdouble lower_bound,
                                                    gdouble upper_bound,
                                                    gdouble scale,
                                                    gdouble abstol,
                                                    gdouble default_val,
                                                    NcmParamType ftype);
```

This function allocates memory for a new **NcmVParam** object and sets its properties to the values from the input arguments.

The *name* parameter is restricted to the interval [*lower\_bound*, *upper\_bound*]. *scale* is an initial step for the statistical algorithms. *abstol* is the absolute error tolerance of the parameter. *ftype* indicates if the parameter will be fitted or not.

**len** : vector length.

**name** : "name".

**symbol** : "symbol".

**lower\_bound** : value of "lower-bound".

**upper\_bound** : value of "upper-bound".

**scale** : value of "scale".

**abstol** : value of "absolute-tolerance".

**default\_val** : value of "default-value".

**ftype** : a **NcmParamType**.

**Returns** : A new **NcmVParam**.

**ncm\_vparam\_copy ()**

```
NcmVParam *          ncm_vparam_copy              (NcmVParam *vparam);
```

Duplicates the **NcmVParam** object setting the same values of the original properties.

**vparam** : a **NcmVParam**.

**Returns** : A new **NcmVParam**. [*transfer full*]

**ncm\_vparam\_free ()**

```
void                 ncm_vparam_free              (NcmVParam *vparam);
```

Atomically decrements the reference count of *vparam* by one. If the reference count drops to 0, all memory allocated by *vparam* is released.

**vparam** : a **NcmVParam**.



This function sets the properties of the  $n$ -th *vparam* component.

**vparam** : a **NcmVParam**.

**n** : vector index.

**name** : "name".

**symbol** : "symbol".

**lower\_bound** : value of "lower-bound".

**upper\_bound** : value of "upper-bound".

**scale** : value of "scale".

**abstol** : value of "absolute-tolerance".

**default\_val** : value of "default-value".

**fctype** : a **NcmParamType**.

#### ncm\_vparam\_peek\_sparam ()

```
NcmSParam *      ncm_vparam_peek_sparam      (const NcmVParam *vparam,
                                              guint n);
```

This function does not increment the reference count of **NcmSParam**.

**vparam** : a **NcmVParam**.

**n** : vector index.

**Returns** : A **NcmSParam**, which is the  $n$ -th component of *vparam*. *[transfer none]*

#### ncm\_vparam\_get\_sparam ()

```
NcmSParam *      ncm_vparam_get_sparam      (NcmVParam *vparam,
                                              guint n);
```

This function returns the  $n$ -th component of *vparam* increasing its reference count.

**vparam** : a **NcmVParam**.

**n** : vector index.

**Returns** : A **NcmSParam**. *[transfer full]*

#### ncm\_vparam\_set\_lower\_bound ()

```
void             ncm_vparam_set_lower_bound (NcmVParam *vparam,
                                              guint n,
                                              const gdouble lb);
```

Sets the value *lb* to the "lower-bound" property of the  $n$ -th component of *vparam*.

**vparam** : a **NcmVParam**.

**n** : vector index.

**lb** : value of "lower-bound".

**ncm\_vparam\_set\_upper\_bound ()**

```
void                ncm_vparam_set_upper_bound      (NcmVParam *vparam,  
                                                    guint n,  
                                                    const gdouble ub);
```

Sets the value *ub* to the "**upper-bound**" property of the *n*-th component of *vparam*.

**vparam** : a **NcmVParam**.

**n** : vector index.

**ub** : value of "**upper-bound**".

**ncm\_vparam\_set\_scale ()**

```
void                ncm_vparam_set_scale           (NcmVParam *vparam,  
                                                    guint n,  
                                                    const gdouble scale);
```

Sets the value *scale* to the "**scale**" property of the *n*-th component of *vparam*.

**vparam** : a **NcmVParam**.

**n** : vector index.

**scale** : value of "**scale**".

**ncm\_vparam\_set\_absolute\_tolerance ()**

```
void                ncm_vparam_set_absolute_tolerance (NcmVParam *vparam,  
                                                    guint n,  
                                                    const gdouble abstol);
```

Sets the value *abstol* to the "**absolute-tolerance**" property of the *n*-th component of *vparam*.

**vparam** : a **NcmVParam**.

**n** : vector index.

**abstol** : value of "**absolute-tolerance**".

**ncm\_vparam\_set\_default\_value ()**

```
void                ncm_vparam_set_default_value   (NcmVParam *vparam,  
                                                    guint n,  
                                                    const gdouble default_val);
```

Sets the value *default\_val* to the "**default-value**" property of the *n*-th component of *vparam*.

**vparam** : a **NcmVParam**.

**n** : vector index.

**default\_val** : value of "**default-value**".

---



**ncm\_vparam\_set\_fit\_type ()**

```
void                ncm_vparam_set_fit_type      (NcmVParam *vparam,
                                                  guint n,
                                                  const NcmParamType ftype);
```

Sets *ftype* to the "**fit-type**" property of the *n*-th component of *vparam*.

***vparam*** : a **NcmVParam**.

***n*** : vector index.

***ftype*** : a **NcmParamType**.

**ncm\_vparam\_get\_lower\_bound ()**

```
gdouble            ncm_vparam_get_lower_bound   (const NcmVParam *vparam,
                                                  guint n);
```

***vparam*** : a **NcmVParam**.

***n*** : vector index.

**Returns** : The value of "**lower-bound**" property of the *n*-th component of *vparam*.

**ncm\_vparam\_get\_upper\_bound ()**

```
gdouble            ncm_vparam_get_upper_bound   (const NcmVParam *vparam,
                                                  guint n);
```

***vparam*** : a **NcmVParam**.

***n*** : vector index.

**Returns** : The value of "**upper-bound**" property of the *n*-th component of *vparam*.

**ncm\_vparam\_get\_scale ()**

```
gdouble            ncm_vparam_get_scale         (const NcmVParam *vparam,
                                                  guint n);
```

***vparam*** : a **NcmVParam**.

***n*** : vector index.

**Returns** : The value of "**scale**" property of the *n*-th component of *vparam*.

**ncm\_vparam\_get\_absolute\_tolerance ()**

```
gdouble            ncm_vparam_get_absolute_tolerance (const NcmVParam *vparam,
                                                       guint n);
```

***vparam*** : a **NcmVParam**.

***n*** : vector index.

**Returns** : The value of "**absolute-tolerance**" property of the *n*-th component of *vparam*.

**ncm\_vparam\_get\_default\_value ()**

gdouble	ncm_vparam_get_default_value	(const NcmVParam *vparam, guint n);
---------	------------------------------	--

**vparam** : a **NcmVParam**.

**n** : vector index.

**Returns** : The value of "**default-value**" property of the *n*-th component of *vparam*.

**ncm\_vparam\_get\_fit\_type ()**

NcmParamType	ncm_vparam_get_fit_type	(const NcmVParam *vparam, guint n);
--------------	-------------------------	--

**vparam** : a **NcmVParam**.

**n** : vector index.

**Returns** : The value of "**fit-type**" property of the *n*-th component of *vparam*.

**Property Details****The "default-sparam" property**

"default-sparam"	NcmSParam*	: Read / Write / Construct Only
------------------	------------	---------------------------------

Default sparam for the vector components.

**2.6 Reparametrization Abstract Class**

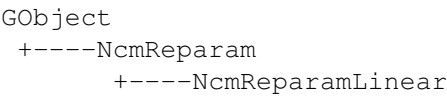
Reparametrization Abstract Class — Base class for model reparametrization

**Synopsis**

gboolean	(*NcmReparamV)	(NcmReparam *reparam, struct _NcmModel *model, NcmVector *src, NcmVector *dest);
gboolean	(*NcmReparamJ)	(NcmReparam *reparam, struct _NcmModel *model, NcmMatrix *jac);
struct	NcmReparamClass;	
struct	NcmReparam;	
NcmReparam *	ncm_reparam_copy	(NcmReparam *reparam);
NcmReparam *	ncm_reparam_ref	(NcmReparam *reparam);
void	ncm_reparam_copyto	(NcmReparam *reparam, NcmReparam *reparam_dest);
void	ncm_reparam_old2new	(NcmReparam *reparam, struct _NcmModel *model, NcmVector *src,

		NcmVector *dest);
void	ncm_reparam_new2old	(NcmReparam *reparam,
		struct _NcmModel *model,
		NcmVector *src,
		NcmVector *dest);
void	ncm_reparam_jac	(NcmReparam *reparam,
		struct _NcmModel *model,
		NcmMatrix *jac);
void	ncm_reparam_grad_old2new	(NcmReparam *reparam,
		struct _NcmModel *model,
		NcmMatrix *jac,
		NcmVector *old_grad,
		NcmVector *new_grad);
void	ncm_reparam_M_old2new	(NcmReparam *reparam,
		struct _NcmModel *model,
		NcmMatrix *jac,
		NcmMatrix *old_M,
		NcmMatrix *new_M);
void	ncm_reparam_free	(NcmReparam *reparam);
void	ncm_reparam_clear	(NcmReparam **reparam);

Object Hierarchy



Properties

"length"	guint	: Read / Write / Construct Only
----------	-------	---------------------------------

Description

FIXME

Details

NcmReparamV ()

gboolean	(*NcmReparamV)	(NcmReparam *reparam,
		struct _NcmModel *model,
		NcmVector *src,
		NcmVector *dest);

FIXME

**reparam** : FIXME

**model** : FIXME

**src** : FIXME

**dest** : FIXME

**NcmReparamJ ()**

```
gboolean          (*NcmReparamJ)          (NcmReparam *reparam,
                                           struct _NcmModel *model,
                                           NcmMatrix *jac);
```

FIXME

**reparam** : FIXME

**model** : FIXME

**jac** : FIXME

**struct NcmReparamClass**

```
struct NcmReparamClass {
};
```

**struct NcmReparam**

```
struct NcmReparam;
```

**ncm\_reparam\_copy ()**

```
NcmReparam *      ncm_reparam_copy      (NcmReparam *reparam);
```

FIXME

**reparam** : a **NcmReparam**

**Returns** : FIXME

**ncm\_reparam\_ref ()**

```
NcmReparam *      ncm_reparam_ref      (NcmReparam *reparam);
```

FIXME

**reparam** : a **NcmReparam**

**Returns** : FIXME. *[transfer full]*

**ncm\_reparam\_copyto ()**

```
void              ncm_reparam_copyto    (NcmReparam *reparam,
                                           NcmReparam *reparam_dest);
```

FIXME

**reparam** : a **NcmReparam**

**reparam\_dest** : a **NcmReparam**

**ncm\_reparam\_old2new ()**

```
void                ncm_reparam_old2new                (NcmReparam *reparam,  
                                                        struct _NcmModel *model,  
                                                        NcmVector *src,  
                                                        NcmVector *dest);
```

FIXME

***reparam*** : a **NcmReparam*****model*** : a **NcmModel*****src*** : a **NcmVector*****dest*** : a **NcmVector****ncm\_reparam\_new2old ()**

```
void                ncm_reparam_new2old                (NcmReparam *reparam,  
                                                        struct _NcmModel *model,  
                                                        NcmVector *src,  
                                                        NcmVector *dest);
```

FIXME

***reparam*** : a **NcmReparam*****model*** : a **NcmModel*****src*** : a **NcmVector*****dest*** : a **NcmVector****ncm\_reparam\_jac ()**

```
void                ncm_reparam_jac                    (NcmReparam *reparam,  
                                                        struct _NcmModel *model,  
                                                        NcmMatrix *jac);
```

FIXME

***reparam*** : a **NcmReparam*****model*** : a **NcmModel*****jac*** : a **NcmMatrix****ncm\_reparam\_grad\_old2new ()**

```
void                ncm_reparam_grad_old2new            (NcmReparam *reparam,  
                                                        struct _NcmModel *model,  
                                                        NcmMatrix *jac,  
                                                        NcmVector *old_grad,  
                                                        NcmVector *new_grad);
```

FIXME

*reparam* : a **NcmReparam**  
*model* : FIXME  
*jac* : a **NcmMatrix**  
*old\_grad* : a **NcmVector**  
*new\_grad* : a **NcmVector**

**ncm\_reparam\_M\_old2new ()**

```
void          ncm_reparam_M_old2new          (NcmReparam *reparam,
                                              struct _NcmModel *model,
                                              NcmMatrix *jac,
                                              NcmMatrix *old_M,
                                              NcmMatrix *new_M);
```

FIXME  
  
*reparam* : a **NcmReparam**  
*model* : FIXME  
*jac* : a **NcmMatrix**  
*old\_M* : a **NcmMatrix**  
*new\_M* : a **NcmMatrix**

**ncm\_reparam\_free ()**

```
void          ncm_reparam_free              (NcmReparam *reparam);
```

FIXME  
  
*reparam* : a **NcmReparam**

**ncm\_reparam\_clear ()**

```
void          ncm_reparam_clear             (NcmReparam **reparam);
```

FIXME  
  
*reparam* : a **NcmReparam**

**Property Details**

**The "length" property**

"length"	guint	: Read / Write / Construct Only
----------	-------	---------------------------------

System's length.  
Default value: 0

## 2.7 Linear Reparametrization

Linear Reparametrization — Linear reparametrization object

### Synopsis

```
struct          NcmReparamLinearClass;
struct          NcmReparamLinear;
NcmReparamLinear * ncm_reparam_linear_new      (guint size,
                                                NcmMatrix *T,
                                                NcmVector *v);
```

### Object Hierarchy

```
GObject
+----NcmReparam
      +----NcmReparamLinear
```

### Properties

"matrix"	NcmMatrix*	: Read / Write / Construct Only
"vector"	NcmVector*	: Read / Write / Construct Only

### Description

FIXME

### Details

#### struct NcmReparamLinearClass

```
struct NcmReparamLinearClass {
};
```

#### struct NcmReparamLinear

```
struct NcmReparamLinear;
```

#### ncm\_reparam\_linear\_new ()

```
NcmReparamLinear * ncm_reparam_linear_new      (guint size,
                                                NcmMatrix *T,
                                                NcmVector *v);
```

### Property Details

#### The "matrix" property

"matrix"	NcmMatrix*	: Read / Write / Construct Only
----------	------------	---------------------------------

Matrix of parameters mixing.





---

void	ncm_model_class_check_params_info	(NcmModelClass *model_class);
NcmModel *	ncm_model_copy	(NcmModel *model);
void	ncm_model_copyto	(NcmModel *model, NcmModel *model_dest);
void	ncm_model_free	(NcmModel *model);
void	ncm_model_clear	(NcmModel **model);
void	ncm_model_set_reparam	(NcmModel *model, NcmReparam *reparam);
gboolean	ncm_model_is_equal	(NcmModel *model1, NcmModel *model2);
NcmModel *	ncm_model_ref	(NcmModel *model);
NcmModelID	ncm_model_id	(NcmModel *model);
NcmModelID	ncm_model_id_by_type	(GType model_type);
guint64	ncm_model_impl	(NcmModel *model);
guint	ncm_model_len	(NcmModel *model);
guint	ncm_model_sparam_len	(NcmModel *model);
guint	ncm_model_vparam_array_len	(NcmModel *model);
guint	ncm_model_vparam_index	(NcmModel *model, guint n, guint i);
guint	ncm_model_vparam_len	(NcmModel *model, guint n);
const gchar *	ncm_model_name	(NcmModel *model);
const gchar *	ncm_model_nick	(NcmModel *model);
NcmReparam *	ncm_model_peek_reparam	(NcmModel *model);
gboolean	ncm_model_params_finite	(NcmModel *model);
gboolean	ncm_model_param_finite	(NcmModel *model, guint i);
void	ncm_model_params_update	(NcmModel *model);
void	ncm_model_orig_params_update	(NcmModel *model);
void	ncm_model_param_set	(NcmModel *model, guint n, gdouble val);
void	ncm_model_param_set_default	(NcmModel *model, guint n);
void	ncm_model_orig_param_set	(NcmModel *model, guint n, gdouble val);
void	ncm_model_orig_vparam_set	(NcmModel *model, guint n, guint i, gdouble val);
void	ncm_model_orig_vparam_set_vector	(NcmModel *model, guint n, NcmVector *val);
gdouble	ncm_model_param_get	(NcmModel *model, guint n);
gdouble	ncm_model_orig_param_get	(NcmModel *model, guint n);
gdouble	ncm_model_orig_vparam_get	(NcmModel *model, guint n, guint i);
NcmVector *	ncm_model_orig_vparam_get_vector	(NcmModel *model, guint n);
void	ncm_model_params_copyto	(NcmModel *model, NcmModel *model_dest);
void	ncm_model_params_set_default	(NcmModel *model);

---

---

```

void          ncm_model_params_save_as_default (NcmModel *model);
void          ncm_model_params_set_all        (NcmModel *model,
...);
void          ncm_model_params_set_all_data    (NcmModel *model,
gdouble *data);
void          ncm_model_params_set_vector      (NcmModel *model,
NcmVector *v);
void          ncm_model_params_set_model       (NcmModel *model,
NcmModel *model_src);
void          ncm_model_params_print_all       (NcmModel *model,
FILE *out);
void          ncm_model_params_log_all         (NcmModel *model);
NcmVector *   ncm_model_params_get_all        (NcmModel *model);
gboolean      ncm_model_params_valid         (NcmModel *model);
guint         ncm_model_param_index_from_name (NcmModel *model,
gchar *param_name);
const gchar * ncm_model_param_name           (NcmModel *model,
guint n);
const gchar * ncm_model_param_symbol         (NcmModel *model,
guint n);
gdouble       ncm_model_param_get_scale      (NcmModel *model,
guint n);
gdouble       ncm_model_param_get_lower_bound (NcmModel *model,
guint n);
gdouble       ncm_model_param_get_upper_bound (NcmModel *model,
guint n);
gdouble       ncm_model_param_get_abstol     (NcmModel *model,
guint n);
NcmParamType  ncm_model_param_get_ftype      (NcmModel *model,
guint n);
void          ncm_model_param_set_scale       (NcmModel *model,
guint n,
const gdouble scale);
void          ncm_model_param_set_lower_bound (NcmModel *model,
guint n,
const gdouble lb);
void          ncm_model_param_set_upper_bound (NcmModel *model,
guint n,
const gdouble ub);
void          ncm_model_param_set_abstol     (NcmModel *model,
guint n,
const gdouble abstol);
void          ncm_model_param_set_ftype      (NcmModel *model,
guint n,
const NcmParamType ptype);
void          ncm_model_reparam_df           (NcmModel *model,
NcmVector *fv,
NcmVector *v);
void          ncm_model_reparam_J            (NcmModel *model,
NcmMatrix *fJ,
NcmMatrix *J);
#define        NCM_MODEL_SET_IMPL_FUNC      (NS_NAME,
NsName,
ns_name,
type,
name)
#define        NCM_MODEL_FUNC0_IMPL        (NS_NAME,

```

---

```
#define NCM_MODEL_FUNC1_IMPL NsName, ns_name, name) (NS_NAME, NsName, ns_name, name)
```

Object Hierarchy

```
GObject
+----NcmModel
      +----NcClusterMass
      +----NcHICosmo
      +----NcSNIADistCov
```

Properties

"implementation"	guint64	: Read
"name"	gchar*	: Read
"nick"	gchar*	: Read
"params"	NcmVector*	: Read
"params-types"	GArray*	: Read
"reparam"	NcmReparam*	: Read / Write
"scalar-params-len"	guint	: Read
"vector-params-len"	guint	: Read

Description

FIXME

Details

NcmModelID

```
typedef gint NcmModelID;
```

struct NcmModelClass

```
struct NcmModelClass {
};
```

struct NcmModel

```
struct NcmModel;
```

Base class for models.

**NcmModelFunc0 ()**

```
gdouble (*NcmModelFunc0) (NcmModel *model);
```

**NcmModelFunc1 ()**

```
gdouble (*NcmModelFunc1) (NcmModel *model,  
const gdouble x);
```

**ncm\_model\_class\_set\_property ()**

```
void ncm_model_class_set_property (GObject *object,  
guint prop_id,  
const GValue *value,  
GParamSpec *pspec);
```

FIXME

**object** : a **NcmModelClass**.**prop\_id** : FIXME**value** : FIXME**pspec** : FIXME**ncm\_model\_class\_get\_property ()**

```
void ncm_model_class_get_property (GObject *object,  
guint prop_id,  
GValue *value,  
GParamSpec *pspec);
```

FIXME

**object** : a **NcmModelClass**.**prop\_id** : FIXME**value** : FIXME**pspec** : FIXME**ncm\_model\_class\_add\_params ()**

```
void ncm_model_class_add_params (NcmModelClass *model_class,  
guint sparam_len,  
guint vparam_len,  
guint nonparam_prop_len);
```

FIXME

**model\_class** : a **NcmModelClass**.**sparam\_len** : FIXME**vparam\_len** : FIXME**nonparam\_prop\_len** : FIXME

### ncm\_model\_class\_set\_name\_nick()

```
void ncm_model_class_set_name_nick(NcmModelClass *model_class,
                                   const gchar *name,
                                   const gchar *nick);
```

FIXME

**model\_class:** a **NcmModelClass**.

**name :** FIXME

*nick*: FIXME

```
ncm_model class set sparam ()
```

```
void ncm_model_class_set_sparam(NcmModelClass *model_class,
                                guint sparam_id,
                                gchar *symbol,
                                gchar *name,
                                gdouble lower_bound,
                                gdouble upper_bound,
                                gdouble scale,
                                gdouble abstol,
                                gdouble default_value,
                                NcmParamType ppt);
```

FIXME

```
model class: a NcmModelClass.
```

*sparam id*: FIXME

*symbol* : FIXME

```
name : FIXME
```

*lower bound*: FIXME

*upper bound*: FIXME

***scale***: FIXME

*abstol*: FIXME

*default value:* FIXME

*ppt* : FIXME

**ncm\_model\_class\_set\_vparam ()**

```
void ncm_model_class_set_vparam(NcmModelClass *model_class,
                                guint vparam_id,
                                guint default_length,
                                gchar *symbol,
                                gchar *name,
                                gdouble lower_bound,
                                gdouble upper_bound,
                                gdouble scale,
                                gdouble abstol,
                                gdouble default_value,
                                NcmParamType ppt);
```

FIXME

*model\_class*: a **NcmModelClass**

*vparam\_id*: FIXME

*default\_length*: FIXME

*symbol*: FIXME

*name*: FIXME

*lower\_bound*: FIXME

*upper\_bound*: FIXME

*scale*: FIXME

*abstol*: FIXME

*default\_value*: FIXME

*ppt*: FIXME

**ncm\_model\_class\_check\_params\_info ()**

```
void                ncm_model_class_check_params_info    (NcmModelClass *model_class);
```

FIXME

*model\_class*: a **NcmModelClass**.

**ncm\_model\_copy ()**

```
NcmModel *          ncm_model_copy                      (NcmModel *model);
```

FIXME

*model*: a **NcmModel**.

**Returns**: FIXME

**ncm\_model\_copyto ()**

```
void                ncm_model_copyto                    (NcmModel *model,
                                                         NcmModel *model_dest);
```

FIXME

*model*: a **NcmModel**.

*model\_dest*: a **NcmModel**.

**ncm\_model\_free ()**

```
void                ncm_model_free                      (NcmModel *model);
```

Atomically decrements the reference count of *model* by one. If the reference count drops to 0, all memory allocated by *model* is released.

*model*: a **NcmModel**.

**ncm\_model\_clear ()**

```
void                ncm_model_clear                (NcmModel **model);
```

Atomically decrements the reference count of *model* by one. If the reference count drops to 0, all memory allocated by *model* is released. Set pointer to NULL.

**model** : a **NcmModel**.

**ncm\_model\_set\_reparam ()**

```
void                ncm_model_set_reparam          (NcmModel *model,
                                                    NcmReparam *reparam);
```

FIXME

**model** : a **NcmModel**.

**reparam** : a **NcmReparam**.

**ncm\_model\_is\_equal ()**

```
gboolean           ncm_model_is_equal             (NcmModel *model1,
                                                    NcmModel *model2);
```

Compares if model1 and model2 are the same, with same dimension and reparametrization.

**model1** : a **NcmModel**.

**model2** : a **NcmModel**.

**ncm\_model\_ref ()**

```
NcmModel *         ncm_model_ref                 (NcmModel *model);
```

FIXME

**model** : a **NcmModel**

**Returns** : FIXME. *[transfer full]*

**ncm\_model\_id ()**

```
NcmModelID         ncm_model_id                 (NcmModel *model);
```

FIXME

**model** : a **NcmModel**

**Returns** : FIXME

**ncm\_model\_id\_by\_type ()**

```
NcmModelID         ncm_model_id_by_type         (GType model_type);
```

**ncm\_model\_impl ()**

```
guint64          ncm_model_impl          (NcmModel *model);
```

FIXME

*model* : a **NcmModel**

**Returns** : FIXME

**ncm\_model\_len ()**

```
guint          ncm_model_len          (NcmModel *model);
```

FIXME

*model* : a **NcmModel**

**Returns** : FIXME

**ncm\_model\_sparam\_len ()**

```
guint          ncm_model_sparam_len   (NcmModel *model);
```

FIXME

*model* : a **NcmModel**

**Returns** : FIXME

**ncm\_model\_vparam\_array\_len ()**

```
guint          ncm_model_vparam_array_len   (NcmModel *model);
```

FIXME

*model* : a **NcmModel**

**Returns** : FIXME

**ncm\_model\_vparam\_index ()**

```
guint          ncm_model_vparam_index   (NcmModel *model,
                                         guint n,
                                         guint i);
```

FIXME

*model* : a **NcmModel**

*n* : vector index

*i* : vector component index

**Returns** : index of the i-th component of the n-th vector



**ncm\_model\_vparam\_len ()**

```
guint          ncm_model_vparam_len      (NcmModel *model,
                                           guint n);
```

FIXME

**model** : a **NcmModel**

**n** : vector index

**Returns** : length of the n-th vector

**ncm\_model\_name ()**

```
const gchar *   ncm_model_name          (NcmModel *model);
```

FIXME

**model** : a **NcmModel**

**Returns** : FIXME. *[transfer none]*

**ncm\_model\_nick ()**

```
const gchar *   ncm_model_nick          (NcmModel *model);
```

FIXME

**model** : a **NcmModel**

**Returns** : FIXME. *[transfer none]*

**ncm\_model\_peek\_reparam ()**

```
NcmReparam *    ncm_model_peek_reparam  (NcmModel *model);
```

FIXME

**model** : a **NcmModel**

**Returns** : FIXME. *[transfer none]*

**ncm\_model\_params\_finite ()**

```
gboolean         ncm_model_params_finite (NcmModel *model);
```

FIXME

**model** : a **NcmModel**

**Returns** : FIXME

**ncm\_model\_param\_finite ()**

```
gboolean          ncm_model_param_finite          (NcmModel *model,
                                                    guint i);
```

FIXME

*model* : a **NcmModel**

*i* : FIXME

**Returns** : FIXME

**ncm\_model\_params\_update ()**

```
void              ncm_model_params_update         (NcmModel *model);
```

Force the parameters to the update its internal flags and update the original parameters if necessary.

*model* : a **NcmModel**

**ncm\_model\_orig\_params\_update ()**

```
void              ncm_model_orig_params_update    (NcmModel *model);
```

Update the new parameters. It causes an error to call this function with a model without reparametrization.

*model* : a **NcmModel**

**ncm\_model\_param\_set ()**

```
void              ncm_model_param_set             (NcmModel *model,
                                                    guint n,
                                                    gdouble val);
```

FIXME

*model* : a **NcmModel**

*n* : FIXME

*val* : FIXME

**ncm\_model\_param\_set\_default ()**

```
void              ncm_model_param_set_default     (NcmModel *model,
                                                    guint n);
```

FIXME

*model* : a **NcmModel**

*n* : FIXME

**ncm\_model\_orig\_param\_set ()**

```
void                ncm_model_orig_param_set      (NcmModel *model,
                                                    guint n,
                                                    gdouble val);
```

FIXME

**model** : a **NcmModel****n** : FIXME**val** : FIXME**ncm\_model\_orig\_vparam\_set ()**

```
void                ncm_model_orig_vparam_set    (NcmModel *model,
                                                    guint n,
                                                    guint i,
                                                    gdouble val);
```

**ncm\_model\_orig\_vparam\_set\_vector ()**

```
void                ncm_model_orig_vparam_set_vector (NcmModel *model,
                                                       guint n,
                                                       NcmVector *val);
```

FIXME

**model** : a **NcmModel****n** : FIXME**val** : FIXME**ncm\_model\_param\_get ()**

```
gdouble            ncm_model_param_get          (NcmModel *model,
                                                    guint n);
```

FIXME

**model** : a **NcmModel****n** : FIXME**Returns** : FIXME**ncm\_model\_orig\_param\_get ()**

```
gdouble            ncm_model_orig_param_get     (NcmModel *model,
                                                    guint n);
```

FIXME

**model** : a **NcmModel****n** : FIXME**Returns** : FIXME

**ncm\_model\_orig\_vparam\_get ()**

```
gdouble          ncm_model_orig_vparam_get      (NcmModel *model,
                                                  guint n,
                                                  guint i);
```

**ncm\_model\_orig\_vparam\_get\_vector ()**

```
NcmVector *      ncm_model_orig_vparam_get_vector (NcmModel *model,
                                                  guint n);
```

FIXME

**model** : a **NcmModel**

**n** : FIXME

**Returns** : FIXME. *[transfer full]*

**ncm\_model\_params\_copyto ()**

```
void            ncm_model_params_copyto        (NcmModel *model,
                                                  NcmModel *model_dest);
```

FIXME

**model** : a **NcmModel**.

**model\_dest** : a **NcmModel**.

**ncm\_model\_params\_set\_default ()**

```
void            ncm_model_params_set_default   (NcmModel *model);
```

FIXME

**model** : a **NcmModel**.

**ncm\_model\_params\_save\_as\_default ()**

```
void            ncm_model_params_save_as_default (NcmModel *model);
```

FIXME

**model** : a **NcmModel**.

**ncm\_model\_params\_set\_all ()**

```
void            ncm_model_params_set_all       (NcmModel *model,
                                                  ...);
```

FIXME

**model** : a **NcmModel**.

**...** : FIXME

**ncm\_model\_params\_set\_all\_data ()**

```
void                ncm_model_params_set_all_data      (NcmModel *model,  
                                                       gdouble *data);
```

FIXME

**model** : a **NcmModel**.**data** : FIXME**ncm\_model\_params\_set\_vector ()**

```
void                ncm_model_params_set_vector      (NcmModel *model,  
                                                       NcmVector *v);
```

FIXME

**model** : a **NcmModel**.**v** : a **NcmVector**.**ncm\_model\_params\_set\_model ()**

```
void                ncm_model_params_set_model      (NcmModel *model,  
                                                       NcmModel *model_src);
```

FIXME

**model** : a **NcmModel**.**model\_src** : a **NcmModel**.**ncm\_model\_params\_print\_all ()**

```
void                ncm_model_params_print_all      (NcmModel *model,  
                                                       FILE *out);
```

FIXME

**model** : a **NcmModel****out** : FIXME**ncm\_model\_params\_log\_all ()**

```
void                ncm_model_params_log_all        (NcmModel *model);
```

FIXME

**model** : a **NcmModel**

**ncm\_model\_params\_get\_all ()**

```
NcmVector *      ncm_model_params_get_all      (NcmModel *model);
```

FIXME

**model** : a **NcmModel**

**Returns** : FIXME. *[transfer full]*

**ncm\_model\_params\_valid ()**

```
gboolean         ncm_model_params_valid      (NcmModel *model);
```

FIXME

**model** : a **NcmModel**

**Returns** : FIXME

**ncm\_model\_param\_index\_from\_name ()**

```
guint           ncm_model_param_index_from_name (NcmModel *model,  
                                                gchar *param_name);
```

**model** : FIXME

**param\_name** : FIXME

**Returns** : FIXME

**ncm\_model\_param\_name ()**

```
const gchar *   ncm_model_param_name      (NcmModel *model,  
                                           guint n);
```

**ncm\_model\_param\_symbol ()**

```
const gchar *   ncm_model_param_symbol    (NcmModel *model,  
                                           guint n);
```

**ncm\_model\_param\_get\_scale ()**

```
gdouble         ncm_model_param_get_scale  (NcmModel *model,  
                                           guint n);
```

FIXME

**model** : a **NcmModel**

**n** : FIXME

**Returns** : FIXME

---

**ncm\_model\_param\_get\_lower\_bound ()**

gdouble	ncm_model_param_get_lower_bound	(NcmModel *model, guint n);
---------	---------------------------------	--------------------------------

FIXME

**model** : a **NcmModel****n** : FIXME**Returns** : FIXME**ncm\_model\_param\_get\_upper\_bound ()**

gdouble	ncm_model_param_get_upper_bound	(NcmModel *model, guint n);
---------	---------------------------------	--------------------------------

FIXME

**model** : a **NcmModel****n** : FIXME**Returns** : FIXME**ncm\_model\_param\_get\_abstol ()**

gdouble	ncm_model_param_get_abstol	(NcmModel *model, guint n);
---------	----------------------------	--------------------------------

FIXME

**model** : a **NcmModel****n** : FIXME**Returns** : FIXME**ncm\_model\_param\_get\_fctype ()**

NcmParamType	ncm_model_param_get_fctype	(NcmModel *model, guint n);
--------------	----------------------------	--------------------------------

FIXME

**model** : a **NcmModel****n** : FIXME**Returns** : FIXME

**ncm\_model\_param\_set\_scale ()**

```
void                ncm_model_param_set_scale      (NcmModel *model,  
                                                    guint n,  
                                                    const gdouble scale);
```

FIXME

**model** : a **NcmModel****n** : FIXME**scale** : FIXME**ncm\_model\_param\_set\_lower\_bound ()**

```
void                ncm_model_param_set_lower_bound (NcmModel *model,  
                                                    guint n,  
                                                    const gdouble lb);
```

FIXME

**model** : a **NcmModel****n** : FIXME**lb** : FIXME**ncm\_model\_param\_set\_upper\_bound ()**

```
void                ncm_model_param_set_upper_bound (NcmModel *model,  
                                                    guint n,  
                                                    const gdouble ub);
```

FIXME

**model** : a **NcmModel****n** : FIXME**ub** : FIXME**ncm\_model\_param\_set\_abstol ()**

```
void                ncm_model_param_set_abstol    (NcmModel *model,  
                                                    guint n,  
                                                    const gdouble abstol);
```

FIXME

**model** : a **NcmModel****n** : FIXME**abstol** : FIXME



**ncm\_model\_param\_set\_ftype ()**

```
void                ncm_model_param_set_ftype      (NcmModel *model,
                                                    guint n,
                                                    const NcmParamType ptype);
```

FIXME

*model* : a **NcmModel**

*n* : FIXME

*ptype* : FIXME

**ncm\_model\_reparam\_df ()**

```
void                ncm_model_reparam_df          (NcmModel *model,
                                                    NcmVector *fv,
                                                    NcmVector *v);
```

FIXME

*model* : a **NcmModel**.

*fv* : a **NcmVector**.

*v* : a **NcmVector**.

**ncm\_model\_reparam\_J ()**

```
void                ncm_model_reparam_J          (NcmModel *model,
                                                    NcmMatrix *fJ,
                                                    NcmMatrix *J);
```

FIXME

*model* : a **NcmModel**.

*fJ* : a **NcmMatrix**.

*J* : a **NcmMatrix**.

**NCM\_MODEL\_SET\_IMPL\_FUNC()**

```
#define                NCM_MODEL_SET_IMPL_FUNC (NS_NAME, NsName, ns_name, type, name)
```

**NCM\_MODEL\_FUNC0\_IMPL()**

```
#define                NCM_MODEL_FUNC0_IMPL (NS_NAME, NsName, ns_name, name)
```

**NCM\_MODEL\_FUNC1\_IMPL()**

```
#define                NCM_MODEL_FUNC1_IMPL (NS_NAME, NsName, ns_name, name)
```

## Property Details

### The "implementation" property

"implementation"	guint64	: Read
------------------	---------	--------

Bitwise specification of functions implementation.

Default value: 0

### The "name" property

"name"	gchar*	: Read
--------	--------	--------

Model's name.

Default value: NULL

### The "nick" property

"nick"	gchar*	: Read
--------	--------	--------

Model's nick.

Default value: NULL

### The "params" property

"params"	NcmVector*	: Read
----------	------------	--------

Parameters vector.

### The "params-types" property

"params-types"	GArray*	: Read
----------------	---------	--------

Parameters' types.

### The "reparam" property

"reparam"	NcmReparam*	: Read / Write
-----------	-------------	----------------

Model reparametrization.

### The "scalar-params-len" property

"scalar-params-len"	guint	: Read
---------------------	-------	--------

Number of scalar parameters.

Default value: 0

---

The "vector-params-len" property

"vector-params-len"	guint	: Read
---------------------	-------	--------

Number of vector parameters.  
Default value: 0

2.9 Model Update Control Object

Model Update Control Object — Control object for testing updates on model status

Synopsis

```
struct          NcmModelCtrlClass;
struct          NcmModelCtrl;
NcmModelCtrl *  ncm_model_ctrl_new          (NcmModel *model);
NcmModelCtrl *  ncm_model_ctrl_copy        (NcmModelCtrl *ctrl);
gboolean        ncm_model_ctrl_set_model    (NcmModelCtrl *ctrl,
                                             NcmModel *model);

NcmModel *      ncm_model_ctrl_get_model    (NcmModelCtrl *ctrl);
void            ncm_model_ctrl_force_update (NcmModelCtrl *ctrl);
void            ncm_model_ctrl_free         (NcmModelCtrl *ctrl);
void            ncm_model_ctrl_clear        (NcmModelCtrl **ctrl);
gboolean        ncm_model_ctrl_update       (NcmModelCtrl *ctrl,
                                             NcmModel *model);
gboolean        ncm_model_ctrl_model_update (NcmModelCtrl *ctrl,
                                             NcmModel *model);
```

Object Hierarchy



Properties

"model"	NcmModel*	: Read / Write
---------	-----------	----------------

Description

FIXME

Details

struct NcmModelCtrlClass

```
struct NcmModelCtrlClass {
};
```

**struct NcmModelCtrl**

```
struct NcmModelCtrl;
```

**ncm\_model\_ctrl\_new ()**

```
NcmModelCtrl *      ncm_model_ctrl_new          (NcmModel *model);
```

FIXME

**model** : FIXME. *[allow-none]*

**Returns** : FIXME

**ncm\_model\_ctrl\_copy ()**

```
NcmModelCtrl *      ncm_model_ctrl_copy        (NcmModelCtrl *ctrl);
```

FIXME

**ctrl** : FIXME

**Returns** : FIXME. *[transfer full]*

**ncm\_model\_ctrl\_set\_model ()**

```
gboolean            ncm_model_ctrl_set_model   (NcmModelCtrl *ctrl,  
                                                NcmModel *model);
```

FIXME

**ctrl** : FIXME

**model** : FIXME

**Returns** : FIXME

**ncm\_model\_ctrl\_get\_model ()**

```
NcmModel *          ncm_model_ctrl_get_model   (NcmModelCtrl *ctrl);
```

FIXME

**ctrl** : FIXME

**Returns** : FIXME. *[transfer full]*

**ncm\_model\_ctrl\_force\_update ()**

```
void                ncm_model_ctrl_force_update (NcmModelCtrl *ctrl);
```

FIXME

**ctrl** : FIXME

**Returns** : FIXME

---

**ncm\_model\_ctrl\_free ()**

```
void                ncm_model_ctrl_free                (NcmModelCtrl *ctrl);
```

**ncm\_model\_ctrl\_clear ()**

```
void                ncm_model_ctrl_clear                (NcmModelCtrl **ctrl);
```

**ncm\_model\_ctrl\_update ()**

```
gboolean            ncm_model_ctrl_update                (NcmModelCtrl *ctrl,  
                                                         NcmModel *model);
```

FIXME

*ctrl* : FIXME

*model* : FIXME

*Returns* : FIXME

**ncm\_model\_ctrl\_model\_update ()**

```
gboolean            ncm_model_ctrl_model_update                (NcmModelCtrl *ctrl,  
                                                         NcmModel *model);
```

FIXME

*ctrl* : FIXME

*model* : FIXME

*Returns* : FIXME

**Property Details**

**The "model" property**

"model"	NcmModel*	: Read / Write
---------	-----------	----------------

Last Model used.

**2.10 A Set of NcmModels**

A Set of NcmModels — Object representing a set of different NcmModel objects

## Synopsis

```

#define          NCM_MODEL_MAX_ID
struct          NcmMSetModelDesc;
struct          NcmMSetClass;
struct          NcmMSet;
struct          NcmMSetPIndex;
void            ncm_mset_model_register_id      (NcmModelClass *model_class,
                                                gchar *ns,
                                                gchar *desc,
                                                gchar *long_desc);

#define          NCM_MSET_MODEL_ID_FUNC
#define          NCM_MSET_MODEL_DECLARE_ID
#define          NCM_MSET_MODEL_REGISTER_ID

NcmMSetPIndex * ncm_mset_pindex_new            (NcmModelID mid,
                                                guint pid);
NcmMSetPIndex * ncm_mset_pindex_dup           (NcmMSetPIndex *pi);
void            ncm_mset_pindex_free          (NcmMSetPIndex *pi);
NcmMSet *       ncm_mset_empty_new            (void);
NcmMSet *       ncm_mset_new                  (NcmModel *model0,
                                                ...);
NcmMSet *       ncm_mset_newv                 (NcmModel *model0,
                                                va_list ap);
NcmMSet *       ncm_mset_new_array            (NcmModel **model);
NcmMSet *       ncm_mset_ref                  (NcmMSet *mset);
NcmMSet *       ncm_mset_dup                  (NcmMSet *mset);
NcmMSet *       ncm_mset_copy                 (NcmMSet *mset);
void            ncm_mset_copyto               (NcmMSet *mset_src,
                                                NcmMSet *mset_dest);
void            ncm_mset_free                  (NcmMSet *mset);
void            ncm_mset_clear                 (NcmMSet **mset);
NcmModel *      ncm_mset_peek                 (NcmMSet *mset,
                                                NcmModelID mid);
NcmModel *      ncm_mset_get                  (NcmMSet *mset,
                                                NcmModelID mid);
void            ncm_mset_remove                (NcmMSet *mset,
                                                NcmModelID mid);
void            ncm_mset_set                   (NcmMSet *mset,
                                                NcmModel *model);
gboolean        ncm_mset_exists                (NcmMSet *mset,
                                                NcmModel *model);
void            ncm_mset_prepare_fparam_map    (NcmMSet *mset);
guint           ncm_mset_total_len             (NcmMSet *mset);
guint           ncm_mset_fparam_len           (NcmMSet *mset);
guint           ncm_mset_max_param_name        (NcmMSet *mset);
guint           ncm_mset_max_fparam_name       (NcmMSet *mset);
guint           ncm_mset_max_model_nick        (NcmMSet *mset);
void            ncm_mset_pretty_log            (NcmMSet *mset);
void            ncm_mset_params_pretty_print   (NcmMSet *mset,
                                                FILE *out,
                                                gchar *header);
void            ncm_mset_params_log_vals       (NcmMSet *mset);
void            ncm_mset_params_print_vals     (NcmMSet *mset,
                                                FILE *out);
gboolean        ncm_mset_params_valid          (NcmMSet *mset);
gboolean        ncm_mset_cmp                   (NcmMSet *mset0,

```

		NcmMSet *mset1, gboolean cmp_model);
void	ncm_mset_param_set	(NcmMSet *mset, NcmModelID mid, guint pid, const gdouble x);
gdouble	ncm_mset_param_get	(NcmMSet *mset, NcmModelID mid, guint pid);
gdouble	ncm_mset_orig_param_get	(NcmMSet *mset, NcmModelID mid, guint pid);
guint	ncm_mset_param_len	(NcmMSet *mset);
const gchar *	ncm_mset_param_name	(NcmMSet *mset, NcmModelID mid, guint pid);
void	ncm_mset_param_set_ftype	(NcmMSet *mset, NcmModelID mid, guint pid, NcmParamType ftype);
void	ncm_mset_param_set_all_ftype	(NcmMSet *mset, NcmParamType ftype);
void	ncm_mset_param_set_vector	(NcmMSet *mset, NcmVector *params);
void	ncm_mset_param_get_vector	(NcmMSet *mset, NcmVector *params);
gdouble	ncm_mset_param_get_scale	(NcmMSet *mset, NcmModelID mid, guint pid);
gdouble	ncm_mset_param_get_lower_bound	(NcmMSet *mset, NcmModelID mid, guint pid);
gdouble	ncm_mset_param_get_upper_bound	(NcmMSet *mset, NcmModelID mid, guint pid);
gdouble	ncm_mset_param_get_abstol	(NcmMSet *mset, NcmModelID mid, guint pid);
NcmParamType	ncm_mset_param_get_ftype	(NcmMSet *mset, NcmModelID mid, guint pid);
void	ncm_mset_param_set_pi	(NcmMSet *mset, NcmMSetPIndex *pi, const gdouble *x, guint n);
void	ncm_mset_param_get_pi	(NcmMSet *mset, NcmMSetPIndex *pi, gdouble *x, guint n);
void	ncm_mset_fparams_get_vector	(NcmMSet *mset, NcmVector *x);
void	ncm_mset_fparams_set_vector	(NcmMSet *mset, const NcmVector *x);
void	ncm_mset_fparams_set_array	(NcmMSet *mset, const gdouble *x);
void	ncm_mset_fparams_set_gsl_vector	(NcmMSet *mset, const gsl_vector *x);

guint	ncm_mset_fparams_len	(NcmMSet *mset);
const gchar *	ncm_mset_fparam_name	(NcmMSet *mset,
		guint n);
gdouble	ncm_mset_fparam_get_scale	(NcmMSet *mset,
		guint n);
gdouble	ncm_mset_fparam_get_lower_bound	(NcmMSet *mset,
		guint n);
gdouble	ncm_mset_fparam_get_upper_bound	(NcmMSet *mset,
		guint n);
gdouble	ncm_mset_fparam_get_abstol	(NcmMSet *mset,
		guint n);
gdouble	ncm_mset_fparam_get	(NcmMSet *mset,
		guint n);
void	ncm_mset_fparam_set	(NcmMSet *mset,
		guint n,
		const gdouble x);
NcmMSetPIndex *	ncm_mset_fparam_get_pi	(NcmMSet *mset,
		guint n);
gint	ncm_mset_fparam_get_fpi	(NcmMSet *mset,
		NcmModelID mid,
		guint pid);
void	ncm_mset_save	(NcmMSet *mset,
		gchar *filename,
		gboolean save_comment);
NcmMSet *	ncm_mset_load	(gchar *filename);

## Object Hierarchy

```

GObject
+----NcmMSet

GBoxed
+----NcmMSetPIndex

```

## Description

FIXME

## Details

### NCM\_MODEL\_MAX\_ID

```
#define NCM_MODEL_MAX_ID 5
```

### struct NcmMSetModelDesc

```
struct NcmMSetModelDesc {
};
```

### struct NcmMSetClass

```
struct NcmMSetClass {
};
```



**struct NcmMSet**

```
struct NcmMSet;
```

**struct NcmMSetPIndex**

```
struct NcmMSetPIndex {
};
```

**ncm\_mset\_model\_register\_id ()**

```
void          ncm_mset_model_register_id      (NcmModelClass *model_class,
                                              gchar *ns,
                                              gchar *desc,
                                              gchar *long_desc);
```

FIXME

**model\_class** : FIXME

**ns** : Model namespace.

**desc** : Short description.

**long\_desc** : Long description.

**NCM\_MSET\_MODEL\_ID\_FUNC()**

```
#define NCM_MSET_MODEL_ID_FUNC(model_ns) model_ns##_id
```

FIXME

**model\_ns** : FIXME

**NCM\_MSET\_MODEL\_DECLARE\_ID()**

```
#define NCM_MSET_MODEL_DECLARE_ID(model_ns) gint32 NCM_MSET_MODEL_ID_FUNC(model_ns) (void) ↵
      G_GNUC_CONST
```

FIXME

**model\_ns** : FIXME

**NCM\_MSET\_MODEL\_REGISTER\_ID()**

```
#define          NCM_MSET_MODEL_REGISTER_ID(model_ns,typemacro)
```

FIXME

**model\_ns** : FIXME

**typemacro** : FIXME

**ncm\_mset\_pindex\_new ()**

```
NcmMSetPIndex *      ncm_mset_pindex_new      (NcmModelID mid,  
                                                guint pid);
```

FIXME

*mid*: FIXME

*pid*: FIXME

**Returns** : FIXME

**ncm\_mset\_pindex\_dup ()**

```
NcmMSetPIndex *      ncm_mset_pindex_dup      (NcmMSetPIndex *pi);
```

FIXME

*pi*: FIXME

**Returns** : FIXME

**ncm\_mset\_pindex\_free ()**

```
void                  ncm_mset_pindex_free      (NcmMSetPIndex *pi);
```

FIXME

*pi*: FIXME

**ncm\_mset\_empty\_new ()**

```
NcmMSet *             ncm_mset_empty_new      (void);
```

FIXME

**Returns** : FIXME

**ncm\_mset\_new ()**

```
NcmMSet *             ncm_mset_new            (NcmModel *model0,  
                                                ...);
```

FIXME

*model0*: FIXME

*...*: FIXME

**Returns** : FIXME

**ncm\_mset\_newv ()**

```
NcmMSet *          ncm_mset_newv          (NcmModel *model0,
                                           va_list ap);
```

FIXME

**model0** : FIXME

**ap** : FIXME

**Returns** : FIXME. *[transfer full]*

**ncm\_mset\_new\_array ()**

```
NcmMSet *          ncm_mset_new_array      (NcmModel **model);
```

**ncm\_mset\_ref ()**

```
NcmMSet *          ncm_mset_ref           (NcmMSet *mset);
```

FIXME

**mset** : a **NcmMSet**.

**Returns** : a new **NcmMSet**. *[transfer full]*

**ncm\_mset\_dup ()**

```
NcmMSet *          ncm_mset_dup           (NcmMSet *mset);
```

FIXME

**mset** : a **NcmMSet**.

**Returns** : a new **NcmMSet**. *[transfer full]*

**ncm\_mset\_copy ()**

```
NcmMSet *          ncm_mset_copy         (NcmMSet *mset);
```

FIXME

**mset** : a **NcmMSet**.

**Returns** : a new **NcmMSet**. *[transfer full]*

**ncm\_mset\_copyto ()**

```
void              ncm_mset_copyto        (NcmMSet *mset_src,
                                           NcmMSet *mset_dest);
```

FIXME

**mset\_src** : a **NcmMSet**.

**mset\_dest** : a **NcmMSet**.

**ncm\_mset\_free ()**

```
void                ncm_mset_free                (NcmMSet *mset);
```

FIXME

***mset*** : FIXME**ncm\_mset\_clear ()**

```
void                ncm_mset_clear               (NcmMSet **mset);
```

FIXME

***mset*** : FIXME**ncm\_mset\_peek ()**

```
NcmModel *         ncm_mset_peek               (NcmMSet *mset,  
                                                NcmModelID mid);
```

FIXME

***mset*** : a **NcmMSet**.***mid*** : a **NcmModelID**.***Returns*** : FIXME. [*transfer none*]**ncm\_mset\_get ()**

```
NcmModel *         ncm_mset_get               (NcmMSet *mset,  
                                                NcmModelID mid);
```

FIXME

***mset*** : a **NcmMSet**.***mid*** : a **NcmModelID**.***Returns*** : FIXME. [*transfer full*]**ncm\_mset\_remove ()**

```
void                ncm_mset_remove            (NcmMSet *mset,  
                                                NcmModelID mid);
```

FIXME

***mset*** : a **NcmMSet**.***mid*** : a **NcmModelID**.

**ncm\_mset\_set ()**

```
void                ncm_mset_set                (NcmMSet *mset,  
                                                NcmModel *model);
```

FIXME

*mset* : FIXME

*model* : FIXME

**ncm\_mset\_exists ()**

```
gboolean            ncm_mset_exists            (NcmMSet *mset,  
                                                NcmModel *model);
```

FIXME

*mset* : FIXME

*model* : FIXME

*Returns* : FIXME

**ncm\_mset\_prepare\_fparam\_map ()**

```
void                ncm_mset_prepare_fparam_map (NcmMSet *mset);
```

FIXME

*mset* : FIXME

**ncm\_mset\_total\_len ()**

```
guint               ncm_mset_total_len         (NcmMSet *mset);
```

FIXME

*mset* : FIXME

*Returns* : FIXME

**ncm\_mset\_fparam\_len ()**

```
guint               ncm_mset_fparam_len       (NcmMSet *mset);
```

FIXME

*mset* : FIXME

*Returns* : FIXME

---

**ncm\_mset\_max\_param\_name ()**

```
guint          ncm_mset_max_param_name          (NcmMSet *mset);
```

FIXME

***mset*** : FIXME

***Returns*** : FIXME

**ncm\_mset\_max\_fparam\_name ()**

```
guint          ncm_mset_max_fparam_name         (NcmMSet *mset);
```

FIXME

***mset*** : FIXME

***Returns*** : FIXME

**ncm\_mset\_max\_model\_nick ()**

```
guint          ncm_mset_max_model_nick          (NcmMSet *mset);
```

FIXME

***mset*** : FIXME

***Returns*** : FIXME

**ncm\_mset\_pretty\_log ()**

```
void           ncm_mset_pretty_log              (NcmMSet *mset);
```

FIXME

***mset*** : FIXME

**ncm\_mset\_params\_pretty\_print ()**

```
void           ncm_mset_params_pretty_print     (NcmMSet *mset,  
                                                FILE *out,  
                                                gchar *header);
```

This function print the command line (first line, commented), the model nick and parameters' names (second line, commented) and their values indicating if they are fixed or free.

***mset*** : a **NcmMSet**

***out*** : name of the file

***header*** : pointer to the command line

---

**ncm\_mset\_params\_log\_vals ()**

```
void                ncm_mset_params_log_vals        (NcmMSet *mset);
```

FIXME

***mset*** : FIXME

**ncm\_mset\_params\_print\_vals ()**

```
void                ncm_mset_params_print_vals      (NcmMSet *mset,
                                                    FILE *out);
```

FIXME

***mset*** : FIXME

***out*** : FIXME

**ncm\_mset\_params\_valid ()**

```
gboolean            ncm_mset_params_valid          (NcmMSet *mset);
```

FIXME

***mset*** : FIXME

***Returns*** : FIXME

**ncm\_mset\_cmp ()**

```
gboolean            ncm_mset_cmp                   (NcmMSet *mset0,
                                                    NcmMSet *mset1,
                                                    gboolean cmp_model);
```

Compares *mset0* and *mset1* and returns TRUE if both contains the same models. If *cmp\_model* is TRUE compare also if the models are the same.

***mset0*** : FIXME

***mset1*** : FIXME

***cmp\_model*** : FIXME

***Returns*** : FIXME

**ncm\_mset\_param\_set ()**

```
void                ncm_mset_param_set             (NcmMSet *mset,
                                                    NcmModelID mid,
                                                    guint pid,
                                                    const gdouble x);
```

FIXME

***mset*** : a **NcmMSet**

***mid*** : FIXME

***pid*** : FIXME

***x*** : FIXME

**ncm\_mset\_param\_get ()**

gdouble	ncm_mset_param_get	(NcmMSet *mset, NcmModelID mid, guint pid);
---------	--------------------	---

FIXME

***mset*** : a **NcmMSet*****mid*** : FIXME***pid*** : FIXME***Returns*** : FIXME**ncm\_mset\_orig\_param\_get ()**

gdouble	ncm_mset_orig_param_get	(NcmMSet *mset, NcmModelID mid, guint pid);
---------	-------------------------	---

FIXME

***mset*** : a **NcmMSet*****mid*** : FIXME***pid*** : FIXME***Returns*** : FIXME**ncm\_mset\_param\_len ()**

guint	ncm_mset_param_len	(NcmMSet *mset);
-------	--------------------	------------------

FIXME

***mset*** : FIXME***Returns*** : FIXME**ncm\_mset\_param\_name ()**

const gchar *	ncm_mset_param_name	(NcmMSet *mset, NcmModelID mid, guint pid);
---------------	---------------------	---

FIXME

***mset*** : a **NcmMSet*****mid*** : FIXME***pid*** : FIXME***Returns*** : FIXME



**ncm\_mset\_param\_set\_ftype ()**

```
void                ncm_mset_param_set_ftype      (NcmMSet *mset,  
                                                    NcmModelID mid,  
                                                    guint pid,  
                                                    NcmParamType ftype);
```

FIXME

***mset*** : a **NcmMSet**

***mid*** : FIXME

***pid*** : FIXME

***ftype*** : FIXME

**ncm\_mset\_param\_set\_all\_ftype ()**

```
void                ncm_mset_param_set_all_ftype (NcmMSet *mset,  
                                                    NcmParamType ftype);
```

FIXME

***mset*** : a **NcmMSet**

***ftype*** : FIXME

**ncm\_mset\_param\_set\_vector ()**

```
void                ncm_mset_param_set_vector    (NcmMSet *mset,  
                                                    NcmVector *params);
```

Set the models parameters using values from *params*.

***mset*** : a **NcmMSet**

***params*** : FIXME

**ncm\_mset\_param\_get\_vector ()**

```
void                ncm_mset_param_get_vector    (NcmMSet *mset,  
                                                    NcmVector *params);
```

Set the components of *params* using the models parameters.

***mset*** : a **NcmMSet**

***params*** : FIXME

---

**ncm\_mset\_param\_get\_scale ()**

gdouble	ncm_mset_param_get_scale	(NcmMSet *mset, NcmModelID mid, guint pid);
---------	--------------------------	---

FIXME

**mset** : a **NcmMSet****mid** : FIXME**pid** : FIXME**Returns** : FIXME**ncm\_mset\_param\_get\_lower\_bound ()**

gdouble	ncm_mset_param_get_lower_bound	(NcmMSet *mset, NcmModelID mid, guint pid);
---------	--------------------------------	---

FIXME

**mset** : a **NcmMSet****mid** : FIXME**pid** : FIXME**Returns** : FIXME**ncm\_mset\_param\_get\_upper\_bound ()**

gdouble	ncm_mset_param_get_upper_bound	(NcmMSet *mset, NcmModelID mid, guint pid);
---------	--------------------------------	---

FIXME

**mset** : a **NcmMSet****mid** : FIXME**pid** : FIXME**Returns** : FIXME**ncm\_mset\_param\_get\_abstol ()**

gdouble	ncm_mset_param_get_abstol	(NcmMSet *mset, NcmModelID mid, guint pid);
---------	---------------------------	---

FIXME

**mset** : a **NcmMSet****mid** : FIXME**pid** : FIXME**Returns** : FIXME

**ncm\_mset\_param\_get\_ftype ()**

```
NcmParamType      ncm_mset_param_get_ftype      (NcmMSet *mset,  
                                                  NcmModelID mid,  
                                                  guint pid);
```

FIXME

***mset*** : a **NcmMSet**.

***mid*** : a **NcmModelID**.

***pid*** : FIXME

**Returns** : FIXME

**ncm\_mset\_param\_set\_pi ()**

```
void              ncm_mset_param_set_pi        (NcmMSet *mset,  
                                                NcmMSetPIndex *pi,  
                                                const gdouble *x,  
                                                guint n);
```

FIXME

***mset*** : a **NcmMSet**

***pi*** : FIXME

***x*** : FIXME

***n*** : FIXME

**ncm\_mset\_param\_get\_pi ()**

```
void              ncm_mset_param_get_pi        (NcmMSet *mset,  
                                                NcmMSetPIndex *pi,  
                                                gdouble *x,  
                                                guint n);
```

FIXME

***mset*** : a **NcmMSet**

***pi*** : FIXME

***x*** : FIXME

***n*** : FIXME

**ncm\_mset\_fparams\_get\_vector ()**

```
void              ncm_mset_fparams_get_vector  (NcmMSet *mset,  
                                                NcmVector *x);
```

FIXME

***mset*** : a **NcmMSet**

***x*** : FIXME

---

**ncm\_mset\_fparams\_set\_vector ()**

```
void                ncm_mset_fparams_set_vector      (NcmMSet *mset,
                                                    const NcmVector *x);
```

FIXME

***mset*** : a **NcmMSet**

***x*** : FIXME

**ncm\_mset\_fparams\_set\_array ()**

```
void                ncm_mset_fparams_set_array      (NcmMSet *mset,
                                                    const gdouble *x);
```

FIXME

***mset*** : a **NcmMSet**

***x*** : FIXME

**ncm\_mset\_fparams\_set\_gsl\_vector ()**

```
void                ncm_mset_fparams_set_gsl_vector (NcmMSet *mset,
                                                    const gsl_vector *x);
```

FIXME

***mset*** : a **NcmMSet**.

***x*** : FIXME

**ncm\_mset\_fparams\_len ()**

```
guint              ncm_mset_fparams_len            (NcmMSet *mset);
```

FIXME

***mset*** : a **NcmMSet**

***Returns*** : FIXME

**ncm\_mset\_fparam\_name ()**

```
const gchar *      ncm_mset_fparam_name           (NcmMSet *mset,
                                                    guint n);
```

FIXME

***mset*** : a **NcmMSet**

***n*** : FIXME

***Returns*** : FIXME

**ncm\_mset\_fparam\_get\_scale ()**

gdouble	ncm_mset_fparam_get_scale	(NcmMSet *mset, guint n);
---------	---------------------------	------------------------------

FIXME

***mset*** : a **NcmMSet*****n*** : FIXME***Returns*** : FIXME**ncm\_mset\_fparam\_get\_lower\_bound ()**

gdouble	ncm_mset_fparam_get_lower_bound	(NcmMSet *mset, guint n);
---------	---------------------------------	------------------------------

FIXME

***mset*** : a **NcmMSet*****n*** : FIXME***Returns*** : FIXME**ncm\_mset\_fparam\_get\_upper\_bound ()**

gdouble	ncm_mset_fparam_get_upper_bound	(NcmMSet *mset, guint n);
---------	---------------------------------	------------------------------

FIXME

***mset*** : a **NcmMSet*****n*** : FIXME***Returns*** : FIXME**ncm\_mset\_fparam\_get\_abstol ()**

gdouble	ncm_mset_fparam_get_abstol	(NcmMSet *mset, guint n);
---------	----------------------------	------------------------------

FIXME

***mset*** : a **NcmMSet*****n*** : FIXME***Returns*** : FIXME

**ncm\_mset\_fparam\_get ()**

gdouble	ncm_mset_fparam_get	(NcmMSet *mset, guint n);
---------	---------------------	------------------------------

FIXME

**mset** : a **NcmMSet****n** : FIXME**Returns** : FIXME**ncm\_mset\_fparam\_set ()**

void	ncm_mset_fparam_set	(NcmMSet *mset, guint n, const gdouble x);
------	---------------------	--

FIXME

**mset** : a **NcmMSet****n** : FIXME**x** : FIXME**ncm\_mset\_fparam\_get\_pi ()**

NcmMSetPIndex *	ncm_mset_fparam_get_pi	(NcmMSet *mset, guint n);
-----------------	------------------------	------------------------------

FIXME

**mset** : a **NcmMSet****n** : FIXME**Returns** : FIXME**ncm\_mset\_fparam\_get\_fpi ()**

gint	ncm_mset_fparam_get_fpi	(NcmMSet *mset, NcmModelID mid, guint pid);
------	-------------------------	---

FIXME

**mset** : a **NcmMSet**.**mid** : a **NcmModelID**.**pid** : FIXME**Returns** : FIXME

**ncm\_mset\_save ()**

```
void                ncm_mset_save                (NcmMSet *mset,
                                                  gchar *filename,
                                                  gboolean save_comment);
```

FIXME

**mset** : FIXME

**filename** : FIXME

**save\_comment** : FIXME

**ncm\_mset\_load ()**

```
NcmMSet *          ncm_mset_load                (gchar *filename);
```

FIXME

**filename** : FIXME

**Returns** : FIXME. *[transfer full]*

## 2.11 A Function of NcmMSet

A Function of NcmMSet — Object representing a function on NcmMSet

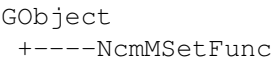
### Synopsis

```
void                (*NcmMSetFuncN)                (NcmMSet *mset,
                                                  gpointer obj,
                                                  const gdouble *x,
                                                  gdouble *f);

struct              NcmMSetFuncClass;
struct              NcmMSetFunc;
NcmMSetFunc *       ncm_mset_func_new                (NcmMSetFuncN func,
                                                  guint np,
                                                  guint dim,
                                                  gpointer obj,
                                                  GDestroyNotify free);
NcmMSetFunc *       ncm_mset_func_ref                (NcmMSetFunc *func);
void                ncm_mset_func_free                (NcmMSetFunc *func);
GPtrArray *         ncm_mset_func_array_new            (void);
gdouble             ncm_mset_func_eval                (NcmMSetFunc *func,
                                                  NcmMSet *mset,
                                                  const gdouble *x);
gdouble             ncm_mset_func_eval0                (NcmMSetFunc *func,
                                                  NcmMSet *mset);
gdouble             ncm_mset_func_eval1                (NcmMSetFunc *func,
                                                  NcmMSet *mset,
                                                  const gdouble x);
gboolean            ncm_mset_func_is_scalar            (NcmMSetFunc *func);
gboolean            ncm_mset_func_is_vector            (NcmMSetFunc *func);
```

```
gboolean          ncm_mset_func_is_const      guint dim);
gboolean          ncm_mset_func_has_params    (NcmMSetFunc *func);
NcmVector *       ncm_mset_func_numdiff_fparams (NcmMSetFunc *func,
                                                NcmMSet *mset,
                                                const gdouble *x,
                                                NcmVector *out);
```

Object Hierarchy



Description

FIXME

Details

NcmMSetFuncN ()

```
void              (*NcmMSetFuncN)             (NcmMSet *mset,
                                                gpointer obj,
                                                const gdouble *x,
                                                gdouble *f);
```

struct NcmMSetFuncClass

```
struct NcmMSetFuncClass {
};
```

struct NcmMSetFunc

```
struct NcmMSetFunc;
```

ncm\_mset\_func\_new ()

```
NcmMSetFunc *     ncm_mset_func_new            (NcmMSetFuncN func,
                                                guint np,
                                                guint dim,
                                                gpointer obj,
                                                GDestroyNotify free);
```

FIXME

*func* : FIXME

*np* : FIXME

*dim* : FIXME



**obj** : FIXME

**free** : FIXME

**Returns** : FIXME

**ncm\_mset\_func\_ref ()**

```
NcmMSetFunc *      ncm_mset_func_ref      (NcmMSetFunc *func);
```

FIXME

**func** : a **NcmMSetFunc**.

**Returns** : FIXME. *[transfer full]*

**ncm\_mset\_func\_free ()**

```
void      ncm_mset_func_free      (NcmMSetFunc *func);
```

FIXME

**func** : a **NcmMSetFunc**.

**ncm\_mset\_func\_array\_new ()**

```
GPtrArray *      ncm_mset_func_array_new      (void);
```

FIXME

**Returns** : FIXME. *[element-type NcmMSetFunc][transfer full]*

**ncm\_mset\_func\_eval ()**

```
gdouble      ncm_mset_func_eval      (NcmMSetFunc *func,  
                                       NcmMSet *mset,  
                                       const gdouble *x);
```

FIXME

**func** : FIXME

**mset** : FIXME

**x** : FIXME

**Returns** : FIXME

**ncm\_mset\_func\_eval0 ()**

```
gdouble      ncm_mset_func_eval0      (NcmMSetFunc *func,  
                                       NcmMSet *mset);
```

FIXME

**func** : FIXME

**mset** : FIXME

**Returns** : FIXME

**ncm\_mset\_func\_eval1 ()**

gdouble	ncm_mset_func_eval1	(NcmMSetFunc *func, NcmMSet *mset, const gdouble x);
---------	---------------------	--

FIXME

**func** : FIXME**mset** : FIXME**x** : FIXME**Returns** : FIXME**ncm\_mset\_func\_is\_scalar ()**

gboolean	ncm_mset_func_is_scalar	(NcmMSetFunc *func);
----------	-------------------------	----------------------

FIXME

**func** : FIXME**Returns** : FIXME**ncm\_mset\_func\_is\_vector ()**

gboolean	ncm_mset_func_is_vector	(NcmMSetFunc *func, guint dim);
----------	-------------------------	------------------------------------

FIXME

**func** : FIXME**dim** : FIXME**Returns** : FIXME**ncm\_mset\_func\_is\_const ()**

gboolean	ncm_mset_func_is_const	(NcmMSetFunc *func);
----------	------------------------	----------------------

FIXME

**func** : FIXME**Returns** : FIXME**ncm\_mset\_func\_has\_params ()**

gboolean	ncm_mset_func_has_params	(NcmMSetFunc *func, guint np);
----------	--------------------------	-----------------------------------

FIXME

**func** : FIXME**np** : FIXME**Returns** : FIXME

**ncm\_mset\_func\_numdiff\_fparams ()**

```
NcmVector *          ncm_mset_func_numdiff_fparams      (NcmMSetFunc *func,
                                                         NcmMSet *mset,
                                                         const gdouble *x,
                                                         NcmVector *out);
```

FIXME

**func** : FIXME

**mset** : FIXME

**x** : FIXME

**out** : FIXME. *[out][transfer full]*

**Returns** : FIXME. *[transfer full]*

## 2.12 Function Evaluator

Function Evaluator — A general purpose multi-threaded function evaluator

### Synopsis

```
void                (*NcmLoopFunc)                (glong i,
                                                    glong f,
                                                    gpointer data);

void                ncm_func_eval_set_max_threads  (gint mt);
void                ncm_func_eval_threaded_loop    (NcmLoopFunc lfunc,
                                                    glong i,
                                                    glong f,
                                                    gpointer data);
```

### Description

FIXME

### Details

#### NcmLoopFunc ()

```
void                (*NcmLoopFunc)                (glong i,
                                                    glong f,
                                                    gpointer data);
```

#### ncm\_func\_eval\_set\_max\_threads ()

```
void                ncm_func_eval_set_max_threads  (gint mt);
```

Set the new maximum number of threads to be used by the pool

**mt** : new max threads to be used in the pool, -1 means unlimited

**ncm\_func\_eval\_threaded\_loop()**

```
void                ncm_func_eval_threaded_loop      (NcmLoopFunc lfunc,
                                                    glong i,
                                                    glong f,
                                                    gpointer data);
```

Using the thread pool, evaluate  $f_l$  in each value of  $(f-i)/nthreads$

**lfunc** : **NcmLoopFunc** to be evaluated in threads. *[scope notified]*

**i** : initial index

**f** : final index

**data** : pointer to be passed to  $f_l$

## 2.13 Logarithm Fast Fourier Algorithm

Logarithm Fast Fourier Algorithm — Object implementing logarithm fast fourier transform

### Synopsis

```
struct              NcmFftlogClass;
struct              NcmFftlog;
void                ncm_fftlog_set_name              (NcmFftlog *fftlog,
                                                    const gchar *name);
gchar *             ncm_fftlog_peek_name             (NcmFftlog *fftlog);
void                ncm_fftlog_set_size             (NcmFftlog *fftlog,
                                                    guint n);
guint               ncm_fftlog_get_size             (NcmFftlog *fftlog);
void                ncm_fftlog_set_length           (NcmFftlog *fftlog,
                                                    gdouble L);
gdouble             ncm_fftlog_get_length           (NcmFftlog *fftlog);
```

### Object Hierarchy

```
GObject
+-----NcmFftlog
```

### Properties

"L"	gdouble	: Read / Write / Construct Only
"N"	guint	: Read / Write / Construct Only
"k0"	gdouble	: Read / Write / Construct Only
"name"	gchar*	: Read / Write / Construct Only
"r0"	gdouble	: Read / Write / Construct Only

### Description

FIXME

## Details

### struct NcmFftlogClass

```
struct NcmFftlogClass {  
};
```

### struct NcmFftlog

```
struct NcmFftlog;
```

FIXME

### ncm\_fftlog\_set\_name ()

```
void                ncm_fftlog_set_name          (NcmFftlog *fftlog,  
                                                  const gchar *name);
```

FIXME

***fftlog***: FIXME

***name***: FIXME

### ncm\_fftlog\_peek\_name ()

```
gchar *            ncm_fftlog_peek_name          (NcmFftlog *fftlog);
```

FIXME

***fftlog***: FIXME

***Returns***: FIXME. *[transfer none]*

### ncm\_fftlog\_set\_size ()

```
void                ncm_fftlog_set_size          (NcmFftlog *fftlog,  
                                                  guint n);
```

FIXME

***fftlog***: FIXME

***n***: FIXME

### ncm\_fftlog\_get\_size ()

```
guint              ncm_fftlog_get_size          (NcmFftlog *fftlog);
```

FIXME

***fftlog***: FIXME

***Returns***: FIXME

---

**ncm\_fftlog\_set\_length ()**

```
void                ncm_fftlog_set_length                (NcmFftlog *fftlog,
                                                         gdouble L);
```

FIXME

*fftlog* : FIXME

*L* : FIXME

**ncm\_fftlog\_get\_length ()**

```
gdouble            ncm_fftlog_get_length                (NcmFftlog *fftlog);
```

FIXME

*fftlog* : FIXME

*Returns* : FIXME

**Property Details**

**The "L" property**

"L"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

Function log-period.

Default value: 1

**The "N" property**

"N"	guint	: Read / Write / Construct Only
-----	-------	---------------------------------

Number of knots.

Default value: 10

**The "k0" property**

"k0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

Center value for k.

Default value: 0

**The "name" property**

"name"	gchar*	: Read / Write / Construct Only
--------	--------	---------------------------------

FFTW Plan wisdown name.

Default value: "fftlog\_default\_wisdown"

**The "r0" property**

"r0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

Center value for r.

Default value: 0

**2.14 Library Configuration**

Library Configuration — Library configuration and helper functions

**Synopsis**

void	ncm_cfg_init	(void);
void	ncm_cfg_enable_gsl_err_handler	(void);
void	ncm_cfg_register_obj	(GType obj);
gchar *	ncm_cfg_get_fullpath	(const gchar *filename, ...);
void	ncm_cfg_keyfile_to_arg	(GKeyFile *kfile, gchar *group_name, GOptionEntry *entries, gchar **argv, gint *argc);
void	ncm_cfg_entries_to_keyfile	(GKeyFile *kfile, gchar *group_name, GOptionEntry *entries);
gchar *	ncm_cfg_string_to_comment	(const gchar *str);
const GEnumValue *	ncm_cfg_get_enum_by_id_name_nick	(GType enum_type, const gchar *id_name_nick);
void	ncm_cfg_enum_print_all	(GType enum_type, gchar *header);
gboolean	ncm_cfg_load_fftw_wisdom	(gchar *filename, ...);
gboolean	ncm_cfg_save_fftw_wisdom	(gchar *filename, ...);
gboolean	ncm_cfg_exists	(gchar *filename, ...);
void	ncm_cfg_set_logfile	(gchar *filename);
void	ncm_cfg_logfile	(gboolean on);
void	ncm_cfg_logfile_flush	(gboolean on);
void	ncm_message	(gchar *msg, ...);
gchar *	ncm_string_ww	(const gchar *msg, const gchar *first, const gchar *rest, guint ncols);
void	ncm_message_ww	(const gchar *msg, const gchar *first, const gchar *rest, guint ncols);
void	ncm_cfg_msg_sepa	(void);
gsl_rng *	ncm_cfg_rng_get	(void);
gchar *	ncm_cfg_rng_get_state	(void);
void	ncm_cfg_rng_set_state	(gchar *state);

gboolean	ncm_cfg_is_named_instance	(gpointer obj);
gboolean	ncm_cfg_exist_named_instance	(gchar *name);
guint	ncm_cfg_count_named_instance	(void);
gpointer	ncm_cfg_get_named_instance	(gchar *name);
gchar *	ncm_cfg_named_instance_peek_name	(gpointer obj);
void	ncm_cfg_set_named_instance	(gpointer obj, gchar *name, gboolean overwrite);
void	ncm_cfg_free_named_instances	(void);
gboolean	ncm_cfg_object_is_named	(const gchar *serobj, gchar **name);
FILE *	ncm_cfg_fopen	(gchar *filename, gchar *mode, ...);
FILE *	ncm_cfg_vfopen	(gchar *filename, gchar *mode, va_list ap);
gboolean	ncm_cfg_load_spline	(gchar *filename, const gsl_interp_type *stype, NcmSpline **s, ...);
gboolean	ncm_cfg_save_spline	(gchar *filename, NcmSpline *s, ...);
gboolean	ncm_cfg_load_vector	(gchar *filename, gsl_vector *v, ...);
gboolean	ncm_cfg_save_vector	(gchar *filename, gsl_vector *v, ...);
gboolean	ncm_cfg_load_matrix	(gchar *filename, gsl_matrix *M, ...);
gboolean	ncm_cfg_save_matrix	(gchar *filename, gsl_matrix *M, ...);
#define	LOAD_SAVE_VECTOR_MATRIX_DEF	(typen)
sqlite3 *	ncm_cfg_get_default_sqlite3	(void);
gchar *	ncm_cfg_command_line	(gchar *argv[], gint argc);
void	ncm_cfg_object_set_property	(GObject *obj, const gchar *prop_str);
GObject *	ncm_cfg_create_from_name_params	(const gchar *obj_name, GVariant *params);
GObject *	ncm_cfg_create_from_string	(const gchar *obj_ser);
GVariant *	ncm_cfg_gvalue_to_gvariant	(GValue *val);
GVariant *	ncm_cfg_serialize_to_variant	(GObject *obj);
gchar *	ncm_cfg_serialize_to_string	(GObject *obj, gboolean valid_variant);
#define	NCM_RETURN_IF_INF	(a)
#define	NCM_FLOOR_TRUNC	(a, b)
#define	NCM_CEIL_TRUNC	(a, b)
#define	NCM_ROUND_TRUNC	(a, b)
#define	NCM_TEST_GSL_RESULT	(func,

---



```

ret)

#define NCM_ZERO_LIMIT
#define NCM_DEFAULT_PRECISION
#define NCM_THREAD_POOL_MAX
#define NCM_MAP_ALM_SIZE (lmax)
#define NCM_MAP_N_IND_PLM (lmax)
#define NCM_MAP_MAX_RING_SIZE (nside)
#define NCM_MAP_N_DIFFERENT_SIZED_RINGS (nside)
#define NCM_MAP_RING_PLAN_INDEX (nside,
ring_n)

#define NCM_MAP_RING_SIZE (nside,
ring_n)

#define NCM_MAP_N_RINGS (nside)
#define NCM_MAP_ALM_M_START (lmax,
m)

#define NCM_MAP_ALM_INDEX (lmax,
l,
m)

#define mpz_inits
#define mpz_clears
#define NCM_COMPLEX_INC_MUL_REAL_TEST (a,
b,
c)

#define NCM_COMPLEX_INC_MUL_REAL (a,
b,
c)

#define NCM_COMPLEX_INC_MUL (a,
b,
c)

#define NCM_COMPLEX_INC_MUL_MUL_REAL (a,
b,
c,
d)

#define NCM_COMPLEX_MUL_REAL (a,
b,
c)

#define NCM_COMPLEX_MUL (a,
b)

#define NCM_COMPLEX_ADD (a,
b)

#define NCM_COMPLEX_MUL_CONJUGATE (a,
b)

#define NCM_CFG_DEFAULT_SQLITE3_FILENAME
#define NCM_WRITE_INT32 (_ff,
_ii)

#define NCM_WRITE_UINT32 (_ff,
_ii)

#define NCM_WRITE_INT64 (_ff,
_ii)

#define NCM_WRITE_UINT64 (_ff,
_ii)

#define NCM_WRITE_DOUBLE (_ff,
_ii)

#define NCM_READ_INT32 (_ff,
_ii)

#define NCM_READ_UINT32 (_ff,
_ii)

```

---

```
#define          NCM_READ_INT64          (_ff,
                                         _ii)
#define          NCM_READ_UINT64        (_ff,
                                         _ii)
#define          NCM_READ_DOUBLE        (_ff,
                                         _ii)
#define          g_clear_object          (obj)
```

**Description**

FIXME

**Details**

**ncm\_cfg\_init ()**

```
void          ncm_cfg_init          (void);
```

FIXME

**ncm\_cfg\_enable\_gsl\_err\_handler ()**

```
void          ncm_cfg_enable_gsl_err_handler    (void);
```

FIXME

**ncm\_cfg\_register\_obj ()**

```
void          ncm_cfg_register_obj    (GType obj);
```

FIXME

*obj*: FIXME

**ncm\_cfg\_get\_fullpath ()**

```
gchar *          ncm_cfg_get_fullpath    (const gchar *filename,
                                         ...);
```

FIXME

***filename***: FIXME

***...***: FIXME

***Returns***: FIXME

**ncm\_cfg\_keyfile\_to\_arg ()**

```
void                ncm_cfg_keyfile_to_arg      (GKeyFile *kfile,  
                                                gchar *group_name,  
                                                GOptionEntry *entries,  
                                                gchar **argv,  
                                                gint *argc);
```

FIXME

**kfile**: FIXME

**group\_name**: FIXME

**entries**: FIXME

**argv**: FIXME

**argc**: FIXME

**Returns**: FIXME

**ncm\_cfg\_entries\_to\_keyfile ()**

```
void                ncm_cfg_entries_to_keyfile  (GKeyFile *kfile,  
                                                gchar *group_name,  
                                                GOptionEntry *entries);
```

FIXME

**kfile**: FIXME

**group\_name**: FIXME

**entries**: FIXME

**ncm\_cfg\_string\_to\_comment ()**

```
gchar *             ncm_cfg_string_to_comment  (const gchar *str);
```

FIXME

**str**: FIXME

**Returns**: FIXME. *[transfer full]*

**ncm\_cfg\_get\_enum\_by\_id\_name\_nick ()**

```
const GEnumValue *   ncm_cfg_get_enum_by_id_name_nick  (GType enum_type,  
                                                         const gchar *id_name_nick);
```

FIXME

**enum\_type**: FIXME

**id\_name\_nick**: FIXME

**Returns**: FIXME

---

**ncm\_cfg\_enum\_print\_all ()**

```
void                ncm_cfg_enum_print_all      (GType enum_type,  
                                                gchar *header);
```

FIXME

**enum\_type** : FIXME

**header** : FIXME

**ncm\_cfg\_load\_fftw\_wisdom ()**

```
gboolean            ncm_cfg_load_fftw_wisdom    (gchar *filename,  
                                                ...);
```

FIXME

**filename** : FIXME

**...** : FIXME

**Returns** : FIXME

**ncm\_cfg\_save\_fftw\_wisdom ()**

```
gboolean            ncm_cfg_save_fftw_wisdom    (gchar *filename,  
                                                ...);
```

FIXME

**filename** : FIXME

**...** : FIXME

**Returns** : FIXME

**ncm\_cfg\_exists ()**

```
gboolean            ncm_cfg_exists              (gchar *filename,  
                                                ...);
```

FIXME

**filename** : FIXME

**...** : FIXME

**Returns** : FIXME

**ncm\_cfg\_set\_logfile ()**

```
void                ncm_cfg_set_logfile         (gchar *filename);
```

FIXME

**filename** : FIXME

---

**ncm\_cfg\_logfile ()**

```
void          ncm_cfg_logfile          (gboolean on);
```

FIXME

**on** : FIXME

**ncm\_cfg\_logfile\_flush ()**

```
void          ncm_cfg_logfile_flush    (gboolean on);
```

FIXME

**on** : FIXME

**ncm\_message ()**

```
void          ncm_message              (gchar *msg,  
                                       ...);
```

**ncm\_string\_ww ()**

```
gchar *       ncm_string_ww           (const gchar *msg,  
                                       const gchar *first,  
                                       const gchar *rest,  
                                       guint ncols);
```

FIXME

**msg** : FIXME

**first** : FIXME

**rest** : FIXME

**ncols** : FIXME

**Returns** : word wrapped string *msg*. *[transfer full]*

**ncm\_message\_ww ()**

```
void          ncm_message_ww          (const gchar *msg,  
                                       const gchar *first,  
                                       const gchar *rest,  
                                       guint ncols);
```

FIXME

**msg** : FIXME

**first** : FIXME

**rest** : FIXME

**ncols** : FIXME

---

**ncm\_cfg\_msg\_sepa ()**

```
void                ncm_cfg_msg_sepa                (void);
```

Log a message separator.

**ncm\_cfg\_rng\_get ()**

```
gsl_rng *          ncm_cfg_rng_get                (void);
```

FIXME

**Returns :** FIXME

**ncm\_cfg\_rng\_get\_state ()**

```
gchar *           ncm_cfg_rng_get_state          (void);
```

FIXME

**Returns :** FIXME. *[transfer full]*

**ncm\_cfg\_rng\_set\_state ()**

```
void                ncm_cfg_rng_set_state        (gchar *state);
```

FIXME

**state :** FIXME

**ncm\_cfg\_is\_named\_instance ()**

```
gboolean           ncm_cfg_is_named_instance    (gpointer obj);
```

FIXME

**obj :** FIXME. *[type GObject]*

**Returns :** FIXME

**ncm\_cfg\_exist\_named\_instance ()**

```
gboolean           ncm_cfg_exist_named_instance (gchar *name);
```

FIXME

**name :** FIXME

**Returns :** FIXME

---

**ncm\_cfg\_count\_named\_instance ()**

```
guint          ncm_cfg_count_named_instance      (void);
```

**ncm\_cfg\_get\_named\_instance ()**

```
gpointer        ncm_cfg_get_named_instance      (gchar *name);
```

FIXME

**name** : FIXME

**Returns** : FIXME. *[transfer full][type GObject]*

**ncm\_cfg\_named\_instance\_peek\_name ()**

```
gchar *         ncm_cfg_named_instance_peek_name (gpointer obj);
```

FIXME

**obj** : FIXME. *[type GObject]*

**Returns** : FIXME. *[transfer none]*

**ncm\_cfg\_set\_named\_instance ()**

```
void            ncm_cfg_set_named_instance      (gpointer obj,  
                                                gchar *name,  
                                                gboolean overwrite);
```

FIXME

**obj** : FIXME. *[type GObject]*

**name** : FIXME

**overwrite** : FIXME

**ncm\_cfg\_free\_named\_instances ()**

```
void            ncm_cfg_free_named_instances    (void);
```

FIXME

**ncm\_cfg\_object\_is\_named ()**

```
gboolean        ncm_cfg_object_is_named        (const gchar *serobj,  
                                                gchar **name);
```

FIXME

**serobj** : FIXME

**name** : FIXME. *[out][transfer full]*

**Returns** : FIXME

---

**ncm\_cfg\_fopen ()**

```
FILE *          ncm_cfg_fopen          (gchar *filename,  
                                         gchar *mode,  
                                         ...);
```

FIXME

***filename*** : FIXME***mode*** : FIXME***...*** : FIXME***Returns*** : FIXME**ncm\_cfg\_vfopen ()**

```
FILE *          ncm_cfg_vfopen         (gchar *filename,  
                                         gchar *mode,  
                                         va_list ap);
```

FIXME

***filename*** : FIXME***mode*** : FIXME***ap*** : FIXME***Returns*** : FIXME**ncm\_cfg\_load\_spline ()**

```
gboolean        ncm_cfg_load_spline    (gchar *filename,  
                                         const gsl_interp_type *stype,  
                                         NcmSpline **s,  
                                         ...);
```

FIXME

***filename*** : FIXME***stype*** : FIXME***s*** : FIXME***...*** : FIXME***Returns*** : FIXME



**ncm\_cfg\_save\_spline ()**

gboolean	ncm_cfg_save_spline	(gchar *filename, NcmSpline *s, ...);
----------	---------------------	---

FIXME

***filename*** : FIXME***s*** : FIXME***...*** : FIXME***Returns*** : FIXME**ncm\_cfg\_load\_vector ()**

gboolean	ncm_cfg_load_vector	(gchar *filename, gsl_vector *v, ...);
----------	---------------------	--

FIXME

***filename*** : FIXME***v*** : FIXME***...*** : FIXME***Returns*** : FIXME**ncm\_cfg\_save\_vector ()**

gboolean	ncm_cfg_save_vector	(gchar *filename, gsl_vector *v, ...);
----------	---------------------	--

FIXME

***filename*** : FIXME***v*** : FIXME***...*** : FIXME***Returns*** : FIXME**ncm\_cfg\_load\_matrix ()**

gboolean	ncm_cfg_load_matrix	(gchar *filename, gsl_matrix *M, ...);
----------	---------------------	--

FIXME

***filename*** : FIXME***M*** : FIXME***...*** : FIXME***Returns*** : FIXME

**ncm\_cfg\_save\_matrix ()**

```
gboolean          ncm_cfg_save_matrix          (gchar *filename,
                                                gsl_matrix *M,
                                                ...);
```

FIXME

**filename** : FIXME

**M** : FIXME

**...** : FIXME

**Returns** : FIXME

**LOAD\_SAVE\_VECTOR\_MATRIX\_DEF()**

```
#define          LOAD_SAVE_VECTOR_MATRIX_DEF (typen)
```

**ncm\_cfg\_get\_default\_sqlite3 ()**

```
sqlite3 *          ncm_cfg_get_default_sqlite3          (void);
```

FIXME

**Returns** : FIXME

**ncm\_cfg\_command\_line ()**

```
gchar *          ncm_cfg_command_line          (gchar *argv[],
                                                gint argc);
```

FIXME

**argv** : FIXME

**argc** : FIXME

**Returns** : FIXME

**ncm\_cfg\_object\_set\_property ()**

```
void          ncm_cfg_object_set_property          (GObject *obj,
                                                const gchar *prop_str);
```

**ncm\_cfg\_create\_from\_name\_params ()**

```
GObject *          ncm_cfg_create_from_name_params          (const gchar *obj_name,
                                                            GVariant *params);
```

FIXME

**obj\_name** : string containing the object name.

**params** : a **GVariant** containing the object parameters.

**Returns** : A new **GObject**. *[transfer full]*

**ncm\_cfg\_create\_from\_string ()**

```
GObject *          ncm_cfg_create_from_string      (const gchar *obj_ser);
```

Parses the serialized and returns the newly created object.

**obj\_ser** : String containing the serialized version of the object.

**Returns** : A new **GObject**. *[transfer full]*

**ncm\_cfg\_gvalue\_to\_gvariant ()**

```
GVariant *          ncm_cfg_gvalue_to_gvariant    (GValue *val);
```

FIXME

**val** : a **GValue**.

**Returns** : A **GVariant** conversion of *val*. *[transfer full]*

**ncm\_cfg\_serialize\_to\_variant ()**

```
GVariant *          ncm_cfg_serialize_to_variant  (GObject *obj);
```

FIXME

**obj** : a **GObject**.

**Returns** : A **GVariant** dictionary describing the *obj*. *[transfer full]*

**ncm\_cfg\_serialize\_to\_string ()**

```
gchar *          ncm_cfg_serialize_to_string      (GObject *obj,  
                                                  gboolean valid_variant);
```

FIXME

**obj** : a **GObject**.

**valid\_variant** : FIXME.

**Returns** : A string containing the serialized version of *obj*. *[transfer full]*

**NCM\_RETURN\_IF\_INF()**

```
#define NCM_RETURN_IF_INF(a) if (gsl_isinf(a)) return a
```

**NCM\_FLOOR\_TRUNC()**

```
#define NCM_FLOOR_TRUNC(a,b) (floor ((b) * (a)) / (b))
```

**NCM\_CEIL\_TRUNC()**

```
#define NCM_CEIL_TRUNC(a,b) (ceil ((b) * (a)) / (b))
```

**NCM\_ROUND\_TRUNC()**

```
#define NCM_ROUND_TRUNC(a,b) (round ((b) * (a)) / (b))
```

**NCM\_TEST\_GSL\_RESULT()**

```
#define NCM_TEST_GSL_RESULT(func,ret) if (ret != GSL_SUCCESS) g_error ("%s: %s", func, ↵  
    gsl_strerror (ret))
```

**NCM\_ZERO\_LIMIT**

```
#define NCM_ZERO_LIMIT 1e-13
```

**NCM\_DEFAULT\_PRECISION**

```
#define NCM_DEFAULT_PRECISION 1e-7
```

**NCM\_THREAD\_POOL\_MAX**

```
#define NCM_THREAD_POOL_MAX 5
```

**NCM\_MAP\_ALM\_SIZE()**

```
#define NCM_MAP_ALM_SIZE(lmax) (((lmax)*(lmax) + 3*(lmax) + 2)/2)
```

**NCM\_MAP\_N\_IND\_PLM()**

```
#define NCM_MAP_N_IND_PLM(lmax) (((lmax)*(lmax) + 3*(lmax) + 2)/2)
```

**NCM\_MAP\_MAX\_RING\_SIZE()**

```
#define NCM_MAP_MAX_RING_SIZE(nside) (4*(nside))
```

**NCM\_MAP\_N\_DIFFERENT\_SIZED\_RINGS()**

```
#define NCM_MAP_N_DIFFERENT_SIZED_RINGS(nside) (nside)
```

**NCM\_MAP\_RING\_PLAN\_INDEX()**

```
#define NCM_MAP_RING_PLAN_INDEX(nside,ring_n) (((ring_n) < (nside)) ? (ring_n) : ((ring_n) ←  
    >=(3*(nside)) ? (4*(nside)-(ring_n)-2) : ((nside)-1)))
```

**NCM\_MAP\_RING\_SIZE()**

```
#define NCM_MAP_RING_SIZE(nside,ring_n) (4*(NCM_MAP_RING_PLAN_INDEX(nside,ring_n)+1))
```

**NCM\_MAP\_N\_RINGS()**

```
#define NCM_MAP_N_RINGS(nside) (4*(nside)-1)
```

**NCM\_MAP\_ALM\_M\_START()**

```
#define NCM_MAP_ALM_M_START(lmax,m) ((2*(lmax)*(m)-(m)*(m)+3*(m))/2)
```

**NCM\_MAP\_ALM\_INDEX()**

```
#define NCM_MAP_ALM_INDEX(lmax,l,m) (((l) >= (m)) ? (NCM_MAP_ALM_M_START(lmax,m) + (l) - (m ←  
    )) : (-1))
```

**mpz\_inits**

```
#define mpz_inits ncm_mpz_inits
```

**mpz\_clears**

```
#define mpz_clears ncm_mpz_clears
```

**NCM\_COMPLEX\_INC\_MUL\_REAL\_TEST()**

```
#define NCM_COMPLEX_INC_MUL_REAL_TEST(a,b,c)
```

**NCM\_COMPLEX\_INC\_MUL\_REAL()**

```
#define NCM_COMPLEX_INC_MUL_REAL(a,b,c)
```

**NCM\_COMPLEX\_INC\_MUL()**

```
#define NCM_COMPLEX_INC_MUL(a,b,c)
```

---

**NCM\_COMPLEX\_INC\_MUL\_MUL\_REAL()**

```
#define NCM_COMPLEX_INC_MUL_MUL_REAL(a,b,c,d)
```

**NCM\_COMPLEX\_MUL\_REAL()**

```
#define NCM_COMPLEX_MUL_REAL(a,b,c)
```

**NCM\_COMPLEX\_MUL()**

```
#define NCM_COMPLEX_MUL(a,b)
```

**NCM\_COMPLEX\_ADD()**

```
#define NCM_COMPLEX_ADD(a,b)
```

**NCM\_COMPLEX\_MUL\_CONJUGATE()**

```
#define NCM_COMPLEX_MUL_CONJUGATE(a,b)
```

**NCM\_CFG\_DEFAULT\_SQLITE3\_FILENAME**

```
#define NCM_CFG_DEFAULT_SQLITE3_FILENAME "data_observation.sqlite3"
```

**NCM\_WRITE\_INT32()**

```
#define NCM_WRITE_INT32(_ff,_ii) do { gint32 _temp_i = GINT32_TO_BE ((_ii)); if (fwrite (& ←  
_temp_i, sizeof(gint32), (1), _ff) != 1) g_error ("NCM_WRITE_INT32: io error"); } while (←  
(FALSE))
```

**NCM\_WRITE\_UINT32()**

```
#define NCM_WRITE_UINT32(_ff,_ii) do { guint32 _temp_i = GUINT32_TO_BE ((_ii)); if (fwrite ←  
(&_temp_i, sizeof(guint32), (1), _ff) != 1) g_error ("NCM_WRITE_UINT32: io error"); } ←  
while (FALSE)
```

**NCM\_WRITE\_INT64()**

```
#define NCM_WRITE_INT64(_ff,_ii) do { gint64 _temp_i = GINT64_TO_BE ((_ii)); if (fwrite (& ←  
_temp_i, sizeof(gint64), (1), _ff) != 1) g_error ("NCM_WRITE_INT64: io error"); } while (←  
(FALSE))
```

**NCM\_WRITE\_UINT64()**

```
#define NCM_WRITE_UINT64(_ff,_ii) do { guint64 _temp_i = GUINT64_TO_BE ((_ii)); if (fwrite (<
    (&_temp_i, sizeof(guint64), (1), _ff) != 1) g_error ("NCM_WRITE_INT64: io error"); } <
while (FALSE)
```

**NCM\_WRITE\_DOUBLE()**

```
#define NCM_WRITE_DOUBLE(_ff,_ii) do { NcmDoubleInt64 _iii; _iii.x = _ii; _iii.i = <
    GINT64_TO_BE ((_iii.i)); if (fwrite (&_iii.i, sizeof(gint64), (1), _ff) != 1) g_error (" <
    NCM_WRITE_DOUBLE: io error"); } while (FALSE)
```

**NCM\_READ\_INT32()**

```
#define NCM_READ_INT32(_ff,_ii) do { gint32 _temp_i; if (fread (&_temp_i, sizeof(gint32), <
    (1), _ff) != 1) g_error ("NCM_READ_INT32: io error"); _ii = GINT32_FROM_BE (_temp_i); } <
while (FALSE)
```

**NCM\_READ\_UINT32()**

```
#define NCM_READ_UINT32(_ff,_ii) do { guint32 _temp_i; if (fread (&_temp_i, sizeof(guint32) <
    , (1), _ff) != 1) g_error ("NCM_READ_UINT32: io error"); _ii = GUINT32_FROM_BE (_temp_i) <
    ; } while (FALSE)
```

**NCM\_READ\_INT64()**

```
#define NCM_READ_INT64(_ff,_ii) do { gint64 _temp_i; if (fread (&_temp_i, sizeof(gint64), <
    (1), _ff) != 1) g_error ("NCM_READ_INT64: io error"); _ii = GINT64_FROM_BE (_temp_i); } <
while (FALSE)
```

**NCM\_READ\_UINT64()**

```
#define NCM_READ_UINT64(_ff,_ii) do { guint64 _temp_i; if (fread (&_temp_i, sizeof(guint64) <
    , (1), _ff) != 1) g_error ("NCM_READ_UINT64: io error"); _ii = GUINT64_FROM_BE (_temp_i) <
    ; } while (FALSE)
```

**NCM\_READ\_DOUBLE()**

```
#define NCM_READ_DOUBLE(_ff,_ii) do { NcmDoubleInt64 _iii; if (fread (&_iii.i, sizeof( <
    gint64), (1), _ff) != 1) g_error ("NCM_READ_DOUBLE: io error"); _iii.i = GINT64_FROM_BE <
    (_iii.i); _ii = _iii.x; } while (FALSE)
```

**g\_clear\_object()**

```
#define g_clear_object(obj)
```

## 2.15 Numerical and Physical Constants

Numerical and Physical Constants — Numerical constants

### Synopsis

```

struct          NcmCClass;
struct          NcmC;
long double     ncm_c_sqrt_1_4pi                (void);
long double     ncm_c_sqrt_2pi                  (void);
long double     ncm_c_sqrt_3_4pi                (void);
long double     ncm_c_lnpi_4                    (void);
long double     ncm_c_ln2pi                     (void);
long double     ncm_c_pi                        (void);
long double     ncm_c_tan_1arcsec               (void);
gdouble         ncm_c_degree_to_radian          (const gdouble d);
gdouble         ncm_c_radian_to_degree          (const gdouble r);
gdouble         ncm_c_radian_0_2pi             (const gdouble r);
gdouble         ncm_c_sign_sin                  (const gdouble r);
gdouble         ncm_c_c                         (void);
gdouble         ncm_c_h                         (void);
gdouble         ncm_c_hbar                      (void);
gdouble         ncm_c_fine_struct              (void);
gdouble         ncm_c_kb                       (void);
gdouble         ncm_c_G                        (void);
gdouble         ncm_c_planck_length            (void);
gdouble         ncm_c_thomson_cs               (void);
gdouble         ncm_c_stefan_boltzmann         (void);
gdouble         ncm_c_mass_e                   (void);
gdouble         ncm_c_mass_p                   (void);
gdouble         ncm_c_mass_n                   (void);
gdouble         ncm_c_mass_ratio_alpha_p       (void);
gdouble         ncm_c_hc                       (void);
gdouble         ncm_c_fine_struct_square       (void);
gdouble         ncm_c_kpc                      (void);
gdouble         ncm_c_Mpc                      (void);
gdouble         ncm_c_AR                       (void);
gdouble         ncm_c_c2                       (void);
gdouble         ncm_c_planck_length2           (void);
gdouble         ncm_c_rest_energy_e            (void);
gdouble         ncm_c_rest_energy_p            (void);
gdouble         ncm_c_rest_energy_n            (void);
gdouble         ncm_c_decay_H_rate_2s_1s       (void);
gdouble         ncm_c_decay_He_rate_2s_1s     (void);
gdouble         ncm_c_HeI_bind_1s              (void);
gdouble         ncm_c_HeII_bind_1s             (void);
gdouble         ncm_c_HeI_Lyman_2s            (void);
gdouble         ncm_c_HeI_Lyman_2p            (void);
gdouble         ncm_c_HeI_Lyman_2s_wl         (void);
gdouble         ncm_c_HeI_Lyman_2p_wl         (void);
gdouble         ncm_c_HeI_bind_2s              (void);
gdouble         ncm_c_HeI_bind_2p              (void);
gdouble         ncm_c_HeI_2s_m_2p             (void);
gdouble         ncm_c_HeI_2s_m_2p_kb          (void);
gdouble         ncm_c_HeI_Lyman_2s_wl3_8pi    (void);

```

---



gdouble	ncm_c_HeI_Lyman_2p_wl3_8pi	(void);
gdouble	ncm_c_H_reduced_mass	(void);
gdouble	ncm_c_H_reduced_energy	(void);
gdouble	ncm_c_H_bind	(const gint n, const gint j);
gdouble	ncm_c_H_bind_1s	(void);
gdouble	ncm_c_H_bind_2s	(void);
gdouble	ncm_c_H_bind_2p	(void);
gdouble	ncm_c_H_Lyman_series	(const gint n, const gint j);
gdouble	ncm_c_H_Lyman_2s	(void);
gdouble	ncm_c_H_Lyman_2p	(void);
gdouble	ncm_c_H_Lyman_series_wl	(const gint n, const gint j);
gdouble	ncm_c_H_Lyman_2s_wl	(void);
gdouble	ncm_c_H_Lyman_2p_wl	(void);
gdouble	ncm_c_H_Lyman_2s_wl3_8pi	(void);
gdouble	ncm_c_H_Lyman_2p_wl3_8pi	(void);
gdouble	ncm_c_thermal_wl_e	(void);
gdouble	ncm_c_thermal_wl_p	(void);
gdouble	ncm_c_thermal_wl_n	(void);
gdouble	ncm_c_thermal_wn_e	(void);
gdouble	ncm_c_thermal_wn_p	(void);
gdouble	ncm_c_thermal_wn_n	(void);
gdouble	ncm_c_boltzmann_factor_H_1s	(const gdouble T);
gdouble	ncm_c_boltzmann_factor_H_2s	(const gdouble T);
gdouble	ncm_c_boltzmann_factor_H_2p	(const gdouble T);
gdouble	ncm_c_boltzmann_factor_HeI_1s	(const gdouble T);
gdouble	ncm_c_boltzmann_factor_HeI_2s	(const gdouble T);
gdouble	ncm_c_boltzmann_factor_HeI_2p	(const gdouble T);
gdouble	ncm_c_AU	(void);
gdouble	ncm_c_pc	(void);
gdouble	ncm_c_mass_solar	(void);
long double	ncm_c_stats_1sigma	(void);
long double	ncm_c_stats_2sigma	(void);
long double	ncm_c_stats_3sigma	(void);
gdouble	ncm_c_wmap3_cmb_z	(void);
gdouble	ncm_c_wmap3_cmb_R	(void);
gdouble	ncm_c_wmap3_cmb_sigma_R	(void);
gdouble	ncm_c_wmap5_cmb_z	(void);
gdouble	ncm_c_wmap5_cmb_R	(void);
gdouble	ncm_c_wmap5_cmb_sigma_R	(void);
gdouble	ncm_c_wmap7_cmb_z	(void);
gdouble	ncm_c_wmap7_cmb_R	(void);
gdouble	ncm_c_wmap7_cmb_sigma_R	(void);
gdouble	ncm_c_wmap5_coadded_I_K	(void);
gdouble	ncm_c_wmap5_coadded_I_Ka	(void);
gdouble	ncm_c_wmap5_coadded_I_Q	(void);
gdouble	ncm_c_wmap5_coadded_I_V	(void);
gdouble	ncm_c_wmap5_coadded_I_W	(void);
gdouble	ncm_c_bao_eisenstein_z	(void);
gdouble	ncm_c_bao_eisenstein_A	(void);
gdouble	ncm_c_bao_eisenstein_sigma_A	(void);
gdouble	ncm_c_bao_eisenstein_DV	(void);
gdouble	ncm_c_bao_eisenstein_sigma_DV	(void);
gdouble	ncm_c_bao_percival2007_DV_DV	(void);
gdouble	ncm_c_bao_percival2007_sigma_DV_DV	(void);

```

gdouble      ncm_c_bao_percival2010_DV_DV      (void);
gdouble      ncm_c_bao_percival2010_sigma_DV_DV (void);
gdouble      ncm_c_hubble_cte_wmap              (void);
gdouble      ncm_c_hubble_cte_hst              (void);
gdouble      ncm_c_hubble_cte_msa              (void);
gdouble      ncm_c_neutrino_n_eff              (void);
gdouble      ncm_c_prim_He_Yp                  (void);
gdouble      ncm_c_prim_H_Yp                   (void);
gdouble      ncm_c_prim_XHe                    (void);
gdouble      ncm_c_hubble_radius               (void);
gdouble      ncm_c_hubble_radius_planck        (void);
gdouble      ncm_c_crit_density                (void);
gdouble      ncm_c_crit_mass_density           (void);
gdouble      ncm_c_crit_mass_density_solar_Mpc (void);
gdouble      ncm_c_crit_number_density_p       (void);
gdouble      ncm_c_crit_number_density_n       (void);
gdouble      ncm_c_blackbody_energy_density    (void);
gdouble      ncm_c_radiation_temp_to_h2omega_r (const gdouble T);
gdouble      ncm_c_radiation_h2Omega_r_to_temp (const gdouble omr);

```

## Object Hierarchy

```

GObject
+-----NcmC

```

## Description

Mathematical and physical constants and constants manipulation functions. Using 2006 CODATA recommended values, see constants.txt.

## Details

### struct NcmCClass

```

struct NcmCClass {
};

```

### struct NcmC

```

struct NcmC;

```

### ncm\_c\_sqrt\_1\_4pi ()

```

long double      ncm_c_sqrt_1_4pi              (void);

```

**Returns :** sqrt (1 / (4 \* pi))

### ncm\_c\_sqrt\_2pi ()

```

long double      ncm_c_sqrt_2pi              (void);

```

**Returns :** sqrt (2 \* pi)

---

**ncm\_c\_sqrt\_3\_4pi ()**

```
long double      ncm_c_sqrt_3_4pi      (void);
```

**Returns :**  $\sqrt{3 / (4 * \pi)}$

**ncm\_c\_lnpi\_4 ()**

```
long double      ncm_c_lnpi_4          (void);
```

**Returns :**  $\ln(\pi) / 4$  1.8378770664093454835606594728112352797227949472755668256343

**ncm\_c\_ln2pi ()**

```
long double      ncm_c_ln2pi           (void);
```

**Returns :**  $\ln(2\pi)$

**ncm\_c\_pi ()**

```
long double      ncm_c_pi              (void);
```

**Returns :**  $\pi$

**ncm\_c\_tan\_1arcsec ()**

```
long double      ncm_c_tan_1arcsec     (void);
```

**Returns :**  $\tan(2 * \pi / (360 * 60 * 60))$

**ncm\_c\_degree\_to\_radian ()**

```
gdouble          ncm_c_degree_to_radian (const gdouble d);
```

**d :** angle in degrees.

**Returns :**  $d * \pi / 180$

**ncm\_c\_radian\_to\_degree ()**

```
gdouble          ncm_c_radian_to_degree (const gdouble r);
```

**r :** angle in radians

**Returns :**  $r * 180 / \pi$

---

**ncm\_c\_radian\_0\_2pi ()**

```
gdouble          ncm_c_radian_0_2pi          (const gdouble r);
```

***r*** : angle in radians

**Returns** : the angle in the interval [0, 2pi]

**ncm\_c\_sign\_sin ()**

```
gdouble          ncm_c_sign_sin              (const gdouble r);
```

***r*** : angle in radias

**Returns** : the sign of the value of sin(d).

**ncm\_c\_c ()**

```
gdouble          ncm_c_c                    (void);
```

**Returns** : Speed of light.

**ncm\_c\_h ()**

```
gdouble          ncm_c_h                    (void);
```

**Returns** : Planck constant.

**ncm\_c\_hbar ()**

```
gdouble          ncm_c_hbar                 (void);
```

**Returns** : Planck constant over 2 pi.

**ncm\_c\_fine\_struct ()**

```
gdouble          ncm_c_fine_struct          (void);
```

**Returns** : Fine structure constant.

**ncm\_c\_kb ()**

```
gdouble          ncm_c_kb                   (void);
```

**Returns** : Boltzmann constant.

---

**ncm\_c\_G ()**

gdouble	ncm_c_G	(void);
---------	---------	---------

**Returns :** Newton constant.

**ncm\_c\_planck\_length ()**

gdouble	ncm_c_planck_length	(void);
---------	---------------------	---------

**Returns :** Planck length.

**ncm\_c\_thomson\_cs ()**

gdouble	ncm_c_thomson_cs	(void);
---------	------------------	---------

**Returns :** Thomson cross section.

**ncm\_c\_stefan\_boltzmann ()**

gdouble	ncm_c_stefan_boltzmann	(void);
---------	------------------------	---------

**Returns :** Stefan Boltzmann constant.

**ncm\_c\_mass\_e ()**

gdouble	ncm_c_mass_e	(void);
---------	--------------	---------

**Returns :** Electron mass.

**ncm\_c\_mass\_p ()**

gdouble	ncm_c_mass_p	(void);
---------	--------------	---------

**Returns :** Proton mass.

**ncm\_c\_mass\_n ()**

gdouble	ncm_c_mass_n	(void);
---------	--------------	---------

**Returns :** Neutron mass.

**ncm\_c\_mass\_ratio\_alpha\_p ()**

gdouble	ncm_c_mass_ratio_alpha_p	(void);
---------	--------------------------	---------

**Returns :** The proton and alpha particle (helium-4) mass ratio.

---

**ncm\_c\_hc ()**

gdouble	ncm_c_hc	(void);
---------	----------	---------

**Returns :** Planck constant times the speed of light.

**ncm\_c\_fine\_struct\_square ()**

gdouble	ncm_c_fine_struct_square	(void);
---------	--------------------------	---------

**Returns :** The square of the fine struct constant.

**ncm\_c\_kpc ()**

gdouble	ncm_c_kpc	(void);
---------	-----------	---------

**Returns :** One kilo parsec.

**ncm\_c\_Mpc ()**

gdouble	ncm_c_Mpc	(void);
---------	-----------	---------

**Returns :** One mega parsec.

**ncm\_c\_AR ()**

gdouble	ncm_c_AR	(void);
---------	----------	---------

**Returns :** Radiation constant AR.

**ncm\_c\_c2 ()**

gdouble	ncm_c_c2	(void);
---------	----------	---------

**Returns :** Square of the speed of light.

**ncm\_c\_planck\_length2 ()**

gdouble	ncm_c_planck_length2	(void);
---------	----------------------	---------

**Returns :** Square of the Planck length.

**ncm\_c\_rest\_energy\_e ()**

gdouble	ncm_c_rest_energy_e	(void);
---------	---------------------	---------

**Returns :** Electron's rest energy.

---

**ncm\_c\_rest\_energy\_p ()**

```
gdouble          ncm_c_rest_energy_p          (void) ;
```

**Returns :** Proton's rest energy.

**ncm\_c\_rest\_energy\_n ()**

```
gdouble          ncm_c_rest_energy_n          (void) ;
```

**Returns :** Neutron's rest energy.

**ncm\_c\_decay\_H\_rate\_2s\_1s ()**

```
gdouble          ncm_c_decay_H_rate_2s_1s     (void) ;
```

FIXME: Cite source.

**Returns :** Decay rate of Hydrogen from 2s -> 1s.

**ncm\_c\_decay\_He\_rate\_2s\_1s ()**

```
gdouble          ncm_c_decay_He_rate_2s_1s    (void) ;
```

FIXME: Cite source.

**Returns :** Decay rate of Helium from 2s -> 1s.

**ncm\_c\_HeI\_bind\_1s ()**

```
gdouble          ncm_c_HeI_bind_1s            (void) ;
```

FIXME: Cite source.

**Returns :** HeI binding energy 1s.

**ncm\_c\_HeII\_bind\_1s ()**

```
gdouble          ncm_c_HeII_bind_1s           (void) ;
```

FIXME: Cite source.

**Returns :** HeII binding energy 1s.

**ncm\_c\_HeI\_Lyman\_2s ()**

```
gdouble          ncm_c_HeI_Lyman_2s          (void) ;
```

FIXME: Cite source.

**Returns :** HeI Lyman 2s energy.

---

**ncm\_c\_HeI\_Lyman\_2p ()**

```
gdouble          ncm_c_HeI_Lyman_2p          (void);
```

FIXME: Cite source.

**Returns :** HeI Lyman 2p energy.

**ncm\_c\_HeI\_Lyman\_2s\_wl ()**

```
gdouble          ncm_c_HeI_Lyman_2s_wl       (void);
```

FIXME: Cite source.

**Returns :** HeI Lyman 2s wave length.

**ncm\_c\_HeI\_Lyman\_2p\_wl ()**

```
gdouble          ncm_c_HeI_Lyman_2p_wl       (void);
```

FIXME: Cite source.

**Returns :** HeI Lyman 2p wave length.

**ncm\_c\_HeI\_bind\_2s ()**

```
gdouble          ncm_c_HeI_bind_2s           (void);
```

FIXME: Cite source.

**Returns :** HeI binding energy 2s.

**ncm\_c\_HeI\_bind\_2p ()**

```
gdouble          ncm_c_HeI_bind_2p           (void);
```

FIXME: Cite source.

**Returns :** HeI binding energy 2p.

**ncm\_c\_HeI\_2s\_m\_2p ()**

```
gdouble          ncm_c_HeI_2s_m_2p           (void);
```

FIXME: Cite source.

**Returns :** HeI energy difference between states 2s and 2p.

---



**ncm\_c\_HeI\_2s\_m\_2p\_kb ()**

```
gdouble          ncm_c_HeI_2s_m_2p_kb          (void);
```

FIXME: Cite source.

**Returns :** HeI energy difference between states 2s and 2p divided by Boltzmann constant.

**ncm\_c\_HeI\_Lyman\_2s\_wl3\_8pi ()**

```
gdouble          ncm_c_HeI_Lyman_2s_wl3_8pi    (void);
```

FIXME: Cite source.

**Returns :** Cubic power of HeI Lyman 2s wave length divided by (8 \* pi).

**ncm\_c\_HeI\_Lyman\_2p\_wl3\_8pi ()**

```
gdouble          ncm_c_HeI_Lyman_2p_wl3_8pi    (void);
```

FIXME: Cite source.

**Returns :** Cubic power of HeI Lyman 2p wave length divided by (8 \* pi).

**ncm\_c\_H\_reduced\_mass ()**

```
gdouble          ncm_c_H_reduced_mass          (void);
```

FIXME: Cite source.

**Returns :** Hydrogen reduced mass.

**ncm\_c\_H\_reduced\_energy ()**

```
gdouble          ncm_c_H_reduced_energy        (void);
```

FIXME: Cite source.

**Returns :** Hydrogen reduced energy.

**ncm\_c\_H\_bind ()**

```
gdouble          ncm_c_H_bind                  (const gint n,  
                                                const gint j);
```

**n :** FIXME

**j :** FIXME FIXME: Cite source.

**Returns :** Hydrogen binding energy.

---

**ncm\_c\_H\_bind\_1s ()**

```
gdouble          ncm_c_H_bind_1s          (void);
```

FIXME: Cite source.

**Returns :** Hydrogen 1s binding energy.

**ncm\_c\_H\_bind\_2s ()**

```
gdouble          ncm_c_H_bind_2s          (void);
```

FIXME: Cite source.

**Returns :** Hydrogen 2s binding energy.

**ncm\_c\_H\_bind\_2p ()**

```
gdouble          ncm_c_H_bind_2p          (void);
```

FIXME: Cite source.

**Returns :** Hydrogen 2p binding energy.

**ncm\_c\_H\_Lyman\_series ()**

```
gdouble          ncm_c_H_Lyman_series     (const gint n,  
                                           const gint j);
```

Energy difference between levels 1s and n,j. FIXME: Cite source.

**n :** FIXME

**j :** FIXME

**Returns :** Hydrogen Lyman series.

**ncm\_c\_H\_Lyman\_2s ()**

```
gdouble          ncm_c_H_Lyman_2s        (void);
```

FIXME: Cite source.

**Returns :** Energy difference between levels 1s and 2s.

**ncm\_c\_H\_Lyman\_2p ()**

```
gdouble          ncm_c_H_Lyman_2p        (void);
```

FIXME: Cite source.

**Returns :** Energy difference between levels 1s and 2p.

---

**ncm\_c\_H\_Lyman\_series\_wl ()**

```
gdouble          ncm_c_H_Lyman_series_wl          (const gint n,
                                                    const gint j);
```

FIXME: Cite source.

*n* : FIXME

*j* : FIXME

**Returns** : Wavelength relative to the energy difference between levels 1s and n,j.

**ncm\_c\_H\_Lyman\_2s\_wl ()**

```
gdouble          ncm_c_H_Lyman_2s_wl              (void);
```

FIXME: Cite source.

**Returns** : Wavelength relative to the energy difference between levels 1s and 2s.

**ncm\_c\_H\_Lyman\_2p\_wl ()**

```
gdouble          ncm_c_H_Lyman_2p_wl              (void);
```

FIXME: Cite source.

**Returns** : Wavelength relative to the energy difference between levels 1s and 2s.

**ncm\_c\_H\_Lyman\_2s\_wl3\_8pi ()**

```
gdouble          ncm_c_H_Lyman_2s_wl3_8pi         (void);
```

FIXME: Cite source.

**Returns** : Cubic power of the Wavelength relative to the energy difference between levels 1s and 2s divided by (8\*pi).

**ncm\_c\_H\_Lyman\_2p\_wl3\_8pi ()**

```
gdouble          ncm_c_H_Lyman_2p_wl3_8pi         (void);
```

FIXME: Cite source.

**Returns** : Cubic power of the wavelength relative to the energy difference between levels 1s and 2p divided by (8\*pi).

**ncm\_c\_thermal\_wl\_e ()**

```
gdouble          ncm_c_thermal_wl_e               (void);
```

FIXME: Cite source.

**Returns** : Thermal electron wavelength.

**ncm\_c\_thermal\_wl\_p ()**

```
gdouble          ncm_c_thermal_wl_p          (void);
```

FIXME: Cite source.

**Returns :** Thermal proton wavelenght.

**ncm\_c\_thermal\_wl\_n ()**

```
gdouble          ncm_c_thermal_wl_n          (void);
```

FIXME: Cite source.

**Returns :** Thermal neutron wavelenght.

**ncm\_c\_thermal\_wn\_e ()**

```
gdouble          ncm_c_thermal_wn_e          (void);
```

FIXME: Cite source.

**Returns :** Thermal eletron wavenumber.

**ncm\_c\_thermal\_wn\_p ()**

```
gdouble          ncm_c_thermal_wn_p          (void);
```

FIXME: Cite source.

**Returns :** Thermal proton wavenumber.

**ncm\_c\_thermal\_wn\_n ()**

```
gdouble          ncm_c_thermal_wn_n          (void);
```

FIXME: Cite source.

**Returns :** Thermal neutron wavenumber.

**ncm\_c\_boltzmann\_factor\_H\_1s ()**

```
gdouble          ncm_c_boltzmann_factor_H_1s (const gdouble T);
```

FIXME: Cite source.

**T :** temperature.

**Returns :** Boltzmann factor for Hydrogen 1s level.

---

**ncm\_c\_boltzmann\_factor\_H\_2s ()**

```
gdouble          ncm_c_boltzmann_factor_H_2s          (const gdouble T);
```

FIXME: Cite source.

$T$  : temperature.

**Returns** : Boltzmann factor for Hydrogen 2s level.

**ncm\_c\_boltzmann\_factor\_H\_2p ()**

```
gdouble          ncm_c_boltzmann_factor_H_2p          (const gdouble T);
```

FIXME: Cite source.

$T$  : temperature.

**Returns** : Boltzmann factor for Hydrogen 2p level.

**ncm\_c\_boltzmann\_factor\_HeI\_1s ()**

```
gdouble          ncm_c_boltzmann_factor_HeI_1s        (const gdouble T);
```

FIXME: Cite source.

$T$  : temperature.

**Returns** : Boltzmann factor for HeI 1s level.

**ncm\_c\_boltzmann\_factor\_HeI\_2s ()**

```
gdouble          ncm_c_boltzmann_factor_HeI_2s        (const gdouble T);
```

FIXME: Cite source.

$T$  : temperature.

**Returns** : Boltzmann factor for HeI 2s level.

**ncm\_c\_boltzmann\_factor\_HeI\_2p ()**

```
gdouble          ncm_c_boltzmann_factor_HeI_2p        (const gdouble T);
```

FIXME: Cite source.

$T$  : temperature.

**Returns** : Boltzmann factor for HeI 2p level.

**ncm\_c\_AU ()**

```
gdouble          ncm_c_AU                              (void);
```

**Returns** : Astronomical unit (<http://ssd.jpl.nasa.gov/?constants>).

---

**ncm\_c\_pc ()**

```
gdouble          ncm_c_pc          (void);
```

**Returns :** Parsec unit 1 AU / tan (1 arcsec) - Copied from CAMB/constants.f90 to facilitate comparison.

**ncm\_c\_mass\_solar ()**

```
gdouble          ncm_c_mass_solar  (void);
```

**Returns :** One solar mass.

**ncm\_c\_stats\_1sigma ()**

```
long double      ncm_c_stats_1sigma (void);
```

The integral of a gaussian distribution with mean mu and standard deviation sigma in (mu - 1 \* sigma, mu + 1 \* sigma)

**Returns :** P (mu - 1 \* sigma, mu + 1 \* sigma)

**ncm\_c\_stats\_2sigma ()**

```
long double      ncm_c_stats_2sigma (void);
```

The integral of a gaussian distribution with mean mu and standard deviation sigma in (mu - 2 \* sigma, mu + 2 \* sigma)

**Returns :** P (mu - 2 \* sigma, mu + 2 \* sigma)

**ncm\_c\_stats\_3sigma ()**

```
long double      ncm_c_stats_3sigma (void);
```

The integral of a gaussian distribution with mean mu and standard deviation sigma in (mu - 3 \* sigma, mu + 3 \* sigma)

**Returns :** P (mu - 3 \* sigma, mu + 3 \* sigma)

**ncm\_c\_wmap3\_cmb\_z ()**

```
gdouble          ncm_c_wmap3_cmb_z (void);
```

**Returns :** Wmap3 last scattering redshift.

**ncm\_c\_wmap3\_cmb\_R ()**

```
gdouble          ncm_c_wmap3_cmb_R (void);
```

**Returns :** Wmap3 last scattering shift parameter.

---

**ncm\_c\_wmap3\_cmb\_sigma\_R ()**

```
gdouble          ncm_c_wmap3_cmb_sigma_R          (void);
```

**Returns :** Wmap3 last scattering shift parameter standard deviation.

**ncm\_c\_wmap5\_cmb\_z ()**

```
gdouble          ncm_c_wmap5_cmb_z                (void);
```

**Returns :** Wmap5 last scattering redshift.

**ncm\_c\_wmap5\_cmb\_R ()**

```
gdouble          ncm_c_wmap5_cmb_R                (void);
```

**Returns :** Wmap5 last scattering shift parameter.

**ncm\_c\_wmap5\_cmb\_sigma\_R ()**

```
gdouble          ncm_c_wmap5_cmb_sigma_R          (void);
```

**Returns :** Wmap5 last scattering shift parameter standard deviation.

**ncm\_c\_wmap7\_cmb\_z ()**

```
gdouble          ncm_c_wmap7_cmb_z                (void);
```

**ncm\_c\_wmap7\_cmb\_R ()**

```
gdouble          ncm_c_wmap7_cmb_R                (void);
```

**ncm\_c\_wmap7\_cmb\_sigma\_R ()**

```
gdouble          ncm_c_wmap7_cmb_sigma_R          (void);
```

**ncm\_c\_wmap5\_coadded\_I\_K ()**

```
gdouble          ncm_c_wmap5_coadded_I_K          (void);
```

**Returns :** FIXME

**ncm\_c\_wmap5\_coadded\_I\_Ka ()**

```
gdouble          ncm_c_wmap5_coadded_I_Ka         (void);
```

**Returns :** FIXME

---

**ncm\_c\_wmap5\_coadded\_I\_Q ()**

gdouble	ncm_c_wmap5_coadded_I_Q	(void);
---------	-------------------------	---------

**Returns :** FIXME

**ncm\_c\_wmap5\_coadded\_I\_V ()**

gdouble	ncm_c_wmap5_coadded_I_V	(void);
---------	-------------------------	---------

**Returns :** FIXME

**ncm\_c\_wmap5\_coadded\_I\_W ()**

gdouble	ncm_c_wmap5_coadded_I_W	(void);
---------	-------------------------	---------

**Returns :** FIXME

**ncm\_c\_bao\_eisenstein\_z ()**

gdouble	ncm_c_bao_eisenstein_z	(void);
---------	------------------------	---------

**Returns :** FIXME

**ncm\_c\_bao\_eisenstein\_A ()**

gdouble	ncm_c_bao_eisenstein_A	(void);
---------	------------------------	---------

**Returns :** FIXME

**ncm\_c\_bao\_eisenstein\_sigma\_A ()**

gdouble	ncm_c_bao_eisenstein_sigma_A	(void);
---------	------------------------------	---------

**Returns :** FIXME

**ncm\_c\_bao\_eisenstein\_DV ()**

gdouble	ncm_c_bao_eisenstein_DV	(void);
---------	-------------------------	---------

**Returns :** FIXME

**ncm\_c\_bao\_eisenstein\_sigma\_DV ()**

gdouble	ncm_c_bao_eisenstein_sigma_DV	(void);
---------	-------------------------------	---------

**Returns :** FIXME

---



**ncm\_c\_bao\_percival2007\_DV\_DV ()**

gdouble	ncm_c_bao_percival2007_DV_DV	(void);
---------	------------------------------	---------

**ncm\_c\_bao\_percival2007\_sigma\_DV\_DV ()**

gdouble	ncm_c_bao_percival2007_sigma_DV_DV	(void);
---------	------------------------------------	---------

**ncm\_c\_bao\_percival2010\_DV\_DV ()**

gdouble	ncm_c_bao_percival2010_DV_DV	(void);
---------	------------------------------	---------

**ncm\_c\_bao\_percival2010\_sigma\_DV\_DV ()**

gdouble	ncm_c_bao_percival2010_sigma_DV_DV	(void);
---------	------------------------------------	---------

**ncm\_c\_hubble\_cte\_wmap ()**

gdouble	ncm_c_hubble_cte_wmap	(void);
---------	-----------------------	---------

FIXME

**Returns :** FIXME

**ncm\_c\_hubble\_cte\_hst ()**

gdouble	ncm_c_hubble_cte_hst	(void);
---------	----------------------	---------

FIXME

**Returns :** FIXME

**ncm\_c\_hubble\_cte\_msa ()**

gdouble	ncm_c_hubble_cte_msa	(void);
---------	----------------------	---------

FIXME

**Returns :** FIXME

**ncm\_c\_neutrino\_n\_eff ()**

gdouble	ncm_c_neutrino_n_eff	(void);
---------	----------------------	---------

FIXME

**Returns :** FIXME

---

**ncm\_c\_prim\_He\_Yp ()**

```
gdouble          ncm_c_prim_He_Yp          (void);
```

The primordial helium mass fraction  $Y_p = \frac{m_{\text{He}} n_{\text{He}}}{m_{\text{He}} n_{\text{He}} + m_{\text{H}} n_{\text{H}}}$ , where  $m_{\text{He}}$ ,  $n_{\text{He}}$ ,  $m_{\text{H}}$  and  $n_{\text{H}}$  are respectively helium mass and number density and hydrogen mass and number density.

**Returns :** The primordial helium mass abundance.

**ncm\_c\_prim\_H\_Yp ()**

```
gdouble          ncm_c_prim_H_Yp          (void);
```

The primordial hydrogen mass fraction  $Y_{\text{H}} = 1 - Y_p$ , where  $Y_p$  is the helium mass fraction, see [ncm\\_c\\_prim\\_He\\_Yp\(\)](#).

**Returns :** The primordial hydrogen mass abundance.

**ncm\_c\_prim\_XHe ()**

```
gdouble          ncm_c_prim_XHe          (void);
```

The primordial helium to hydrogen ratio  $X_{\text{He}} = \frac{n_{\text{He}}}{n_{\text{H}}} = \frac{m_{\text{H}}}{m_{\text{He}}} \frac{Y_p}{Y_{\text{H}}}$ , see [ncm\\_c\\_prim\\_H\\_Yp\(\)](#) and [ncm\\_c\\_prim\\_He\\_Yp\(\)](#).

**Returns :** The primordial helium to hydrogen ratio.

**ncm\_c\_hubble\_radius ()**

```
gdouble          ncm_c_hubble_radius      (void);
```

FIXME

**Returns :** Hubble radius

**ncm\_c\_hubble\_radius\_planck ()**

```
gdouble          ncm_c_hubble_radius_planck (void);
```

FIXME

**Returns :** Hubble radius

**ncm\_c\_crit\_density ()**

```
gdouble          ncm_c_crit_density      (void);
```

FIXME

**Returns :** Critical density in ... units.

**ncm\_c\_crit\_mass\_density ()**

```
gdouble          ncm_c_crit_mass_density      (void);
```

FIXME

**Returns :** Critical mass density in ... units.

**ncm\_c\_crit\_mass\_density\_solar\_Mpc ()**

```
gdouble          ncm_c_crit_mass_density_solar_Mpc  (void);
```

FIXME

**Returns :** Critical mass density in ... units.

**ncm\_c\_crit\_number\_density\_p ()**

```
gdouble          ncm_c_crit_number_density_p      (void);
```

FIXME

**Returns :** Critical proton number density in ... units.

**ncm\_c\_crit\_number\_density\_n ()**

```
gdouble          ncm_c_crit_number_density_n      (void);
```

FIXME

**Returns :** Critical neutron number density in ... units.

**ncm\_c\_blackbody\_energy\_density ()**

```
gdouble          ncm_c_blackbody_energy_density    (void);
```

FIXME

**Returns :** Blackbody energy density in ... units.

**ncm\_c\_radiation\_temp\_to\_h2omega\_r ()**

```
gdouble          ncm_c_radiation_temp_to_h2omega_r  (const gdouble T);
```

**ncm\_c\_radiation\_h2Omega\_r\_to\_temp ()**

```
gdouble          ncm_c_radiation_h2Omega_r_to_temp  (const gdouble omr);
```

FIXME

**omr :** FIXME

**Returns :** .

## 2.16 Splines 1D

### 2.16.1 Spline Abstract Class

### Spline Abstract Class — Base class for implementing splines

## Synopsis

```
struct NcmSplineClass;
struct NcmSpline;
NcmSpline * ncm_spline_copy_empty (const NcmSpline *s);
NcmSpline * ncm_spline_copy (const NcmSpline *s);
NcmSpline * ncm_spline_new (const NcmSpline *s,
                             NcmVector *xv,
                             NcmVector *yv,
                             const gboolean init);
NcmSpline * ncm_spline_new_array (const NcmSpline *s,
                                   GArray *x,
                                   GArray *y,
                                   const gboolean init);
NcmSpline * ncm_spline_new_data (const NcmSpline *s,
                                  gdoube *x,
                                  gdoube *y,
                                  const gsize len,
                                  const gboolean init);
NcmSpline * ncm_spline_set (NcmSpline *s,
                              NcmVector *xv,
                              NcmVector *yv,
                              gboolean init);
NcmSpline * ncm_spline_ref (NcmSpline *s);
void ncm_spline_set_xv (NcmSpline *s,
                        NcmVector *xv,
                        gboolean init);
void ncm_spline_set_yv (NcmSpline *s,
                        NcmVector *yv,
                        gboolean init);
void ncm_spline_set_array (NcmSpline *s,
                           GArray *x,
                           GArray *y,
                           gboolean init);
void ncm_spline_set_data_static (NcmSpline *s,
                                 gdoube *x,
                                 gdoube *y,
                                 gsize len,
                                 gboolean init);
NcmVector * ncm_spline_get_xv (NcmSpline *s);
NcmVector * ncm_spline_get_yv (NcmSpline *s);
void ncm_spline_free (NcmSpline *s);
void ncm_spline_clear (NcmSpline **s);
void ncm_spline_prepare (NcmSpline *s);
void ncm_spline_prepare_base (NcmSpline *s);
gdoube ncm_spline_eval (const NcmSpline *s,
                         const gdoube x);
gdoube ncm_spline_eval_deriv (const NcmSpline *s,
                               const gdoube x);
```

gdouble	ncm_spline_eval_deriv2	(const NcmSpline *s, const gdouble x);
gdouble	ncm_spline_eval_deriv_nmax	(const NcmSpline *s, const gdouble x);
gdouble	ncm_spline_eval_integ	(const NcmSpline *s, const gdouble x0, const gdouble x1);
gboolean	ncm_spline_is_empty	(const NcmSpline *s);
gsize	ncm_spline_min_size	(const NcmSpline *s);
guint	ncm_spline_get_index	(const NcmSpline *s, const gdouble x);
const gdouble	r1;	
const gdouble	r2;	
const gdouble	r12;	
const gdouble	bterm;	
const gdouble	cterm;	
const gdouble	dterm;	

## Object Hierarchy

```

GObject
+----NcmSpline
      +----NcmSplineCubic
      +----NcmSplineGsl

```

## Description

This class comprises all functions to provide a **NcmSpline**, together with all necessary methods.

## Details

### struct NcmSplineClass

```

struct NcmSplineClass {
};

```

### struct NcmSpline

```

struct NcmSpline;

```

### ncm\_spline\_copy\_empty ()

```

NcmSpline *      ncm_spline_copy_empty      (const NcmSpline *s);

```

This function copies the spline *s* into an initialized empty **NcmSpline** of a specific type.

**s** : a constant **NcmSpline**.

**Returns** : A **NcmSpline**. *[transfer full]*



This function returns a new **NcmSpline**, where the knots of this new spline are given in the array  $x$  and the values of the function, at those knots, to be interpolated are given in the array  $y$ .

**$s$**  : a constant **NcmSpline**.

**$x$**  : array of knots.

**$y$**  : array of the values of the function, to be interpolated, computed at  $x$ .

**len** : lenght of  $x$  and  $y$ .

**init** : TRUE to prepare the new **NcmSpline** or FALSE to not prepare it.

**Returns** : A new **NcmSpline**. *[transfer full]*

### ncm\_spline\_set ()

```
NcmSpline *      ncm_spline_set      (NcmSpline *s,
                                       NcmVector *xv,
                                       NcmVector *yv,
                                       gboolean init);
```

This funtion sets both  $xv$  and  $yv$  vectors to  $s$ . The two vectors must have the same length.

**$s$**  : a **NcmSpline**.

**$xv$**  : **NcmVector** of knots.

**$yv$**  : **NcmVector** of the values of the function, to be interpolated, computed at  $xv$ .

**init** : TRUE to prepare  $s$  or FALSE to not prepare it.

**Returns** : FIXME. *[transfer none]*

### ncm\_spline\_ref ()

```
NcmSpline *      ncm_spline_ref      (NcmSpline *s);
```

FIXME

**$s$**  : a **NcmSpline**.

**Returns** : FIXME. *[transfer full]*

### ncm\_spline\_set\_xv ()

```
void             ncm_spline_set_xv   (NcmSpline *s,
                                       NcmVector *xv,
                                       gboolean init);
```

This function sets  $xv$  as the knot vector of the spline.

**$s$**  : a **NcmSpline**.

**$xv$**  : **NcmVector** of knots.

**init** : TRUE to prepare  $s$  or FALSE to not prepare it.

**ncm\_spline\_set\_yv ()**

```
void                ncm_spline_set_yv                (NcmSpline *s,
                                                    NcmVector *yv,
                                                    gboolean init);
```

This function sets  $yv$  as the function values vector. This **NcmVector**  $yv$  comprises the function values computed at the knots of the spline.

**s** : a **NcmSpline**.

**yv** : **NcmVector** of the values of the function to be interpolated.

**init** : TRUE to prepare  $s$  or FALSE to not prepare it.

**ncm\_spline\_set\_array ()**

```
void                ncm_spline_set_array            (NcmSpline *s,
                                                    GArray *x,
                                                    GArray *y,
                                                    gboolean init);
```

This function sets  $x$  as the knot vector and  $y$  as the function values vector of the spline.

**s** : a **NcmSpline**.

**x** : GArray of knots. *[element-type double]*

**y** : GArray of the values of the function, to be interpolated, computed at  $x$ . *[element-type double]*

**init** : TRUE to prepare  $s$  or FALSE to not prepare it.

**ncm\_spline\_set\_data\_static ()**

```
void                ncm_spline_set_data_static      (NcmSpline *s,
                                                    gdouble *x,
                                                    gdouble *y,
                                                    gsize len,
                                                    gboolean init);
```

This function sets  $x$  as the knot vector and  $y$  as the function values vector of the spline.

**s** : a **NcmSpline**.

**x** : array of knots.

**y** : array of the values of the function, to be interpolated, computed at  $x$ .

**len** : lenght of  $x$  and  $y$ .

**init** : TRUE to prepare  $s$  or FALSE to not prepare it.

**ncm\_spline\_get\_xv ()**

```
NcmVector *        ncm_spline_get_xv              (NcmSpline *s);
```

This function returns the  $s$  **NcmVector** of knots.

**s** : a **NcmSpline**

**Returns** : A **NcmVector**. *[transfer full]*



**ncm\_spline\_get\_yv ()**

```
NcmVector *      ncm_spline_get_yv      (NcmSpline *s);
```

This function returns the  $s$  **NcmVector** of the values of the function to be interpolated.

**$s$**  : a **NcmSpline**.

**Returns** : A **NcmVector**. *[transfer full]*

**ncm\_spline\_free ()**

```
void      ncm_spline_free      (NcmSpline *s);
```

Atomically decrements the reference count of  $s$  by one. If the reference count drops to 0, all memory allocated by  $s$  is released.

**$s$**  : a **NcmSpline**.

**ncm\_spline\_clear ()**

```
void      ncm_spline_clear      (NcmSpline **s);
```

Atomically decrements the reference count of  $s$  by one. If the reference count drops to 0, all memory allocated by  $s$  is released. The pointer is set to NULL.

**$s$**  : a **NcmSpline**.

**ncm\_spline\_prepare ()**

```
void      ncm_spline_prepare      (NcmSpline *s);
```

This function prepares the spline  $s$  such that one can evaluate it (**ncm\_spline\_eval**), as well as to compute its first and second derivatives (**ncm\_spline\_eval\_deriv**, **ncm\_spline\_eval\_deriv2**) and integration (**ncm\_spline\_eval\_integ**).

**$s$**  : a **NcmSpline**.

**ncm\_spline\_prepare\_base ()**

```
void      ncm_spline_prepare_base      (NcmSpline *s);
```

This function computes the second derivatives of  $s$  and it is used to prepare a bidimensional spline.

**$s$**  : a **NcmSpline**.

**ncm\_spline\_eval ()**

```
gdouble      ncm_spline_eval      (const NcmSpline *s,  
                                   const gdouble x);
```

**$s$**  : a constant **NcmSpline**.

**$x$**  : x-coordinate value.

**Returns** : The interpolated value of a function computed at  $x$ .

**ncm\_spline\_eval\_deriv ()**

gdouble	ncm_spline_eval_deriv	(const NcmSpline *s, const gdouble x);
---------	-----------------------	---

**s** : a constant **NcmSpline**.

**x** : x-coordinate value.

**Returns** : The derivative of an interpolated function computed at  $x$ .

**ncm\_spline\_eval\_deriv2 ()**

gdouble	ncm_spline_eval_deriv2	(const NcmSpline *s, const gdouble x);
---------	------------------------	---

**s** : a constant **NcmSpline**.

**x** : x-coordinate value.

**Returns** : The second derivative of an interpolated function computed at  $x$ .

**ncm\_spline\_eval\_deriv\_nmax ()**

gdouble	ncm_spline_eval_deriv_nmax	(const NcmSpline *s, const gdouble x);
---------	----------------------------	---

**s** : a constant **NcmSpline**.

**x** : x-coordinate value.

**Returns** : The highest non null derivative of an interpolated function computed at  $x$ .

**ncm\_spline\_eval\_integ ()**

gdouble	ncm_spline_eval_integ	(const NcmSpline *s, const gdouble x0, const gdouble x1);
---------	-----------------------	---

**s** : a constant **NcmSpline**.

**x0** : lower integration limit.

**x1** : upper integration limit.

**Returns** : The numerical integral of an interpolated function over the range  $[x0, x1]$ .

**ncm\_spline\_is\_empty ()**

gboolean	ncm_spline_is_empty	(const NcmSpline *s);
----------	---------------------	-----------------------

**ncm\_spline\_min\_size ()**

```
gsize          ncm_spline_min_size          (const NcmSpline *s);
```

**s** : a constant **NcmSpline**.

**Returns** : Minimum number of knots required.

**ncm\_spline\_get\_index ()**

```
guint          ncm_spline_get_index          (const NcmSpline *s,  
                                              const gdouble x);
```

**s** : a constant **NcmSpline**.

**x** : a value of the abscissa axis.

**Returns** : The index of the lower knot of the interval  $x$  belongs to.

**r1**

```
const gdouble r1 = a - xi;
```

**r2**

```
const gdouble r2 = b - xi;
```

**r12**

```
const gdouble r12 = r1 + r2;
```

**bterm**

```
const gdouble bterm = 0.5 * bi * r12;
```

**cterm**

```
const gdouble cterm = (1.0 / 3.0) * ci * (r1 * r1 + r2 * r2 + r1 * r2);
```

**dterm**

```
const gdouble dterm = 0.25 * di * r12 * (r1 * r1 + r2 * r2);
```

**2.16.2 GSL Spline**

GSL Spline — GSL spline object wrapper

## Synopsis

```

enum          NcmSplineGslType;
struct        NcmSplineGslClass;
struct        NcmSplineGsl;
NcmSpline *   ncm_spline_gsl_new          (const gsl_interp_type *type);
NcmSpline *   ncm_spline_gsl_new_full    (const gsl_interp_type *type,
                                           NcmVector *xv,
                                           NcmVector *yv,
                                           gboolean init);

void          ncm_spline_gsl_set_type     (NcmSplineGsl *sg,
                                           const gsl_interp_type *type);

void          ncm_spline_gsl_set_type_by_id (NcmSplineGsl *sg,
                                           NcmSplineGslType type_id);

```

## Object Hierarchy

```

GObject
+-----NcmSpline
        +-----NcmSplineGsl

```

## Properties

"type"	NcmSplineGslType	: Read / Write
"type-name"	gchar*	: Write

## Description

This object comprises the proper functions to use the GNU Scientific Library (GSL) spline functions and interpolation methods.

## Details

### enum NcmSplineGslType

```

typedef enum {
    NCM_SPLINE_GSL_LINEAR = 0,
    NCM_SPLINE_GSL_POLYNOMIAL,
    NCM_SPLINE_GSL_CSPLINE,
    NCM_SPLINE_GSL_CSPLINE_PERIODIC,
    NCM_SPLINE_GSL_AKIMA,
} NcmSplineGslType;

```

FIXME

**NCM\_SPLINE\_GSL\_LINEAR** FIXME

**NCM\_SPLINE\_GSL\_POLYNOMIAL** FIXME

**NCM\_SPLINE\_GSL\_CSPLINE** FIXME

**NCM\_SPLINE\_GSL\_CSPLINE\_PERIODIC** FIXME

**NCM\_SPLINE\_GSL\_AKIMA** FIXME

**NCM\_SPLINE\_GSL\_AKIMA\_PERIODIC** FIXME

**struct NcmSplineGslClass**

```
struct NcmSplineGslClass {
};
```

**struct NcmSplineGsl**

```
struct NcmSplineGsl;
```

**ncm\_spline\_gsl\_new ()**

```
NcmSpline *          ncm_spline_gsl_new          (const gsl_interp_type *type);
```

This function returns a new gsl **NcmSpline** which will use *type* interpolation method.

**type** : gsl interpolation method.

**Returns** : a new **NcmSpline**.

**ncm\_spline\_gsl\_new\_full ()**

```
NcmSpline *          ncm_spline_gsl_new_full      (const gsl_interp_type *type,
                                                    NcmVector *xv,
                                                    NcmVector *yv,
                                                    gboolean init);
```

This function returns a new gsl **NcmSpline** setting all its members.

**type** : gsl interpolation method.

**xv** : **NcmVector** of knots.

**yv** : **NcmVector** of the values of the function, to be interpolated, computed at *xv*.

**init** : TRUE to prepare the new **NcmSpline** or FALSE to not prepare it.

**Returns** : a new **NcmSpline**.

**ncm\_spline\_gsl\_set\_type ()**

```
void                  ncm_spline_gsl_set_type      (NcmSplineGsl *sg,
                                                    const gsl_interp_type *type);
```

This function sets the interpolation method *type* to *sg*.

**sg** : a **NcmSplineGsl**.

**type** : gsl interpolation method.

**ncm\_spline\_gsl\_set\_type\_by\_id ()**

```
void                  ncm_spline_gsl_set_type_by_id (NcmSplineGsl *sg,
                                                    NcmSplineGslType type_id);
```

This function sets the interpolation method *type\_id* to *sg*.

**sg** : a **NcmSplineGsl**.

**type\_id** : gsl interpolation method id.

Property Details

The "type" property

"type"	NcmSplineGslType	: Read / Write
--------	------------------	----------------

GSL Interpolation method.

Default value: NCM\_SPLINE\_GSL\_CSPLINE

The "type-name" property

"type-name"	gchar*	: Write
-------------	--------	---------

GSL Interpolation method name.

Default value: NULL

2.16.3 Cubic Spline Abstract Class

Cubic Spline Abstract Class — Base class for implementing cubic splines

Synopsis

```
struct          NcmSplineCubicClass;
struct          NcmSplineCubic;
```

Object Hierarchy

```
GObject
+----NcmSpline
      +----NcmSplineCubic
            +----NcmSplineCubicNotaknot
```

Description

This class implements the functions which use a polynomial interpolation method of third degree.

Details

struct NcmSplineCubicClass

```
struct NcmSplineCubicClass {
};
```

struct NcmSplineCubic

```
struct NcmSplineCubic;
```

2.16.4 Notaknot Cubic Spline

Notaknot Cubic Spline — Cubic spline with 'not a knot' boundary conditions

## Synopsis

```

struct          NcmSplineCubicNotaknotClass;
struct          NcmSplineCubicNotaknot;
NcmSpline *     ncm_spline_cubic_notaknot_new      (void);
NcmSpline *     ncm_spline_cubic_notaknot_new_full (NcmVector *xv,
                                                    NcmVector *yv,
                                                    gboolean init);

```

## Object Hierarchy

```

GObject
+----NcmSpline
      +----NcmSplineCubic
            +----NcmSplineCubicNotaknot

```

## Description

This object implements the necessary functions to compute a cubic spline with boundary conditions obtained with the 'not a knot' method.

## Details

### struct NcmSplineCubicNotaknotClass

```

struct NcmSplineCubicNotaknotClass {
};

```

### struct NcmSplineCubicNotaknot

```

struct NcmSplineCubicNotaknot;

```

### ncm\_spline\_cubic\_notaknot\_new ()

```

NcmSpline *     ncm_spline_cubic_notaknot_new      (void);

```

This function returns a new cubic **NcmSpline**.

**Returns** : a new **NcmSpline**.

### ncm\_spline\_cubic\_notaknot\_new\_full ()

```

NcmSpline *     ncm_spline_cubic_notaknot_new_full (NcmVector *xv,
                                                    NcmVector *yv,
                                                    gboolean init);

```

This function returns a new **NcmSpline** setting all its members.

**xv** : **NcmVector** of knots.

**yv** : **NcmVector** of the values of the function, to be interpolated, computed at **xv**.

**init** : TRUE to prepare the new **NcmSpline** or FALSE to not prepare it.

**Returns** : a new **NcmSpline**.

## 2.16.5 Spline Autoknots

Spline Autoknots — Automatic generation of the knots of a spline

### Synopsis

```
enum                NcmSplineFuncType;
void                ncm_spline_set_func      (NcmSpline *s,
                                              NcmSplineFuncType ftype,
                                              gsl_function *F,
                                              gdouble xi,
                                              gdouble xf,
                                              gsize max_nodes,
                                              gdouble rel_error);

#define             NCM_SPLINE_FUNC_DEFAULT_MAX_NODES
#define             NCM_SPLINE_KNOT_DIFF_TOL
```

### Description

This set of functions implements 4 different methods to automatically determine the **NcmVector** of knots of a **NcmSpline** given a relative error between the function to be interpolated and the spline result.

### Details

#### enum NcmSplineFuncType

```
typedef enum {
    NCM_SPLINE_FUNCTION_4POINTS,
    NCM_SPLINE_FUNCTION_2x2POINTS,
    NCM_SPLINE_FUNCTION_SPLINE,
    NCM_SPLINE_FUNCTION_SPLINE_LNKNOT,
} NcmSplineFuncType;
```

FIXME

**NCM\_SPLINE\_FUNCTION\_4POINTS** FIXME

**NCM\_SPLINE\_FUNCTION\_2x2POINTS** FIXME

**NCM\_SPLINE\_FUNCTION\_SPLINE** FIXME

**NCM\_SPLINE\_FUNCTION\_SPLINE\_LNKNOT** FIXME

#### ncm\_spline\_set\_func ()

```
void                ncm_spline_set_func      (NcmSpline *s,
                                              NcmSplineFuncType ftype,
                                              gsl_function *F,
                                              gdouble xi,
                                              gdouble xf,
                                              gsize max_nodes,
                                              gdouble rel_error);
```

This function automatically determines the knots of *s* in the interval [*xi*, *xf*] given a *ftype* and *rel\_error*.

**s** : a **NcmSpline**.



***ftype*** : a **NcmSplineFuncType**.

***f*** : function to be approximated by spline functions.

***xi*** : lower knot.

***xf*** : upper knot.

***max\_nodes*** : maximum number of knots.

***rel\_error*** : relative error between the function to be interpolated and the spline result.

## NCM\_SPLINE\_FUNC\_DEFAULT\_MAX\_NODES

```
#define NCM_SPLINE_FUNC_DEFAULT_MAX_NODES 10000
```

## NCM\_SPLINE\_KNOT\_DIFF\_TOL

```
#define NCM_SPLINE_KNOT_DIFF_TOL (GSL_DBL_EPSILON * 1.0e2)
```

## 2.16.6 ODE Spline Interpolation

ODE Spline Interpolation — Automatic generation of splines from ODE solutions

### Synopsis

```
gdouble (*NcmOdeSplineDydx) (gdouble y,
                              gdouble x,
                              gpointer userdata);

struct NcmOdeSpline * ncm_ode_spline_new (NcmSpline *s,
                                           NcmOdeSplineDydx dydx,
                                           gpointer userdata,
                                           gdouble yi,
                                           gdouble xi,
                                           gdouble xf);

void ncm_ode_spline_prepare (NcmOdeSpline *os,
                             gpointer userdata);

void ncm_ode_spline_free (NcmOdeSpline *os);

void ncm_ode_spline_clear (NcmOdeSpline **os);
```

### Description

FIXME

### Details

#### NcmOdeSplineDydx ()

```
gdouble (*NcmOdeSplineDydx) (gdouble y,
                              gdouble x,
                              gpointer userdata);
```

**struct NcmOdeSpline**

```
struct NcmOdeSpline {  
};
```

FIXME

**ncm\_ode\_spline\_new ()**

```
NcmOdeSpline *      ncm_ode_spline_new      (NcmSpline *s,  
                                              NcmOdeSplineDydx dydx,  
                                              gpointer userdata,  
                                              gdouble yi,  
                                              gdouble xi,  
                                              gdouble xf);
```

FIXME

***s*** : a **NcmSpline*****dydx*** : a **NcmOdeSplineDydx*****userdata*** : FIXME***yi*** : FIXME***xi*** : FIXME***xf*** : FIXME***Returns*** : FIXME**ncm\_ode\_spline\_prepare ()**

```
void                ncm_ode_spline_prepare  (NcmOdeSpline *os,  
                                              gpointer userdata);
```

FIXME

***os*** : a **NcmOdeSpline*****userdata*** : FIXME**ncm\_ode\_spline\_free ()**

```
void                ncm_ode_spline_free    (NcmOdeSpline *os);
```

FIXME

***os*** : a **NcmOdeSpline****ncm\_ode\_spline\_clear ()**

```
void                ncm_ode_spline_clear    (NcmOdeSpline **os);
```

FIXME

***os*** : a **NcmOdeSpline**

## 2.17 Splines 2D

### 2.17.1 Bidimensional Spline Abstract Class

Bidimensional Spline Abstract Class — Base class for implementing bidimensional splines

#### Synopsis

```

struct          NcmSpline2dClass;
struct          NcmSpline2d;
void            ncm_spline2d_set          (NcmSpline2d *s2d,
                                           NcmVector *xv,
                                           NcmVector *yv,
                                           NcmMatrix *zm,
                                           gboolean init);

void            ncm_spline2d_set_function (NcmSpline2d *s2d,
                                           NcmSplineFuncType ftype,
                                           gsl_function *Fx,
                                           gsl_function *Fy,
                                           gdouble xl,
                                           gdouble xu,
                                           gdouble yl,
                                           gdouble yu,
                                           gdouble rel_err);

void            ncm_spline2d_prepare      (NcmSpline2d *s2d);
guint           ncm_spline2d_min_size     (NcmSpline2d *s2d);
NcmSpline2d *   ncm_spline2d_copy_empty   (const NcmSpline2d *s2d);
NcmSpline2d *   ncm_spline2d_copy        (NcmSpline2d *s2d);
NcmSpline2d *   ncm_spline2d_new          (const NcmSpline2d *s2d,
                                           NcmVector *xv,
                                           NcmVector *yv,
                                           NcmMatrix *zm,
                                           gboolean init);

void            ncm_spline2d_free          (NcmSpline2d *s2d);
void            ncm_spline2d_clear         (NcmSpline2d **s2d);
gdouble         ncm_spline2d_eval          (NcmSpline2d *s2d,
                                           gdouble x,
                                           gdouble y);

gdouble         ncm_spline2d_integ_dx      (NcmSpline2d *s2d,
                                           gdouble xl,
                                           gdouble xu,
                                           gdouble y);

gdouble         ncm_spline2d_integ_dy      (NcmSpline2d *s2d,
                                           gdouble x,
                                           gdouble yl,
                                           gdouble yu);

gdouble         ncm_spline2d_integ_dxdy   (NcmSpline2d *s2d,
                                           gdouble xl,
                                           gdouble xu,
                                           gdouble yl,
                                           gdouble yu);

NcmSpline *     ncm_spline2d_integ_dx_spline (NcmSpline2d *s2d,
                                           gdouble xl,
                                           gdouble xu);

NcmSpline *     ncm_spline2d_integ_dy_spline (NcmSpline2d *s2d,
```

gdouble	ncm_spline2d_integ_dx_spline_val	gdouble yl, gdouble yu); (NcmSpline2d *s2d, gdouble xl, gdouble xu, gdouble y);
gdouble	ncm_spline2d_integ_dy_spline_val	(NcmSpline2d *s2d, gdouble x, gdouble yl, gdouble yu);
gdouble	ncm_spline2d_integ_dxdy_spline_x	(NcmSpline2d *s2d, gdouble xl, gdouble xu, gdouble yl, gdouble yu);
gdouble	ncm_spline2d_integ_dxdy_spline_y	(NcmSpline2d *s2d, gdouble xl, gdouble xu, gdouble yl, gdouble yu);
gdouble	ncm_spline2dim_integ_total	(NcmSpline2d *s2d);

## Object Hierarchy

```
GObject
+----NcmSpline2d
      +----NcmSpline2dBicubic
      +----NcmSpline2dGsl
      +----NcmSpline2dSpline
```

## Properties

```
"spline"                NcmSpline*                : Read / Write / Construct Only
```

## Description

This class comprises all functions to provide a **NcmSpline2d**, get its properties and evaluate it given an interpolation method.

## Details

### struct NcmSpline2dClass

```
struct NcmSpline2dClass {
};
```

### struct NcmSpline2d

```
struct NcmSpline2d;
```

**ncm\_spline2d\_set ()**

```
void                ncm_spline2d_set                (NcmSpline2d *s2d,
                                                    NcmVector *xv,
                                                    NcmVector *yv,
                                                    NcmMatrix *zm,
                                                    gboolean init);
```

This funtion sets  $xv$  and  $yv$  vectors and  $zm$  matrix to  $s$ .

**$s2d$**  : a **NcmSpline2d**

**$xv$**  : a **NcmVector** of knots.

**$yv$**  : a **NcmVector** of knots.

**$zm$**  : a **NcmMatrix** of the values of the function, to be interpolated, computed at  $xv$  and  $yv$ .

**$init$**  : TRUE to prepare the **NcmSpline2d** or FALSE to not prepare it.

**ncm\_spline2d\_set\_function ()**

```
void                ncm_spline2d_set_function        (NcmSpline2d *s2d,
                                                    NcmSplineFuncType ftype,
                                                    gsl_function *Fx,
                                                    gsl_function *Fy,
                                                    gdouble xl,
                                                    gdouble xu,
                                                    gdouble yl,
                                                    gdouble yu,
                                                    gdouble rel_err);
```

This function automatically determines the knots of  $s2d$  in the intervals  $[xl, xu]$  and  $[yl, yu]$  given a  $ftype$  and  $rel\_error$ .

The functions  $Fx$  and  $Fy$  are the bidimensional function given at specific values of  $y$  and  $x$ , respectively. These  $x$  and  $y$  values must be in the the intervals  $[xl, xu]$  and  $[yl, yu]$ .

**$s2d$**  : a **NcmSpline2d**.

**$ftype$**  : a **NcmSplineFuncType**.

**$Fx$**  : function of  $x$  variable to be approximated by spline functions.

**$Fy$**  : function of  $y$  variable to be approximated by spline functions.

**$xl$**  : lower knot of  $x$ -coordinate.

**$xu$**  : upper knot of  $x$ -coordinate.

**$yl$**  : lower knot of  $y$ -coordinate.

**$yu$**  : upper knot of  $y$ -coordinate.

**$rel\_err$**  : relative error between the function to be interpolated and the spline result.

**ncm\_spline2d\_prepare ()**

```
void                ncm_spline2d_prepare            (NcmSpline2d *s2d);
```

This function prepares the bidimensional spline  $s2d$  such that one can evaluate it (**ncm\_spline2d\_eval**), as well as to compute its integration in  $x$ ,  $y$  or both directions.

**$s2d$**  : a **NcmSpline2d**.

**ncm\_spline2d\_min\_size ()**

```
guint          ncm_spline2d_min_size          (NcmSpline2d *s2d);
```

**s2d** : a **NcmSpline2d**.

**Returns** : The size of the **NcmSpline** member of *s2d*.

**ncm\_spline2d\_copy\_empty ()**

```
NcmSpline2d *  ncm_spline2d_copy_empty        (const NcmSpline2d *s2d);
```

This function copies the bidimensional spline *s2d* into an initialized empty **NcmSpline2d** of a specific type.

**s2d** : a **NcmSpline2d**.

**Returns** : a **NcmSpline2d**. *[transfer full]*

**ncm\_spline2d\_copy ()**

```
NcmSpline2d *  ncm_spline2d_copy              (NcmSpline2d *s2d);
```

This function copies the two **NcmVector** and the **NcmMatrix** of the bidimensional spline *s2d* into those two **NcmVector** and **NcmMatrix** of a new **NcmSpline2d**.

**s2d** : a **NcmSpline2d**.

**Returns** : A **NcmSpline2d**. *[transfer full]*

**ncm\_spline2d\_new ()**

```
NcmSpline2d *  ncm_spline2d_new               (const NcmSpline2d *s2d,
                                                NcmVector *xv,
                                                NcmVector *yv,
                                                NcmMatrix *zm,
                                                gboolean init);
```

This function returns a new **NcmSpline2d**, where the knots of this new spline are given in the **NcmVector** *xv* and *yv*. The values of the function, at those knots, to be interpolated are given in the **NcmMatrix** *zm*.

**s2d** : a constant **NcmSpline2d**.

**xv** : **NcmVector** of knots.

**yv** : **NcmVector** of knots.

**zm** : **NcmMatrix** of the values of the function, to be interpolated, computed at *xv* and *yv*.

**init** : TRUE to prepare the new **NcmSpline2d** or FALSE to not prepare it.

**Returns** : A new **NcmSpline2d**. *[transfer full]*

**ncm\_spline2d\_free ()**

```
void          ncm_spline2d_free              (NcmSpline2d *s2d);
```

Atomically decrements the reference count of *s2d* by one. If the reference count drops to 0, all memory allocated by *s2d* is released.

**s2d** : a **NcmSpline2d**.

**ncm\_spline2d\_clear ()**

```
void                ncm_spline2d_clear                (NcmSpline2d **s2d);
```

Atomically decrements the reference count of *s2d* by one. If the reference count drops to 0, all memory allocated by *s2d* is released. Set pointer to NULL.

**s2d** : a **NcmSpline2d**.

**ncm\_spline2d\_eval ()**

```
gdouble            ncm_spline2d_eval                (NcmSpline2d *s2d,  
                                                    gdouble x,  
                                                    gdouble y);
```

**s2d** : a **NcmSpline2d**.

**x** : x-coordinate value.

**y** : y-coordinate value.

**Returns** : The interpolated value of a function computed at the point (*x*, *y*).

**ncm\_spline2d\_integ\_dx ()**

```
gdouble            ncm_spline2d_integ_dx            (NcmSpline2d *s2d,  
                                                    gdouble x1,  
                                                    gdouble xu,  
                                                    gdouble y);
```

This function computes the integration in x over the interval [*x1*, *xu*] and at *y*.

**s2d** : a **NcmSpline2d**.

**x1** : lower limit of integration.

**xu** : upper limit of integration.

**y** : y-coordinate value.

**Returns** : The numerical integral in x of an interpolated function over the range [*x1*, *xu*] and at *y*.

**ncm\_spline2d\_integ\_dy ()**

```
gdouble            ncm_spline2d_integ_dy            (NcmSpline2d *s2d,  
                                                    gdouble x,  
                                                    gdouble y1,  
                                                    gdouble yu);
```

This function computes the integration in y over the interval [*y1*, *yu*] and at *x*.

**s2d** : a **NcmSpline2d**.

**x** : x-coordinate value.

**y1** : lower limit of integration.

**yu** : upper limit of integration.

**Returns** : The numerical integral in y of an interpolated function over the range [*y1*, *yu*] and at *x*.

**ncm\_spline2d\_integ\_dxdy ()**

```

gdouble          ncm_spline2d_integ_dxdy      (NcmSpline2d *s2d,
                                                gdouble x1,
                                                gdouble xu,
                                                gdouble y1,
                                                gdouble yu);

```

This function computes the integration in both x and y directions over the intervals  $[x1, xu]$  and  $[y1, yu]$ .

**s2d** : a **NcmSpline2d**

**x1** : lower limit of integration in the x-direction.

**xu** : upper limit of integration in the x-direction.

**y1** : lower limit of integration in the y-direction.

**yu** : upper limit of integration in the y-direction.

**Returns** : The numerical integral in x and y of an interpolated function over the ranges  $[x1, xu]$  and  $[y1, yu]$ .

**ncm\_spline2d\_integ\_dx\_spline ()**

```

NcmSpline *      ncm_spline2d_integ_dx_spline  (NcmSpline2d *s2d,
                                                gdouble x1,
                                                gdouble xu);

```

This function computes the integral in x of the bidimensional interpolated function over the range  $[x1, xu]$  resulting in a one dimensional function.

**s2d** : a **NcmSpline2d**.

**x1** : lower limit of integration x.

**xu** : upper limit of integration x.

**Returns** : A **NcmSpline**. *[transfer full]*

**ncm\_spline2d\_integ\_dy\_spline ()**

```

NcmSpline *      ncm_spline2d_integ_dy_spline  (NcmSpline2d *s2d,
                                                gdouble y1,
                                                gdouble yu);

```

This function computes the integral in y of the bidimensional interpolated function over the range  $[y1, yu]$  resulting in a one dimensional function.

**s2d** : a **NcmSpline2d**.

**y1** : lower limit of integration.

**yu** : upper limit of integration.

**Returns** : A **NcmSpline**. *[transfer full]*



**ncm\_spline2d\_integ\_dx\_spline\_val ()**

gdouble	ncm_spline2d_integ_dx_spline_val	(NcmSpline2d *s2d, gdouble x1, gdouble xu, gdouble y);
---------	----------------------------------	---

This function calls `ncm_spline2d_integ_dx_spline` and evaluates the resulting `NcmSpline` at  $y$ .

**s2d** : a `NcmSpline2d`.

**x1** : lower limit of integration.

**xu** : upper limit of integration.

**y** : y-coordinate value.

**Returns** : The value of  $s2d$  integrated in  $x$  over the range  $[x1, xu]$  and computed at  $y$ .

**ncm\_spline2d\_integ\_dy\_spline\_val ()**

gdouble	ncm_spline2d_integ_dy_spline_val	(NcmSpline2d *s2d, gdouble x, gdouble y1, gdouble yu);
---------	----------------------------------	---

This function calls `ncm_spline2d_integ_dy_spline` and evaluates the resulting `NcmSpline` at  $x$ .

**s2d** : a `NcmSpline2d`.

**x** : x-coordinate value.

**y1** : lower limit of integration.

**yu** : upper limit of integration.

**Returns** : The value of  $s2d$  integrated in  $y$  over the range  $[y1, yu]$  and computed at  $x$ .

**ncm\_spline2d\_integ\_dxdy\_spline\_x ()**

gdouble	ncm_spline2d_integ_dxdy_spline_x	(NcmSpline2d *s2d, gdouble x1, gdouble xu, gdouble y1, gdouble yu);
---------	----------------------------------	---

This function calls `ncm_spline2d_integ_dx_spline` and integrates the resulting `NcmSpline` over the interval  $[y1, yu]$ .

**s2d** : a `NcmSpline2d`.

**x1** : lower limit of integration in the x-direction.

**xu** : upper limit of integration in the x-direction.

**y1** : lower limit of integration in the y-direction.

**yu** : upper limit of integration in the y-direction.

**Returns** : The value of  $s2d$  integrated in  $x$  and  $y$  over the ranges  $[x1, xu]$  and  $[y1, yu]$ , respectively.

**ncm\_spline2d\_integ\_dxdy\_spline\_y ()**

```
gdouble          ncm_spline2d_integ_dxdy_spline_y      (NcmSpline2d *s2d,
                                                         gdouble x1,
                                                         gdouble xu,
                                                         gdouble y1,
                                                         gdouble yu);
```

This function calls `ncm_spline2d_integ_dy_spline` and integrates the resulting `NcmSpline` over the interval  $[x1, xu]$ .

**s2d** : a `NcmSpline2d`.

**x1** : lower limit of integration in the x-direction.

**xu** : upper limit of integration in the x-direction.

**y1** : lower limit of integration in the y-direction.

**yu** : upper limit of integration in the y-direction.

**Returns** : The value of *s2d* integrated in x and y over the ranges  $[x1, xu]$  and  $[y1, yu]$ , respectively.

**ncm\_spline2dim\_integ\_total ()**

```
gdouble          ncm_spline2dim_integ_total            (NcmSpline2d *s2d);
```

**s2d** : a `NcmSpline2d`.

**Returns** : The numerical integral in both x and y directions of an interpolated function over the entire valid ranges of x and y coordinates.

**Property Details****The "spline" property**

"spline"	NcmSpline*	: Read / Write / Construct Only
----------	------------	---------------------------------

`NcmSpline` object used internally.

**2.17.2 Bidimensional Spline from Spline**

Bidimensional Spline from Spline — Implements bidimensional splines from splines method.

**Synopsis**

```
struct          NcmSpline2dSplineClass;
struct          NcmSpline2dSpline;
NcmSpline2d *   ncm_spline2d_spline_new                (NcmSpline *s);
```

**Object Hierarchy**

```
GObject
+----NcmSpline2d
      +----NcmSpline2dSpline
```

**Description**

FIXME

**Details****struct NcmSpline2dSplineClass**

```
struct NcmSpline2dSplineClass {
};
```

**struct NcmSpline2dSpline**

```
struct NcmSpline2dSpline;
```

**ncm\_spline2d\_spline\_new ()**

```
NcmSpline2d *      ncm_spline2d_spline_new      (NcmSpline *s);
```

This function initializes a **NcmSpline2d** **FIXME**

**s** : a **NcmSpline**.

**Returns** : A new **NcmSpline2d**.

**2.17.3 Bidimensional Bicubic Spline**

Bidimensional Bicubic Spline — Implements a bidimensional bicubic spline.

**Synopsis**

```
struct          NcmSpline2dBicubicCoeffs;
struct          _NcmSpline2dBicubicOptimizeInt;
struct          NcmSpline2dBicubicClass;
struct          NcmSpline2dBicubic;
NcmSpline2d *   ncm_spline2d_bicubic_new      (NcmSpline *s);
NcmSpline2d *   ncm_spline2d_bicubic_notaknot_new  ();
#define         NCM_SPLINE2D_BICUBIC_00
#define         NCM_SPLINE2D_BICUBIC_10
#define         NCM_SPLINE2D_BICUBIC_01
#define         NCM_SPLINE2D_BICUBIC_11
#define         NCM_SPLINE2D_BICUBIC_F
#define         NCM_SPLINE2D_BICUBIC_FX
#define         NCM_SPLINE2D_BICUBIC_FY
#define         NCM_SPLINE2D_BICUBIC_FXY
#define         NCM_SPLINE2D_BICUBIC_COEFF_INDEX      (s2d,
                                                         i,
                                                         j)
#define         NCM_SPLINE2D_BICUBIC_COEFF            (s2d,
                                                         i,
                                                         j)
#define         NCM_SPLINE2D_BICUBIC_STRUCT          (s2d,
```

		i, j)
gdouble	ncm_spline2d_bicubic_eval_poly	(const NcmSpline2dBicubicCoeffs *s, const gdouble x, const gdouble y);
void	ncm_spline2d_bicubic_fij_to_aij	(NcmSpline2dBicubicCoeffs *sf, const gdouble dx, const gdouble dy, NcmSpline2dBicubicCoeffs *sa);
gdouble	ncm_spline2d_bicubic_bi	(NcmSplineCubic *sc, NcmVector *xv, NcmVector *yv, gsize i);
void	ncm_spline2d_bicubic_bi_bipl	(NcmSplineCubic *sc, NcmVector *xv, NcmVector *yv, gsize i, gdouble *b_i, gdouble *b_ip1);
void	ncm_spline2d_bicubic_integ_dx_coeffs	(NcmSpline2dBicubicCoeffs *aij, gdouble dy, gdouble *coeffs);
void	ncm_spline2d_bicubic_integ_dy_coeffs	(NcmSpline2dBicubicCoeffs *aij, gdouble dx, gdouble *coeffs);
gdouble	ncm_spline2d_bicubic_integ_eval2d	(NcmSpline2dBicubicCoeffs *aij, const gdouble x0, const gdouble xl, const gdouble xu, const gdouble y0, const gdouble yl, const gdouble yu);

## Object Hierarchy

```
GObject
+----NcmSpline2d
      +----NcmSpline2dBicubic
```

## Description

This class implements the functions which use a bicubic interpolation method.

## Details

### struct NcmSpline2dBicubicCoeffs

```
struct NcmSpline2dBicubicCoeffs {
};
```

FIXME

**struct \_NcmSpline2dBicubicOptimizeInt**

```
struct _NcmSpline2dBicubicOptimizeInt {
};
```

**struct NcmSpline2dBicubicClass**

```
struct NcmSpline2dBicubicClass {
};
```

**struct NcmSpline2dBicubic**

```
struct NcmSpline2dBicubic;
```

**ncm\_spline2d\_bicubic\_new ()**

```
NcmSpline2d *      ncm_spline2d_bicubic_new      (NcmSpline *s);
```

This function initializes a **NcmSpline2d** of bicubic type given *s*.

**s** : a **NcmSplineCubic** derived **NcmSpline**.

**Returns** : A new **NcmSpline2d**.

**ncm\_spline2d\_bicubic\_notaknot\_new ()**

```
NcmSpline2d *      ncm_spline2d_bicubic_notaknot_new      ();
```

This function initializes a **NcmSpline2d** of bicubic notaknot type.

**Returns** : A new **NcmSpline2d**.

**NCM\_SPLINE2D\_BICUBIC\_00**

```
#define NCM_SPLINE2D_BICUBIC_00 (0)
```

**NCM\_SPLINE2D\_BICUBIC\_10**

```
#define NCM_SPLINE2D_BICUBIC_10 (1)
```

**NCM\_SPLINE2D\_BICUBIC\_01**

```
#define NCM_SPLINE2D_BICUBIC_01 (2)
```

**NCM\_SPLINE2D\_BICUBIC\_11**

```
#define NCM_SPLINE2D_BICUBIC_11 (3)
```

```
#define NCM_SPLINE2D_BICUBIC_F      (0)
```

```
#define NCM_SPLINE2D_BICUBIC_FX (1)
```

```
#define NCM_SPLINE2D_BICUBIC_FY (2)
```

```
#define NCM_SPLINE2D_BICUBIC_FXY (3)
```

```
#define NCM_SPLINE2D_BICUBIC_COEFF_INDEX(s2d,i,j) ((s2d)->z_x->len * (i) + (j))
```

```
#define NCM_SPLINE2D_BICUBIC_COEFF(s2d,i,j) ((s2d)->bicoeff[ ←  
      NCM_SPLINE2D_BICUBIC_COEFF_INDEX(s2d,i,j)].ij)
```

```
#define NCM_SPLINE2D_BICUBIC_STRUCT(s2d,i,j) ((s2d)->bicoeff[ ←  
    NCM_SPLINE2D_BICUBIC_COEFF_INDEX(s2d,i,j)])
```

[illegible][illegible]

**ncm\_spline2d\_bicubic\_bi ()**

```
gdouble          ncm_spline2d_bicubic_bi      (NcmSplineCubic *sc,
                                                NcmVector *xv,
                                                NcmVector *yv,
                                                gsize i);
```

**ncm\_spline2d\_bicubic\_bi\_bip1 ()**

```
void             ncm_spline2d_bicubic_bi_bip1 (NcmSplineCubic *sc,
                                                NcmVector *xv,
                                                NcmVector *yv,
                                                gsize i,
                                                gdouble *b_i,
                                                gdouble *b_ip1);
```

**ncm\_spline2d\_bicubic\_integ\_dx\_coeffs ()**

```
void             ncm_spline2d_bicubic_integ_dx_coeffs (NcmSpline2dBicubicCoeffs *aij,
                                                        gdouble dy,
                                                        gdouble *coeffs);
```

**ncm\_spline2d\_bicubic\_integ\_dy\_coeffs ()**

```
void             ncm_spline2d_bicubic_integ_dy_coeffs (NcmSpline2dBicubicCoeffs *aij,
                                                        gdouble dx,
                                                        gdouble *coeffs);
```

**ncm\_spline2d\_bicubic\_integ\_eval2d ()**

```
gdouble          ncm_spline2d_bicubic_integ_eval2d (NcmSpline2dBicubicCoeffs *aij,
                                                      const gdouble x0,
                                                      const gdouble xl,
                                                      const gdouble xu,
                                                      const gdouble y0,
                                                      const gdouble yl,
                                                      const gdouble yu);
```

**2.17.4 Bidimensional Spline from Spline (GSL)**

Bidimensional Spline from Spline (GSL) — Implements spline from spline method using GSL as base splines

**Synopsis**

```
struct           NcmSpline2dGslClass;
struct           NcmSpline2dGsl;
NcmSpline2d *    ncm_spline2d_gsl_new          (NcmSpline *s);
NcmSpline2d *    ncm_spline2d_gsl_natural_new  ();
```

## Object Hierarchy

```
GObject
+----NcmSpline2d
      +----NcmSpline2dGsl
```

## Description

This object implements spline from spline methods using GSL splines as base objects.

## Details

### struct NcmSpline2dGslClass

```
struct NcmSpline2dGslClass {
};
```

### struct NcmSpline2dGsl

```
struct NcmSpline2dGsl;
```

### ncm\_spline2d\_gsl\_new ()

```
NcmSpline2d *      ncm_spline2d_gsl_new      (NcmSpline *s);
```

This function initializes a **NcmSpline2d** of gsl type given s.

**s** : a **NcmSplineGsl** derived **NcmSpline**.

**Returns** : A new **NcmSpline2d**.

### ncm\_spline2d\_gsl\_natural\_new ()

```
NcmSpline2d *      ncm_spline2d_gsl_natural_new      ();
```

This function initializes a **NcmSpline2d** of gsl type.

**Returns** : A new **NcmSpline2d**.

## 2.18 Special Functions

### 2.18.1 Trigonometric Integrals

Trigonometric Integrals — Sin integral implementation with support for multiple precision calculation

#### Synopsis

```
void      ncm_mpsf_sin_int_mpfr      (mpq_t q,
                                       mpfr_ptr res,
                                       mp_rnd_t rnd);
```



**Description**

FIXME

**Details****ncm\_mpsf\_sin\_int\_mpfr ()**

```
void                ncm_mpsf_sin_int_mpfr                (mpq_t q,
                                                         mpfr_ptr res,
                                                         mp_rnd_t rnd);
```

FIXME

***q*** : FIXME

***res*** : FIXME

***rnd*** : FIXME

**2.18.2 Hypergeometric 0F1**

Hypergeometric 0F1 — Hypergeometric 0F1 multiple precision implementation

**Synopsis**

```
void                ncm_mpsf_0F1_q                (mpq_t b,
                                                         mpq_t x,
                                                         mpfr_ptr res,
                                                         mp_rnd_t rnd);

void                ncm_mpsf_0F1_d                (gdouble b,
                                                         gdouble x,
                                                         mpfr_ptr res,
                                                         mp_rnd_t rnd);

gdouble            ncm_sf_0F1                (gdouble b,
                                                         gdouble x);
```

**Description**

FIXME

**Details****ncm\_mpsf\_0F1\_q ()**

```
void                ncm_mpsf_0F1_q                (mpq_t b,
                                                         mpq_t x,
                                                         mpfr_ptr res,
                                                         mp_rnd_t rnd);
```

FIXME

***b*** : FIXME

***x*** : FIXME

***res*** : FIXME

***rnd*** : FIXME

**ncm\_mpsf\_0F1\_d ()**

```
void          ncm_mpsf_0F1_d          (gdouble b,
                                       gdouble x,
                                       mpfr_ptr res,
                                       mp_rnd_t rnd);
```

FIXME

**b** : FIXME

**x** : FIXME

**res** : FIXME

**rnd** : FIXME

**ncm\_sf\_0F1 ()**

```
gdouble       ncm_sf_0F1              (gdouble b,
                                       gdouble x);
```

FIXME

**b** : FIXME

**x** : FIXME

**2.18.3 Spherical Bessel**

Spherical Bessel — Spherical bessel implementation with support for multiple precision calculation

**Synopsis**

```
struct          NcmMpsfSBesselRecur;
void           ncm_mpsf_sbessel

void           ncm_mpsf_sbessel_d

NcmMpsfSBesselRecur * ncm_mpsf_sbessel_recur_new
NcmMpsfSBesselRecur * ncm_mpsf_sbessel_recur_read
void           ncm_mpsf_sbessel_recur_free
void           ncm_mpsf_sbessel_recur_set_q

void           ncm_mpsf_sbessel_recur_set_d

void           ncm_mpsf_sbessel_recur_next

(gulong l,
 mpq_t q,
 mpfr_ptr res,
 mp_rnd_t rnd);
(gulong l,
 gdouble x,
 mpfr_ptr res,
 mp_rnd_t rnd);
(gulong prec);
(FILE *f);
(NcmMpsfSBesselRecur *jlrec);
(NcmMpsfSBesselRecur *jlrec,
 gulong l,
 mpq_ptr q,
 mp_rnd_t rnd);
(NcmMpsfSBesselRecur *jlrec,
 gulong l,
 gdouble x,
 mp_rnd_t rnd);
(NcmMpsfSBesselRecur *jlrec,
 mp_rnd_t rnd);
```

```
void          ncm_mpsf_sbessel_recur_previous (NcmMpsfSBesselRecur *jlrec,
mp_rnd_t rnd);
void          ncm_mpsf_sbessel_recur_goto    (NcmMpsfSBesselRecur *jlrec,
glong l,
mp_rnd_t rnd);
void          ncm_mpsf_sbessel_recur_write   (NcmMpsfSBesselRecur *jlrec,
FILE *f);
```

**Description**

FIXME

**Details**

**struct NcmMpsfSBesselRecur**

```
struct NcmMpsfSBesselRecur {
};
```

FIXME

**ncm\_mpsf\_sbessel ()**

```
void          ncm_mpsf_sbessel                (gulong l,
mpq_t q,
mpfr_ptr res,
mp_rnd_t rnd);
```

FIXME

***l*** : FIXME

***q*** : FIXME

***res*** : FIXME

***rnd*** : FIXME

**ncm\_mpsf\_sbessel\_d ()**

```
void          ncm_mpsf_sbessel_d              (gulong l,
gdouble x,
mpfr_ptr res,
mp_rnd_t rnd);
```

FIXME

***l*** : FIXME

***x*** : FIXME

***res*** : FIXME

***rnd*** : FIXME

**ncm\_mpsf\_sbessel\_recur\_new ()**

```
NcmMpsfSBesselRecur * ncm_mpsf_sbessel_recur_new      (gulong prec);
```

FIXME

*prec* : FIXME

*Returns* : FIXME

**ncm\_mpsf\_sbessel\_recur\_read ()**

```
NcmMpsfSBesselRecur * ncm_mpsf_sbessel_recur_read    (FILE *f);
```

FIXME

*f* : FIXME

*Returns* : FIXME

**ncm\_mpsf\_sbessel\_recur\_free ()**

```
void                  ncm_mpsf_sbessel_recur_free    (NcmMpsfSBesselRecur *jlrec);
```

FIXME

*jlrec* : a **NcmMpsfSBesselRecur**

**ncm\_mpsf\_sbessel\_recur\_set\_q ()**

```
void                  ncm_mpsf_sbessel_recur_set_q    (NcmMpsfSBesselRecur *jlrec,
                                                       glong l,
                                                       mpq_ptr q,
                                                       mp_rnd_t rnd);
```

FIXME

*jlrec* : a **NcmMpsfSBesselRecur**

*l* : FIXME

*q* : FIXME

*rnd* : FIXME

**ncm\_mpsf\_sbessel\_recur\_set\_d ()**

```
void                  ncm_mpsf_sbessel_recur_set_d    (NcmMpsfSBesselRecur *jlrec,
                                                       glong l,
                                                       gdouble x,
                                                       mp_rnd_t rnd);
```

FIXME

*jlrec* : a **NcmMpsfSBesselRecur**

*l* : FIXME

*x* : FIXME

*rnd* : FIXME

**ncm\_mpsf\_sbessel\_recur\_next ()**

```
void                ncm_mpsf_sbessel_recur_next      (NcmMpsfSBesselRecur *jlrec,
                                                       mp_rnd_t rnd);
```

FIXME

**jlrec** : a **NcmMpsfSBesselRecur**

**rnd** : FIXME

**ncm\_mpsf\_sbessel\_recur\_previous ()**

```
void                ncm_mpsf_sbessel_recur_previous  (NcmMpsfSBesselRecur *jlrec,
                                                       mp_rnd_t rnd);
```

FIXME

**jlrec** : a **NcmMpsfSBesselRecur**

**rnd** : FIXME

**ncm\_mpsf\_sbessel\_recur\_goto ()**

```
void                ncm_mpsf_sbessel_recur_goto     (NcmMpsfSBesselRecur *jlrec,
                                                       glong l,
                                                       mp_rnd_t rnd);
```

FIXME

**jlrec** : a **NcmMpsfSBesselRecur**

**l** : FIXME

**rnd** : FIXME

**ncm\_mpsf\_sbessel\_recur\_write ()**

```
void                ncm_mpsf_sbessel_recur_write    (NcmMpsfSBesselRecur *jlrec,
                                                       FILE *f);
```

FIXME

**jlrec** : a **NcmMpsfSBesselRecur**

**f** : FIXME

**2.18.4 Spherical Bessel Integral**

Spherical Bessel Integral — Spherical bessel integrals implementation with support for multiple precision calculation



```

NcmMpsfSBesselIntSpline * ncm_mpsf_sbessel_jl_xj_integrate_spline_new
                                (gulong prec,
                                 NcmGrid *x,
                                 NcmGrid *k);
NcmMpsfSBesselIntSpline * ncm_mpsf_sbessel_jl_xj_integrate_spline_new_from_sections
                                (gulong prec,
                                 NcmGridSection *x_secs,
                                 NcmGridSection *k_secs);
NcmMpsfSBesselIntSpline * ncm_mpsf_sbessel_jl_xj_integrate_spline_cached_new
                                (gulong prec,
                                 glong l,
                                 NcmGridSection *x_secs,
                                 NcmGridSection *k_secs,
                                 mp_rnd_t rnd);
NcmMpsfSBesselIntSpline * ncm_mpsf_sbessel_jl_xj_integrate_spline_load
                                (gchar *filename);
void ncm_mpsf_sbessel_jl_xj_integrate_spline_prepare
                                (NcmMpsfSBesselIntSpline *int_jlsp,
                                 glong l,
                                 mp_rnd_t rnd);
void ncm_mpsf_sbessel_jl_xj_integrate_spline_prepare_new
                                (NcmMpsfSBesselIntSpline *int_jlsp,
                                 glong l,
                                 mp_rnd_t rnd);
void ncm_mpsf_sbessel_jl_xj_integrate_spline_save
                                (NcmMpsfSBesselIntSpline *int_jlsp,
                                 gchar *filename);
void ncm_mpsf_sbessel_jl_xj_integrate_spline_next
                                (NcmMpsfSBesselIntSpline *int_jlsp,
                                 mp_rnd_t rnd);
void ncm_mpsf_sbessel_jl_xj_integrate_spline_previous
                                (NcmMpsfSBesselIntSpline *int_jlsp,
                                 mp_rnd_t rnd);
void ncm_mpsf_sbessel_jl_xj_integrate_spline_goto
                                (NcmMpsfSBesselIntSpline *int_jlsp,
                                 gulong l,
                                 mp_rnd_t rnd);

```

## Description

FIXME

## Details

### struct NcmMpsfSBesselIntegRecur

```

struct NcmMpsfSBesselIntegRecur {
};

```

FIXME

### struct NcmMpsfSBesselIntSpline

```

struct NcmMpsfSBesselIntSpline {
};

```

FIXME

### NCM\_MPSF\_SBESSEL\_INT\_MAP()

```
#define NCM_MPSF_SBESSEL_INT_MAP(int_jlspline,xi,kj) ((int_jlspline)->map_ij2r[((int_jlspline)->row * kj + xi)])
```

### ncm\_mpsf\_sbessel\_jl\_xj\_integral()

```
void ncm_mpsf_sbessel_jl_xj_integral(gint l,
                                     gint j,
                                     gdouble x,
                                     mpfr_t res,
                                     mp_rnd_t rnd);
```

FIXME

***l*** : FIXME

***j*** : FIXME

***x*** : FIXME

***res*** : FIXME

***rnd*** : FIXME

### ncm\_mpsf\_sbessel\_jl\_xj\_integral\_q()

```
void ncm_mpsf_sbessel_jl_xj_integral_q(gint l,
                                         gint j,
                                         mpq_t q,
                                         mpfr_t res,
                                         mp_rnd_t rnd);
```

FIXME

***l*** : FIXME

***j*** : FIXME

***q*** : FIXME

***res*** : FIXME

***rnd*** : FIXME

### ncm\_mpsf\_sbessel\_integrate()

```
gdouble ncm_mpsf_sbessel_integrate(NcmMpsfSBesselIntSpline * ↵
    int_jlspline,
    NcmSpline *s,
    gint l,
    guint ki,
    guint xi,
    gint diff);
```

FIXME



**int\_jlspline** : a **NcmMpsfSBesselIntSpline**

**s** : a **NcmSpline**

**l** : FIXME

**ki** : FIXME

**xi** : FIXME

**diff** : FIXME

**Returns** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_new ()**

```
NcmMpsfSBesselIntegRecur * ncm_mpsf_sbessel_jl_xj_integral_recur_new
                             (gulong prec,
                              NcmMpsfSBesselRecur *jlrec);
```

FIXME

**prec** : FIXME

**jlrec** : a **NcmMpsfSBesselRecur**

**Returns** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_read ()**

```
NcmMpsfSBesselIntegRecur * ncm_mpsf_sbessel_jl_xj_integral_recur_read
                             (FILE *f);
```

FIXME

**f** : FIXME

**Returns** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_free ()**

```
void                          ncm_mpsf_sbessel_jl_xj_integral_recur_free
                             (NcmMpsfSBesselIntegRecur *xnjlrec) ↵
                             ;
```

FIXME

**xnjlrec** : a **NcmMpsfSBesselIntegRecur**

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_next ()**

```
void                          ncm_mpsf_sbessel_jl_xj_integral_recur_next
                             (NcmMpsfSBesselIntegRecur *xnjlrec,
                              mp_rnd_t rnd);
```

FIXME

**xnjlrec** : a **NcmMpsfSBesselIntegRecur**

**rnd** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_previous ()**

```
void                ncm_mpsf_sbessel_jl_xj_integral_recur_previous
                    (NcmMpsfSBesselIntegRecur *xnjlrec,
                     mp_rnd_t rnd);
```

FIXME

**xnjlrec** : a NcmMpsfSBesselIntegRecur

**rnd** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_goto ()**

```
void                ncm_mpsf_sbessel_jl_xj_integral_recur_goto
                    (NcmMpsfSBesselIntegRecur *xnjlrec,
                     glong l,
                     mp_rnd_t rnd);
```

FIXME

**xnjlrec** : a NcmMpsfSBesselIntegRecur

**l** : FIXME

**rnd** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_calc\_djl ()**

```
void                ncm_mpsf_sbessel_jl_xj_integral_recur_calc_djl
                    (NcmMpsfSBesselIntegRecur *xnjlrec,
                     glong n,
                     mpfr_ptr rule,
                     mp_rnd_t rnd);
```

FIXME

**xnjlrec** : a NcmMpsfSBesselIntegRecur

**n** : FIXME

**rule** : FIXME

**rnd** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_calc\_d2jl ()**

```
void                ncm_mpsf_sbessel_jl_xj_integral_recur_calc_d2jl
                    (NcmMpsfSBesselIntegRecur *xnjlrec,
                     glong n,
                     mpfr_ptr rule,
                     mp_rnd_t rnd);
```

FIXME

**xnjlrec** : a NcmMpsfSBesselIntegRecur

**n** : FIXME

**rule** : FIXME

**rnd** : FIXME

---

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_write ()**

```
void                ncm_mpsf_sbessel_jl_xj_integral_recur_write
                    (NcmMpsfSBesselIntegRecur *xnjlrec,
                     FILE *f);
```

FIXME

***xnjlrec*** : a NcmMpsfSBesselIntegRecur

***f*** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integral\_a\_b ()**

```
void                ncm_mpsf_sbessel_jl_xj_integral_a_b (NcmMpsfSBesselIntSpline * ↵
    int_jlspline,
                    guint ki,
                    guint xi,
                    mp_rnd_t rnd);
```

FIXME

***int\_jlspline*** : a NcmMpsfSBesselIntSpline

***ki*** : FIXME

***xi*** : FIXME

***rnd*** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_new ()**

```
NcmMpsfSBesselIntSpline * ncm_mpsf_sbessel_jl_xj_integrate_spline_new
                           (gulong prec,
                            NcmGrid *x,
                            NcmGrid *k);
```

FIXME

***prec*** : FIXME

***x*** : a NcmGrid

***k*** : a NcmGrid

**Returns** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_new\_from\_sections ()**

```
NcmMpsfSBesselIntSpline * ncm_mpsf_sbessel_jl_xj_integrate_spline_new_from_sections
                           (gulong prec,
                            NcmGridSection *x_secs,
                            NcmGridSection *k_secs);
```

FIXME

***prec*** : FIXME

***x\_secs*** : a NcmGridSection

***k\_secs*** : a NcmGridSection

**Returns** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_cached\_new ()**

```
NcmMpsfSBesselIntSpline * ncm_mpsf_sbessel_jl_xj_integrate_spline_cached_new
                                                                    (gulong prec,
                                                                    glong l,
                                                                    NcmGridSection *x_secs,
                                                                    NcmGridSection *k_secs,
                                                                    mp_rnd_t rnd);
```

FIXME

**prec** : FIXME

**l** : FIXME

**x\_secs** : a **NcmGridSection**

**k\_secs** : a **NcmGridSection**

**rnd** : FIXME

**Returns** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_load ()**

```
NcmMpsfSBesselIntSpline * ncm_mpsf_sbessel_jl_xj_integrate_spline_load
                                                                    (gchar *filename);
```

FIXME

**filename** : FIXME

**Returns** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_prepare ()**

```
void                                ncm_mpsf_sbessel_jl_xj_integrate_spline_prepare
                                                                    (NcmMpsfSBesselIntSpline * ↵
                                                                    int_jlspline,
                                                                    glong l,
                                                                    mp_rnd_t rnd);
```

FIXME

**int\_jlspline** : a **NcmMpsfSBesselIntSpline**

**l** : FIXME

**rnd** : FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_prepare\_new ()**

```
void                                ncm_mpsf_sbessel_jl_xj_integrate_spline_prepare_new
                                                                    (NcmMpsfSBesselIntSpline * ↵
                                                                    int_jlspline,
                                                                    glong l,
                                                                    mp_rnd_t rnd);
```

FIXME

***int\_jlspline***: a **NcmMpsfSBesselIntSpline**

***l***: FIXME

***rnd***: FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_save ()**

```
void                ncm_mpsf_sbessel_jl_xj_integrate_spline_save
                    (NcmMpsfSBesselIntSpline * ←
                     int_jlspline,
                     gchar *filename);
```

FIXME

***int\_jlspline***: a **NcmMpsfSBesselIntSpline**

***filename***: FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_next ()**

```
void                ncm_mpsf_sbessel_jl_xj_integrate_spline_next
                    (NcmMpsfSBesselIntSpline * ←
                     int_jlspline,
                     mp_rnd_t rnd);
```

FIXME

***int\_jlspline***: a **NcmMpsfSBesselIntSpline**

***rnd***: FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_previous ()**

```
void                ncm_mpsf_sbessel_jl_xj_integrate_spline_previous
                    (NcmMpsfSBesselIntSpline * ←
                     int_jlspline,
                     mp_rnd_t rnd);
```

FIXME

***int\_jlspline***: a **NcmMpsfSBesselIntSpline**

***rnd***: FIXME

**ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_goto ()**

```
void                ncm_mpsf_sbessel_jl_xj_integrate_spline_goto
                    (NcmMpsfSBesselIntSpline * ←
                     int_jlspline,
                     gulong l,
                     mp_rnd_t rnd);
```

FIXME

***int\_jlspline***: a **NcmMpsfSBesselIntSpline**

***l***: FIXME

***rnd***: FIXME

## 2.18.5 Spherical Bessel -- Double Precision

Spherical Bessel -- Double Precision — Double precision spherical bessel implementation

### Synopsis

```

struct          NcmSFSBesselRecur;
NcmSFSBesselRecur * ncm_sf_sbessel_recur_new          (NcmGrid *x_grid);
NcmSFSBesselRecur * ncm_sf_sbessel_recur_read          (FILE *f);
void            ncm_sf_sbessel_recur_free              (NcmSFSBesselRecur *jlrec,
                                                       gboolean free_grid);
void            ncm_sf_sbessel_recur_set                (NcmSFSBesselRecur *jlrec,
                                                       gulong l);
void            ncm_sf_sbessel_recur_next                (NcmSFSBesselRecur *jlrec);
void            ncm_sf_sbessel_recur_previous            (NcmSFSBesselRecur *jlrec);
void            ncm_sf_sbessel_recur_goto                (NcmSFSBesselRecur *jlrec,
                                                       gulong l);
void            ncm_sf_sbessel_taylor_coeff_jl_jlp1      (NcmSFSBesselRecur *jlrec,
                                                       guint n,
                                                       gdouble *djl,
                                                       gdouble *djlp1);
void            ncm_sf_sbessel_recur_write                (NcmSFSBesselRecur *jlrec,
                                                       FILE *f);
gdouble         ncm_sf_sbessel                            (gulong l,
                                                       gdouble x);
void            ncm_sf_sbessel_taylor                    (gulong l,
                                                       gdouble x,
                                                       gdouble *djl);
void            ncm_sf_sbessel_deriv                    (gulong l,
                                                       gdouble x,
                                                       gdouble jl,
                                                       gdouble jlp1,
                                                       gdouble *djl);
NcmSpline *      ncm_sf_sbessel_spline                    (gulong l,
                                                       gdouble xi,
                                                       gdouble xf,
                                                       gdouble reltol);

```

### Description

FIXME

### Details

#### struct NcmSFSBesselRecur

```

struct NcmSFSBesselRecur {
};

```

FIXME

#### ncm\_sf\_sbessel\_recur\_new ()

```

NcmSFSBesselRecur * ncm_sf_sbessel_recur_new          (NcmGrid *x_grid);

```

FIXME

***x\_grid***: a **NcmGrid**

**Returns** : FIXME

**ncm\_sf\_sbessel\_recur\_read ()**

```
NcmSFSBesselRecur * ncm_sf_sbessel_recur_read      (FILE *f);
```

FIXME

***f*** : FIXME

**Returns** : FIXME

**ncm\_sf\_sbessel\_recur\_free ()**

```
void                ncm_sf_sbessel_recur_free      (NcmSFSBesselRecur *jlrec,  
                                                    gboolean free_grid);
```

FIXME

***jlrec***: a **NcmSFSBesselRecur**

***free\_grid***: FIXME

**ncm\_sf\_sbessel\_recur\_set ()**

```
void                ncm_sf_sbessel_recur_set      (NcmSFSBesselRecur *jlrec,  
                                                    glong l);
```

FIXME

***jlrec***: a **NcmSFSBesselRecur**

***l*** : FIXME

**ncm\_sf\_sbessel\_recur\_next ()**

```
void                ncm_sf_sbessel_recur_next     (NcmSFSBesselRecur *jlrec);
```

FIXME

***jlrec***: a **NcmSFSBesselRecur**

**ncm\_sf\_sbessel\_recur\_previous ()**

```
void                ncm_sf_sbessel_recur_previous (NcmSFSBesselRecur *jlrec);
```

FIXME

***jlrec***: a **NcmSFSBesselRecur**

---

**ncm\_sf\_sbessel\_recur\_goto ()**

```
void                ncm_sf_sbessel_recur_goto      (NcmSFSBesselRecur *jlrec,
                                                    glong l);
```

FIXME

**jlrec** : a **NcmSFSBesselRecur**

**l** : FIXME

**ncm\_sf\_sbessel\_taylor\_coeff\_jl\_jlp1 ()**

```
void                ncm_sf_sbessel_taylor_coeff_jl_jlp1 (NcmSFSBesselRecur *jlrec,
                                                         guint n,
                                                         gdouble *dj1,
                                                         gdouble *djlp1);
```

FIXME

**jlrec** : a **NcmSFSBesselRecur**

**n** : FIXME

**dj1** : FIXME

**djlp1** : FIXME

**ncm\_sf\_sbessel\_recur\_write ()**

```
void                ncm_sf_sbessel_recur_write      (NcmSFSBesselRecur *jlrec,
                                                    FILE *f);
```

FIXME

**jlrec** : a **NcmSFSBesselRecur**

**f** : FIXME

**ncm\_sf\_sbessel ()**

```
gdouble            ncm_sf_sbessel                  (gulong l,
                                                    gdouble x);
```

FIXME

**l** : FIXME

**x** : FIXME

**Returns** : FIXME



**ncm\_sf\_sbessel\_taylor ()**

```
void          ncm_sf_sbessel_taylor      (gulong l,
                                           gdouble x,
                                           gdouble *dj1);
```

FIXME

***l*** : FIXME

***x*** : FIXME

***dj1*** : FIXME

**ncm\_sf\_sbessel\_deriv ()**

```
void          ncm_sf_sbessel_deriv      (gulong l,
                                           gdouble x,
                                           gdouble j1,
                                           gdouble j1p1,
                                           gdouble *dj1);
```

FIXME

***l*** : FIXME

***x*** : FIXME

***j1*** : FIXME

***j1p1*** : FIXME

***dj1*** : FIXME

**ncm\_sf\_sbessel\_spline ()**

```
NcmSpline *   ncm_sf_sbessel_spline      (gulong l,
                                           gdouble xi,
                                           gdouble xf,
                                           gdouble reltol);
```

FIXME

***l*** : FIXME

***xi*** : FIXME

***xf*** : FIXME

***reltol*** : FIXME

***Returns*** : FIXME. *[transfer full]*

**2.18.6 Spherical Bessel Integral -- Double Precision**

Spherical Bessel Integral -- Double Precision — Double precision spherical bessel integrals implementation



```

void                ncm_sf_sbessel_jl_xj_integral_a_b      (NcmSFSphericalBesselIntSpline *in
                                                         gdouble x0,
                                                         gdouble x1,
                                                         gdouble w,
                                                         gdouble *xnjl_rules,
                                                         gdouble *xndjl_rules,
                                                         gdouble *xnd2jl_rules);

gdouble            ncm_sf_sbessel_jl_xj_integrate_spline_eval
                                                         (NcmSFSphericalBesselIntSpline *in
                                                         gint d,
                                                         gdouble x);

gdouble            ncm_sf_sbessel_jl_xj_integral_spline
                                                         (NcmSFSphericalBesselIntSpline *in
                                                         NcmSpline *s0,
                                                         NcmSpline *s1,
                                                         NcmSpline *s2,
                                                         gdouble w);

```

**Description**

FIXME

**Details****struct NcmSFSphericalBesselIntegRecur**

```

struct NcmSFSphericalBesselIntegRecur {
};

```

FIXME

**struct NcmSFSphericalBesselIntSpline**

```

struct NcmSFSphericalBesselIntSpline {
};

```

FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral ()**

```

gdouble            ncm_sf_sbessel_jl_xj_integral      (gint l,
                                                         gint j,
                                                         gdouble x);

```

FIXME

***l*** : FIXME

***j*** : FIXME

***x*** : FIXME

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_new ()**

```
NcmSFSphericalBesselIntegRecur * ncm_sf_sbessel_jl_xj_integral_recur_new
                                   (NcmSFSBesselRecur *jlrec,
                                   NcmGrid *x_grid);
```

FIXME

**jlrec** : a **NcmSFSBesselRecur**

**x\_grid** : a **NcmGrid**

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_new\_from\_section ()**

```
NcmSFSphericalBesselIntegRecur * ncm_sf_sbessel_jl_xj_integral_recur_new_from_section
                                   (NcmGridSection *x_sec);
```

FIXME

**x\_sec** : a **NcmGridSection**

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_load ()**

```
NcmSFSphericalBesselIntegRecur * ncm_sf_sbessel_jl_xj_integral_recur_load
                                   (gchar *filename);
```

FIXME

**filename** : FIXME

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_read ()**

```
NcmSFSphericalBesselIntegRecur * ncm_sf_sbessel_jl_xj_integral_recur_read
                                   (FILE *f);
```

FIXME

**f** : FIXME

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_cached\_new ()**

```
NcmSFSphericalBesselIntegRecur * ncm_sf_sbessel_jl_xj_integral_recur_cached_new
                                   (glong l,
                                   NcmGridSection *x_sec);
```

FIXME

**l** : FIXME

**x\_sec** : a **NcmGridSection**

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_free ()**

```
void                ncm_sf_sbessel_jl_xj_integral_recur_free
                                (NcmSFSphericalBesselIntegRecur * ↔
                                xnjlrec,
                                gboolean free_grid);
```

FIXME

**xnjlrec** : a **NcmSFSphericalBesselIntegRecur**

**free\_grid** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_set ()**

```
void                ncm_sf_sbessel_jl_xj_integral_recur_set
                                (NcmSFSphericalBesselIntegRecur * ↔
                                xnjlrec,
                                glong l);
```

FIXME

**xnjlrec** : a **NcmSFSphericalBesselIntegRecur**

**l** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_next ()**

```
glong                ncm_sf_sbessel_jl_xj_integral_recur_next
                                (NcmSFSphericalBesselIntegRecur * ↔
                                xnjlrec);
```

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_previous ()**

```
glong                ncm_sf_sbessel_jl_xj_integral_recur_previous
                                (NcmSFSphericalBesselIntegRecur * ↔
                                xnjlrec);
```

FIXME

**xnjlrec** : a **NcmSFSphericalBesselIntegRecur**

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_goto ()**

```
glong                ncm_sf_sbessel_jl_xj_integral_recur_goto
                                (NcmSFSphericalBesselIntegRecur * ↔
                                xnjlrec,
                                glong l);
```

FIXME

**xnjlrec** : a **NcmSFSphericalBesselIntegRecur**

**l** : FIXME

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_save ()**

```
void                ncm_sf_sbessel_jl_xj_integral_recur_save
                                (NcmSFSphericalBesselIntegRecur * ↵
                                xnjlrec,
                                gchar *filename);
```

FIXME

**xnjlrec** : a **NcmSFSphericalBesselIntegRecur**

**filename** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_write ()**

```
void                ncm_sf_sbessel_jl_xj_integral_recur_write
                                (NcmSFSphericalBesselIntegRecur * ↵
                                xnjlrec,
                                FILE *f);
```

FIXME

**xnjlrec** : a **NcmSFSphericalBesselIntegRecur**

**f** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_new ()**

```
NcmSFSphericalBesselIntSpline * ncm_sf_sbessel_jl_xj_integrate_spline_new
                                (NcmSFSphericalBesselIntegRecur * ↵
                                xnjlrec,
                                gboolean init);
```

FIXME

**xnjlrec** : a **NcmSFSphericalBesselIntegRecur**

**init** : FIXME

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_cached\_new ()**

```
NcmSFSphericalBesselIntSpline * ncm_sf_sbessel_jl_xj_integrate_spline_cached_new
                                (glong l,
                                NcmGridSection *x_sec,
                                gboolean init);
```

FIXME

**l** : FIXME

**x\_sec** : a **NcmGridSection**

**init** : FIXME

**Returns** : FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_reset ()**

```
void          ncm_sf_sbessel_jl_xj_integrate_spline_reset
               (NcmSFSphericalBesselIntSpline * ↔
                int_jlspline);
```

FIXME

**int\_jlspline**: a **NcmSFSphericalBesselIntSpline****ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_set ()**

```
void          ncm_sf_sbessel_jl_xj_integrate_spline_set
               (NcmSFSphericalBesselIntSpline * ↔
                int_jlspline,
                glong l);
```

FIXME

**int\_jlspline**: a **NcmSFSphericalBesselIntSpline****l**: FIXME**ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_next ()**

```
void          ncm_sf_sbessel_jl_xj_integrate_spline_next
               (NcmSFSphericalBesselIntSpline * ↔
                int_jlspline);
```

FIXME

**int\_jlspline**: a **NcmSFSphericalBesselIntSpline****ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_previous ()**

```
void          ncm_sf_sbessel_jl_xj_integrate_spline_previous
               (NcmSFSphericalBesselIntSpline * ↔
                int_jlspline);
```

FIXME

**int\_jlspline**: a **NcmSFSphericalBesselIntSpline****ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_goto ()**

```
void          ncm_sf_sbessel_jl_xj_integrate_spline_goto
               (NcmSFSphericalBesselIntSpline * ↔
                int_jlspline,
                glong l);
```

FIXME

**int\_jlspline**: a **NcmSFSphericalBesselIntSpline****l**: FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_a\_b ()**

```
void          ncm_sf_sbessel_jl_xj_integral_a_b  (NcmSFSphericalBesselIntSpline * ←
    int_jlspline,
                                                    gdouble x0,
                                                    gdouble x1,
                                                    gdouble w,
                                                    gdouble *xnjl_rules,
                                                    gdouble *xndjl_rules,
                                                    gdouble *xnd2jl_rules);
```

FIXME

**int\_jlspline**: a **NcmSFSphericalBesselIntSpline**

**x0**: FIXME

**x1**: FIXME

**w**: FIXME

**xnjl\_rules**: FIXME

**xndjl\_rules**: FIXME

**xnd2jl\_rules**: FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_eval ()**

```
gdouble          ncm_sf_sbessel_jl_xj_integrate_spline_eval
                                                    (NcmSFSphericalBesselIntSpline * ←
    int_jlspline,
    gint d,
    gdouble x);
```

FIXME

**int\_jlspline**: a **NcmSFSphericalBesselIntSpline**

**d**: FIXME

**x**: FIXME

**Returns**: FIXME

**ncm\_sf\_sbessel\_jl\_xj\_integral\_spline ()**

```
gdouble          ncm_sf_sbessel_jl_xj_integral_spline
                                                    (NcmSFSphericalBesselIntSpline * ←
    int_jlspline,
    NcmSpline *s0,
    NcmSpline *s1,
    NcmSpline *s2,
    gdouble w);
```

FIXME

**int\_jlspline**: a **NcmSFSphericalBesselIntSpline**

**s0**: a **NcmSpline**



**s1** : a **NcmSpline**

**s2** : a **NcmSpline**

**w** : FIXME

**Returns** : FIXME

## 2.19 Data Objects

### 2.19.1 Data Abstract Class

Data Abstract Class — Base class for implementing data objects

#### Synopsis

```

struct          NcmDataClass;
struct          NcmData;
NcmData *       ncm_data_ref          (NcmData *data);
void           ncm_data_free          (NcmData *data);
void           ncm_data_clear         (NcmData **data);
NcmData *       ncm_data_dup          (NcmData *data);
quint          ncm_data_get_length   (NcmData *data);
quint          ncm_data_get_dof       (NcmData *data);
void           ncm_data_set_init      (NcmData *data);
void           ncm_data_prepare       (NcmData *data,
                                       NcmMSet *mset);
void           ncm_data_resample      (NcmData *data,
                                       NcmMSet *mset);
void           ncm_data_leastquares_f (NcmData *data,
                                       NcmMSet *mset,
                                       NcmVector *f);
void           ncm_data_leastquares_J (NcmData *data,
                                       NcmMSet *mset,
                                       NcmMatrix *J);
void           ncm_data_leastquares_f_J (NcmData *data,
                                       NcmMSet *mset,
                                       NcmVector *f,
                                       NcmMatrix *J);
void           ncm_data_m2lnL_val     (NcmData *data,
                                       NcmMSet *mset,
                                       gdouble *m2lnL);
void           ncm_data_m2lnL_grad    (NcmData *data,
                                       NcmMSet *mset,
                                       NcmVector *grad);
void           ncm_data_m2lnL_val_grad (NcmData *data,
                                       NcmMSet *mset,
                                       gdouble *m2lnL,
                                       NcmVector *grad);
gchar *        ncm_data_get_desc     (NcmData *data);

```

#### Object Hierarchy

```

GObject
+----NcmData

```

```
+----NcmDataGaussDiag
+----NcmDataGauss
+----NcDataClusterNCount
+----NcmDataPoisson
+----NcmDataGaussCov
+----NcmDataDist1d
```

Properties

"name" gchar\* : Read

Description

FIXME

Details

struct NcmDataClass

```
struct NcmDataClass {
};
```

struct NcmData

```
struct NcmData;
```

ncm\_data\_ref ()

```
NcmData * ncm_data_ref (NcmData *data);
```

FIXME

**data** : a **NcmData**.

**Returns** : FIXME. *[transfer full]*

ncm\_data\_free ()

```
void ncm_data_free (NcmData *data);
```

FIXME

**data** : a **NcmData**.

ncm\_data\_clear ()

```
void ncm_data_clear (NcmData **data);
```

FIXME

**data** : a **NcmData**.

---

**ncm\_data\_dup ()**

```
NcmData *          ncm_data_dup          (NcmData *data);
```

FIXME

**data** : a **NcmData**.**Returns** : FIXME. *[transfer full]***ncm\_data\_get\_length ()**

```
guint          ncm_data_get_length      (NcmData *data);
```

FIXME

**data** : a **NcmData**.**Returns** : FIXME**ncm\_data\_get\_dof ()**

```
guint          ncm_data_get_dof        (NcmData *data);
```

Calculates the degrees of freedom associated with the data.

**data** : a **NcmData**.**Returns** : FIXME**ncm\_data\_set\_init ()**

```
void          ncm_data_set_init        (NcmData *data);
```

FIXME

**data** : a **NcmData**.**ncm\_data\_prepare ()**

```
void          ncm_data_prepare        (NcmData *data,  
                                       NcmMSet *mset);
```

FIXME

**data** : a **NcmData**.**mset** : a **NcmMSet**.**ncm\_data\_resample ()**

```
void          ncm_data_resample        (NcmData *data,  
                                       NcmMSet *mset);
```

FIXME

**data** : a **NcmData**.**mset** : a **NcmMSet**.

**ncm\_data\_leastquares\_f ()**

```
void                ncm_data_leastquares_f      (NcmData *data,  
                                                NcmMSet *mset,  
                                                NcmVector *f);
```

FIXME

**data** : a **NcmData**.**mset** : a **NcmMSet**.**f** : FIXME**ncm\_data\_leastquares\_J ()**

```
void                ncm_data_leastquares_J     (NcmData *data,  
                                                NcmMSet *mset,  
                                                NcmMatrix *J);
```

FIXME

**data** : a **NcmData**.**mset** : a **NcmMSet**.**J** : FIXME**ncm\_data\_leastquares\_f\_J ()**

```
void                ncm_data_leastquares_f_J   (NcmData *data,  
                                                NcmMSet *mset,  
                                                NcmVector *f,  
                                                NcmMatrix *J);
```

FIXME

**data** : a **NcmData**.**mset** : a **NcmMSet**.**f** : FIXME**J** : FIXME**ncm\_data\_m2lnL\_val ()**

```
void                ncm_data_m2lnL_val        (NcmData *data,  
                                                NcmMSet *mset,  
                                                gdouble *m2lnL);
```

FIXME

**data** : a **NcmData**.**mset** : a **NcmMSet**.**m2lnL** : FIXME. *[out]*

**ncm\_data\_m2lnL\_grad ()**

```
void          ncm_data_m2lnL_grad          (NcmData *data,
                                             NcmMSet *mset,
                                             NcmVector *grad);
```

FIXME

**data** : a **NcmData**.

**mset** : a **NcmMSet**.

**grad** : FIXME

**ncm\_data\_m2lnL\_val\_grad ()**

```
void          ncm_data_m2lnL_val_grad      (NcmData *data,
                                             NcmMSet *mset,
                                             gdouble *m2lnL,
                                             NcmVector *grad);
```

FIXME

**data** : a **NcmData**.

**mset** : a **NcmMSet**.

**m2lnL** : FIXME. *[out]*

**grad** : FIXME

**ncm\_data\_get\_desc ()**

```
gchar *       ncm_data_get_desc            (NcmData *data);
```

FIXME

**data** : a **NcmData**.

**Returns** : FIXME. *[transfer none]*

**Property Details**

**The "name" property**

"name"	gchar*	: Read
--------	--------	--------

Data type name.

Default value: NULL

**2.19.2 Data Set**

Data Set — Object representing a set of NcmData objects

## Synopsis

```

struct          NcmDatasetClass;
struct          NcmDataset;
NcmDataset *    ncm_dataset_new                (void);
NcmDataset *    ncm_dataset_dup                (NcmDataset *dset);
NcmDataset *    ncm_dataset_ref                (NcmDataset *dset);
NcmDataset *    ncm_dataset_copy               (NcmDataset *dset);
void            ncm_dataset_free               (NcmDataset *dset);
void            ncm_dataset_clear               (NcmDataset **dset);
guint           ncm_dataset_get_length          (NcmDataset *dset);
guint           ncm_dataset_get_n              (NcmDataset *dset);
guint           ncm_dataset_get_dof            (NcmDataset *dset);
gboolean        ncm_dataset_all_init           (NcmDataset *dset);
void            ncm_dataset_append_data        (NcmDataset *dset,
                                                NcmData *data);
NcmData *       ncm_dataset_get_data           (NcmDataset *dset,
                                                guint n);
NcmData *       ncm_dataset_peek_data          (NcmDataset *dset,
                                                guint n);
guint           ncm_dataset_get_ndata          (NcmDataset *dset);
void            ncm_dataset_resample           (NcmDataset *dset,
                                                NcmMSet *mset);
void            ncm_dataset_log_info           (NcmDataset *dset);
gboolean        ncm_dataset_has_least_squares_f (NcmDataset *dset);
gboolean        ncm_dataset_has_least_squares_J (NcmDataset *dset);
gboolean        ncm_dataset_has_least_squares_f_J (NcmDataset *dset);
gboolean        ncm_dataset_has_m2lnL_val      (NcmDataset *dset);
gboolean        ncm_dataset_has_m2lnL_grad     (NcmDataset *dset);
gboolean        ncm_dataset_has_m2lnL_val_grad (NcmDataset *dset);
void            ncm_dataset_least_squares_f    (NcmDataset *dset,
                                                NcmMSet *mset,
                                                NcmVector *f);
void            ncm_dataset_least_squares_J    (NcmDataset *dset,
                                                NcmMSet *mset,
                                                NcmMatrix *J);
void            ncm_dataset_least_squares_f_J  (NcmDataset *dset,
                                                NcmMSet *mset,
                                                NcmVector *f,
                                                NcmMatrix *J);
void            ncm_dataset_m2lnL_val          (NcmDataset *dset,
                                                NcmMSet *mset,
                                                gdouble *m2lnL);
void            ncm_dataset_m2lnL_grad         (NcmDataset *dset,
                                                NcmMSet *mset,
                                                NcmVector *grad);
void            ncm_dataset_m2lnL_val_grad     (NcmDataset *dset,
                                                NcmMSet *mset,
                                                gdouble *m2lnL,
                                                NcmVector *grad);

```

## Object Hierarchy

```

GObject
+----NcmDataset

```

## Description

FIXME

## Details

### struct NcmDatasetClass

```
struct NcmDatasetClass {  
};
```

### struct NcmDataset

```
struct NcmDataset;
```

### ncm\_dataset\_new ()

```
NcmDataset *      ncm_dataset_new      (void);
```

FIXME

**Returns :** FIXME

### ncm\_dataset\_dup ()

```
NcmDataset *      ncm_dataset_dup      (NcmDataset *dset);
```

Duplicates the object and all of its content.

**dset :** pointer to type defined by **NcmDataset**

**Returns :** FIXME. *[transfer full]*

### ncm\_dataset\_ref ()

```
NcmDataset *      ncm_dataset_ref      (NcmDataset *dset);
```

FIXME

**dset :** pointer to type defined by **NcmDataset**

**Returns :** FIXME. *[transfer full]*

### ncm\_dataset\_copy ()

```
NcmDataset *      ncm_dataset_copy     (NcmDataset *dset);
```

Duplicates the object getting a reference of its content.

**dset :** pointer to type defined by **NcmDataset**

**Returns :** FIXME. *[transfer full]*

---

**ncm\_dataset\_free ()**

```
void                ncm_dataset_free                (NcmDataset *dset);
```

FIXME

**dset** : pointer to type defined by **NcmDataset**

**ncm\_dataset\_clear ()**

```
void                ncm_dataset_clear                (NcmDataset **dset);
```

FIXME

**dset** : pointer to type defined by **NcmDataset**

**ncm\_dataset\_get\_length ()**

```
guint                ncm_dataset_get_length                (NcmDataset *dset);
```

FIXME

**dset** : pointer to type defined by **NcmDataset**

**Returns** : number of NcmData objects in the set

**ncm\_dataset\_get\_n ()**

```
guint                ncm_dataset_get_n                (NcmDataset *dset);
```

Calculate the total number of data set points

**dset** : pointer to type defined by **NcmDataset**

**Returns** : FIXME

**ncm\_dataset\_get\_dof ()**

```
guint                ncm_dataset_get_dof                (NcmDataset *dset);
```

Calculate the total degrees of freedom associated with all **NcmData** objects.

**dset** : pointer to type defined by **NcmDataset**

**Returns** : FIXME

**ncm\_dataset\_all\_init ()**

```
gboolean                ncm_dataset_all_init                (NcmDataset *dset);
```

FIXME

**dset** : pointer to type defined by **NcmDataset**

**Returns** : FIXME



**ncm\_dataset\_append\_data ()**

```
void                ncm_dataset_append_data      (NcmDataset *dset,  
                                                NcmData *data);
```

FIXME

**dset** : pointer to type defined by **NcmDataset****data** : **NcmData** object to be appended to **NcmDataset****Returns** : FIXME**ncm\_dataset\_get\_data ()**

```
NcmData *          ncm_dataset_get_data        (NcmDataset *dset,  
                                                guint n);
```

FIXME

**dset** : pointer to type defined by **NcmDataset****n** : FIXME**Returns** : FIXME. *[transfer full]***ncm\_dataset\_peek\_data ()**

```
NcmData *          ncm_dataset_peek_data      (NcmDataset *dset,  
                                                guint n);
```

FIXME

**dset** : pointer to type defined by **NcmDataset****n** : FIXME**Returns** : FIXME. *[transfer none]***ncm\_dataset\_get\_ndata ()**

```
guint              ncm_dataset_get_ndata      (NcmDataset *dset);
```

**ncm\_dataset\_resample ()**

```
void                ncm_dataset_resample      (NcmDataset *dset,  
                                                NcmMSet *mset);
```

FIXME

**dset** : a **NcmDataset**.**mset** : a **NcmMSet**.

**ncm\_dataset\_log\_info ()**

```
void                ncm_dataset_log_info                (NcmDataset *dset);
```

FIXME

**dset** : a **NcmDataset****ncm\_dataset\_has\_leastquares\_f ()**

```
gboolean            ncm_dataset_has_leastquares_f      (NcmDataset *dset);
```

FIXME

**dset** : a **NcmDataset****Returns** : FIXME**ncm\_dataset\_has\_leastquares\_J ()**

```
gboolean            ncm_dataset_has_leastquares_J      (NcmDataset *dset);
```

FIXME

**dset** : a **NcmDataset****Returns** : FIXME**ncm\_dataset\_has\_leastquares\_f\_J ()**

```
gboolean            ncm_dataset_has_leastquares_f_J    (NcmDataset *dset);
```

FIXME

**dset** : a **NcmDataset****Returns** : FIXME**ncm\_dataset\_has\_m2lnL\_val ()**

```
gboolean            ncm_dataset_has_m2lnL_val          (NcmDataset *dset);
```

FIXME

**dset** : a **NcmDataset****Returns** : FIXME**ncm\_dataset\_has\_m2lnL\_grad ()**

```
gboolean            ncm_dataset_has_m2lnL_grad         (NcmDataset *dset);
```

FIXME

**dset** : a **NcmDataset****Returns** : FIXME

**ncm\_dataset\_has\_m2lnL\_val\_grad ()**

```
gboolean          ncm_dataset_has_m2lnL_val_grad      (NcmDataset *dset);
```

FIXME

**dset** : a **NcmDataset**

**Returns** : FIXME

**ncm\_dataset\_leastquares\_f ()**

```
void              ncm_dataset_leastquares_f          (NcmDataset *dset,
                                                      NcmMSet *mset,
                                                      NcmVector *f);
```

**ncm\_dataset\_leastquares\_J ()**

```
void              ncm_dataset_leastquares_J          (NcmDataset *dset,
                                                      NcmMSet *mset,
                                                      NcmMatrix *J);
```

FIXME

**dset** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**J** : a **NcmMatrix**.

**ncm\_dataset\_leastquares\_f\_J ()**

```
void              ncm_dataset_leastquares_f_J        (NcmDataset *dset,
                                                      NcmMSet *mset,
                                                      NcmVector *f,
                                                      NcmMatrix *J);
```

FIXME

**dset** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**f** : a **NcmVector**.

**J** : a **NcmMatrix**.

**ncm\_dataset\_m2lnL\_val ()**

```
void              ncm_dataset_m2lnL_val              (NcmDataset *dset,
                                                      NcmMSet *mset,
                                                      gdouble *m2lnL);
```

FIXME

**dset** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**m2lnL** : FIXME. *[out]*

**ncm\_dataset\_m2lnL\_grad ()**

```
void                ncm_dataset_m2lnL_grad      (NcmDataset *dset,
                                                NcmMSet *mset,
                                                NcmVector *grad);
```

FIXME

**dset** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**grad** : a **NcmVector**.

**ncm\_dataset\_m2lnL\_val\_grad ()**

```
void                ncm_dataset_m2lnL_val_grad (NcmDataset *dset,
                                                NcmMSet *mset,
                                                gdouble *m2lnL,
                                                NcmVector *grad);
```

FIXME

**dset** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**m2lnL** : FIXME. *[out]*

**grad** : a **NcmVector**.

**2.19.3 Gaussian Data - InvCov**

Gaussian Data - InvCov — Gaussian data object, inverse covariance

**Synopsis**

```
struct              NcmDataGaussClass;
struct              NcmDataGauss;
void                ncm_data_gauss_set_size    (NcmDataGauss *gauss,
                                                quint np);
quint               ncm_data_gauss_get_size    (NcmDataGauss *gauss);
```

**Object Hierarchy**

```
GObject
+----NcmData
      +----NcmDataGauss
            +----NcDataBaoRDV
            +----NcDataCMBDistPriors
```

**Properties**

```
"n-points"          quint          : Read / Write
```

Description

Gaussian distribution which uses the inverse covariance matrix as input.

Details

struct NcmDataGaussClass

```
struct NcmDataGaussClass {  
};
```

struct NcmDataGauss

```
struct NcmDataGauss;
```

ncm\_data\_gauss\_set\_size ()

```
void                ncm_data_gauss_set_size      (NcmDataGauss *gauss,  
                                                  guint np);
```

FIXME

*gauss* : a **NcmDataGauss**

*np* : FIXME

ncm\_data\_gauss\_get\_size ()

```
guint              ncm_data_gauss_get_size      (NcmDataGauss *gauss);
```

FIXME

*gauss* : a **NcmDataGauss**

*Returns* : FIXME

Property Details

The "n-points" property

```
"n-points"          guint          : Read / Write
```

Data sample size.

Default value: 0

2.19.4 Gaussian Data - DiagCov

Gaussian Data - DiagCov — Gaussian data object, diagonal covariance

## Synopsis

```

struct          NcmDataGaussDiagClass;
struct          NcmDataGaussDiag;
void            ncm_data_gauss_diag_set_size      (NcmDataGaussDiag *diag,
                                                    quint np);
quint           ncm_data_gauss_diag_get_size      (NcmDataGaussDiag *diag);

```

## Object Hierarchy

```

GObject
+-----NcmData
      +-----NcmDataGaussDiag
            +-----NcDataBaoA
            +-----NcDataBaoDV
            +-----NcDataBaoDVDV
            +-----NcDataCMBShiftParam
            +-----NcDataDistMu
            +-----NcDataHubbleBao
            +-----NcDataHubble

```

## Properties

"n-points"	quint	: Read / Write
"w-mean"	gboolean	: Read / Write

## Description

Gaussian distribution which uses a diagonal covariance matrix as input.

## Details

### struct NcmDataGaussDiagClass

```

struct NcmDataGaussDiagClass {
};

```

### struct NcmDataGaussDiag

```

struct NcmDataGaussDiag;

```

### ncm\_data\_gauss\_diag\_set\_size ()

```

void            ncm_data_gauss_diag_set_size      (NcmDataGaussDiag *diag,
                                                    quint np);

```

FIXME

**diag** : a **NcmDataGauss**

**np** : FIXME

**ncm\_data\_gauss\_diag\_get\_size ()**

```
guint          ncm_data_gauss_diag_get_size          (NcmDataGaussDiag *diag);
```

FIXME

*diag* : a **NcmDataGauss**  
*Returns* : FIXME

**Property Details**

**The "n-points" property**

"n-points"	guint	: Read / Write
------------	-------	----------------

Data sample size.  
Default value: 0

**The "w-mean" property**

"w-mean"	gboolean	: Read / Write
----------	----------	----------------

Whether to minimize analytically over the weighted mean.  
Default value: FALSE

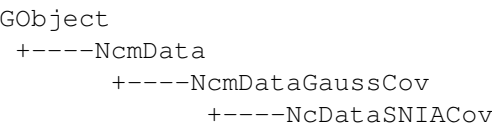
**2.19.5 Gaussian Data - Cov**

Gaussian Data - Cov — Gaussian data object, covariance

**Synopsis**

```
struct          NcmDataGaussCovClass;  
struct          NcmDataGaussCov;  
void            ncm_data_gauss_cov_set_size          (NcmDataGaussCov *gauss,  
                                                    guint np);  
guint           ncm_data_gauss_cov_get_size          (NcmDataGaussCov *gauss);
```

**Object Hierarchy**



**Properties**

"n-points"	guint	: Read / Write
"use-det"	gboolean	: Read / Write

Description

Generic gaussian distribution which uses the covariance matrix as input.

Details

struct NcmDataGaussCovClass

```
struct NcmDataGaussCovClass {  
};
```

struct NcmDataGaussCov

```
struct NcmDataGaussCov;
```

ncm\_data\_gauss\_cov\_set\_size ()

```
void ncm_data_gauss_cov_set_size (NcmDataGaussCov *gauss,  
guint np);
```

FIXME

*gauss* : a **NcmDataGauss**

*np* : FIXME

ncm\_data\_gauss\_cov\_get\_size ()

```
guint ncm_data_gauss_cov_get_size (NcmDataGaussCov *gauss);
```

FIXME

*gauss* : a **NcmDataGauss**

*Returns* : FIXME

Property Details

The "n-points" property

```
"n-points"          guint          : Read / Write
```

Data sample size.

Default value: 0

The "use-det" property

```
"use-det"           gboolean       : Read / Write
```

Use determinant to calculate -2lnL.

Default value: FALSE



## 2.19.6 Poisson Data

Poisson Data — Poisson data

### Synopsis

```
enum                NcmDataPoissonType;
struct              NcmDataPoissonClass;
struct              NcmDataPoisson;
void                ncm_data_poisson_init_from_vector (NcmData *data,
                                                         NcmVector *nodes,
                                                         gsl_vector_ulong *N);

void                ncm_data_poisson_init_from_histogram (NcmData *data,
                                                         gsl_histogram *h);

void                ncm_data_poisson_init_zero (NcmData *data,
                                                         NcmVector *nodes);
```

### Object Hierarchy

```
GObject
+----NcmData
      +----NcmDataPoisson
            +----NcDataClusterPoisson
```

### Properties

```
"n-points"          guint          : Read / Write
```

### Description

FIXME

### Details

#### enum NcmDataPoissonType

```
typedef enum {
    NCM_DATA_POISSON_INT,
} NcmDataPoissonType;
```

FIXME

**NCM\_DATA\_POISSON\_INT** FIXME

#### struct NcmDataPoissonClass

```
struct NcmDataPoissonClass {
};
```

**struct NcmDataPoisson**

```
struct NcmDataPoisson;
```

FIXME

**ncm\_data\_poisson\_init\_from\_vector ()**

```
void ncm_data_poisson_init_from_vector (NcmData *data,
                                       NcmVector *nodes,
                                       gsl_vector_ulong *N);
```

FIXME

***data*** : FIXME

***nodes*** : FIXME

***N*** : FIXME

***Returns*** : FIXME

**ncm\_data\_poisson\_init\_from\_histogram ()**

```
void ncm_data_poisson_init_from_histogram (NcmData *data,
                                           gsl_histogram *h);
```

FIXME

***data*** : FIXME

***h*** : FIXME

***Returns*** : FIXME

**ncm\_data\_poisson\_init\_zero ()**

```
void ncm_data_poisson_init_zero (NcmData *data,
                                 NcmVector *nodes);
```

FIXME

***data*** : FIXME

***nodes*** : FIXME

***Returns*** : FIXME

**Property Details**

**The "n-points" property**

"n-points"	guint	: Read / Write
------------	-------	----------------

Data sample size.

Default value: 0

### 2.19.7 One Variable Distribution Data

One Variable Distribution Data — Object representing a one variable distribution data

#### Synopsis

```
struct          NcmDataDist1dClass;
struct          NcmDataDist1d;
void            ncm_data_dist1d_set_size      (NcmDataDist1d *dist1d,
                                              guint np);
guint           ncm_data_dist1d_get_size      (NcmDataDist1d *dist1d);
```

#### Object Hierarchy

```
GObject
+----NcmData
      +----NcmDataDist1d
```

#### Properties

"n-points"	guint	: Read / Write
------------	-------	----------------

#### Description

FIXME

#### Details

##### struct NcmDataDist1dClass

```
struct NcmDataDist1dClass {
};
```

##### struct NcmDataDist1d

```
struct NcmDataDist1d;
```

##### ncm\_data\_dist1d\_set\_size ()

```
void            ncm_data_dist1d_set_size      (NcmDataDist1d *dist1d,
                                              guint np);
```

FIXME

***dist1d***: a **NcmDataDist1d**

***np***: FIXME

---

```
guint ncm_data_dist1d_get_size (NcmDataDist1d *dist1d);
```

**dist1d:** a **NcmDataDist1d**

## Property Details

"n-points"	quint	: Read / Write
------------	-------	----------------

Default value: 0

### 2.20.1 Likelihood

## Synopsis

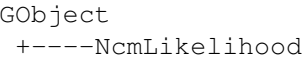
```

struct      NcmLikelihoodClass;
struct      NcmLikelihood;
NcmLikelihood * ncm_likelihoood_new          (NcmDataset *dset);
NcmLikelihood * ncm_likelihoood_ref          (NcmLikelihood *lh);
NcmLikelihood * ncm_likelihoood_dup          (NcmLikelihood *lh);
NcmLikelihood * ncm_likelihoood_copy         (NcmLikelihood *lh);
void         ncm_likelihoood_free            (NcmLikelihood *lh);
void         ncm_likelihoood_clear           (NcmLikelihood **lh);
void         ncm_likelihoood_priors_add      (NcmLikelihood *lh,
                                              NcmMSetFunc *prior);
NcmMSetFunc * ncm_likelihoood_priors_peek    (NcmLikelihood *lh,
                                              quint i);
quint         ncm_likelihoood_priors_length   (NcmLikelihood *lh);
gboolean      ncm_likelihoood_has_leastqsquares_J (NcmLikelihood *lh);
gboolean      ncm_likelihoood_has_m2lnL_grad   (NcmLikelihood *lh);
void          ncm_likelihoood_priors_leastqsquares_f
                                              (NcmLikelihood *lh,
                                              NcmMSet *mset,
                                              NcmVector *priors_f);
void          ncm_likelihoood_leastqsquares_f (NcmLikelihood *lh,
                                              NcmMSet *mset,
                                              NcmVector *f);
void          ncm_likelihoood_leastqsquares_J (NcmLikelihood *lh,
                                              NcmMSet *mset,
                                              NcmMatrix *J);
void          ncm_likelihoood_leastqsquares_f_J (NcmLikelihood *lh,

```

```
void ncm_likelihood_priors_m2lnL_val(NcmMSet *mset, NcmVector *f, NcmMatrix *J);
void ncm_likelihood_m2lnL_val(NcmLikelihood *lh, NcmMSet *mset, gdouble *priors_m2lnL);
void ncm_likelihood_m2lnL_grad(NcmLikelihood *lh, NcmMSet *mset, gdouble *m2lnL);
void ncm_likelihood_m2lnL_val_grad(NcmLikelihood *lh, NcmMSet *mset, NcmVector *grad);
```

Object Hierarchy



Properties

"dataset"	NcmDataset*	: Read / Write / Construct
-----------	-------------	----------------------------

Description

FIXME

Details

struct NcmLikelihoodClass

```
struct NcmLikelihoodClass {
};
```

struct NcmLikelihood

```
struct NcmLikelihood;
```

ncm\_likelihood\_new ()

```
NcmLikelihood * ncm_likelihood_new (NcmDataset *dset);
```

FIXME

**dset** : a **NcmDataset**.

**Returns** : FIXME

**ncm\_likelihood\_ref ()**

```
NcmLikelihood *      ncm_likelihood_ref      (NcmLikelihood *lh);
```

FIXME

**lh** : FIXME

**Returns** : FIXME. *[transfer full]*

**ncm\_likelihood\_dup ()**

```
NcmLikelihood *      ncm_likelihood_dup      (NcmLikelihood *lh);
```

Duplicates the object and all of its content.

**lh** : a **NcmLikelihood**.

**Returns** : A duplicate of *lh*. *[transfer full]*

**ncm\_likelihood\_copy ()**

```
NcmLikelihood *      ncm_likelihood_copy      (NcmLikelihood *lh);
```

Duplicates the object and gets a reference for its content.

**lh** : a **NcmLikelihood**.

**Returns** : A copy of *lh*. *[transfer full]*

**ncm\_likelihood\_free ()**

```
void                  ncm_likelihood_free      (NcmLikelihood *lh);
```

FIXME

**lh** : FIXME

**ncm\_likelihood\_clear ()**

```
void                  ncm_likelihood_clear      (NcmLikelihood **lh);
```

FIXME

**lh** : FIXME

**ncm\_likelihood\_priors\_add ()**

```
void                  ncm_likelihood_priors_add      (NcmLikelihood *lh,  
                                                       NcmMSetFunc *prior);
```

FIXME

**lh** : FIXME

**prior** : FIXME

---

**ncm\_likelihoood\_priors\_peek ()**

```
NcmMSetFunc *      ncm_likelihoood_priors_peek      (NcmLikelihood *lh,  
                                                    quint i);
```

FIXME

**lh** : FIXME

**i** : FIXME

**Returns** : FIXME. *[transfer none]*

**ncm\_likelihoood\_priors\_length ()**

```
quint              ncm_likelihoood_priors_length      (NcmLikelihood *lh);
```

FIXME

**lh** : FIXME

**Returns** : FIXME

**ncm\_likelihoood\_has\_leastqsquares\_J ()**

```
gboolean           ncm_likelihoood_has_leastqsquares_J      (NcmLikelihood *lh);
```

FIXME

**lh** : FIXME

**Returns** : FIXME

**ncm\_likelihoood\_has\_m2lnL\_grad ()**

```
gboolean           ncm_likelihoood_has_m2lnL_grad      (NcmLikelihood *lh);
```

FIXME

**lh** : FIXME

**Returns** : FIXME

**ncm\_likelihoood\_priors\_leastqsquares\_f ()**

```
void               ncm_likelihoood_priors_leastqsquares_f      (NcmLikelihood *lh,  
                                                                NcmMSet *mset,  
                                                                NcmVector *priors_f);
```

FIXME

**lh** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**priors\_f** : a **NcmVector**.

---

**ncm\_likelihood\_least\_squares\_f ()**

```
void          ncm_likelihood_least_squares_f      (NcmLikelihood *lh,
                                                  NcmMSet *mset,
                                                  NcmVector *f);
```

FIXME

**lh** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**f** : a **NcmVector**.

**ncm\_likelihood\_least\_squares\_J ()**

```
void          ncm_likelihood_least_squares_J      (NcmLikelihood *lh,
                                                  NcmMSet *mset,
                                                  NcmMatrix *J);
```

FIXME

**lh** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**J** : a **NcmMatrix**.

**ncm\_likelihood\_least\_squares\_f\_J ()**

```
void          ncm_likelihood_least_squares_f_J    (NcmLikelihood *lh,
                                                  NcmMSet *mset,
                                                  NcmVector *f,
                                                  NcmMatrix *J);
```

FIXME

**lh** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**f** : a **NcmVector**.

**J** : a **NcmMatrix**.

**ncm\_likelihood\_priors\_m2lnL\_val ()**

```
void          ncm_likelihood_priors_m2lnL_val     (NcmLikelihood *lh,
                                                  NcmMSet *mset,
                                                  gdouble *priors_m2lnL);
```

FIXME

**lh** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**priors\_m2lnL** : FIXME. [out]



**ncm\_likelihood\_m2lnL\_val ()**

```
void          ncm_likelihood_m2lnL_val          (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                gdouble *m2lnL);
```

FIXME

**lh** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**m2lnL** : FIXME. *[out]*

**ncm\_likelihood\_m2lnL\_grad ()**

```
void          ncm_likelihood_m2lnL_grad        (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                NcmVector *grad);
```

FIXME

**lh** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**grad** : a **NcmVector**.

**ncm\_likelihood\_m2lnL\_val\_grad ()**

```
void          ncm_likelihood_m2lnL_val_grad    (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                gdouble *m2lnL,
                                                NcmVector *grad);
```

FIXME

**lh** : a **NcmLikelihood**.

**mset** : a **NcmMSet**.

**m2lnL** : FIXME. *[out]*

**grad** : a **NcmVector**.

**Property Details****The "dataset" property**

"dataset"	NcmDataset*	: Read / Write / Construct
-----------	-------------	----------------------------

Dataset object.

**2.20.2 Statistical Priors**

Statistical Priors — General statistical priors

**Synopsis**

```

struct      NcmPriorGauss;
void        ncm_prior_add_oneside_a_inf_param (NcmLikelihood *lh,
                                                NcmModelID mid,
                                                guint pid,
                                                gdouble a,
                                                gdouble s);

void        ncm_prior_add_oneside_a_inf_const_func (NcmLikelihood *lh,
                                                    NcmMSetFunc *func,
                                                    gdouble mean,
                                                    gdouble s);

void        ncm_prior_add_oneside_a_inf_func (NcmLikelihood *lh,
                                              NcmMSetFunc *func,
                                              gdouble z,
                                              gdouble mean,
                                              gdouble s);

void        ncm_prior_add_oneside_a_inf (NcmLikelihood *lh,
                                          NcmPriorGauss *gp);

void        ncm_prior_add_twoside_a_b (NcmLikelihood *lh,
                                       NcmModelID mid,
                                       guint pid,
                                       gdouble a,
                                       gdouble b,
                                       gdouble s);

void        ncm_prior_add_positive (NcmLikelihood *lh,
                                    NcmModelID mid,
                                    guint pid);

void        ncm_prior_add_gaussian (NcmLikelihood *lh,
                                    NcmPriorGauss *gp);

void        ncm_prior_add_gaussian_data (NcmLikelihood *lh,
                                         NcmModelID mid,
                                         guint pid,
                                         gdouble mean,
                                         gdouble sigma);

void        ncm_prior_add_gaussian_func (NcmLikelihood *lh,
                                         NcmMSetFunc *func,
                                         gdouble z,
                                         gdouble mean,
                                         gdouble sigma);

void        ncm_prior_add_gaussian_const_func (NcmLikelihood *lh,
                                                NcmMSetFunc *func,
                                                gdouble mean,
                                                gdouble sigma);

```

**Description**

FIXME

**Details****struct NcmPriorGauss**

```

struct NcmPriorGauss {
};

```

FIXME

### **ncm\_prior\_add\_oneside\_a\_inf\_param ()**

```
void                ncm_prior_add_oneside_a_inf_param    (NcmLikelihood *lh,  
                                                         NcmModelID mid,  
                                                         guint pid,  
                                                         gdouble a,  
                                                         gdouble s);
```

FIXME

**lh** : FIXME

**mid** : FIXME

**pid** : FIXME

**a** : FIXME

**s** : FIXME

### **ncm\_prior\_add\_oneside\_a\_inf\_const\_func ()**

```
void                ncm_prior_add_oneside_a_inf_const_func (NcmLikelihood *lh,  
                                                         NcmMSetFunc *func,  
                                                         gdouble mean,  
                                                         gdouble s);
```

FIXME

**lh** : FIXME

**func** : FIXME

**mean** : FIXME

**s** : FIXME

### **ncm\_prior\_add\_oneside\_a\_inf\_func ()**

```
void                ncm_prior_add_oneside_a_inf_func    (NcmLikelihood *lh,  
                                                         NcmMSetFunc *func,  
                                                         gdouble z,  
                                                         gdouble mean,  
                                                         gdouble s);
```

FIXME

**lh** : FIXME

**func** : FIXME

**z** : FIXME

**mean** : FIXME

**s** : FIXME

---

**ncm\_prior\_add\_oneside\_a\_inf ()**

```
void                ncm_prior_add_oneside_a_inf      (NcmLikelihood *lh,  
                                                    NcmPriorGauss *gp);
```

FIXME

*lh* : FIXME

*gp* : FIXME

**ncm\_prior\_add\_twoside\_a\_b ()**

```
void                ncm_prior_add_twoside_a_b      (NcmLikelihood *lh,  
                                                    NcmModelID mid,  
                                                    guint pid,  
                                                    gdouble a,  
                                                    gdouble b,  
                                                    gdouble s);
```

FIXME

*lh* : FIXME

*mid* : FIXME

*pid* : FIXME

*a* : FIXME

*b* : FIXME

*s* : FIXME

**ncm\_prior\_add\_positive ()**

```
void                ncm_prior_add_positive        (NcmLikelihood *lh,  
                                                    NcmModelID mid,  
                                                    guint pid);
```

FIXME

*lh* : FIXME

*mid* : FIXME

*pid* : FIXME

**ncm\_prior\_add\_gaussian ()**

```
void                ncm_prior_add_gaussian        (NcmLikelihood *lh,  
                                                    NcmPriorGauss *gp);
```

FIXME

*lh* : FIXME

*gp* : FIXME

---

**ncm\_prior\_add\_gaussian\_data ()**

```
void          ncm_prior_add_gaussian_data      (NcmLikelihood *lh,
                                                NcmModelID mid,
                                                guint pid,
                                                gdouble mean,
                                                gdouble sigma);
```

FIXME

**lh**: FIXME

**mid**: FIXME

**pid**: FIXME

**mean**: FIXME

**sigma**: FIXME

**ncm\_prior\_add\_gaussian\_func ()**

```
void          ncm_prior_add_gaussian_func      (NcmLikelihood *lh,
                                                NcmMSetFunc *func,
                                                gdouble z,
                                                gdouble mean,
                                                gdouble sigma);
```

FIXME

**lh**: FIXME

**func**: FIXME

**z**: FIXME

**mean**: FIXME

**sigma**: FIXME

**ncm\_prior\_add\_gaussian\_const\_func ()**

```
void          ncm_prior_add_gaussian_const_func (NcmLikelihood *lh,
                                                NcmMSetFunc *func,
                                                gdouble mean,
                                                gdouble sigma);
```

FIXME

**lh**: FIXME

**func**: FIXME

**mean**: FIXME

**sigma**: FIXME

**2.20.3 Fitting State**

Fitting State — Object representing the current state of a NcmFit object

## Synopsis

```

struct          NcmFitStateClass;
struct          NcmFitState;
NcmFitState *   ncm_fit_state_new          (guint data_len,
                                           guint fparam_len,
                                           gint dof,
                                           gboolean is_least_squares);

NcmFitState *   ncm_fit_state_ref          (NcmFitState *fstate);
void            ncm_fit_state_free         (NcmFitState *fstate);
void            ncm_fit_state_clear        (NcmFitState **fstate);
void            ncm_fit_state_set_all      (NcmFitState *fstate,
                                           guint data_len,
                                           guint fparam_len,
                                           gint dof,
                                           gboolean is_least_squares);

void            ncm_fit_state_reset        (NcmFitState *fstate);

```

## Object Hierarchy

```

GObject
+----NcmFitState

```

## Properties

"data-len"	guint	: Read / Write / Construct
"dof"	gint	: Read / Write / Construct
"fparam-len"	guint	: Read / Write / Construct
"func-eval"	guint	: Read
"grad-eval"	guint	: Read
"is-best-fit"	gboolean	: Read
"is-least-squares"	gboolean	: Read / Write / Construct
"niters"	guint	: Read

## Description

FIXME

## Details

### struct NcmFitStateClass

```

struct NcmFitStateClass {
};

```

### struct NcmFitState

```

struct NcmFitState;

```

**ncm\_fit\_state\_new ()**

```
NcmFitState *      ncm_fit_state_new      (guint data_len,  
                                           guint fparam_len,  
                                           gint dof,  
                                           gboolean is_least_squares);
```

FIXME

***data\_len***: FIXME

***fparam\_len***: FIXME

***dof***: FIXME

***is\_least\_squares***: FIXME

***Returns***: FIXME

**ncm\_fit\_state\_ref ()**

```
NcmFitState *      ncm_fit_state_ref      (NcmFitState *fstate);
```

FIXME

***fstate***: FIXME

***Returns***: FIXME. *[transfer full]*

**ncm\_fit\_state\_free ()**

```
void               ncm_fit_state_free      (NcmFitState *fstate);
```

FIXME

***fstate***: FIXME

**ncm\_fit\_state\_clear ()**

```
void               ncm_fit_state_clear      (NcmFitState **fstate);
```

FIXME

***fstate***: FIXME

**ncm\_fit\_state\_set\_all ()**

```
void               ncm_fit_state_set_all    (NcmFitState *fstate,  
                                           guint data_len,  
                                           guint fparam_len,  
                                           gint dof,  
                                           gboolean is_least_squares);
```

FIXME

***fstate***: FIXME

---

*data\_len* : FIXME  
*fparam\_len* : FIXME  
*dof* : FIXME  
*is\_least\_squares* : FIXME  
*Returns* : FIXME

**ncm\_fit\_state\_reset()**

```
void ncm_fit_state_reset(NcmFitState *fstate);
```

FIXME  
*fstate* : FIXME  
*Returns* : FIXME

**Property Details**

**The "data-len" property**

"data-len"	guint	: Read / Write / Construct
------------	-------	----------------------------

Data length.  
Default value: 0

**The "dof" property**

"dof"	gint	: Read / Write / Construct
-------	------	----------------------------

Degrees of freedom.  
Default value: 0

**The "fparam-len" property**

"fparam-len"	guint	: Read / Write / Construct
--------------	-------	----------------------------

Free parameters length.  
Default value: 0

**The "func-eval" property**

"func-eval"	guint	: Read
-------------	-------	--------

Number of function evaluations.  
Default value: 0

---



The "grad-eval" property

"grad-eval"	guint	: Read
-------------	-------	--------

Number of gradient evaluations.

Default value: 0

The "is-best-fit" property

"is-best-fit"	gboolean	: Read
---------------	----------	--------

Is a best fit state.

Default value: FALSE

The "is-least-squares" property

"is-least-squares"	gboolean	: Read / Write / Construct
--------------------	----------	----------------------------

Is a least squares fit state.

Default value: FALSE

The "nitters" property

"nitters"	guint	: Read
-----------	-------	--------

Number of iterations.

Default value: 0

2.20.4 Model Fitting Abstract Class

Model Fitting Abstract Class — Class for implementing fitting methods

Synopsis

```
enum          NcmFitType;
enum          NcmFitGradType;
void          (*_NcmFitLSJ)          (NcmFit *fit,
                                     NcmMatrix *J);
void          (*_NcmFitLSFJ)          (NcmFit *fit,
                                     NcmVector *f,
                                     NcmMatrix *J);
void          (*_NcmFitM2lnLGrad)     (NcmFit *fit,
                                     NcmVector *grad);
void          (*_NcmFitM2lnLValGrad)  (NcmFit *fit,
                                     gdouble *m2lnL,
                                     NcmVector *grad);

enum          NcmFitRunMsgs;
struct        NcmFitClass;
struct        NcmFit;
struct        NcmFitConstraint;
NcmFitConstraint * ncm_fit_constraint_new (NcmFit *fit,
```

		NcmMSetFunc *func,
		gdouble tot);
NcmFitConstraint *	ncm_fit_constraint_dup	(NcmFitConstraint *fitc);
void	ncm_fit_constraint_free	(NcmFitConstraint *fitc);
NcmFit *	ncm_fit_new	(NcmFitType ftype,
		gchar *algo_name,
		NcmLikelihood *lh,
		NcmMSet *mset,
		NcmFitGradType gtype);
NcmFit *	ncm_fit_ref	(NcmFit *fit);
NcmFit *	ncm_fit_copy_new	(NcmFit *fit,
		NcmLikelihood *lh,
		NcmMSet *mset,
		NcmFitGradType gtype);
void	ncm_fit_free	(NcmFit *fit);
void	ncm_fit_clear	(NcmFit **fit);
void	ncm_fit_set_grad_type	(NcmFit *fit,
		NcmFitGradType gtype);
void	ncm_fit_ls_set_state	(NcmFit *fit,
		gdouble prec,
		NcmVector *x,
		NcmVector *f,
		NcmMatrix *J);
void	ncm_fit_set_maxiter	(NcmFit *fit,
		guint maxiter);
guint	ncm_fit_get_maxiter	(NcmFit *fit);
void	ncm_fit_add_equality_constraint	(NcmFit *fit,
		NcmMSetFunc *func,
		gdouble tot);
void	ncm_fit_add_inequality_constraint	(NcmFit *fit,
		NcmMSetFunc *func,
		gdouble tot);
void	ncm_fit_remove_equality_constraints	(NcmFit *fit);
void	ncm_fit_remove_inequality_constraints	(NcmFit *fit);
guint	ncm_fit_has_equality_constraints	(NcmFit *fit);
guint	ncm_fit_has_inequality_constraints	(NcmFit *fit);
gboolean	ncm_fit_is_least_squares	(NcmFit *fit);
const gchar *	ncm_fit_get_desc	(NcmFit *fit);
void	ncm_fit_log_info	(NcmFit *fit);
void	ncm_fit_log_covar	(NcmFit *fit);
void	ncm_fit_log_start	(NcmFit *fit);
void	ncm_fit_log_state	(NcmFit *fit);
void	ncm_fit_log_step	(NcmFit *fit);
void	ncm_fit_log_step_error	(NcmFit *fit,
		const gchar *strerror,
		...);
void	ncm_fit_log_end	(NcmFit *fit);
void	ncm_fit_fishermatrix_print	(NcmFit *fit,
		FILE *out,
		gchar *header);
void	ncm_fit_data_m2lnL_val	(NcmFit *fit,
		gdouble *data_m2lnL);
void	ncm_fit_priors_m2lnL_val	(NcmFit *fit,
		gdouble *priors_m2lnL);
void	ncm_fit_m2lnL_val	(NcmFit *fit,
		gdouble *m2lnL);

void	ncm_fit_ls_f	(NcmFit *fit, NcmVector *f);
void	ncm_fit_m2lnL_grad	(NcmFit *fit, NcmVector *df);
void	ncm_fit_m2lnL_grad_an	(NcmFit *fit, NcmVector *df);
void	ncm_fit_m2lnL_grad_nd_fo	(NcmFit *fit, NcmVector *grad);
void	ncm_fit_m2lnL_grad_nd_ce	(NcmFit *fit, NcmVector *grad);
void	ncm_fit_m2lnL_grad_nd_ac	(NcmFit *fit, NcmVector *grad);
void	ncm_fit_m2lnL_val_grad	(NcmFit *fit, gdouble *result, NcmVector *df);
void	ncm_fit_m2lnL_val_grad_an	(NcmFit *fit, gdouble *result, NcmVector *df);
void	ncm_fit_m2lnL_val_grad_nd_fo	(NcmFit *fit, gdouble *m2lnL, NcmVector *grad);
void	ncm_fit_m2lnL_val_grad_nd_ce	(NcmFit *fit, gdouble *m2lnL, NcmVector *grad);
void	ncm_fit_m2lnL_val_grad_nd_ac	(NcmFit *fit, gdouble *m2lnL, NcmVector *grad);
void	ncm_fit_ls_J	(NcmFit *fit, NcmMatrix *J);
void	ncm_fit_ls_J_an	(NcmFit *fit, NcmMatrix *J);
void	ncm_fit_ls_J_nd_fo	(NcmFit *fit, NcmMatrix *J);
void	ncm_fit_ls_J_nd_ce	(NcmFit *fit, NcmMatrix *J);
void	ncm_fit_ls_f_J	(NcmFit *fit, NcmVector *f, NcmMatrix *J);
void	ncm_fit_ls_f_J_an	(NcmFit *fit, NcmVector *f, NcmMatrix *J);
void	ncm_fit_ls_f_J_nd_fo	(NcmFit *fit, NcmVector *f, NcmMatrix *J);
void	ncm_fit_ls_f_J_nd_ce	(NcmFit *fit, NcmVector *f, NcmMatrix *J);
void	ncm_fit_numdiff_m2lnL_hessian	(NcmFit *fit, NcmMatrix *H);
void	ncm_fit_numdiff_m2lnL_covar	(NcmFit *fit);
void	ncm_fit_ls_covar	(NcmFit *fit);
gdouble	ncm_fit_covar_var	(NcmFit *fit, NcmModelID mid, guint pid);
gdouble	ncm_fit_covar_sd	(NcmFit *fit, NcmModelID mid, guint pid);

---

gdouble	ncm_fit_covar_cov	(NcmFit *fit, NcmModelID mid1, guint pid1, NcmModelID mid2, guint pid2);
gdouble	ncm_fit_covar_cor	(NcmFit *fit, NcmModelID mid1, guint pid1, NcmModelID mid2, guint pid2);
gdouble	ncm_fit_covar_fparam_var	(NcmFit *fit, guint fpi);
gdouble	ncm_fit_covar_fparam_sd	(NcmFit *fit, guint fpi);
gdouble	ncm_fit_covar_fparam_cov	(NcmFit *fit, guint fpi1, guint fpi2);
gdouble	ncm_fit_covar_fparam_cor	(NcmFit *fit, guint fpi1, guint fpi2);
void	ncm_fit_lr_test_range	(NcmFit *fit, NcmModelID mid, guint pid, gdouble start, gdouble stop, gdouble step);
void	ncm_fit_dprob	(NcmFit *fit, NcmModelID mid, guint pid, gdouble a, gdouble b, gdouble step, gdouble norm);
gdouble	ncm_fit_lr_test	(NcmFit *fit, NcmModelID mid, guint pid, gdouble val, gint dof);
gdouble	ncm_fit_prob	(NcmFit *fit, NcmModelID mid, guint pid, gdouble a, gdouble b);
gdouble	ncm_fit_chisq_test	(NcmFit *fit, size_t bins);
void	ncm_fit_reset	(NcmFit *fit);
gboolean	ncm_fit_run	(NcmFit *fit, NcmFitRunMsgs mtype);
gdouble	ncm_fit_type_constrain_error	(NcmFit *fit, gdouble p, gint nu, gdouble dir, NcmMSetFunc *func, gdouble z, gboolean walk);
void	ncm_fit_function_error	(NcmFit *fit, NcmMSetFunc *func,

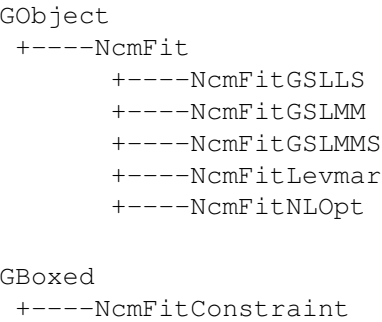
```
gdouble          ncm_fit_function_cov

#define          NCM_FIT_NUMDIFF_SCALE
#define          NCM_FIT_NPARAM
#define          NCM_FIT_DEFAULT_M2LNL_ABSTOL
#define          NCM_FIT_DEFAULT_M2LNL_RELTOL
#define          NCM_FIT_MAXITER

gdouble *x,
gboolean pretty_print,
gdouble *f,
gdouble *sigma_f);
(NcmFit *fit,
NcmMSetFunc *func1,
gdouble z1,
NcmMSetFunc *func2,
gdouble z2,
gboolean pretty_print);

(fit)
```

Object Hierarchy



Properties

"grad-type"	NcmFitGradType	: Read / Write / Construct
"likelihood"	NcmLikelihood*	: Read / Write / Construct
"maxiter"	guint	: Read / Write / Construct
"mset"	NcmMSet*	: Read / Write / Construct

Description

FIXME

Details

enum NcmFitType

```
typedef enum {
    NCM_FIT_TYPE_GSL_LS = 0,
    NCM_FIT_TYPE_GSL_MM,
    NCM_FIT_TYPE_GSL_MMS,
    NCM_FIT_TYPE_LEVMAR,
    NCM_FIT_TYPE_NLOPT,
} NcmFitType;
```

FIXME

**NCM\_FIT\_TYPE\_GSL\_LS** GSL Least Squares

**NCM\_FIT\_TYPE\_GSL\_MM** GSL Multidimensional Minimization

**NCM\_FIT\_TYPE\_GSL\_MMS** GSL Multidimensional Minimization (simplex)

**NCM\_FIT\_TYPE\_LEVMAR** Levmar Least Squares Library

**NCM\_FIT\_TYPE\_NLOPT** Non Linear Optimization (NLOpt)

### enum NcmFitGradType

```
typedef enum {
    NCM_FIT_GRAD_ANALYTICAL = 0,
    NCM_FIT_GRAD_NUMDIFF_FORWARD,
    NCM_FIT_GRAD_NUMDIFF_CENTRAL,
    NCM_FIT_GRAD_NUMDIFF_ACCURATE,
} NcmFitGradType;
```

FIXME

**NCM\_FIT\_GRAD\_ANALYTICAL** FIXME

**NCM\_FIT\_GRAD\_NUMDIFF\_FORWARD** FIXME

**NCM\_FIT\_GRAD\_NUMDIFF\_CENTRAL** FIXME

**NCM\_FIT\_GRAD\_NUMDIFF\_ACCURATE** FIXME

### \_NcmFitLSJ ()

```
void (*_NcmFitLSJ) (NcmFit *fit,
                    NcmMatrix *J);
```

### \_NcmFitLSFJ ()

```
void (*_NcmFitLSFJ) (NcmFit *fit,
                     NcmVector *f,
                     NcmMatrix *J);
```

### \_NcmFitM2lnLGrad ()

```
void (*_NcmFitM2lnLGrad) (NcmFit *fit,
                           NcmVector *grad);
```

### \_NcmFitM2lnLValGrad ()

```
void (*_NcmFitM2lnLValGrad) (NcmFit *fit,
                              gdouble *m2lnL,
                              NcmVector *grad);
```

**enum NcmFitRunMsgs**

```
typedef enum {
    NCM_FIT_RUN_MSGS_NONE = 0,
    NCM_FIT_RUN_MSGS_SIMPLE,
    NCM_FIT_RUN_MSGS_FULLL,
} NcmFitRunMsgs;
```

FIXME

**NCM\_FIT\_RUN\_MSGS\_NONE** FIXME

**NCM\_FIT\_RUN\_MSGS\_SIMPLE** FIXME

**NCM\_FIT\_RUN\_MSGS\_FULLL** FIXME

**struct NcmFitClass**

```
struct NcmFitClass {
};
```

**struct NcmFit**

```
struct NcmFit;
```

**struct NcmFitConstraint**

```
struct NcmFitConstraint {
};
```

**ncm\_fit\_constraint\_new ()**

```
NcmFitConstraint * ncm_fit_constraint_new (NcmFit *fit,
                                           NcmMSetFunc *func,
                                           gdouble tot);
```

FIXME

***fit*** : FIXME

***func*** : FIXME

***tot*** : FIXME

**Returns** : FIXME. *[transfer full]*

**ncm\_fit\_constraint\_dup ()**

```
NcmFitConstraint * ncm_fit_constraint_dup (NcmFitConstraint *fitc);
```

FIXME

***fitc*** : FIXME

**Returns** : FIXME. *[transfer full]*

**ncm\_fit\_constraint\_free ()**

```
void                ncm_fit_constraint_free        (NcmFitConstraint *fitc);
```

FIXME

**fitc** : FIXME

**ncm\_fit\_new ()**

```
NcmFit *            ncm_fit_new                    (NcmFitType ftype,
                                                    gchar *algo_name,
                                                    NcmLikelihood *lh,
                                                    NcmMSet *mset,
                                                    NcmFitGradType gtype);
```

FIXME

**ftype** : a **NcmFitType**

**algo\_name** : name of the algorithm to be used

**lh** : a **NcmLikelihood**

**mset** : a **NcmMSet**

**gtype** : a **NcmFitGradType**

**Returns** : FIXME

**ncm\_fit\_ref ()**

```
NcmFit *            ncm_fit_ref                    (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**.

**Returns** : FIXME. *[transfer full]*

**ncm\_fit\_copy\_new ()**

```
NcmFit *            ncm_fit_copy_new                (NcmFit *fit,
                                                    NcmLikelihood *lh,
                                                    NcmMSet *mset,
                                                    NcmFitGradType gtype);
```

Duplicates the **NcmFit** object with new references for its contents.

**fit** : a **NcmFit**

**lh** : a **NcmLikelihood**

**mset** : a **NcmMSet**

**gtype** : a **NcmFitGradType**

**Returns** : FIXME. *[transfer full]*



**ncm\_fit\_free ()**

```
void          ncm_fit_free          (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**.

**ncm\_fit\_clear ()**

```
void          ncm_fit_clear        (NcmFit **fit);
```

FIXME

**fit** : a **NcmFit**.

**ncm\_fit\_set\_grad\_type ()**

```
void          ncm_fit_set_grad_type (NcmFit *fit,
                                     NcmFitGradType gtype);
```

FIXME

**fit** : a **NcmLikelihood**.

**gtype** : a **NcmFitGradType**.

**ncm\_fit\_ls\_set\_state ()**

```
void          ncm_fit_ls_set_state (NcmFit *fit,
                                     gdouble prec,
                                     NcmVector *x,
                                     NcmVector *f,
                                     NcmMatrix *J);
```

FIXME

**fit** : a **NcmFit**.

**prec** : FIXME

**x** : a **NcmVector**.

**f** : a **NcmVector**.

**J** : a **NcmMatrix**.

**ncm\_fit\_set\_maxiter ()**

```
void          ncm_fit_set_maxiter  (NcmFit *fit,
                                     guint maxiter);
```

FIXME

**fit** : a **NcmFit**.

**maxiter** : FIXME.

**ncm\_fit\_get\_maxiter ()**

```
quint          ncm_fit_get_maxiter          (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**.

**Returns** : FIXME

**ncm\_fit\_add\_equality\_constraint ()**

```
void          ncm_fit_add_equality_constraint (NcmFit *fit,  
                                              NcmMSetFunc *func,  
                                              gdouble tot);
```

FIXME

**fit** : a **NcmFit**.

**func** : FIXME

**tot** : FIXME

**ncm\_fit\_add\_inequality\_constraint ()**

```
void          ncm_fit_add_inequality_constraint (NcmFit *fit,  
                                              NcmMSetFunc *func,  
                                              gdouble tot);
```

FIXME

**fit** : a **NcmFit**.

**func** : FIXME

**tot** : FIXME

**ncm\_fit\_remove\_equality\_constraints ()**

```
void          ncm_fit_remove_equality_constraints (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**.

**ncm\_fit\_remove\_inequality\_constraints ()**

```
void          ncm_fit_remove_inequality_constraints  
                                              (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**.

---

**ncm\_fit\_has\_equality\_constraints ()**

```
guint          ncm_fit_has_equality_constraints    (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**.

**Returns** : FIXME

**ncm\_fit\_has\_inequality\_constraints ()**

```
guint          ncm_fit_has_inequality_constraints  (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**.

**Returns** : FIXME

**ncm\_fit\_is\_least\_squares ()**

```
gboolean       ncm_fit_is_least_squares          (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**

**Returns** : whenever the fit object use a least squares method.

**ncm\_fit\_get\_desc ()**

```
const gchar *  ncm_fit_get_desc                  (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**

**Returns** : fit object description. *[transfer none]*

**ncm\_fit\_log\_info ()**

```
void           ncm_fit_log_info                   (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**

**ncm\_fit\_log\_covar ()**

```
void           ncm_fit_log_covar                  (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**

---

```
void ncm_fit_log_start(NcmFit *fit);
```

**fit** : a NcmFit

```
void ncm_fit_log_state(NcmFit *fit);
```

*fit* : a NcmFit

```
void ncm_fit_log_step(NcmFit *fit);
```

**fit**: a NcmFit

```
void ncm_fit_log_step_error(NcmFit *fit,
                           const gchar *strerror,
                           ...);
```

*fit*: a NcmFit

```
... : FIXME
```

```
void ncm_fit_log_end(NcmFit *fit);
```

*fit*: a NcmFit

```
void ncm_fit_fishermatrix_print(NcmFit *fit, FILE *out, qchar *header);
```

**ncm\_fit\_data\_m2lnL\_val ()**

```
void                ncm_fit_data_m2lnL_val                (NcmFit *fit,
                                                         gdouble *data_m2lnL);
```

This function computes minus two times the logarithm base e of the likelihood using only the data set and not considering any prior. The result is set on *data\_m2lnL*.

**fit** : a **NcmFit**

**data\_m2lnL** : minus two times the logarithm base e of the likelihood. *[out]*

**ncm\_fit\_priors\_m2lnL\_val ()**

```
void                ncm_fit_priors_m2lnL_val              (NcmFit *fit,
                                                         gdouble *priors_m2lnL);
```

This function computes minus two times the logarithm base e of the likelihood using the data set and taking into account the assumed priors. The result is set on *priors\_m2lnL*.

**fit** : a **NcmFit**

**priors\_m2lnL** : minus two times the logarithm base e of the likelihood. *[out]*

**ncm\_fit\_m2lnL\_val ()**

```
void                ncm_fit_m2lnL_val                   (NcmFit *fit,
                                                         gdouble *m2lnL);
```

FIXME

**fit** : a **NcmFit**

**m2lnL** : minus two times the logarithm base e of the likelihood. *[out]*

**ncm\_fit\_ls\_f ()**

```
void                ncm_fit_ls_f                        (NcmFit *fit,
                                                         NcmVector *f);
```

FIXME

**fit** : a **NcmFit**

**f** : a **NcmVector**

**ncm\_fit\_m2lnL\_grad ()**

```
void                ncm_fit_m2lnL_grad                 (NcmFit *fit,
                                                         NcmVector *df);
```

FIXME

**fit** : a **NcmFit**

**df** : a **NcmVector**

**ncm\_fit\_m2lnL\_grad\_an ()**

```
void                ncm_fit_m2lnL_grad_an                (NcmFit *fit,  
                                                         NcmVector *df);
```

Analytical gradient.

**fit** : a **NcmFit**

**df** : a **NcmVector**

**ncm\_fit\_m2lnL\_grad\_nd\_fo ()**

```
void                ncm_fit_m2lnL_grad_nd_fo            (NcmFit *fit,  
                                                         NcmVector *grad);
```

Numerical differentiation (forward).

**fit** : a **NcmFit**

**grad** : a **NcmVector**

**ncm\_fit\_m2lnL\_grad\_nd\_ce ()**

```
void                ncm_fit_m2lnL_grad_nd_ce            (NcmFit *fit,  
                                                         NcmVector *grad);
```

Numerical differentiation (central).

**fit** : a **NcmFit**

**grad** : a **NcmVector**

**ncm\_fit\_m2lnL\_grad\_nd\_ac ()**

```
void                ncm_fit_m2lnL_grad_nd_ac            (NcmFit *fit,  
                                                         NcmVector *grad);
```

Numerical differentiation (accurate).

**fit** : a **NcmFit**

**grad** : a **NcmVector**

**ncm\_fit\_m2lnL\_val\_grad ()**

```
void                ncm_fit_m2lnL_val_grad              (NcmFit *fit,  
                                                         gdouble *result,  
                                                         NcmVector *df);
```

FIXME

**fit** : a **NcmFit**

**result** : FIXME. *[out]*

**df** : a **NcmVector**

---

**ncm\_fit\_m2lnL\_val\_grad\_an ()**

```
void          ncm_fit_m2lnL_val_grad_an      (NcmFit *fit,
                                              gdouble *result,
                                              NcmVector *df);
```

FIXME

**fit** : a **NcmFit**

**result** : FIXME. *[out]*

**df** : a **NcmVector**

**ncm\_fit\_m2lnL\_val\_grad\_nd\_fo ()**

```
void          ncm_fit_m2lnL_val_grad_nd_fo  (NcmFit *fit,
                                              gdouble *m2lnL,
                                              NcmVector *grad);
```

FIXME

**fit** : a **NcmFit**

**m2lnL** : minus two times the logarithm base e of the likelihood. *[out]*

**grad** : a **NcmVector**

**ncm\_fit\_m2lnL\_val\_grad\_nd\_ce ()**

```
void          ncm_fit_m2lnL_val_grad_nd_ce  (NcmFit *fit,
                                              gdouble *m2lnL,
                                              NcmVector *grad);
```

FIXME

**fit** : a **NcmFit**

**m2lnL** : minus two times the logarithm base e of the likelihood.

**grad** : a **NcmVector**

**ncm\_fit\_m2lnL\_val\_grad\_nd\_ac ()**

```
void          ncm_fit_m2lnL_val_grad_nd_ac  (NcmFit *fit,
                                              gdouble *m2lnL,
                                              NcmVector *grad);
```

FIXME

**fit** : a **NcmFit**

**m2lnL** : minus two times the logarithm base e of the likelihood. *[out]*

**grad** : a **NcmVector**

**ncm\_fit\_ls\_J ()**

```
void                ncm_fit_ls_J                (NcmFit *fit,  
                                                NcmMatrix *J);
```

FIXME

***f*it** : a **NcmFit*****J*** : a **NcmMatrix****ncm\_fit\_ls\_J\_an ()**

```
void                ncm_fit_ls_J_an            (NcmFit *fit,  
                                                NcmMatrix *J);
```

FIXME

***f*it** : a **NcmFit*****J*** : a **NcmMatrix****ncm\_fit\_ls\_J\_nd\_fo ()**

```
void                ncm_fit_ls_J_nd_fo        (NcmFit *fit,  
                                                NcmMatrix *J);
```

FIXME

***f*it** : a **NcmFit*****J*** : a **NcmMatrix****ncm\_fit\_ls\_J\_nd\_ce ()**

```
void                ncm_fit_ls_J_nd_ce        (NcmFit *fit,  
                                                NcmMatrix *J);
```

FIXME

***f*it** : a **NcmFit*****J*** : a **NcmMatrix****ncm\_fit\_ls\_f\_J ()**

```
void                ncm_fit_ls_f_J            (NcmFit *fit,  
                                                NcmVector *f,  
                                                NcmMatrix *J);
```

FIXME

***f*it** : a **NcmFit*****f*** : a **NcmVector*****J*** : a **NcmMatrix**



**ncm\_fit\_ls\_f\_J\_an()**

```
void          ncm_fit_ls_f_J_an          (NcmFit *fit,
                                           NcmVector *f,
                                           NcmMatrix *J);
```

FIXME

***f*** : a **NcmFit**

***f*** : a **NcmVector**

***J*** : a **NcmMatrix**

**ncm\_fit\_ls\_f\_J\_nd\_fo()**

```
void          ncm_fit_ls_f_J_nd_fo      (NcmFit *fit,
                                           NcmVector *f,
                                           NcmMatrix *J);
```

FIXME

***f*** : a **NcmFit**

***f*** : a **NcmVector**

***J*** : a **NcmMatrix**

**ncm\_fit\_ls\_f\_J\_nd\_ce()**

```
void          ncm_fit_ls_f_J_nd_ce      (NcmFit *fit,
                                           NcmVector *f,
                                           NcmMatrix *J);
```

FIXME

***f*** : a **NcmFit**

***f*** : a **NcmVector**

***J*** : a **NcmMatrix**

**ncm\_fit\_numdiff\_m2lnL\_hessian()**

```
void          ncm_fit_numdiff_m2lnL_hessian (NcmFit *fit,
                                              NcmMatrix *H);
```

FIXME

***f*** : a **NcmFit**

***H*** : a **NcmMatrix**

**ncm\_fit\_numdiff\_m2lnL\_covar()**

```
void          ncm_fit_numdiff_m2lnL_covar (NcmFit *fit);
```

FIXME

***f*** : a **NcmFit**

**ncm\_fit\_ls\_covar ()**

```
void                ncm_fit_ls_covar                (NcmFit *fit);
```

FIXME

**fit** : a **NcmFit**.

**ncm\_fit\_covar\_var ()**

```
gdouble            ncm_fit_covar_var                (NcmFit *fit,  
                                                    NcmModelID mid,  
                                                    guint pid);
```

FIXME

**fit** : a **NcmFit**.

**mid** : a **NcmModelID**.

**pid** : FIXME

**Returns** : FIXME

**ncm\_fit\_covar\_sd ()**

```
gdouble            ncm_fit_covar_sd                (NcmFit *fit,  
                                                    NcmModelID mid,  
                                                    guint pid);
```

FIXME

**fit** : a **NcmFit**.

**mid** : a **NcmModelID**.

**pid** : FIXME

**Returns** : FIXME

**ncm\_fit\_covar\_cov ()**

```
gdouble            ncm_fit_covar_cov                (NcmFit *fit,  
                                                    NcmModelID mid1,  
                                                    guint pid1,  
                                                    NcmModelID mid2,  
                                                    guint pid2);
```

FIXME

**fit** : a **NcmFit**

**mid1** : a **NcmModelID**.

**pid1** : FIXME

**mid2** : a **NcmModelID**.

**pid2** : FIXME

**Returns** : FIXME

---

**ncm\_fit\_covar\_cor ()**

gdouble	ncm_fit_covar_cor	(NcmFit *fit, NcmModelID mid1, guint pid1, NcmModelID mid2, guint pid2);
---------	-------------------	--

FIXME

**fit** : a **NcmFit**

**mid1** : a **NcmModelID**.

**pid1** : FIXME

**mid2** : a **NcmModelID**.

**pid2** : FIXME

**Returns** : FIXME

**ncm\_fit\_covar\_fparam\_var ()**

gdouble	ncm_fit_covar_fparam_var	(NcmFit *fit, guint fpi);
---------	--------------------------	------------------------------

FIXME

**fit** : a **NcmFit**.

**fpi** : FIXME

**Returns** : FIXME

**ncm\_fit\_covar\_fparam\_sd ()**

gdouble	ncm_fit_covar_fparam_sd	(NcmFit *fit, guint fpi);
---------	-------------------------	------------------------------

FIXME

**fit** : a **NcmFit**.

**fpi** : FIXME

**Returns** : FIXME

**ncm\_fit\_covar\_fparam\_cov ()**

gdouble	ncm_fit_covar_fparam_cov	(NcmFit *fit, guint fpi1, guint fpi2);
---------	--------------------------	--

FIXME

**fit** : a **NcmFit**.

**fpi1** : FIXME

**fpi2** : FIXME

**Returns** : FIXME

**ncm\_fit\_covar\_fparam\_cor ()**

```
gdouble          ncm_fit_covar_fparam_cor      (NcmFit *fit,
                                                guint fpi1,
                                                guint fpi2);
```

FIXME

**fit** : a **NcmFit**.

**fpi1** : FIXME

**fpi2** : FIXME

**Returns** : FIXME

**ncm\_fit\_lr\_test\_range ()**

```
void             ncm_fit_lr_test_range        (NcmFit *fit,
                                                NcmModelID mid,
                                                guint pid,
                                                gdouble start,
                                                gdouble stop,
                                                gdouble step);
```

FIXME

**fit** : a **NcmFit**

**mid** : a **NcmModelID**.

**pid** : FIXME

**start** : FIXME

**stop** : FIXME

**step** : FIXME

**ncm\_fit\_dprob ()**

```
void             ncm_fit_dprob               (NcmFit *fit,
                                                NcmModelID mid,
                                                guint pid,
                                                gdouble a,
                                                gdouble b,
                                                gdouble step,
                                                gdouble norm);
```

FIXME

**fit** : a **NcmFit**.

**mid** : a **NcmModelID**.

**pid** : FIXME

**a** : FIXME

**b** : FIXME

**step** : FIXME

**norm** : FIXME

**Returns** : FIXME

**ncm\_fit\_lr\_test ()**

gdouble	ncm_fit_lr_test	(NcmFit *fit, NcmModelID mid, guint pid, gdouble val, gint dof);
---------	-----------------	--

FIXME

**fit** : a **NcmFit**.**mid** : a **NcmModelID**.**pid** : FIXME**val** : FIXME**dof** : FIXME**Returns** : FIXME**ncm\_fit\_prob ()**

gdouble	ncm_fit_prob	(NcmFit *fit, NcmModelID mid, guint pid, gdouble a, gdouble b);
---------	--------------	---

FIXME

**fit** : a **NcmFit**.**mid** : a **NcmModelID**.**pid** : FIXME**a** : FIXME**b** : FIXME**Returns** : FIXME**ncm\_fit\_chisq\_test ()**

gdouble	ncm_fit_chisq_test	(NcmFit *fit, size_t bins);
---------	--------------------	--------------------------------

**ncm\_fit\_reset ()**

void	ncm_fit_reset	(NcmFit *fit);
------	---------------	----------------

FIXME

**fit** : a **NcmFit**

**ncm\_fit\_run ()**

gboolean	ncm_fit_run	(NcmFit *fit, NcmFitRunMsgs mtype);
----------	-------------	--

FIXME

**fit** : a **NcmFit****mtype** : a **NcmFitRunMsgs****Returns** : FIXME**ncm\_fit\_type\_constrain\_error ()**

gdouble	ncm_fit_type_constrain_error	(NcmFit *fit, gdouble p, gint nu, gdouble dir, NcmMSetFunc *func, gdouble z, gboolean walk);
---------	------------------------------	--

**ncm\_fit\_function\_error ()**

void	ncm_fit_function_error	(NcmFit *fit, NcmMSetFunc *func, gdouble *x, gboolean pretty_print, gdouble *f, gdouble *sigma_f);
------	------------------------	---

FIXME

**fit** : a **NcmFit****func** : a **NcmMSetFunc****x** : FIXME**pretty\_print** : FIXME**f** : FIXME. [out]**sigma\_f** : FIXME. [out]**ncm\_fit\_function\_cov ()**

gdouble	ncm_fit_function_cov	(NcmFit *fit, NcmMSetFunc *func1, gdouble z1, NcmMSetFunc *func2, gdouble z2, gboolean pretty_print);
---------	----------------------	--

FIXME

**fit** : a **NcmFit**

*func1* : a **NcmMSetFunc**  
*z1* : FIXME  
*func2* : a **NcmMSetFunc**  
*z2* : FIXME  
*pretty\_print* : FIXME  
*Returns* : FIXME

**NCM\_FIT\_NUMDIFF\_SCALE**

```
#define NCM_FIT_NUMDIFF_SCALE (1.0e-4)
```

**NCM\_FIT\_NPARAM()**

```
#define NCM_FIT_NPARAM(fit) ((fit)->pt->nfree)
```

**NCM\_FIT\_DEFAULT\_M2LNL\_ABSTOL**

```
#define NCM_FIT_DEFAULT_M2LNL_ABSTOL (0.0)
```

**NCM\_FIT\_DEFAULT\_M2LNL\_RELTOL**

```
#define NCM_FIT_DEFAULT_M2LNL_RELTOL (1e-13)
```

**NCM\_FIT\_MAXITER**

```
#define NCM_FIT_MAXITER 10000
```

**Property Details**

**The "grad-type" property**

"grad-type"	NcmFitGradType	: Read / Write / Construct
-------------	----------------	----------------------------

Differentiation method.  
Default value: NCM\_FIT\_GRAD\_NUMDIFF\_FORWARD

**The "likelihood" property**

"likelihood"	NcmLikelihood*	: Read / Write / Construct
--------------	----------------	----------------------------

Likelihood object.

---

The "maxiter" property

"maxiter"	guint	: Read / Write / Construct
-----------	-------	----------------------------

Maximum number of iterations.

Default value: 10000

The "mset " property

"mset "	NcmMSet*	: Read / Write / Construct
---------	----------	----------------------------

Model set object.

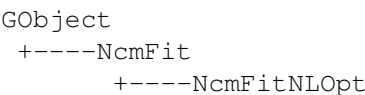
2.20.5 NLOpt Interface Object

NLOpt Interface Object — Interface for NLOpt optimization library

Synopsis

struct	NcmFitNLOptClass;	
struct	NcmFitNLOpt;	
NcmFit *	ncm_fit_nlopt_new	(NcmLikelihood *lh, NcmMSet *mset, NcmFitGradType gtype, nlopt_algorithm algo);
NcmFit *	ncm_fit_nlopt_local_new	(NcmLikelihood *lh, NcmMSet *mset, NcmFitGradType gtype, nlopt_algorithm algo, nlopt_algorithm local_algo);
void	ncm_fit_nlopt_set_algo	(NcmFitNLOpt *fit_nlopt, nlopt_algorithm algo);
void	ncm_fit_nlopt_set_local_algo	(NcmFitNLOpt *fit_nlopt, nlopt_algorithm algo);
NcmFit *	ncm_fit_nlopt_new_by_name	(NcmLikelihood *lh, NcmMSet *mset, NcmFitGradType gtype, gchar *algo_name);
NcmFit *	ncm_fit_nlopt_new_default	(NcmLikelihood *lh, NcmMSet *mset, NcmFitGradType gtype);

Object Hierarchy



Properties

"algorithm"	NcmFitNloptAlgorithm	: Read / Write / Construct
"local-algorithm"	NcmFitNloptAlgorithm	: Read / Write



**Description**

FIXME

**Details****struct NcmFitNLOptClass**

```
struct NcmFitNLOptClass {
};
```

**struct NcmFitNLOpt**

```
struct NcmFitNLOpt;
```

**ncm\_fit\_nlopt\_new ()**

```
NcmFit *          ncm_fit_nlopt_new          (NcmLikelihood *lh,
                                              NcmMSet *mset,
                                              NcmFitGradType gtype,
                                              nlopt_algorithm algo);
```

FIXME

**lh** : FIXME

**mset** : FIXME

**gtype** : FIXME

**algo** : FIXME

**Returns** : FIXME

**ncm\_fit\_nlopt\_local\_new ()**

```
NcmFit *          ncm_fit_nlopt_local_new    (NcmLikelihood *lh,
                                              NcmMSet *mset,
                                              NcmFitGradType gtype,
                                              nlopt_algorithm algo,
                                              nlopt_algorithm local_algo);
```

FIXME

**lh** : FIXME

**mset** : FIXME

**gtype** : FIXME

**algo** : FIXME

**local\_algo** : FIXME

**Returns** : FIXME

**ncm\_fit\_nlopt\_set\_algo ()**

```
void                ncm_fit_nlopt_set_algo                (NcmFitNLOpt *fit_nlopt,
                                                         nlopt_algorithm algo);
```

FIXME

**fit\_nlopt** : a **NcmFitNLOpt**.

**algo** : a **nlopt\_algorithm**.

**ncm\_fit\_nlopt\_set\_local\_algo ()**

```
void                ncm_fit_nlopt_set_local_algo          (NcmFitNLOpt *fit_nlopt,
                                                         nlopt_algorithm algo);
```

FIXME

**fit\_nlopt** : a **NcmFitNLOpt**.

**algo** : a **nlopt\_algorithm**.

**ncm\_fit\_nlopt\_new\_by\_name ()**

```
NcmFit *            ncm_fit_nlopt_new_by_name             (NcmLikelihood *lh,
                                                         NcmMSet *mset,
                                                         NcmFitGradType gtype,
                                                         gchar *algo_name);
```

FIXME

**lh** : FIXME

**mset** : FIXME

**gtype** : FIXME

**algo\_name** : FIXME

**Returns** : FIXME

**ncm\_fit\_nlopt\_new\_default ()**

```
NcmFit *            ncm_fit_nlopt_new_default            (NcmLikelihood *lh,
                                                         NcmMSet *mset,
                                                         NcmFitGradType gtype);
```

FIXME

**lh** : FIXME

**mset** : FIXME

**gtype** : FIXME

**Returns** : FIXME

## Property Details

### The "algorithm" property

"algorithm"	NcmFitNloptAlgorithm	: Read / Write / Construct
-------------	----------------------	----------------------------

NLOpt algorithm.

Default value: NLOPT\_LN\_NELDERMEAD

### The "local-algorithm" property

"local-algorithm"	NcmFitNloptAlgorithm	: Read / Write
-------------------	----------------------	----------------

NLOpt local algorithm.

Default value: NLOPT\_LN\_NELDERMEAD

## 2.20.6 Least Squares -- GSL

Least Squares -- GSL — Interface for GSL least squares algorithms

### Synopsis

```

struct          NcmFitGSLLSClass;
struct          NcmFitGSLLS;
NcmFit *        ncm_fit_gsl_ls_new          (NcmLikelihood *lh,
                                              NcmMSet *mset,
                                              NcmFitGradType gtype);

```

### Object Hierarchy

```

GObject
+----NcmFit
      +----NcmFitGSLLS

```

### Description

FIXME

### Details

#### struct NcmFitGSLLSClass

```

struct NcmFitGSLLSClass {
};

```

#### struct NcmFitGSLLS

```

struct NcmFitGSLLS;

```

---

**ncm\_fit\_gsl\_ls\_new ()**

```
NcmFit *          ncm_fit_gsl_ls_new          (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                NcmFitGradType gtype);
```

FIXME

**lh** : FIXME

**mset** : FIXME

**gtype** : FIXME

**Returns** : FIXME

**2.20.7 Non-linear Minimization -- GSL**

Non-linear Minimization -- GSL — Interface for non-linear minimization GSL algorithms

**Synopsis**

```
enum              NcmFitGSLMMAlgos;
struct            NcmFitGSLMMClass;
struct            NcmFitGSLMM;
NcmFit *          ncm_fit_gsl_mm_new          (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                NcmFitGradType gtype,
                                                NcmFitGSLMMAlgos algo);

NcmFit *          ncm_fit_gsl_mm_new_default  (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                NcmFitGradType gtype);

NcmFit *          ncm_fit_gsl_mm_new_by_name  (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                NcmFitGradType gtype,
                                                gchar *algo_name);

void              ncm_fit_gsl_mm_set_algo     (NcmFitGSLMM *fit_gsl_mm,
                                                NcmFitGSLMMAlgos algo);
```

**Object Hierarchy**

```
GObject
+-----NcmFit
+-----NcmFitGSLMM
```

**Properties**

```
"algorithm"          NcmFitGSLMMAlgos          : Read / Write / Construct
```

**Description**

FIXME

**Details****enum NcmFitGSLMMAlgos**

```
typedef enum {
    NCM_FIT_GSL_MM_CONJUGATE_FR = 0,
    NCM_FIT_GSL_MM_CONJUGATE_PR,
    NCM_FIT_GSL_MM_VECTOR_BFGS,
    NCM_FIT_GSL_MM_VECTOR_BFGS2,
} NcmFitGSLMMAlgos;
```

FIXME

**NCM\_FIT\_GSL\_MM\_CONJUGATE\_FR** FIXME

**NCM\_FIT\_GSL\_MM\_CONJUGATE\_PR** FIXME

**NCM\_FIT\_GSL\_MM\_VECTOR\_BFGS** FIXME

**NCM\_FIT\_GSL\_MM\_VECTOR\_BFGS2** FIXME

**NCM\_FIT\_GSL\_MM\_STEEPEST\_DESCENT** FIXME

**struct NcmFitGSLMMClass**

```
struct NcmFitGSLMMClass {
};
```

**struct NcmFitGSLMM**

```
struct NcmFitGSLMM;
```

**ncm\_fit\_gsl\_mm\_new ()**

```
NcmFit *          ncm_fit_gsl_mm_new          (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                NcmFitGradType gtype,
                                                NcmFitGSLMMAlgos algo);
```

FIXME

**lh** : FIXME

**mset** : FIXME

**gtype** : FIXME

**algo** : FIXME

**Returns** : FIXME

**ncm\_fit\_gsl\_mm\_new\_default ()**

```
NcmFit *          ncm_fit_gsl_mm_new_default      (NcmLikelihood *lh,
                                                    NcmMSet *mset,
                                                    NcmFitGradType gtype);
```

FIXME

*lh* : FIXME

*mset* : FIXME

*gtype* : FIXME

*Returns* : FIXME

**ncm\_fit\_gsl\_mm\_new\_by\_name ()**

```
NcmFit *          ncm_fit_gsl_mm_new_by_name      (NcmLikelihood *lh,
                                                    NcmMSet *mset,
                                                    NcmFitGradType gtype,
                                                    gchar *algo_name);
```

FIXME

*lh* : FIXME

*mset* : FIXME

*gtype* : FIXME

*algo\_name* : FIXME

*Returns* : FIXME

**ncm\_fit\_gsl\_mm\_set\_algo ()**

```
void              ncm_fit_gsl_mm_set_algo          (NcmFitGSLMM *fit_gsl_mm,
                                                    NcmFitGSLMMAlgos algo);
```

FIXME

*fit\_gsl\_mm* : a **NcmFitGSLMM**.

*algo* : a **gsl\_mm\_algorithm**.

**Property Details****The "algorithm" property**

"algorithm"	NcmFitGSLMMAlgos	: Read / Write / Construct
-------------	------------------	----------------------------

GSL multidimensional minimization algorithm.

Default value: NCM\_FIT\_GSL\_MM\_VECTOR\_BFGS2

**2.20.8 Non-linear Simplex Minimization -- GSL**

Non-linear Simplex Minimization -- GSL — Interface for GSL non-linear minimization (simplex) algorithms

## Synopsis

```

enum          NcmFitGSLMMSAlgos;
struct        NcmFitGSLMMSClass;
struct        NcmFitGSLMMS;
NcmFit *      ncm_fit_gsl_mms_new          (NcmLikelihood *lh,
                                           NcmMSet *mset,
                                           NcmFitGradType gtype,
                                           NcmFitGSLMMSAlgos algo);

NcmFit *      ncm_fit_gsl_mms_new_default (NcmLikelihood *lh,
                                           NcmMSet *mset,
                                           NcmFitGradType gtype);

NcmFit *      ncm_fit_gsl_mms_new_by_name (NcmLikelihood *lh,
                                           NcmMSet *mset,
                                           NcmFitGradType gtype,
                                           gchar *algo_name);

void          ncm_fit_gsl_mms_set_algo    (NcmFitGSLMMS *fit_gsl_mms,
                                           NcmFitGSLMMSAlgos algo);

```

## Object Hierarchy

```

GObject
+-----NcmFit
+-----NcmFitGSLMMS

```

## Properties

```

"algorithm"          NcmFitGSLMMSAlgos          : Read / Write / Construct

```

## Description

FIXME

## Details

### enum NcmFitGSLMMSAlgos

```

typedef enum {
    NCM_FIT_GSL_MMS_NMSIMPLEX2 = 0,
    NCM_FIT_GSL_MMS_NMSIMPLEX,
} NcmFitGSLMMSAlgos;

```

FIXME

**NCM\_FIT\_GSL\_MMS\_NMSIMPLEX2** FIXME

**NCM\_FIT\_GSL\_MMS\_NMSIMPLEX** FIXME

**NCM\_FIT\_GSL\_MMS\_NMSIMPLES2RAND** FIXME

### struct NcmFitGSLMMSClass

```

struct NcmFitGSLMMSClass {
};

```

**struct NcmFitGSLMMS**

```
struct NcmFitGSLMMS;
```

**ncm\_fit\_gsl\_mms\_new ()**

```
NcmFit *          ncm_fit_gsl_mms_new          (NcmLikelihood *lh,  
                                                NcmMSet *mset,  
                                                NcmFitGradType gtype,  
                                                NcmFitGSLMMSAlgos algo);
```

FIXME

*lh* : FIXME

*mset* : FIXME

*gtype* : FIXME

*algo* : FIXME

*Returns* : FIXME

**ncm\_fit\_gsl\_mms\_new\_default ()**

```
NcmFit *          ncm_fit_gsl_mms_new_default  (NcmLikelihood *lh,  
                                                NcmMSet *mset,  
                                                NcmFitGradType gtype);
```

FIXME

*lh* : FIXME

*mset* : FIXME

*gtype* : FIXME

*Returns* : FIXME

**ncm\_fit\_gsl\_mms\_new\_by\_name ()**

```
NcmFit *          ncm_fit_gsl_mms_new_by_name (NcmLikelihood *lh,  
                                                NcmMSet *mset,  
                                                NcmFitGradType gtype,  
                                                gchar *algo_name);
```

FIXME

*lh* : FIXME

*mset* : FIXME

*gtype* : FIXME

*algo\_name* : FIXME

*Returns* : FIXME



**ncm\_fit\_gsl\_mms\_set\_algo ()**

```
void                ncm_fit_gsl_mms_set_algo                (NcmFitGSLMMS *fit_gsl_mms,
                                                            NcmFitGSLMMSAlgos algo);
```

FIXME

**fit\_gsl\_mms**: a [NcmFitGSLMMS](#).

**algo**: a [gsl\\_mms\\_algorithm](#).

**Property Details****The "algorithm" property**

"algorithm"	NcmFitGSLMMSAlgos	: Read / Write / Construct
-------------	-------------------	----------------------------

GSL multidimensional minimization algorithm [simplex].

Default value: NCM\_FIT\_GSL\_MMS\_NMSIMPLEX2

**2.20.9 Least Squares -- Levmar**

Least Squares -- Levmar — Interface for Levenberg-Marquardt nonlinear least squares algorithm library

**Synopsis**

```
enum                NcmFitLevmarAlgos;
struct              NcmFitLevmarClass;
struct              NcmFitLevmar;
NcmFit *            ncm_fit_levmar_new                    (NcmLikelihood *lh,
                                                            NcmMSet *mset,
                                                            NcmFitGradType gtype,
                                                            NcmFitLevmarAlgos algo);

NcmFit *            ncm_fit_levmar_new_default            (NcmLikelihood *lh,
                                                            NcmMSet *mset,
                                                            NcmFitGradType gtype);

NcmFit *            ncm_fit_levmar_new_by_name            (NcmLikelihood *lh,
                                                            NcmMSet *mset,
                                                            NcmFitGradType gtype,
                                                            gchar *algo_name);

void                ncm_fit_levmar_set_algo                (NcmFitLevmar *fit_levmar,
                                                            NcmFitLevmarAlgos algo);
```

**Object Hierarchy**

```
GObject
+----NcmFit
      +----NcmFitLevmar
```

**Properties**

"algorithm"	NcmFitLevmarAlgos	: Read / Write / Construct
-------------	-------------------	----------------------------

**Description**

FIXME

**Details****enum NcmFitLevmarAlgos**

```
typedef enum {
    NCM_FIT_LEVMAR_DER = 0,
} NcmFitLevmarAlgos;
```

FIXME

**NCM\_FIT\_LEVMAR\_DER** FIXME

**NCM\_FIT\_LEVMAR\_DIF** FIXME

**struct NcmFitLevmarClass**

```
struct NcmFitLevmarClass {
};
```

**struct NcmFitLevmar**

```
struct NcmFitLevmar;
```

**ncm\_fit\_levmar\_new ()**

```
NcmFit *          ncm_fit_levmar_new          (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                NcmFitGradType gtype,
                                                NcmFitLevmarAlgos algo);
```

FIXME

**lh** : FIXME

**mset** : FIXME

**gtype** : FIXME

**algo** : FIXME

**Returns** : FIXME

**ncm\_fit\_levmar\_new\_default ()**

```
NcmFit *          ncm_fit_levmar_new_default (NcmLikelihood *lh,
                                                NcmMSet *mset,
                                                NcmFitGradType gtype);
```

FIXME

**lh** : FIXME

**mset** : FIXME

**gtype** : FIXME

**Returns** : FIXME



```
void          ncm_fit_mc_print          (NcmFitMC *mc);
void          ncm_fit_mc_mean_covar    (NcmFitMC *mc);
void          ncm_fit_mc_gof_pdf       (NcmFitMC *mc);
void          ncm_fit_mc_gof_pdf_print (NcmFitMC *mc);
gdouble       ncm_fit_mc_gof_pdf_pvalue (NcmFitMC *mc,
                                         gdouble m2lnL,
                                         gboolean both);
```

Object Hierarchy

```
GObject
+-----NcmFitMC
```

Properties

"fit"	NcmFit*	: Read / Write / Construct Only
-------	---------	---------------------------------

Description

FIXME

Details

struct NcmFitMCClass

```
struct NcmFitMCClass {
};
```

struct NcmFitMC

```
struct NcmFitMC;
```

ncm\_fit\_mc\_new ()

```
NcmFitMC *      ncm_fit_mc_new          (NcmFit *fit);
```

FIXME

*fit* : FIXME

*Returns* : FIXME

ncm\_fit\_mc\_free ()

```
void          ncm_fit_mc_free          (NcmFitMC *mc);
```

FIXME

*mc* : FIXME

**ncm\_fit\_mc\_clear ()**

```
void                ncm_fit_mc_clear                (NcmFitMC **mc);
```

FIXME

*mc* : FIXME

**ncm\_fit\_mc\_run ()**

```
void                ncm_fit_mc_run                (NcmFitMC *mc,  
                                                  NcmMSet *fiduc,  
                                                  guint ni,  
                                                  guint nf,  
                                                  NcmFitRunMsgs mtype);
```

FIXME

*mc* : FIXME

*fiduc* : FIXME

*ni* : FIXME

*nf* : FIXME

*mtype* : FIXME

**ncm\_fit\_mc\_print ()**

```
void                ncm_fit_mc_print                (NcmFitMC *mc);
```

FIXME

*mc* : FIXME

**ncm\_fit\_mc\_mean\_covar ()**

```
void                ncm_fit_mc_mean_covar                (NcmFitMC *mc);
```

FIXME

*mc* : FIXME

**ncm\_fit\_mc\_gof\_pdf ()**

```
void                ncm_fit_mc_gof_pdf                (NcmFitMC *mc);
```

FIXME

*mc* : FIXME

---



## Object Hierarchy

```
GObject
+----NcmLHRatio1d
```

## Properties

"constraint"	NcmMSetFunc*	: Read / Write / Construct
"fit"	NcmFit*	: Write / Construct Only
"pi"	NcmMSetPIndex*	: Read / Write / Construct

## Description

FIXME

## Details

### enum NcmLHRatio1dRoot

```
typedef enum {
    NCM_LH_RATIO1D_ROOT_BRACKET = 0,
    NCM_LH_RATIO1D_ROOT_NUMDIFF,
} NcmLHRatio1dRoot;
```

**NCM\_LH\_RATIO1D\_ROOT\_BRACKET** FIXME

**NCM\_LH\_RATIO1D\_ROOT\_NUMDIFF** FIXME

### struct NcmLHRatio1dClass

```
struct NcmLHRatio1dClass {
};
```

### struct NcmLHRatio1d

```
struct NcmLHRatio1d;
```

### ncm\_lh\_ratio1d\_new ()

```
NcmLHRatio1d *      ncm_lh_ratio1d_new      (NcmFit *fit,
                                              NcmMSetPIndex *pi);
```

FIXME

**fit**: FIXME

**pi**: FIXME

**Returns**: FIXME

**ncm\_lh\_ratio1d\_free ()**

```
void                ncm_lh_ratio1d_free                (NcmLHRatio1d *lhr1d);
```

FIXME

*lhr1d* : FIXME

**ncm\_lh\_ratio1d\_clear ()**

```
void                ncm_lh_ratio1d_clear                (NcmLHRatio1d **lhr1d);
```

FIXME

*lhr1d* : FIXME

**ncm\_lh\_ratio1d\_set\_pindex ()**

```
void                ncm_lh_ratio1d_set_pindex            (NcmLHRatio1d *lhr1d,
                                                         NcmMSetPIndex *pi);
```

FIXME

*lhr1d* : a **NcmLHRatio1d**.

*pi* : FIXME

**ncm\_lh\_ratio1d\_find\_bounds ()**

```
void                ncm_lh_ratio1d_find_bounds            (NcmLHRatio1d *lhr1d,
                                                         gdouble clevel,
                                                         NcmFitRunMsgs mtype,
                                                         gdouble *lb,
                                                         gdouble *ub);
```

FIXME

*lhr1d* : a **NcmLHRatio1d**.

*clevel* : The confidence level (0,1).

*mtype* : FIXME

*lb* : Lower bound. *[out]*

*ub* : Upper bound. *[out]*

**Property Details****The "constraint" property**

"constraint"	NcmMSetFunc*	: Read / Write / Construct
--------------	--------------	----------------------------

Constraint.



**The "fit" property**

"fit"	NcmFit*	: Write / Construct Only
-------	---------	--------------------------

NcmFit object.

**The "pi" property**

"pi"	NcmMSetPIndex*	: Read / Write / Construct
------	----------------	----------------------------

Param index.

**2.20.12 Likelihood Ratio 2D**

Likelihood Ratio 2D — Likelihood ratio object for bidimensional analysis.

**Synopsis**

```
enum                NcmLHRatio2dRoot;
struct              NcmLHRatio2dClass;
struct              NcmLHRatio2d;
struct              NcmLHRatio2dPoint;
struct              NcmLHRatio2dRegion;
NcmLHRatio2d *      ncm_lh_ratio2d_new                (NcmFit *fit,
                                                         NcmMSetPIndex *pi1,
                                                         NcmMSetPIndex *pi2);
void                ncm_lh_ratio2d_free                (NcmLHRatio2d *lhr2d);
void                ncm_lh_ratio2d_clear              (NcmLHRatio2d **lhr2d);
void                ncm_lh_ratio2d_set_pindex         (NcmLHRatio2d *lhr2d,
                                                         NcmMSetPIndex *pi1,
                                                         NcmMSetPIndex *pi2);
NcmLHRatio2dRegion * ncm_lh_ratio2d_conf_region       (NcmLHRatio2d *lhr2d,
                                                         gdouble clevel,
                                                         gdouble expected_np,
                                                         NcmFitRunMsgs mtype);
NcmLHRatio2dRegion * ncm_lh_ratio2d_fisher_border    (NcmLHRatio2d *lhr2d,
                                                         gdouble clevel,
                                                         gdouble expected_np,
                                                         NcmFitRunMsgs mtype);
NcmLHRatio2dRegion * ncm_lh_ratio2d_region_dup       (NcmLHRatio2dRegion *rg);
void                ncm_lh_ratio2d_region_free       (NcmLHRatio2dRegion *rg);
void                ncm_lh_ratio2d_region_clear      (NcmLHRatio2dRegion **rg);
void                ncm_lh_ratio2d_region_print      (NcmLHRatio2dRegion *rg,
                                                         FILE *out);
```

**Object Hierarchy**

```
GObject
+----NcmLHRatio2d

GBoxed
+----NcmLHRatio2dRegion
```

**Properties**

"fit"	NcmFit*	: Write / Construct Only
"pi1"	NcmMSetPIndex*	: Read / Write / Construct
"pi2"	NcmMSetPIndex*	: Read / Write / Construct

**Description**

FIXME

**Details****enum NcmLHRatio2dRoot**

```
typedef enum {
    NCM_LH_RATIO2D_ROOT_BRACKET = 0,
    NCM_LH_RATIO2D_ROOT_NUMDIFF,
} NcmLHRatio2dRoot;
```

**NCM\_LH\_RATIO2D\_ROOT\_BRACKET** FIXME

**NCM\_LH\_RATIO2D\_ROOT\_NUMDIFF** FIXME

**struct NcmLHRatio2dClass**

```
struct NcmLHRatio2dClass {
};
```

**struct NcmLHRatio2d**

```
struct NcmLHRatio2d;
```

**struct NcmLHRatio2dPoint**

```
struct NcmLHRatio2dPoint {
};
```

FIXME

**struct NcmLHRatio2dRegion**

```
struct NcmLHRatio2dRegion {
    guint np;
    NcmVector *p1;
    NcmVector *p2;
    gdouble clevel;
};
```

Object describing a confidence region.

**guint** *np*; Number of points.

**NcmVector** \**p1*; a **NcmVector** containing points of parameter one.

**NcmVector** \**p2*; a **NcmVector** containing points of parameter two.

**gdouble** *clevel*; the confidence level represented by the border.

**ncm\_lh\_ratio2d\_new ()**

```
NcmLHRatio2d *      ncm_lh_ratio2d_new      (NcmFit *fit,
                                             NcmMSetPIndex *pi1,
                                             NcmMSetPIndex *pi2);
```

FIXME

**fit** : FIXME

**pi1** : FIXME

**pi2** : FIXME

**Returns** : FIXME

**ncm\_lh\_ratio2d\_free ()**

```
void              ncm_lh_ratio2d_free      (NcmLHRatio2d *lhr2d);
```

FIXME

**lhr2d** : FIXME

**ncm\_lh\_ratio2d\_clear ()**

```
void              ncm_lh_ratio2d_clear    (NcmLHRatio2d **lhr2d);
```

FIXME

**lhr2d** : FIXME

**ncm\_lh\_ratio2d\_set\_pindex ()**

```
void              ncm_lh_ratio2d_set_pindex (NcmLHRatio2d *lhr2d,
                                             NcmMSetPIndex *pi1,
                                             NcmMSetPIndex *pi2);
```

FIXME

**lhr2d** : a **NcmLHRatio2d**.

**pi1** : FIXME

**pi2** : FIXME

**ncm\_lh\_ratio2d\_conf\_region ()**

```
NcmLHRatio2dRegion * ncm_lh_ratio2d_conf_region (NcmLHRatio2d *lhr2d,
                                                  gdouble clevel,
                                                  gdouble expected_np,
                                                  NcmFitRunMsgs mtype);
```

FIXME

**lhr2d** : a *NcmFit*

**clevel** : FIXME

**expected\_np** : Expected number of points, if lesser than 1 it uses the default value of 100.

**mtime** : FIXME

**Returns** : FIXME. *[transfer full]*

**ncm\_lh\_ratio2d\_fisher\_border ()**

```
NcmLHRatio2dRegion * ncm_lh_ratio2d_fisher_border      (NcmLHRatio2d *lhr2d,
                                                         gdouble clevel,
                                                         gdouble expected_np,
                                                         NcmFitRunMsgs mtype);
```

FIXME

**lhr2d** : a **NcmFit**.

**clevel** : FIXME

**expected\_np** : Expected number of points, if lesser than 1 it uses the default value of 600.

**mtime** : FIXME

**Returns** : FIXME. *[transfer full]*

**ncm\_lh\_ratio2d\_region\_dup ()**

```
NcmLHRatio2dRegion * ncm_lh_ratio2d_region_dup      (NcmLHRatio2dRegion *rg);
```

FIXME

**rg** : a **NcmLHRatio2dRegion**.

**Returns** : FIXME. *[transfer full]*

**ncm\_lh\_ratio2d\_region\_free ()**

```
void                ncm_lh_ratio2d_region_free      (NcmLHRatio2dRegion *rg);
```

FIXME

**rg** : a **NcmLHRatio2dRegion**.

**ncm\_lh\_ratio2d\_region\_clear ()**

```
void                ncm_lh_ratio2d_region_clear      (NcmLHRatio2dRegion **rg);
```

FIXME

**rg** : a **NcmLHRatio2dRegion**.

**ncm\_lh\_ratio2d\_region\_print ()**

```
void                ncm_lh_ratio2d_region_print      (NcmLHRatio2dRegion *rg,
                                                    FILE *out);
```

FIXME

*rg* : FIXME

*out* : FIXME

**Property Details**

**The "fit" property**

"fit"	NcmFit*	: Write / Construct Only
-------	---------	--------------------------

NcmFit object.

**The "pi1" property**

"pi1"	NcmMSetPIndex*	: Read / Write / Construct
-------	----------------	----------------------------

First param index.

**The "pi2" property**

"pi2"	NcmMSetPIndex*	: Read / Write / Construct
-------	----------------	----------------------------

Second param index.

**2.21 GObject introspection compatibility**

**2.21.1 Gir Scanning Compatibility.**

Gir Scanning Compatibility. — Gir scanning types stubs

**Synopsis**

```
#define                mpq_ptr
#define                mpq_t
#define                mp_rnd_t
#define                mpz_t
#define                mpfr_ptr
#define                mpfr_t
#define                CVRhsFn
#define                CVDlsDenseJacFn
#define                fftw_plan
#define                fftw_complex
```

## Description

Stubs to avoid warnings from gir scanning all functions/structs using these types must be skipped using (skip). These types do not represent anything, do not use this documentation.

## Details

### mpq\_ptr

```
#define mpq_ptr gint
```

### mpq\_t

```
#define mpq_t gint
```

### mp\_rnd\_t

```
#define mp_rnd_t gint
```

### mpz\_t

```
#define mpz_t gint
```

### mpfr\_ptr

```
#define mpfr_ptr gint
```

### mpfr\_t

```
#define mpfr_t gint
```

### CVRhsFn

```
#define CVRhsFn gint
```

### CVDlsDenseJacFn

```
#define CVDlsDenseJacFn gint
```

### fftw\_plan

```
#define fftw_plan gint
```

### fftw\_complex

```
#define fftw_complex gint
```

## Chapter 3

# Other Utilities Objects

### 3.1 Healpix

Healpix — Healpix re-implementation

#### Synopsis

NcmSphereMap *	ncm_sphere_healpix_read_map	(gchar *fits_file, NcmSphereMap *map);
gboolean	ncm_sphere_healpix_write_map	(NcmSphereMap *map, gchar *filename, gboolean overwrite);
glong	ncm_sphere_healpix_nest2ring	(gint nside, glong nest_index);
glong	ncm_sphere_healpix_ring2nest	(gint nside, glong ring_index);
void	ncm_sphere_healpix_pix2ang_nest	(gint nside, glong nest_index, gdouble *theta, gdouble *phi);
void	ncm_sphere_healpix_pix2ang_ring	(gint nside, glong ring_index, gdouble *theta, gdouble *phi);
void	ncm_sphere_healpix_pix2vec_nest	(gint nside, glong nest_index, NcmTriVector vec);
void	ncm_sphere_healpix_pix2vec_ring	(gint nside, glong ring_index, NcmTriVector v);
void	ncm_sphere_healpix_vec2pix_ring	(gint nside, NcmTriVector v, glong *i);
#define	HEALPIX_NPIX	(nside)
#define	HEALPIX_INT_TO_XY	(i, x, y)
#define	HEALPIX_XY_TO_INT	(x, y, i)

```
#define NCM_HEALPIX_NULLVAL
```

## Description

FIXME

## Details

### ncm\_sphere\_healpix\_read\_map ()

```
NcmSphereMap * ncm_sphere_healpix_read_map (gchar *fits_file,  
                                             NcmSphereMap *map);
```

FIXME

***fits\_file***: FIXME

***map***: a [NcmSphereMap](#)

***Returns***: FIXME

### ncm\_sphere\_healpix\_write\_map ()

```
gboolean ncm_sphere_healpix_write_map (NcmSphereMap *map,  
                                       gchar *filename,  
                                       gboolean overwrite);
```

FIXME

***map***: a [NcmSphereMap](#)

***filename***: FIXME

***overwrite***: FIXME

***Returns***: FIXME

### ncm\_sphere\_healpix\_nest2ring ()

```
glong ncm_sphere_healpix_nest2ring (gint nside,  
                                     glong nest_index);
```

FIXME

***nside***: FIXME

***nest\_index***: FIXME

***Returns***: FIXME



**ncm\_sphere\_healpix\_ring2nest ()**

```
glong          ncm_sphere_healpix_ring2nest      (gint nside,
                                                  glong ring_index);
```

FIXME

***nside***: FIXME

***ring\_index***: FIXME

***Returns***: FIXME

**ncm\_sphere\_healpix\_pix2ang\_nest ()**

```
void          ncm_sphere_healpix_pix2ang_nest    (gint nside,
                                                  glong nest_index,
                                                  gdouble *theta,
                                                  gdouble *phi);
```

FIXME

***nside***: FIXME

***nest\_index***: FIXME

***theta***: FIXME

***phi***: FIXME

**ncm\_sphere\_healpix\_pix2ang\_ring ()**

```
void          ncm_sphere_healpix_pix2ang_ring    (gint nside,
                                                  glong ring_index,
                                                  gdouble *theta,
                                                  gdouble *phi);
```

FIXME

***nside***: FIXME

***ring\_index***: FIXME

***theta***: FIXME

***phi***: FIXME

**ncm\_sphere\_healpix\_pix2vec\_nest ()**

```
void          ncm_sphere_healpix_pix2vec_nest    (gint nside,
                                                  glong nest_index,
                                                  NcmTriVector vec);
```

FIXME

***nside***: FIXME

***nest\_index***: FIXME

***vec***: a **NcmTriVector**

**ncm\_sphere\_healpix\_pix2vec\_ring ()**

```
void          ncm_sphere_healpix_pix2vec_ring      (gint nside,
                                                    glong ring_index,
                                                    NcmTriVector v);
```

FIXME

**nside**: FIXME

**ring\_index**: FIXME

**v**: a **NcmTriVector**

**ncm\_sphere\_healpix\_vec2pix\_ring ()**

```
void          ncm_sphere_healpix_vec2pix_ring      (gint nside,
                                                    NcmTriVector v,
                                                    glong *i);
```

FIXME

**nside**: FIXME

**v**: a **NcmTriVector**

**i**: FIXME

**HEALPIX\_NPIX()**

```
#define HEALPIX_NPIX(nside) (12*(nside)*(nside))
```

**HEALPIX\_INT\_TO\_XY()**

```
#define          HEALPIX_INT_TO_XY(i,x,y)
```

**HEALPIX\_XY\_TO\_INT()**

```
#define          HEALPIX_XY_TO_INT(x,y,i)
```

**NCM\_HEALPIX\_NULLVAL**

```
#define NCM_HEALPIX_NULLVAL (-1.6375e30) /* check if its ok to copy it here FIXME*/
```

## 3.2 Spherical Shell Map

Spherical Shell Map — Object representing a spherical shell map

## Synopsis

```

enum                NcmSphereMapOrder;
enum                NcmSphereMapType;
struct              NcmSphereMap;
struct              NcmSphereMapAlm;
struct              NcmSphereMapSHT;
NcmSphereMap *      ncm_sphere_map_new          (gint nside);
NcmSphereMap *      ncm_sphere_map_clone        (NcmSphereMap *map);
gboolean            ncm_sphere_map_copy         (NcmSphereMap *dest,
                                                NcmSphereMap *orig);
gboolean            ncm_sphere_map_init_coord   (NcmSphereMap *map);
gboolean            ncm_sphere_map_set_order    (NcmSphereMap *map,
                                                NcmSphereMapOrder order,
                                                gboolean init_coord);

NcmSphereMapAlm *   ncm_sphere_mapalm_new       (void);
gboolean            ncm_sphere_mapalm_init      (NcmSphereMapAlm *mapalm,
                                                gint lmax);
NcmSphereMapSHT *   ncm_sphere_mapsht_new      (NcmSphereMap *map,
                                                NcmSphereMapAlm *mapalm,
                                                guint fftw_flags);
gboolean            ncm_sphere_mapsht_map2alm_circle (NcmSphereMapSHT *mapsht,
                                                gint ring,
                                                gint ring_size,
                                                gdouble norma,
                                                gdouble theta,
                                                gdouble phi,
                                                gint start_m,
                                                gint end_m);
gboolean            ncm_sphere_mapsht_alm2map_circle (NcmSphereMapSHT *mapsht,
                                                gint ring,
                                                gint ring_size,
                                                gdouble theta,
                                                gdouble phi);
gboolean            ncm_sphere_mapsht_map2alm     (NcmSphereMapSHT *mapsht,
                                                gdouble cut);
gboolean            ncm_sphere_mapsht_alm2map     (NcmSphereMapSHT *mapsht);
gdouble            ncm_sphere_map_homogenize_noise (NcmSphereMap *map,
                                                gdouble base_sigma);
gdouble            ncm_sphere_map_rotate_avg     (NcmSphereMap *map,
                                                glong n);

```

## Description

Map manipulation algorithms, Ylm decomposition.

## Details

### enum NcmSphereMapOrder

```

typedef enum {
    NC_SPHERE_MAP_ORDER_NEST,
    NC_SPHERE_MAP_ORDER_RING
} NcmSphereMapOrder;

```

**NC\_SPHERE\_MAP\_ORDER\_NEST** FIXME

**NC\_SPHERE\_MAP\_ORDER\_RING** FIXME

### enum NcmSphereMapType

```
typedef enum {
    NC_SPHERE_MAP_TYPE_TEMPERATURE      = 1 << 0,
    NC_SPHERE_MAP_TYPE_Q_POLARIZATION   = 1 << 1,
    NC_SPHERE_MAP_TYPE_U_POLARISATION   = 1 << 2,
    NC_SPHERE_MAP_TYPE_SPUR_SIGNAL       = 1 << 3,
    NC_SPHERE_MAP_TYPE_N_OBS             = 1 << 4
} NcmSphereMapType;
```

FIXME

**NC\_SPHERE\_MAP\_TYPE\_TEMPERATURE** FIXME

**NC\_SPHERE\_MAP\_TYPE\_Q\_POLARIZATION** FIXME

**NC\_SPHERE\_MAP\_TYPE\_U\_POLARISATION** FIXME

**NC\_SPHERE\_MAP\_TYPE\_SPUR\_SIGNAL** FIXME

**NC\_SPHERE\_MAP\_TYPE\_N\_OBS** FIXME

### struct NcmSphereMap

```
struct NcmSphereMap {
};
```

FIXME

### struct NcmSphereMapAlm

```
struct NcmSphereMapAlm {
};
```

FIXME

### struct NcmSphereMapSHT

```
struct NcmSphereMapSHT {
};
```

FIXME

### ncm\_sphere\_map\_new ()

```
NcmSphereMap *      ncm_sphere_map_new      (gint nside);
```

FIXME

***nside*** : FIXME

***Returns*** : FIXME

**ncm\_sphere\_map\_clone ()**

```
NcmSphereMap *      ncm_sphere_map_clone      (NcmSphereMap *map);
```

FIXME

**map** : a **NcmSphereMap****Returns** : FIXME**ncm\_sphere\_map\_copy ()**

```
gboolean            ncm_sphere_map_copy      (NcmSphereMap *dest,  
                                              NcmSphereMap *orig);
```

FIXME

**dest** : a **NcmSphereMap****orig** : a **NcmSphereMap****Returns** : FIXME**ncm\_sphere\_map\_init\_coord ()**

```
gboolean            ncm_sphere_map_init_coord (NcmSphereMap *map);
```

FIXME

**map** : a **NcmSphereMap****Returns** : FIXME**ncm\_sphere\_map\_set\_order ()**

```
gboolean            ncm_sphere_map_set_order (NcmSphereMap *map,  
                                              NcmSphereMapOrder order,  
                                              gboolean init_coord);
```

FIXME

**map** : a **NcmSphereMap****order** : a **NcmSphereMapOrder****init\_coord** : FIXME**Returns** : FIXME**ncm\_sphere\_mapalm\_new ()**

```
NcmSphereMapAlm *   ncm_sphere_mapalm_new   (void);
```

FIXME

**Returns** : FIXME

**ncm\_sphere\_mapalm\_init ()**

```
gboolean          ncm_sphere_mapalm_init          (NcmSphereMapAlm *mapalm,
                                                    gint lmax);
```

FIXME

**mapalm** : a **NcmSphereMapAlm**

**lmax** : FIXME

**Returns** : FIXME

**ncm\_sphere\_mapsht\_new ()**

```
NcmSphereMapSHT *  ncm_sphere_mapsht_new          (NcmSphereMap *map,
                                                    NcmSphereMapAlm *mapalm,
                                                    guint fftw_flags);
```

FIXME

**map** : a **NcmSphereMap**

**mapalm** : a **NcmSphereMapAlm**

**fftw\_flags** : FIXME

**Returns** : FIXME

**ncm\_sphere\_mapsht\_map2alm\_circle ()**

```
gboolean          ncm_sphere_mapsht_map2alm_circle (NcmSphereMapSHT *mapsht,
                                                    gint ring,
                                                    gint ring_size,
                                                    gdouble norma,
                                                    gdouble theta,
                                                    gdouble phi,
                                                    gint start_m,
                                                    gint end_m);
```

Transform the map to alm circle by circle using fft in each one Copied from gsl-1.11 specfunc/legendre\_poly.c line 596 And then adapted...

**mapsht** : a **NcmSphereMapSHT**

**ring** : FIXME

**ring\_size** : FIXME

**norma** : FIXME

**theta** : FIXME

**phi** : FIXME

**start\_m** : FIXME

**end\_m** : FIXME

**Returns** : FIXME

**ncm\_sphere\_mapsht\_alm2map\_circle ()**

gboolean	ncm_sphere_mapsht_alm2map_circle	(NcmSphereMapSHT *mapsht, gint ring, gint ring_size, gdouble theta, gdouble phi);
----------	----------------------------------	---

Transform the map to alm circle by circle using fft in each one Copied from gsl-1.11 specfunc/legendre\_poly.c line 596 And then adapted... And then adapted again...

**mapsht** : a **NcmSphereMapSHT**

**ring** : FIXME

**ring\_size** : FIXME

**theta** : FIXME

**phi** : FIXME

**Returns** : FIXME

**ncm\_sphere\_mapsht\_map2alm ()**

gboolean	ncm_sphere_mapsht_map2alm	(NcmSphereMapSHT *mapsht, gdouble cut);
----------	---------------------------	--

Transform the map to alm circle by circle using fft in each one

**mapsht** : a **NcmSphereMapSHT**

**cut** : FIXME

**Returns** : FIXME

**ncm\_sphere\_mapsht\_alm2map ()**

gboolean	ncm_sphere_mapsht_alm2map	(NcmSphereMapSHT *mapsht);
----------	---------------------------	----------------------------

FIXME

**mapsht** : a **NcmSphereMapSHT**

**Returns** : FIXME

**ncm\_sphere\_map\_homogenize\_noise ()**

gdouble	ncm_sphere_map_homogenize_noise	(NcmSphereMap *map, gdouble base_sigma);
---------	---------------------------------	---

FIXME

**map** : a **NcmSphereMap**

**base\_sigma** : FIXME

**Returns** : FIXME

**ncm\_sphere\_map\_rotate\_avg ()**

```
gdouble          ncm_sphere_map_rotate_avg      (NcmSphereMap *map,  
                                                  glong n);
```

FIXME

*map* : a **NcmSphereMap**

*n* : FIXME

*Returns* : FIXME



## Chapter 4

# Models

### 4.1 Cosmological Model Abstract Class

Cosmological Model Abstract Class — Class for implementing homogeneous and isotropic cosmological models

#### Synopsis

```
enum                NcHICosmoImpl;
gdouble             (*NcHICosmoFunc0)           (NcHICosmo *cosmo);
gdouble             (*NcHICosmoFunc1)           (NcHICosmo *cosmo,
gdouble             gdouble x);

struct              NcHICosmoClass;
struct              NcHICosmo;
gdouble             nc_hicosmo_H0                (NcHICosmo *cosmo);
gdouble             nc_hicosmo_Omega_b            (NcHICosmo *cosmo);
gdouble             nc_hicosmo_Omega_r            (NcHICosmo *cosmo);
gdouble             nc_hicosmo_Omega_c            (NcHICosmo *cosmo);
gdouble             nc_hicosmo_Omega_t            (NcHICosmo *cosmo);
gdouble             nc_hicosmo_T_gamma0           (NcHICosmo *cosmo);
gdouble             nc_hicosmo_sigma_8           (NcHICosmo *cosmo);
gdouble             nc_hicosmo_z_lss             (NcHICosmo *cosmo);
gdouble             nc_hicosmo_as_drag           (NcHICosmo *cosmo);
gdouble             nc_hicosmo_E2                (NcHICosmo *cosmo,
gdouble             gdouble x);
gdouble             nc_hicosmo_dE2_dz            (NcHICosmo *cosmo,
gdouble             gdouble x);
gdouble             nc_hicosmo_d2E2_dz2          (NcHICosmo *cosmo,
gdouble             gdouble x);
gdouble             nc_hicosmo_cd                (NcHICosmo *cosmo,
gdouble             gdouble x);
gdouble             nc_hicosmo_powspec           (NcHICosmo *cosmo,
gdouble             gdouble x);
gdouble             nc_hicosmo_c_H0              (NcHICosmo *cosmo);
gdouble             nc_hicosmo_Omega_k           (NcHICosmo *cosmo);
gdouble             nc_hicosmo_Omega_m           (NcHICosmo *cosmo);
gdouble             nc_hicosmo_h                 (NcHICosmo *cosmo);
gdouble             nc_hicosmo_h2                (NcHICosmo *cosmo);
gdouble             nc_hicosmo_Omega_bh2         (NcHICosmo *cosmo);
gdouble             nc_hicosmo_Omega_ch2         (NcHICosmo *cosmo);
gdouble             nc_hicosmo_Omega_rh2         (NcHICosmo *cosmo);
```

---

gdouble	nc_hicosmo_Omega_mh2	(NcHICosmo *cosmo);
gdouble	nc_hicosmo_E	(NcHICosmo *cosmo, gdouble z);
gdouble	nc_hicosmo_Em2	(NcHICosmo *cosmo, gdouble z);
gdouble	nc_hicosmo_H	(NcHICosmo *cosmo, gdouble z);
gdouble	nc_hicosmo_dH_dz	(NcHICosmo *cosmo, gdouble z);
gdouble	nc_hicosmo_j	(NcHICosmo *cosmo, gdouble z);
gdouble	nc_hicosmo_qp	(NcHICosmo *cosmo, gdouble z);
gdouble	nc_hicosmo_q	(NcHICosmo *cosmo, gdouble z);
gdouble	nc_hicosmo_dec	(NcHICosmo *cosmo, gdouble z);
gdouble	nc_hicosmo_wec	(NcHICosmo *cosmo, gdouble z);
NcHICosmo *	nc_hicosmo_new_from_name	(GType parent_type, gchar *cosmo_name);
void	nc_hicosmo_log_all_models	(GType parent);
void	nc_hicosmo_free	(NcHICosmo *hic);
NcmMSetFunc *	nc_hicosmo_create_mset_func0	(NcHICosmoFunc0 f0);
NcmMSetFunc *	nc_hicosmo_create_mset_func1	(NcHICosmoFunc1 f1);
void	nc_hicosmo_set_H0_impl	(NcHICosmoClass *model_class, NcmModelFunc0 f);
void	nc_hicosmo_set_Omega_b_impl	(NcHICosmoClass *model_class, NcmModelFunc0 f);
void	nc_hicosmo_set_Omega_r_impl	(NcHICosmoClass *model_class, NcmModelFunc0 f);
void	nc_hicosmo_set_Omega_c_impl	(NcHICosmoClass *model_class, NcmModelFunc0 f);
void	nc_hicosmo_set_Omega_t_impl	(NcHICosmoClass *model_class, NcmModelFunc0 f);
void	nc_hicosmo_set_sigma_8_impl	(NcHICosmoClass *model_class, NcmModelFunc0 f);
void	nc_hicosmo_set_T_gamma0_impl	(NcHICosmoClass *model_class, NcmModelFunc0 f);
void	nc_hicosmo_set_z_lss_impl	(NcHICosmoClass *model_class, NcmModelFunc0 f);
void	nc_hicosmo_set_as_drag_impl	(NcHICosmoClass *model_class, NcmModelFunc0 f);
void	nc_hicosmo_set_E2_impl	(NcHICosmoClass *model_class, NcmModelFunc1 f);
void	nc_hicosmo_set_dE2_dz_impl	(NcHICosmoClass *model_class, NcmModelFunc1 f);
void	nc_hicosmo_set_d2E2_dz2_impl	(NcHICosmoClass *model_class, NcmModelFunc1 f);
void	nc_hicosmo_set_cd_impl	(NcHICosmoClass *model_class, NcmModelFunc1 f);
void	nc_hicosmo_set_powspec_impl	(NcHICosmoClass *model_class, NcmModelFunc1 f);
#define	NC_HICOSMO_DEFAULT_PARAMS_RELTOL	
#define	NC_HICOSMO_DEFAULT_PARAMS_ABSTOL	

---

## Object Hierarchy

```

GObject
+-----NcmModel
+-----NcHICosmo
+-----NcHICosmoDE
+-----NcHICosmoLCDM
+-----NcHICosmoQConst
+-----NcHICosmoQLinear
+-----NcHICosmoQPW
+-----NcHICosmoQSpline

```

## Description

FIXME

## Details

### enum NcHICosmoImpl

```

typedef enum {
    NC_HICOSMO_IMPL_H0           = 1 << 0,
    NC_HICOSMO_IMPL_Omega_b      = 1 << 1,
    NC_HICOSMO_IMPL_Omega_r      = 1 << 2,
    NC_HICOSMO_IMPL_Omega_c      = 1 << 3,
    NC_HICOSMO_IMPL_Omega_t      = 1 << 4,
    NC_HICOSMO_IMPL_sigma_8      = 1 << 5,
    NC_HICOSMO_IMPL_T_gamma0     = 1 << 6,
    NC_HICOSMO_IMPL_z_lss        = 1 << 7,
    NC_HICOSMO_IMPL_as_drag      = 1 << 8,
    NC_HICOSMO_IMPL_E2           = 1 << 9,
    NC_HICOSMO_IMPL_dE2_dz       = 1 << 10,
    NC_HICOSMO_IMPL_d2E2_dz2     = 1 << 11,
    NC_HICOSMO_IMPL_cd           = 1 << 12,
} NcHICosmoImpl;

```

FIXME

**NC\_HICOSMO\_IMPL\_H0** FIXME

**NC\_HICOSMO\_IMPL\_Omega\_b** FIXME

**NC\_HICOSMO\_IMPL\_Omega\_r** FIXME

**NC\_HICOSMO\_IMPL\_Omega\_c** FIXME

**NC\_HICOSMO\_IMPL\_Omega\_t** FIXME

**NC\_HICOSMO\_IMPL\_sigma\_8** FIXME

**NC\_HICOSMO\_IMPL\_T\_gamma0** Radiation temperature today

**NC\_HICOSMO\_IMPL\_z\_lss** Redshift of the last scattering surface

**NC\_HICOSMO\_IMPL\_as\_drag** Acoustic Scale at drag redshift

**NC\_HICOSMO\_IMPL\_E2** Adimensional Hubble function squared

**NC\_HICOSMO\_IMPL\_dE2\_dz** FIXME

**NC\_HICOSMO\_IMPL\_d2E2\_dz2** FIXME

**NC\_HICOSMO\_IMPL\_cd** Comoving distance

**NC\_HICOSMO\_IMPL\_powspec** Perturbations power spectrum

**NcHICosmoFunc0 ()**

```
gdouble (*NcHICosmoFunc0) (NcHICosmo *cosmo);
```

**NcHICosmoFunc1 ()**

```
gdouble (*NcHICosmoFunc1) (NcHICosmo *cosmo,
                             gdouble x);
```

**struct NcHICosmoClass**

```
struct NcHICosmoClass {
};
```

**struct NcHICosmo**

```
struct NcHICosmo;
```

FIXME

**nc\_hicosmo\_H0 ()**

```
gdouble nc_hicosmo_H0 (NcHICosmo *cosmo);
```

The value of the Hubble constant in unity of  $\text{ms}^{-1}\text{kpc}^{-1}$ .

**cosmo** : a **NcHICosmo**.

**Returns** :  $H_0$

**nc\_hicosmo\_Omega\_b ()**

```
gdouble nc_hicosmo_Omega_b (NcHICosmo *cosmo);
```

FIXME

**cosmo** : a **NcHICosmo**.

**Returns** :  $\Omega_b$

**nc\_hicosmo\_Omega\_r ()**

```
gdouble nc_hicosmo_Omega_r (NcHICosmo *cosmo);
```

FIXME

**cosmo** : a **NcHICosmo**.

**Returns** :  $\Omega_r$

**nc\_hicosmo\_Omega\_c ()**

```
gdouble          nc_hicosmo_Omega_c          (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**.*Returns* :  $\Omega_c$ **nc\_hicosmo\_Omega\_t ()**

```
gdouble          nc_hicosmo_Omega_t          (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**.*Returns* :  $\Omega_t$ **nc\_hicosmo\_T\_gamma0 ()**

```
gdouble          nc_hicosmo_T_gamma0         (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**.*Returns* :  $T_{\gamma 0}$ **nc\_hicosmo\_sigma\_8 ()**

```
gdouble          nc_hicosmo_sigma_8          (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**.*Returns* :  $\sigma_8$ **nc\_hicosmo\_z\_lss ()**

```
gdouble          nc_hicosmo_z_lss            (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**.*Returns* : FIXME**nc\_hicosmo\_as\_drag ()**

```
gdouble          nc_hicosmo_as_drag          (NcHICosmo *cosmo);
```

**nc\_hicosmo\_E2 ()**

gdouble	nc_hicosmo_E2	(NcHICosmo *cosmo, gdouble x);
---------	---------------	-----------------------------------

FIXME

*cosmo* : a **NcHICosmo**.*x* : FIXME*Returns* : FIXME**nc\_hicosmo\_dE2\_dz ()**

gdouble	nc_hicosmo_dE2_dz	(NcHICosmo *cosmo, gdouble x);
---------	-------------------	-----------------------------------

FIXME

*cosmo* : a **NcHICosmo**.*x* : FIXME*Returns* : FIXME**nc\_hicosmo\_d2E2\_dz2 ()**

gdouble	nc_hicosmo_d2E2_dz2	(NcHICosmo *cosmo, gdouble x);
---------	---------------------	-----------------------------------

FIXME

*cosmo* : a **NcHICosmo**.*x* : FIXME*Returns* : FIXME**nc\_hicosmo\_cd ()**

gdouble	nc_hicosmo_cd	(NcHICosmo *cosmo, gdouble x);
---------	---------------	-----------------------------------

**nc\_hicosmo\_powspec ()**

gdouble	nc_hicosmo_powspec	(NcHICosmo *cosmo, gdouble x);
---------	--------------------	-----------------------------------

FIXME

*cosmo* : a **NcHICosmo**.*x* : FIXME*Returns* : FIXME

**nc\_hicosmo\_c\_H0 ()**

```
gdouble          nc_hicosmo_c_H0          (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**.

**Returns** : FIXME

**nc\_hicosmo\_Omega\_k ()**

```
gdouble          nc_hicosmo_Omega_k      (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**.

**Returns** : FIXME

**nc\_hicosmo\_Omega\_m ()**

```
gdouble          nc_hicosmo_Omega_m      (NcHICosmo *cosmo);
```

The matter density parameter is given by the baryonic plus the cold dark matter density parameters.

*cosmo* : a **NcHICosmo**.

**Returns** : The matter density parameter at redshift zero.

**nc\_hicosmo\_h ()**

```
gdouble          nc_hicosmo_h            (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**

**Returns** : FIXME

**nc\_hicosmo\_h2 ()**

```
gdouble          nc_hicosmo_h2           (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**

**Returns** : FIXME

**nc\_hicosmo\_Omega\_bh2 ()**

```
gdouble          nc_hicosmo_Omega_bh2          (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**

*Returns* : FIXME

**nc\_hicosmo\_Omega\_ch2 ()**

```
gdouble          nc_hicosmo_Omega_ch2          (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**

*Returns* : FIXME

**nc\_hicosmo\_Omega\_rh2 ()**

```
gdouble          nc_hicosmo_Omega_rh2          (NcHICosmo *cosmo);
```

FIXME

*cosmo* : a **NcHICosmo**

*Returns* : FIXME

**nc\_hicosmo\_Omega\_mh2 ()**

```
gdouble          nc_hicosmo_Omega_mh2          (NcHICosmo *cosmo);
```

**nc\_hicosmo\_E ()**

```
gdouble          nc_hicosmo_E                  (NcHICosmo *cosmo,  
                                                gdouble z);
```

This function computes the normalized Hubble function  $E(z)$ .

*cosmo* : a **NcHICosmo**

*z* : redshift

*Returns* :  $E(z)$ .

**nc\_hicosmo\_Em2 ()**

```
gdouble          nc_hicosmo_Em2                (NcHICosmo *cosmo,  
                                                gdouble z);
```

This function computes the inverse of the square normalized Hubble function.

*cosmo* : a **NcHICosmo**

*z* : redshift

*Returns* :  $E(z)^{-2}$ .



**nc\_hicosmo\_H ()**

gdouble	nc_hicosmo_H	(NcHICosmo *cosmo, gdouble z);
---------	--------------	-----------------------------------

The value of the Hubble function in unity of  $\text{ms}^{-1}\text{kpc}^{-1}$ .

**cosmo** : a **NcHICosmo**.

**z** : FIXME

**Returns** :  $H(z)$

**nc\_hicosmo\_dH\_dz ()**

gdouble	nc_hicosmo_dH_dz	(NcHICosmo *cosmo, gdouble z);
---------	------------------	-----------------------------------

FIXME

**cosmo** : a **NcHICosmo**

**z** : redshift

**Returns** : FIXME

**nc\_hicosmo\_j ()**

gdouble	nc_hicosmo_j	(NcHICosmo *cosmo, gdouble z);
---------	--------------	-----------------------------------

FIXME

**cosmo** : a **NcHICosmo**

**z** : redshift

**Returns** : FIXME

**nc\_hicosmo\_qp ()**

gdouble	nc_hicosmo_qp	(NcHICosmo *cosmo, gdouble z);
---------	---------------	-----------------------------------

FIXME

**cosmo** : a **NcHICosmo**

**z** : redshift

**Returns** : FIXME

**nc\_hicosmo\_q ()**

gdouble	nc_hicosmo_q	(NcHICosmo *cosmo, gdouble z);
---------	--------------	-----------------------------------

FIXME

**cosmo** : a NcHICosmo**z** : redshift**Returns** : FIXME**nc\_hicosmo\_dec ()**

gdouble	nc_hicosmo_dec	(NcHICosmo *cosmo, gdouble z);
---------	----------------	-----------------------------------

FIXME

**cosmo** : a NcHICosmo**z** : redshift**Returns** : FIXME**nc\_hicosmo\_wec ()**

gdouble	nc_hicosmo_wec	(NcHICosmo *cosmo, gdouble z);
---------	----------------	-----------------------------------

FIXME

**cosmo** : a NcHICosmo**z** : redshift**Returns** : FIXME**nc\_hicosmo\_new\_from\_name ()**

NcHICosmo *	nc_hicosmo_new_from_name	(GType parent_type, gchar *cosmo_name);
-------------	--------------------------	--

FIXME

**parent\_type** : FIXME**cosmo\_name** : FIXME**Returns** : FIXME**nc\_hicosmo\_log\_all\_models ()**

void	nc_hicosmo_log_all_models	(GType parent);
------	---------------------------	-----------------

FIXME

**parent** : FIXME

**nc\_hicosmo\_free ()**

```
void nc_hicosmo_free (NcHICosmo *hic);
```

FIXME

**hic** : FIXME

**nc\_hicosmo\_create\_mset\_func0 ()**

```
NcmMSetFunc * nc_hicosmo_create_mset_func0 (NcHICosmoFunc0 f0);
```

**f0** : FIXME. *[scope notified]*

**Returns** : FIXME. *[transfer full]*

**nc\_hicosmo\_create\_mset\_func1 ()**

```
NcmMSetFunc * nc_hicosmo_create_mset_func1 (NcHICosmoFunc1 f1);
```

**f1** : FIXME. *[scope notified]*

**Returns** : FIXME. *[transfer full]*

**nc\_hicosmo\_set\_H0\_impl ()**

```
void nc_hicosmo_set_H0_impl (NcHICosmoClass *model_class,
                             NcmModelFunc0 f);
```

FIXME

**model\_class** : FIXME

**f** : FIXME

**nc\_hicosmo\_set\_Omega\_b\_impl ()**

```
void nc_hicosmo_set_Omega_b_impl (NcHICosmoClass *model_class,
                                   NcmModelFunc0 f);
```

FIXME

**model\_class** : FIXME

**f** : FIXME

**nc\_hicosmo\_set\_Omega\_r\_impl ()**

```
void nc_hicosmo_set_Omega_r_impl (NcHICosmoClass *model_class,
                                   NcmModelFunc0 f);
```

FIXME

**model\_class** : FIXME

**f** : FIXME

**nc\_hicosmo\_set\_Omega\_c\_impl ()**

```
void          nc_hicosmo_set_Omega_c_impl      (NcHICosmoClass *model_class,  
                                                NcmModelFunc0 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

**nc\_hicosmo\_set\_Omega\_t\_impl ()**

```
void          nc_hicosmo_set_Omega_t_impl      (NcHICosmoClass *model_class,  
                                                NcmModelFunc0 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

**nc\_hicosmo\_set\_sigma\_8\_impl ()**

```
void          nc_hicosmo_set_sigma_8_impl      (NcHICosmoClass *model_class,  
                                                NcmModelFunc0 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

**nc\_hicosmo\_set\_T\_gamma0\_impl ()**

```
void          nc_hicosmo_set_T_gamma0_impl     (NcHICosmoClass *model_class,  
                                                NcmModelFunc0 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

**nc\_hicosmo\_set\_z\_lss\_impl ()**

```
void          nc_hicosmo_set_z_lss_impl        (NcHICosmoClass *model_class,  
                                                NcmModelFunc0 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

---

**nc\_hicosmo\_set\_as\_drag\_impl ()**

```
void          nc_hicosmo_set_as_drag_impl      (NcHICosmoClass *model_class,  
                                                NcmModelFunc0 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

**nc\_hicosmo\_set\_E2\_impl ()**

```
void          nc_hicosmo_set_E2_impl          (NcHICosmoClass *model_class,  
                                                NcmModelFunc1 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

**nc\_hicosmo\_set\_dE2\_dz\_impl ()**

```
void          nc_hicosmo_set_dE2_dz_impl      (NcHICosmoClass *model_class,  
                                                NcmModelFunc1 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

**nc\_hicosmo\_set\_d2E2\_dz2\_impl ()**

```
void          nc_hicosmo_set_d2E2_dz2_impl    (NcHICosmoClass *model_class,  
                                                NcmModelFunc1 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

**nc\_hicosmo\_set\_cd\_impl ()**

```
void          nc_hicosmo_set_cd_impl          (NcHICosmoClass *model_class,  
                                                NcmModelFunc1 f);
```

FIXME

**model\_class**: FIXME

**f**: FIXME

---

**nc\_hicosmo\_set\_powspec\_impl()**

```
void          nc_hicosmo_set_powspec_impl      (NcHICosmoClass *model_class,
                                                NcmModelFunc1 f);
```

FIXME

*model\_class*: FIXME

*f*: FIXME

**NC\_HICOSMO\_DEFAULT\_PARAMS\_RELTOL**

```
#define NC_HICOSMO_DEFAULT_PARAMS_RELTOL (1e-7)
```

**NC\_HICOSMO\_DEFAULT\_PARAMS\_ABSTOL**

```
#define NC_HICOSMO_DEFAULT_PARAMS_ABSTOL (0.0)
```

## 4.2 Homogeneous and Isotropic Models Priors

Homogeneous and Isotropic Models Priors — Priors for HICosmo models

### Synopsis

```
struct          NcHICosmoPriorTop;
NcHICosmoPriorTop * nc_hicosmo_prior_top_new      (gdouble z,
                                                    gdouble alpha,
                                                    gdouble sigma_alpha,
                                                    gint n);

void          nc_hicosmo_prior_top_free           (NcHICosmoPriorTop *tp);
void          nc_hicosmo_prior_top_set            (NcHICosmoPriorTop *tp,
                                                    gdouble z,
                                                    gdouble alpha,
                                                    gdouble sigma_alpha,
                                                    gint n);

void          nc_hicosmo_prior_top_add            (NcmLikelihood *lh,
                                                    gdouble z,
                                                    gdouble alpha,
                                                    gdouble sigma_alpha,
                                                    gint n);
```

### Description

FIXME

## Details

### struct NcHICosmoPriorTop

```
struct NcHICosmoPriorTop {  
};
```

FIXME

### nc\_hicosmo\_prior\_top\_new ()

```
NcHICosmoPriorTop * nc_hicosmo_prior_top_new      (gdouble z,  
                                                    gdouble alpha,  
                                                    gdouble sigma_alpha,  
                                                    gint n);
```

FIXME

**z** : FIXME

**alpha** : FIXME

**sigma\_alpha** : FIXME

**n** : FIXME

**Returns** : FIXME

### nc\_hicosmo\_prior\_top\_free ()

```
void          nc_hicosmo_prior_top_free      (NcHICosmoPriorTop *tp);
```

FIXME

**tp** : FIXME

### nc\_hicosmo\_prior\_top\_set ()

```
void          nc_hicosmo_prior_top_set      (NcHICosmoPriorTop *tp,  
                                                    gdouble z,  
                                                    gdouble alpha,  
                                                    gdouble sigma_alpha,  
                                                    gint n);
```

FIXME

**tp** : FIXME

**z** : FIXME

**alpha** : FIXME

**sigma\_alpha** : FIXME

**n** : FIXME

**nc\_hicosmo\_prior\_top\_add ()**

```
void          nc_hicosmo_prior_top_add          (NcmLikelihood *lh,
                                                gdouble z,
                                                gdouble alpha,
                                                gdouble sigma_alpha,
                                                gint n);
```

FIXME

**lh**: a **NcmLikelihood**

**z**: FIXME

**alpha**: FIXME

**sigma\_alpha**: FIXME

**n**: FIXME

## 4.3 $\Lambda$ CDM

$\Lambda$ CDM — Implementation of  $\Lambda$ CDM model

### Synopsis

```
struct          NchICosmoLCDMClass;
struct          NchICosmoLCDM;
NchICosmoLCDM * nc_hicosmo_lcdm_new          (void);
```

### Object Hierarchy

```
GObject
+-----NcmModel
      +-----NchICosmo
            +-----NchICosmoLCDM
```

### Properties

"H0"	gdouble	: Read / Write
"H0-fit"	gboolean	: Read / Write
"Omegab"	gdouble	: Read / Write
"Omegab-fit"	gboolean	: Read / Write
"Omegac"	gdouble	: Read / Write
"Omegac-fit"	gboolean	: Read / Write
"Omegax"	gdouble	: Read / Write
"Omegax-fit"	gboolean	: Read / Write
"Tgamma0"	gdouble	: Read / Write
"Tgamma0-fit"	gboolean	: Read / Write
"ns"	gdouble	: Read / Write
"ns-fit"	gboolean	: Read / Write
"sigma8"	gdouble	: Read / Write
"sigma8-fit"	gboolean	: Read / Write



Description

FIXME

Details

struct NcHICosmoLCDMClass

```
struct NcHICosmoLCDMClass {  
};
```

struct NcHICosmoLCDM

```
struct NcHICosmoLCDM;
```

nc\_hicosmo\_lcdm\_new ()

```
NcHICosmoLCDM *      nc_hicosmo_lcdm_new      (void);
```

FIXME

*Returns* : FIXME

Property Details

The "H0" property

"H0"	gdouble	: Read / Write
------	---------	----------------

H\_0.

Allowed values: [10,500]

Default value: 73

The "H0-fit" property

"H0-fit"	gboolean	: Read / Write
----------	----------	----------------

H\_0:fit.

Default value: FALSE

The "Omegab" property

"Omegab"	gdouble	: Read / Write
----------	---------	----------------

\Omega\_b.

Allowed values: [1e-08,10]

Default value: 0.0432

---

**The "Omegab-fit" property**

"Omegab-fit"	gboolean	: Read / Write
--------------	----------	----------------

\Omega\_b:fit.  
Default value: FALSE

**The "Omegac" property**

"Omegac"	gdouble	: Read / Write
----------	---------	----------------

\Omega\_c.  
Allowed values: [1e-08,10]  
Default value: 0.2568

**The "Omegac-fit" property**

"Omegac-fit"	gboolean	: Read / Write
--------------	----------	----------------

\Omega\_c:fit.  
Default value: TRUE

**The "Omegax" property**

"Omegax"	gdouble	: Read / Write
----------	---------	----------------

\Omega\_x.  
Allowed values: [1e-08,10]  
Default value: 0.7

**The "Omegax-fit" property**

"Omegax-fit"	gboolean	: Read / Write
--------------	----------	----------------

\Omega\_x:fit.  
Default value: TRUE

**The "Tgamma0" property**

"Tgamma0"	gdouble	: Read / Write
-----------	---------	----------------

T\_{\gamma0}.  
Allowed values: [1e-08,10]  
Default value: 2.7245

---

**The "Tgamma0-fit" property**

"Tgamma0-fit"	gboolean	: Read / Write
---------------	----------	----------------

T\_{\gamma 0}:fit.

Default value: FALSE

**The "ns" property**

"ns"	gdouble	: Read / Write
------	---------	----------------

n\_s.

Allowed values: [0.5,1.5]

Default value: 1

**The "ns-fit" property**

"ns-fit"	gboolean	: Read / Write
----------	----------	----------------

n\_s:fit.

Default value: FALSE

**The "sigma8" property**

"sigma8"	gdouble	: Read / Write
----------	---------	----------------

\sigma\_8.

Allowed values: [0.2,1.8]

Default value: 0.9

**The "sigma8-fit" property**

"sigma8-fit"	gboolean	: Read / Write
--------------	----------	----------------

\sigma\_8:fit.

Default value: FALSE

## 4.4 Darkenergy

### 4.4.1 Dark Energy Abstract Class

Dark Energy Abstract Class — Base class for implementing dark energy models

**Synopsis**

```

enum          NchICosmoDEImpl;
enum          NchICosmoDEParams;
#define       NC_HICOSMO_DE_DEFAULT_H0
#define       NC_HICOSMO_DE_DEFAULT_OMEGA_C
#define       NC_HICOSMO_DE_DEFAULT_OMEGA_X
#define       NC_HICOSMO_DE_DEFAULT_OMEGA_B
#define       NC_HICOSMO_DE_DEFAULT_T_GAMMA0
#define       NC_HICOSMO_DE_DEFAULT_SPECINDEX
#define       NC_HICOSMO_DE_DEFAULT_SIGMA8
struct        NchICosmoDEClass;
struct        NchICosmoDE;
void          nc_hicosmo_de_set_wmap5_params      (NchICosmo *cosmo);
void          nc_hicosmo_de_omega_x2omega_k      (NchICosmo *cosmo);
gboolean      nc_hicosmo_de_new_add_bbn          (NcmLikelihood *lh);
void          nc_hicosmo_de_set_weff_impl        (NchICosmoDEClass *cosmo_de_class,
                                                  NcmModelFunc1 f);
void          nc_hicosmo_de_set_dweff_dz_impl    (NchICosmoDEClass *cosmo_de_class,
                                                  NcmModelFunc1 f);
gdouble       nc_hicosmo_de_weff                (NchICosmoDE *cosmo,
                                                  gdouble x);
gdouble       nc_hicosmo_de_dweff_dz            (NchICosmoDE *cosmo,
                                                  gdouble x);

```

**Object Hierarchy**

```

GObject
+----NcmModel
      +----NchICosmo
            +----NchICosmoDE
                  +----NchICosmoDELinder
                  +----NchICosmoDEPad
                  +----NchICosmoDEQe
                  +----NchICosmoDEXcdm

```

**Properties**

"H0"	gdouble	: Read / Write
"H0-fit"	gboolean	: Read / Write
"Omegab"	gdouble	: Read / Write
"Omegab-fit"	gboolean	: Read / Write
"Omegac"	gdouble	: Read / Write
"Omegac-fit"	gboolean	: Read / Write
"Omegax"	gdouble	: Read / Write
"Omegax-fit"	gboolean	: Read / Write
"Tgamma0"	gdouble	: Read / Write
"Tgamma0-fit"	gboolean	: Read / Write
"ns"	gdouble	: Read / Write
"ns-fit"	gboolean	: Read / Write
"sigma8"	gdouble	: Read / Write
"sigma8-fit"	gboolean	: Read / Write

**Description**

FIXME

---

**Details****enum NcHICosmoDEImpl**

```
typedef enum {
    NC_HICOSMO_DE_IMPL_weff      = NC_HICOSMO_IMPL_LAST << 0,
} NcHICosmoDEImpl;
```

FIXME

**NC\_HICOSMO\_DE\_IMPL\_weff** FIXME

**NC\_HICOSMO\_DE\_IMPL\_dweff\_dz** FIXME

**enum NcHICosmoDEParams**

```
typedef enum {
    NC_HICOSMO_DE_H0 = 0,
    NC_HICOSMO_DE_OMEGA_C,
    NC_HICOSMO_DE_OMEGA_X,
    NC_HICOSMO_DE_T_GAMMA0,
    NC_HICOSMO_DE_OMEGA_B,
    NC_HICOSMO_DE_SPECINDEX,
} NcHICosmoDEParams;
```

FIXME

**NC\_HICOSMO\_DE\_H0** FIXME

**NC\_HICOSMO\_DE\_OMEGA\_C** FIXME

**NC\_HICOSMO\_DE\_OMEGA\_X** FIXME

**NC\_HICOSMO\_DE\_T\_GAMMA0** FIXME

**NC\_HICOSMO\_DE\_OMEGA\_B** FIXME

**NC\_HICOSMO\_DE\_SPECINDEX** FIXME

**NC\_HICOSMO\_DE\_SIGMA8** FIXME

**NC\_HICOSMO\_DE\_DEFAULT\_H0**

```
#define NC_HICOSMO_DE_DEFAULT_H0      ncm_c_hubble_cte_wmap ()
```

**NC\_HICOSMO\_DE\_DEFAULT\_OMEGA\_C**

```
#define NC_HICOSMO_DE_DEFAULT_OMEGA_C      (0.2568)
```

**NC\_HICOSMO\_DE\_DEFAULT\_OMEGA\_X**

```
#define NC_HICOSMO_DE_DEFAULT_OMEGA_X      (0.70)
```

**NC\_HICOSMO\_DE\_DEFAULT\_OMEGA\_B**

```
#define NC_HICOSMO_DE_DEFAULT_OMEGA_B (0.0432)
```

**NC\_HICOSMO\_DE\_DEFAULT\_T\_GAMMA0**

```
#define NC_HICOSMO_DE_DEFAULT_T_GAMMA0 (2.7245)
```

**NC\_HICOSMO\_DE\_DEFAULT\_SPECINDEX**

```
#define NC_HICOSMO_DE_DEFAULT_SPECINDEX (1.0)
```

**NC\_HICOSMO\_DE\_DEFAULT\_SIGMA8**

```
#define NC_HICOSMO_DE_DEFAULT_SIGMA8 (0.9)
```

**struct NcHICosmoDEClass**

```
struct NcHICosmoDEClass {  
};
```

**struct NcHICosmoDE**

```
struct NcHICosmoDE;
```

**nc\_hicosmo\_de\_set\_wmap5\_params ()**

```
void nc_hicosmo_de_set_wmap5_params (NcHICosmo *cosmo);
```

**nc\_hicosmo\_de\_omega\_x2omega\_k ()**

```
void nc_hicosmo_de_omega_x2omega_k (NcHICosmo *cosmo);
```

FIXME

*cosmo* : FIXME

**nc\_hicosmo\_de\_new\_add\_bbn ()**

```
gboolean nc_hicosmo_de_new_add_bbn (NcmLikelihood *lh);
```

FIXME

*lh* : FIXME

**Returns** : FIXME

---

**nc\_hicosmo\_de\_set\_weff\_impl ()**

```
void                nc_hicosmo_de_set_weff_impl      (NchICosmoDEClass *cosmo_de_class,
                                                         NcmModelFunc1 f);
```

FIXME

*cosmo\_de\_class*: FIXME

*f*: FIXME

**nc\_hicosmo\_de\_set\_dweff\_dz\_impl ()**

```
void                nc_hicosmo_de_set_dweff_dz_impl  (NchICosmoDEClass *cosmo_de_class,
                                                         NcmModelFunc1 f);
```

FIXME

*cosmo\_de\_class*: FIXME

*f*: FIXME

**nc\_hicosmo\_de\_weff ()**

```
gdouble            nc_hicosmo_de_weff              (NchICosmoDE *cosmo,
                                                         gdouble x);
```

**nc\_hicosmo\_de\_dweff\_dz ()**

```
gdouble            nc_hicosmo_de_dweff_dz          (NchICosmoDE *cosmo,
                                                         gdouble x);
```

**Property Details**

**The "H0" property**

"H0"	gdouble	: Read / Write
------	---------	----------------

H\_0.

Allowed values: [10,500]

Default value: 73

**The "H0-fit" property**

"H0-fit"	gboolean	: Read / Write
----------	----------	----------------

H\_0:fit.

Default value: FALSE

---

**The "Omegab" property**

"Omegab"	gdouble	: Read / Write
----------	---------	----------------

\Omega\_b.

Allowed values: [1e-08,10]

Default value: 0.0432

**The "Omegab-fit" property**

"Omegab-fit"	gboolean	: Read / Write
--------------	----------	----------------

\Omega\_b:fit.

Default value: FALSE

**The "Omegac" property**

"Omegac"	gdouble	: Read / Write
----------	---------	----------------

\Omega\_c.

Allowed values: [1e-08,10]

Default value: 0.2568

**The "Omegac-fit" property**

"Omegac-fit"	gboolean	: Read / Write
--------------	----------	----------------

\Omega\_c:fit.

Default value: TRUE

**The "Omegax" property**

"Omegax"	gdouble	: Read / Write
----------	---------	----------------

\Omega\_x.

Allowed values: [1e-08,10]

Default value: 0.7

**The "Omegax-fit" property**

"Omegax-fit"	gboolean	: Read / Write
--------------	----------	----------------

\Omega\_x:fit.

Default value: TRUE

---



**The "Tgamma0" property**

"Tgamma0"	gdouble	: Read / Write
-----------	---------	----------------

T\_{\gamma0}.

Allowed values: [1e-08,10]

Default value: 2.7245

**The "Tgamma0-fit" property**

"Tgamma0-fit"	gboolean	: Read / Write
---------------	----------	----------------

T\_{\gamma0}:fit.

Default value: FALSE

**The "ns" property**

"ns"	gdouble	: Read / Write
------	---------	----------------

n\_s.

Allowed values: [0.5,1.5]

Default value: 1

**The "ns-fit" property**

"ns-fit"	gboolean	: Read / Write
----------	----------	----------------

n\_s:fit.

Default value: FALSE

**The "sigma8" property**

"sigma8"	gdouble	: Read / Write
----------	---------	----------------

\sigma\_8.

Allowed values: [0.2,1.8]

Default value: 0.9

**The "sigma8-fit" property**

"sigma8-fit"	gboolean	: Read / Write
--------------	----------	----------------

\sigma\_8:fit.

Default value: FALSE

**4.4.2 Dark Energy -- XCDM**

Dark Energy -- XCDM — Constant dark energy equation of state model

## Synopsis

```
enum                NcHICosmoDEXCDMParams;
#define             NC_HICOSMO_DE_XCDM_DEFAULT_W0
#define             NC_HICOSMO_DE_XCDM_N
struct             NcHICosmoDEXcdmClass;
struct             NcHICosmoDEXcdm;
NcHICosmoDEXcdm *  nc_hicosmo_de_xcdm_new          (void);
```

## Object Hierarchy

```
GObject
+----NcmModel
      +----NcHICosmo
            +----NcHICosmoDE
                  +----NcHICosmoDEXcdm
```

## Properties

"w"	gdouble	: Read / Write
"w-fit"	gboolean	: Read / Write

## Description

FIXME

## Details

### enum NcHICosmoDEXCDMParams

```
typedef enum {
} NcHICosmoDEXCDMParams;
```

FIXME

**NC\_HICOSMO\_DE\_XCDM\_W** FIXME

**NC\_HICOSMO\_DE\_XCDM\_DEFAULT\_W0**

```
#define NC_HICOSMO_DE_XCDM_DEFAULT_W0 (-1.0)
```

**NC\_HICOSMO\_DE\_XCDM\_N**

```
#define NC_HICOSMO_DE_XCDM_N (NC_HICOSMO_DE_XCDM_W + 1 - NC_HICOSMO_DE_BASE_N)
```

### struct NcHICosmoDEXcdmClass

```
struct NcHICosmoDEXcdmClass {
};
```

**struct NcHICosmoDEXcdm**

```
struct NcHICosmoDEXcdm;
```

**nc\_hicosmo\_de\_xcdm\_new ()**

```
NcHICosmoDEXcdm * nc_hicosmo_de_xcdm_new (void);
```

FIXME

*Returns* : FIXME

**Property Details**

**The "w" property**

"w"	gdouble	: Read / Write
-----	---------	----------------

w.  
Allowed values: [-10,1]  
Default value: -1

**The "w-fit" property**

"w-fit"	gboolean	: Read / Write
---------	----------	----------------

w:fit.  
Default value: TRUE

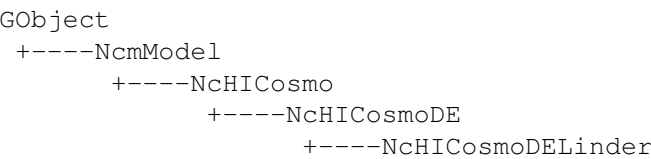
**4.4.3 Dark Energy -- Linder**

Dark Energy -- Linder — Linder dark energy equation of state parametrization

**Synopsis**

```
enum NcHICosmoDELinderParams;
#define NC_HICOSMO_DE_LINDER_DEFAULT_W0
#define NC_HICOSMO_DE_LINDER_DEFAULT_W1
#define NC_HICOSMO_DE_LINDER_N
struct NcHICosmoDELinderClass;
struct NcHICosmoDELinder;
NcHICosmoDELinder * nc_hicosmo_de_linder_new (void);
```

**Object Hierarchy**



**Properties**

"w0"	gdouble	: Read / Write
"w0-fit"	gboolean	: Read / Write
"w1"	gdouble	: Read / Write
"w1-fit"	gboolean	: Read / Write

**Description**

See [Linder \(2003\)](#).

**Details****enum NcHICosmoDELinderParams**

```
typedef enum {
    NC_HICOSMO_DE_LINDER_W0 = NC_HICOSMO_DE_SPARAM_LEN,
} NcHICosmoDELinderParams;
```

FIXME

**NC\_HICOSMO\_DE\_LINDER\_W0** FIXME

**NC\_HICOSMO\_DE\_LINDER\_W1** FIXME

**NC\_HICOSMO\_DE\_LINDER\_DEFAULT\_W0**

```
#define NC_HICOSMO_DE_LINDER_DEFAULT_W0 (-1.0)
```

**NC\_HICOSMO\_DE\_LINDER\_DEFAULT\_W1**

```
#define NC_HICOSMO_DE_LINDER_DEFAULT_W1 ( 0.0)
```

**NC\_HICOSMO\_DE\_LINDER\_N**

```
#define NC_HICOSMO_DE_LINDER_N (NC_HICOSMO_DE_LINDER_W1 + 1 - NC_HICOSMO_DE_BASE_N)
```

**struct NcHICosmoDELinderClass**

```
struct NcHICosmoDELinderClass {
};
```

**struct NcHICosmoDELinder**

```
struct NcHICosmoDELinder;
```

**nc\_hicosmo\_de\_linder\_new ()**

```
NcHICosmoDELinder * nc_hicosmo_de_linder_new (void);
```

FIXME

*Returns* : FIXME

**Property Details**

**The "w0" property**

"w0"	gdouble	: Read / Write
------	---------	----------------

w\_0.  
Allowed values: [-10,1]  
Default value: -1

**The "w0-fit" property**

"w0-fit"	gboolean	: Read / Write
----------	----------	----------------

w\_0:fit.  
Default value: TRUE

**The "w1" property**

"w1"	gdouble	: Read / Write
------	---------	----------------

w\_1.  
Allowed values: [-5,5]  
Default value: 0

**The "w1-fit" property**

"w1-fit"	gboolean	: Read / Write
----------	----------	----------------

w\_1:fit.  
Default value: TRUE

**4.4.4 Dark Energy -- Jassal-Bagla-Padmanabhan**

Dark Energy -- Jassal-Bagla-Padmanabhan — Jassal-Bagla-Padmanabhan dark energy equation of state parametrization

**Synopsis**

```
enum NcHICosmoDEPadParams;
#define NC_HICOSMO_DE_PAD_DEFAULT_W0
#define NC_HICOSMO_DE_PAD_DEFAULT_W1
#define NC_HICOSMO_DE_PM_N
struct NcHICosmoDEPadClass;
struct NcHICosmoDEPad;
NcHICosmoDEPad * nc_hicosmo_de_pad_new (void);
```

## Object Hierarchy

```
GObject
+-----NcmModel
      +-----NcHICosmo
            +-----NcHICosmoDE
                  +-----NcHICosmoDEPad
```

## Properties

"w0"	gdouble	: Read / Write
"w0-fit"	gboolean	: Read / Write
"w1"	gdouble	: Read / Write
"w1-fit"	gboolean	: Read / Write

## Description

See [Jassal et al. \(2005\)](#).

## Details

### enum NcHICosmoDEPadParams

```
typedef enum {
    NC_HICOSMO_DE_PAD_W0 = NC_HICOSMO_DE_SPARAM_LEN,
} NcHICosmoDEPadParams;
```

FIXME

**NC\_HICOSMO\_DE\_PAD\_W0** FIXME

**NC\_HICOSMO\_DE\_PAD\_W1** FIXME

**NC\_HICOSMO\_DE\_PAD\_DEFAULT\_W0**

```
#define NC_HICOSMO_DE_PAD_DEFAULT_W0 (-1.0)
```

**NC\_HICOSMO\_DE\_PAD\_DEFAULT\_W1**

```
#define NC_HICOSMO_DE_PAD_DEFAULT_W1 ( 0.0)
```

**NC\_HICOSMO\_DE\_PM\_N**

```
#define NC_HICOSMO_DE_PM_N (NC_HICOSMO_DE_PAD_W1 + 1 - NC_HICOSMO_DE_BASE_N)
```

### struct NcHICosmoDEPadClass

```
struct NcHICosmoDEPadClass {
};
```

**struct NcHICosmoDEPad**

```
struct NcHICosmoDEPad;
```

**nc\_hicosmo\_de\_pad\_new ()**

```
NcHICosmoDEPad * nc_hicosmo_de_pad_new (void);
```

FIXME

**Returns :** FIXME

**Property Details**

**The "w0" property**

"w0"	gdouble	: Read / Write
------	---------	----------------

w\_0.  
Allowed values: [-10,1]  
Default value: -1

**The "w0-fit" property**

"w0-fit"	gboolean	: Read / Write
----------	----------	----------------

w\_0:fit.  
Default value: TRUE

**The "w1" property**

"w1"	gdouble	: Read / Write
------	---------	----------------

w\_1.  
Allowed values: [-5,5]  
Default value: 0

**The "w1-fit" property**

"w1-fit"	gboolean	: Read / Write
----------	----------	----------------

w\_1:fit.  
Default value: TRUE

**4.4.5 Dark Energy -- Quintessence (Inspired)**

Dark Energy -- Quintessence (Inspired) — FIXME

## Synopsis

```
enum                NcHICosmoDEQEParams;
#define             NC_HICOSMO_DE_QE_DEFAULT_W0
#define             NC_HICOSMO_DE_QE_DEFAULT_W1
#define             NC_HICOSMO_DE_QE_N
struct              NcHICosmoDEQeClass;
struct              NcHICosmoDEQe;
NcHICosmoDEQe *     nc_hicosmo_de_qe_new           (void);
```

## Object Hierarchy

```
GObject
+----NcmModel
      +----NcHICosmo
            +----NcHICosmoDE
                  +----NcHICosmoDEQe
```

## Properties

"w0"	gdouble	: Read / Write
"w0-fit"	gboolean	: Read / Write
"w1"	gdouble	: Read / Write
"w1-fit"	gboolean	: Read / Write

## Description

FIXME

## Details

### enum NcHICosmoDEQEParams

```
typedef enum {
    NC_HICOSMO_DE_QE_W0 = NC_HICOSMO_DE_SPARAM_LEN,
} NcHICosmoDEQEParams;
```

FIXME

**NC\_HICOSMO\_DE\_QE\_W0** FIXME

**NC\_HICOSMO\_DE\_QE\_W1** FIXME

### NC\_HICOSMO\_DE\_QE\_DEFAULT\_W0

```
#define NC_HICOSMO_DE_QE_DEFAULT_W0 (-1.0)
```

### NC\_HICOSMO\_DE\_QE\_DEFAULT\_W1

```
#define NC_HICOSMO_DE_QE_DEFAULT_W1 ( 0.0)
```



**NC\_HICOSMO\_DE\_QE\_N**

```
#define NC_HICOSMO_DE_QE_N (NC_HICOSMO_DE_QE_W1 + 1 - NC_HICOSMO_DE_BASE_N)
```

**struct NcHICosmoDEQeClass**

```
struct NcHICosmoDEQeClass {  
};
```

**struct NcHICosmoDEQe**

```
struct NcHICosmoDEQe;
```

**nc\_hicosmo\_de\_qe\_new ()**

```
NcHICosmoDEQe * nc_hicosmo_de_qe_new (void);
```

FIXME

*Returns* : FIXME

**Property Details**

**The "w0" property**

"w0"	gdouble	: Read / Write
------	---------	----------------

w\_0.  
Allowed values: [-10,1]  
Default value: -1

**The "w0-fit" property**

"w0-fit"	gboolean	: Read / Write
----------	----------	----------------

w\_0:fit.  
Default value: TRUE

**The "w1" property**

"w1"	gdouble	: Read / Write
------	---------	----------------

w\_1.  
Allowed values: [-5,5]  
Default value: 0

---

### The "w1-fit" property

"w1-fit"	gboolean	: Read / Write
----------	----------	----------------

w\_1:fit.

Default value: TRUE

#### 4.4.6 Quantum Gravity Bouncing Model

Quantum Gravity Bouncing Model — FIXME

## Synopsis

NcHICosmo *	nc_hicosmo_qg_new	(void);
void	nc_hicosmo_qg_max_z	(NcmModel *model, gdouble *max, gdouble *trans);
gdouble	nc_hicosmo_qg_get_eta_b	(NcmModel *model, gpointer userdata);
gdouble	nc_hicosmo_qg_gbar2	(NcmModel *model, gdouble x);
gdouble	nc_hicosmo_qg_gbarbar	(NcmModel *model, gdouble x);
gdouble	nc_hicosmo_qg_xbar	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_xxbarzeta2	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_dxxbarzeta2_xxbarzeta2	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_d2sqrtxxbarzeta_sqrtxxbarzeta	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_cs2_xxbar2	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_cs2	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_dcs2	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_beta	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_zeta	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_dzeta_zeta	(NcmModel *model, gdouble x, gpointer userdata);

gdouble	nc_hicosmo_qg_ddzeta_zeta	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_xddzeta_zeta_mxdzeta_zeta2_dzeta_zeta	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_V	(NcmModel *model, gdouble x, gpointer userdata);
gdouble	nc_hicosmo_qg_alphaprime2	(NcmModel *model, gdouble alpha, gpointer data);
gdouble	nc_hicosmo_qg_dalphaprime2_dalpha	(NcmModel *model, gdouble alpha, gpointer data);
gdouble	nc_hicosmo_qg_lambda_x	(NcmModel *model, gdouble x, gpointer userdata);
gboolean	nc_hicosmo_qg_past_sol	(NcmModel *model, gdouble k, gdouble lambda, gsl_matrix *sol);
void	nc_hicosmo_qg_h_to_R_matrix	(NcmModel *model, gdouble x, gsl_matrix *T);
void	nc_hicosmo_qg_R_to_h_matrix	(NcmModel *model, gdouble x, gsl_matrix *T);
gdouble	nc_hicosmo_qg_get_lambda_f	(NcmModel *model, gpointer userdata);
gdouble	nc_hicosmo_qg_get_lambda_i	(NcmModel *model, gpointer userdata);
gdouble	nc_hicosmo_qg_get_lambda_d	(NcmModel *model, gpointer userdata);
gdouble	nc_hicosmo_qg_eta_lambda	(NcmModel *model, gdouble lambda, gboolean deriv);
gdouble	nc_hicosmo_qg_x_lambda	(NcmModel *model, gdouble lambda, gboolean deriv);
gdouble	nc_hicosmo_qg_gbar_lambda	(NcmModel *model, gdouble lambda, gboolean deriv);
gdouble	nc_hicosmo_qg_gbarbar_lambda	(NcmModel *model, gdouble lambda, gboolean deriv);
gdouble	nc_hicosmo_qg_int_1_zeta2_lambda	(NcmModel *model, gdouble lambda, gboolean deriv);
gdouble	nc_hicosmo_qg_cs2zeta2_int_1_zeta2_lambda	(NcmModel *model, gdouble lambda, gboolean deriv);
gdouble	nc_hicosmo_qg_lambda_k_cross	(NcmModel *model, gdouble lambda, gboolean deriv);

gdouble	nc_hicosmo_qg_cs2_lambda	(NcmModel *model, gdouble lambda, gboolean deriv);
gdouble	nc_hicosmo_qg_V_lambda	(NcmModel *model, gdouble lambda, gboolean deriv);
enum struct NcHICosmoQGMode *	NcHICosmoQGPertType; NcHICosmoQGMode; nc_hicosmo_qg_pert_new	(NcmModel *model, gdouble ax_i, gdouble x_i);
gboolean	nc_hicosmo_qg_pert_set_opts	(NcHICosmoQGMode *qgmode);
gboolean	nc_hicosmo_qg_pert_init	(NcHICosmoQGMode *qgmode, gdouble k);
gboolean	nc_hicosmo_qg_pert_evolve	(NcHICosmoQGMode *qgmode);
gboolean	nc_hicosmo_qg_pert_prepare_pw_spline	(NcHICosmoQGMode *qgmode, gboolean verbose);
gdouble	nc_hicosmo_qg_pert_powerspectrum	(NcHICosmoQGMode *qgmode, gdouble x, gdouble *R);
NcHICosmoQGMode *	nc_hicosmo_qg_modefunc	(NcmModel *model, long double k, long double x0, long double xf);
gboolean	nc_hicosmo_qg_modefunc_set_opts	(NcHICosmoQGMode *qgmode);
gboolean	nc_hicosmo_qg_modefunc_init	(NcHICosmoQGMode *qgmode);
gboolean	nc_hicosmo_qg_modefuncm_cvode_init	(NcHICosmoQGMode *qgmode);
gboolean	nc_hicosmo_qg_modefunc_evolve	(NcHICosmoQGMode *qgmode);
void	nc_hicosmo_qg_evolvefunc	(NcmModel *model, long double x, long double *x2d2sqrtxxbarzeta_sq, long double *x2cs2_xxbar2);
void	nc_hicosmo_qg_modefunc_sol	(NcHICosmoQGMode *qgmode, long double x, long double x0, long double *Re_u, long double *Im_u, long double *Re_up, long double *Im_up);
void	nc_hicosmo_qg_pert_R_to_h	(NcHICosmoQGMode *qgmode, gdouble x, gdouble *R);
void	nc_hicosmo_qg_pert_h_to_R	(NcHICosmoQGMode *qgmode, gdouble x, gdouble *h);

**Description**

FIXME

**Details****nc\_hicosmo\_qg\_new ()**

NcHICosmo *	nc_hicosmo_qg_new	(void);
-------------	-------------------	---------

**nc\_hicosmo\_qg\_max\_z ()**

```
void nc_hicosmo_qg_max_z(NcmModel *model,
                          gdouble *max,
                          gdouble *trans);
```

**nc\_hicosmo\_qg\_get\_eta\_b()**

```
gdouble nc_hicosmo_qg_get_eta_b(NcmModel *model,
                                  gpointer userdata);
```

**nc\_hicosmo\_qg\_gbar2 ()**

```
gdouble          nc_hicosmo_qg_gbar2          (NcmModel *model,
                                                gdouble x);
```

**nc\_hicosmo\_qg\_gbarbar ()**

```
gdouble          nc_hicosmo_qg_gbarbar      (NcmModel *model,
                                              gdouble x);
```

**nc\_hicosmo\_qg\_xbar ()**

```
gdouble          nc_hicosmo_qg_xbar          (NcmModel *model,
gdouble x,
gpointer userdata);
```

**nc\_hicosmo\_qg\_xxbarzeta2 ()**

```
gdouble          nc_hicosmo_qg_xxbarzeta2      (NcmModel *model,
gdouble x,
gpointer userdata);
```

## nc\_hicosmo\_qg\_dxxbarzeta2\_xxbarzeta2 ()

[illegible]

**nc\_hicosmo\_qg\_d2sqrtxxbarzeta\_sqrtxxbarzeta ()**

[illegible]

**nc\_hicosmo\_qg\_cs2\_xxbar2 ()**

gdouble	nc_hicosmo_qg_cs2_xxbar2	(NcmModel *model, gdouble x, gpointer userdata);
---------	--------------------------	--

**nc\_hicosmo\_qg\_cs2 ()**

gdouble	nc_hicosmo_qg_cs2	(NcmModel *model, gdouble x, gpointer userdata);
---------	-------------------	--

**nc\_hicosmo\_qg\_dcs2 ()**

gdouble	nc_hicosmo_qg_dcs2	(NcmModel *model, gdouble x, gpointer userdata);
---------	--------------------	--

**nc\_hicosmo\_qg\_beta ()**

gdouble	nc_hicosmo_qg_beta	(NcmModel *model, gdouble x, gpointer userdata);
---------	--------------------	--

**nc\_hicosmo\_qg\_zeta ()**

gdouble	nc_hicosmo_qg_zeta	(NcmModel *model, gdouble x, gpointer userdata);
---------	--------------------	--

**nc\_hicosmo\_qg\_dzeta\_zeta ()**

gdouble	nc_hicosmo_qg_dzeta_zeta	(NcmModel *model, gdouble x, gpointer userdata);
---------	--------------------------	--

**nc\_hicosmo\_qg\_ddzeta\_zeta ()**

gdouble	nc_hicosmo_qg_ddzeta_zeta	(NcmModel *model, gdouble x, gpointer userdata);
---------	---------------------------	--

**nc\_hicosmo\_qg\_xddzeta\_zeta\_mxdzeta\_zeta2\_dzeta\_zeta ()**

gdouble	nc_hicosmo_qg_xddzeta_zeta_mxdzeta_zeta2_dzeta_zeta	(NcmModel *model, gdouble x, gpointer userdata);
---------	---	--

**nc\_hicosmo\_qg\_V ()**

gdouble	nc_hicosmo_qg_V	(NcmModel *model, gdouble x, gpointer userdata);
---------	-----------------	--

**nc\_hicosmo\_qg\_alphaprime2 ()**

gdouble	nc_hicosmo_qg_alphaprime2	(NcmModel *model, gdouble alpha, gpointer data);
---------	---------------------------	--

**nc\_hicosmo\_qg\_dalphaprime2\_dalpha ()**

gdouble	nc_hicosmo_qg_dalphaprime2_dalpha	(NcmModel *model, gdouble alpha, gpointer data);
---------	-----------------------------------	--

**nc\_hicosmo\_qg\_lambda\_x ()**

gdouble	nc_hicosmo_qg_lambda_x	(NcmModel *model, gdouble x, gpointer userdata);
---------	------------------------	--

**nc\_hicosmo\_qg\_past\_sol ()**

gboolean	nc_hicosmo_qg_past_sol	(NcmModel *model, gdouble k, gdouble lambda, gsl_matrix *sol);
----------	------------------------	---

FIXME

**model** : FIXME

**k** : FIXME

**lambda** : FIXME

**sol** : FIXME

**Returns** : FIXME

**nc\_hicosmo\_qg\_h\_to\_R\_matrix ()**

void	nc_hicosmo_qg_h_to_R_matrix	(NcmModel *model, gdouble x, gsl_matrix *T);
------	-----------------------------	--

FIXME

**model** : FIXME

**x** : FIXME

**T** : FIXME

```
void nc_hicosmo_qg_R_to_h_matrix(NcmModel *model,
                                  gdouble x,
                                  gsl_matrix *T);
```

```
model : FIXME
```

**x:** FIXME

***T* : FIXME**

```
gdouble nc_hicosmo_qg_get_lambda_f(NcmModel *model,
                                     gpointer userdata);
```

```
gdouble nc_hicosmo_qg_get_lambda_i(NcmModel *model,
                                     gpointer userdata);
```

```
gdouble nc_hicosmo_qg_get_lambda_d(NcmModel *model,
                                     gpointer userdata);
```

```
gdouble          nc_hicosmo_qg_eta_lambda      (NcmModel *model,
gdouble lambda,
gboolean deriv);
```

[illegible]

```
gdouble          nc_hicosmo_qg_gbar_lambda      (NcmModel *model,
gdouble lambda,
gboolean deriv);
```

[illegible]



**nc\_hicosmo\_qg\_int\_1\_zeta2\_lambda ()**

```
gdouble          nc_hicosmo_qg_int_1_zeta2_lambda      (NcmModel *model,
                                                         gdouble lambda,
                                                         gboolean deriv);
```

**nc\_hicosmo\_qg\_cs2zeta2\_int\_1\_zeta2\_lambda ()**

```
gdouble          nc_hicosmo_qg_cs2zeta2_int_1_zeta2_lambda      (NcmModel *model,
                                                         gdouble lambda,
                                                         gboolean deriv);
```

**nc\_hicosmo\_qg\_lambda\_k\_cross ()**

```
gdouble          nc_hicosmo_qg_lambda_k_cross          (NcmModel *model,
                                                         gdouble lambda,
                                                         gboolean deriv);
```

**nc\_hicosmo\_qg\_cs2\_lambda ()**

```
gdouble          nc_hicosmo_qg_cs2_lambda             (NcmModel *model,
                                                         gdouble lambda,
                                                         gboolean deriv);
```

**nc\_hicosmo\_qg\_V\_lambda ()**

```
gdouble          nc_hicosmo_qg_V_lambda              (NcmModel *model,
                                                         gdouble lambda,
                                                         gboolean deriv);
```

**enum NcHICosmoQGPertType**

```
typedef enum {
    NC_HICOSMO_QG_PERT_CURVATURE = 0,
    NC_HICOSMO_QG_PERT_H,
} NcHICosmoQGPertType;
```

FIXME

**NC\_HICOSMO\_QG\_PERT\_CURVATURE** FIXME

**NC\_HICOSMO\_QG\_PERT\_H** FIXME

**struct NcHICosmoQGMode**

```
struct NcHICosmoQGMode {
};
```

FIXME

**nc\_hicosmo\_qg\_pert\_new ()**

```
NcHICosmoQGMode *   nc_hicosmo_qg_pert_new      (NcmModel *model,
                                                    gdouble ax_i,
                                                    gdouble x_i);
```

FIXME

**model** : FIXME

**ax\_i** : FIXME

**x\_i** : FIXME

**Returns** : FIXME

**nc\_hicosmo\_qg\_pert\_set\_opts ()**

```
gboolean             nc_hicosmo_qg_pert_set_opts  (NcHICosmoQGMode *qgmode);
```

**nc\_hicosmo\_qg\_pert\_init ()**

```
gboolean             nc_hicosmo_qg_pert_init      (NcHICosmoQGMode *qgmode,
                                                    gdouble k);
```

**nc\_hicosmo\_qg\_pert\_evolve ()**

```
gboolean             nc_hicosmo_qg_pert_evolve    (NcHICosmoQGMode *qgmode);
```

**nc\_hicosmo\_qg\_pert\_prepare\_pw\_spline ()**

```
gboolean             nc_hicosmo_qg_pert_prepare_pw_spline
                                                    (NcHICosmoQGMode *qgmode,
                                                    gboolean verbose);
```

**nc\_hicosmo\_qg\_pert\_powerspectrum ()**

```
gdouble             nc_hicosmo_qg_pert_powerspectrum
                                                    (NcHICosmoQGMode *qgmode,
                                                    gdouble x,
                                                    gdouble *R);
```

**nc\_hicosmo\_qg\_modefunc ()**

```
NcHICosmoQGMode *   nc_hicosmo_qg_modefunc      (NcmModel *model,
                                                    long double k,
                                                    long double x0,
                                                    long double xf);
```

FIXME

**model** : FIXME

***k*** : FIXME***x0*** : FIXME***xf*** : FIXME**Returns** : FIXME**nc\_hicosmo\_qg\_modfunc\_set\_opts ()**

```
gboolean          nc_hicosmo_qg_modfunc_set_opts      (NcHICosmoQGMode *qgmode);
```

**nc\_hicosmo\_qg\_modfunc\_init ()**

```
gboolean          nc_hicosmo_qg_modfunc_init          (NcHICosmoQGMode *qgmode);
```

**nc\_hicosmo\_qg\_modfuncm\_cvode\_init ()**

```
gboolean          nc_hicosmo_qg_modfuncm_cvode_init   (NcHICosmoQGMode *qgmode);
```

**nc\_hicosmo\_qg\_modfunc\_evolve ()**

```
gboolean          nc_hicosmo_qg_modfunc_evolve        (NcHICosmoQGMode *qgmode);
```

**nc\_hicosmo\_qg\_evofunc ()**

```
void              nc_hicosmo_qg_evofunc               (NcmModel *model,
long double x,
long double * ←
                x2d2sqrtxxbarzeta_sqrtxxbarzeta ←
                ,
long double *x2cs2_xxbar2);
```

**nc\_hicosmo\_qg\_modfunc\_sol ()**

```
void              nc_hicosmo_qg_modfunc_sol           (NcHICosmoQGMode *qgmode,
long double x,
long double x0,
long double *Re_u,
long double *Im_u,
long double *Re_up,
long double *Im_up);
```

**nc\_hicosmo\_qg\_pert\_R\_to\_h ()**

```
void              nc_hicosmo_qg_pert_R_to_h          (NcHICosmoQGMode *qgmode,
gdouble x,
gdouble *R);
```

**nc\_hicosmo\_qg\_pert\_h\_to\_R()**

```
void          nc_hicosmo_qg_pert_h_to_R          (NchICosmoQGMode *qgmode,
                                                  gdouble x,
                                                  gdouble *h);
```

## 4.5 Kinematical

### 4.5.1 Constant Deceleration Parameter Model

Constant Deceleration Parameter Model — FIXME

**Synopsis**

```
enum          NchICosmoQConstParams;
#define       NC_HICOSMO_QCONST_DEFAULT_H0
#define       NC_HICOSMO_QCONST_DEFAULT_OMEGA_T
#define       NC_HICOSMO_QCONST_DEFAULT_CD
#define       NC_HICOSMO_QCONST_DEFAULT_E
#define       NC_HICOSMO_QCONST_DEFAULT_Q
#define       NC_HICOSMO_QCONST_DEFAULT_Z1
struct        NchICosmoQConstClass;
struct        NchICosmoQConst;
NchICosmoQConst * nc_hicosmo_qconst_new          (void);
```

**Object Hierarchy**

```
GObject
+-----NcmModel
      +-----NchICosmo
            +-----NchICosmoQConst
```

**Properties**

"E"	gdouble	: Read / Write
"E-fit"	gboolean	: Read / Write
"H0"	gdouble	: Read / Write
"H0-fit"	gboolean	: Read / Write
"Omegat"	gdouble	: Read / Write
"Omegat-fit"	gboolean	: Read / Write
"cd"	gdouble	: Read / Write
"cd-fit"	gboolean	: Read / Write
"q"	gdouble	: Read / Write
"q-fit"	gboolean	: Read / Write
"zs"	gdouble	: Read / Write
"zs-fit"	gboolean	: Read / Write

**Description**

FIXME

## Details

### enum NcHICosmoQConstParams

```
typedef enum {  
    NC_HICOSMO_QCONST_H0 = 0,  
    NC_HICOSMO_QCONST_OMEGA_T,  
    NC_HICOSMO_QCONST_CD,  
    NC_HICOSMO_QCONST_E,  
    NC_HICOSMO_QCONST_Q,  
} NcHICosmoQConstParams;
```

**NC\_HICOSMO\_QCONST\_H0** FIXME

**NC\_HICOSMO\_QCONST\_OMEGA\_T** FIXME

**NC\_HICOSMO\_QCONST\_CD** FIXME

**NC\_HICOSMO\_QCONST\_E** FIXME

**NC\_HICOSMO\_QCONST\_Q** FIXME

**NC\_HICOSMO\_QCONST\_Z1** FIXME

**NC\_HICOSMO\_QCONST\_DEFAULT\_H0**

```
#define NC_HICOSMO_QCONST_DEFAULT_H0      ncm_c_hubble_cte_wmap ()
```

**NC\_HICOSMO\_QCONST\_DEFAULT\_OMEGA\_T**

```
#define NC_HICOSMO_QCONST_DEFAULT_OMEGA_T ( 1.0)
```

**NC\_HICOSMO\_QCONST\_DEFAULT\_CD**

```
#define NC_HICOSMO_QCONST_DEFAULT_CD      ( 0.0)
```

**NC\_HICOSMO\_QCONST\_DEFAULT\_E**

```
#define NC_HICOSMO_QCONST_DEFAULT_E      ( 1.0)
```

**NC\_HICOSMO\_QCONST\_DEFAULT\_Q**

```
#define NC_HICOSMO_QCONST_DEFAULT_Q      (-0.5)
```

**NC\_HICOSMO\_QCONST\_DEFAULT\_Z1**

```
#define NC_HICOSMO_QCONST_DEFAULT_Z1      ( 0.0)
```

**struct NcHICosmoQConstClass**

```
struct NcHICosmoQConstClass {  
};
```

**struct NcHICosmoQConst**

```
struct NcHICosmoQConst;
```

**nc\_hicosmo\_qconst\_new ()**

```
NcHICosmoQConst * nc_hicosmo_qconst_new (void);
```

FIXME

*Returns* : FIXME

**Property Details**

**The "E" property**

"E"	gdouble	: Read / Write
-----	---------	----------------

E.  
Allowed values: [0,50]  
Default value: 1

**The "E-fit" property**

"E-fit"	gboolean	: Read / Write
---------	----------	----------------

E:fit.  
Default value: FALSE

**The "H0" property**

"H0"	gdouble	: Read / Write
------	---------	----------------

H\_0.  
Allowed values: [10,500]  
Default value: 73

**The "H0-fit" property**

"H0-fit"	gboolean	: Read / Write
----------	----------	----------------

H\_0:fit.  
Default value: FALSE

---

**The "Omegat" property**

"Omegat"	gdouble	: Read / Write
----------	---------	----------------

\Omega\_t.

Allowed values: [-5,5]

Default value: 1

**The "Omegat-fit" property**

"Omegat-fit"	gboolean	: Read / Write
--------------	----------	----------------

\Omega\_t:fit.

Default value: FALSE

**The "cd" property**

"cd"	gdouble	: Read / Write
------	---------	----------------

d\_c.

Allowed values: [-50,50]

Default value: 0

**The "cd-fit" property**

"cd-fit"	gboolean	: Read / Write
----------	----------	----------------

d\_c:fit.

Default value: FALSE

**The "q" property**

"q"	gdouble	: Read / Write
-----	---------	----------------

q.

Allowed values: [-50,50]

Default value: -0.5

**The "q-fit" property**

"q-fit"	gboolean	: Read / Write
---------	----------	----------------

q:fit.

Default value: TRUE

---

The "zs" property

"zs"	gdouble	: Read / Write
------	---------	----------------

z\_\star.

Allowed values: [0,5]

Default value: 0

The "zs-fit" property

"zs-fit"	gboolean	: Read / Write
----------	----------	----------------

z\_\star:fit.

Default value: FALSE

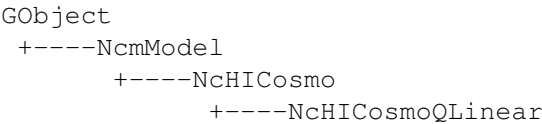
4.5.2 Linear Desceleration Parameter Model

Linear Desceleration Parameter Model — FIXME

Synopsis

```
enum                NcHICosmoQLinearParams;
#define              NC_HICOSMO_QLINEAR_DEFAULT_H0
#define              NC_HICOSMO_QLINEAR_DEFAULT_OMEGA_T
#define              NC_HICOSMO_QLINEAR_DEFAULT_CD
#define              NC_HICOSMO_QLINEAR_DEFAULT_E
#define              NC_HICOSMO_QLINEAR_DEFAULT_Q
#define              NC_HICOSMO_QLINEAR_DEFAULT_QP
#define              NC_HICOSMO_QLINEAR_DEFAULT_Z1
struct              NcHICosmoQLinearClass;
struct              NcHICosmoQLinear;
NcHICosmoQLinear *  nc_hicosmo_qlinear_new                (void);
gdouble             nc_hicosmo_qlinear_dE                (gdouble z2,
                                                         gdouble z1,
                                                         gdouble q,
                                                         gdouble qp);
```

Object Hierarchy



Properties

"E"	gdouble	: Read / Write
"E-fit"	gboolean	: Read / Write
"H0"	gdouble	: Read / Write
"H0-fit"	gboolean	: Read / Write
"Omegat"	gdouble	: Read / Write
"Omegat-fit"	gboolean	: Read / Write



"cd"	gdouble	: Read / Write
"cd-fit"	gboolean	: Read / Write
"q"	gdouble	: Read / Write
"q-fit"	gboolean	: Read / Write
"qp"	gdouble	: Read / Write
"qp-fit"	gboolean	: Read / Write
"zs"	gdouble	: Read / Write
"zs-fit"	gboolean	: Read / Write

## Description

FIXME

## Details

### enum NcHICosmoQLinearParams

```
typedef enum {
    NC_HICOSMO_QLINEAR_H0 = 0,
    NC_HICOSMO_QLINEAR_OMEGA_T,
    NC_HICOSMO_QLINEAR_CD,
    NC_HICOSMO_QLINEAR_E,
    NC_HICOSMO_QLINEAR_Q,
    NC_HICOSMO_QLINEAR_QP,
} NcHICosmoQLinearParams;
```

**NC\_HICOSMO\_QLINEAR\_H0** FIXME

**NC\_HICOSMO\_QLINEAR\_OMEGA\_T** FIXME

**NC\_HICOSMO\_QLINEAR\_CD** FIXME

**NC\_HICOSMO\_QLINEAR\_E** FIXME

**NC\_HICOSMO\_QLINEAR\_Q** FIXME

**NC\_HICOSMO\_QLINEAR\_QP** FIXME

**NC\_HICOSMO\_QLINEAR\_Z1** FIXME

### NC\_HICOSMO\_QLINEAR\_DEFAULT\_H0

```
#define NC_HICOSMO_QLINEAR_DEFAULT_H0          ncm_c_hubble_cte_wmap ()
```

### NC\_HICOSMO\_QLINEAR\_DEFAULT\_OMEGA\_T

```
#define NC_HICOSMO_QLINEAR_DEFAULT_OMEGA_T ( 1.0)
```

### NC\_HICOSMO\_QLINEAR\_DEFAULT\_CD

```
#define NC_HICOSMO_QLINEAR_DEFAULT_CD          ( 0.0)
```

**NC\_HICOSMO\_QLINEAR\_DEFAULT\_E**

```
#define NC_HICOSMO_QLINEAR_DEFAULT_E      ( 1.0)
```

**NC\_HICOSMO\_QLINEAR\_DEFAULT\_Q**

```
#define NC_HICOSMO_QLINEAR_DEFAULT_Q      (-0.5)
```

**NC\_HICOSMO\_QLINEAR\_DEFAULT\_QP**

```
#define NC_HICOSMO_QLINEAR_DEFAULT_QP     ( 1.0)
```

**NC\_HICOSMO\_QLINEAR\_DEFAULT\_Z1**

```
#define NC_HICOSMO_QLINEAR_DEFAULT_Z1     ( 0.0)
```

**struct NcHICosmoQLinearClass**

```
struct NcHICosmoQLinearClass {
};
```

**struct NcHICosmoQLinear**

```
struct NcHICosmoQLinear;
```

**nc\_hicosmo\_qlinear\_new ()**

```
NcHICosmoQLinear * nc_hicosmo_qlinear_new      (void);
```

FIXME

**Returns :** FIXME

**nc\_hicosmo\_qlinear\_dE ()**

```
gdouble          nc_hicosmo_qlinear_dE      (gdouble z2,
                                              gdouble z1,
                                              gdouble q,
                                              gdouble qp);
```

FIXME

**z2 :** FIXME

**z1 :** FIXME

**q :** FIXME

**qp :** FIXME

**Returns :** FIXME

Property Details

The "E" property

"E"	gdouble	: Read / Write
-----	---------	----------------

E.  
Allowed values: [0,50]  
Default value: 1

The "E-fit" property

"E-fit"	gboolean	: Read / Write
---------	----------	----------------

E:fit.  
Default value: FALSE

The "H0" property

"H0"	gdouble	: Read / Write
------	---------	----------------

H\_0.  
Allowed values: [10,500]  
Default value: 73

The "H0-fit" property

"H0-fit"	gboolean	: Read / Write
----------	----------	----------------

H\_0:fit.  
Default value: FALSE

The "Omegat" property

"Omegat"	gdouble	: Read / Write
----------	---------	----------------

\Omega\_t.  
Allowed values: [-5,5]  
Default value: 1

The "Omegat-fit" property

"Omegat-fit"	gboolean	: Read / Write
--------------	----------	----------------

\Omega\_t:fit.  
Default value: FALSE

---

**The "cd" property**

"cd"	gdouble	: Read / Write
------	---------	----------------

d\_c.  
Allowed values: [-50,50]  
Default value: 0

**The "cd-fit" property**

"cd-fit"	gboolean	: Read / Write
----------	----------	----------------

d\_c:fit.  
Default value: FALSE

**The "q" property**

"q"	gdouble	: Read / Write
-----	---------	----------------

q.  
Allowed values: [-50,50]  
Default value: -0.5

**The "q-fit" property**

"q-fit"	gboolean	: Read / Write
---------	----------	----------------

q:fit.  
Default value: TRUE

**The "qp" property**

"qp"	gdouble	: Read / Write
------	---------	----------------

q<sup>prime</sup>.  
Allowed values: [-50,50]  
Default value: 1

**The "qp-fit" property**

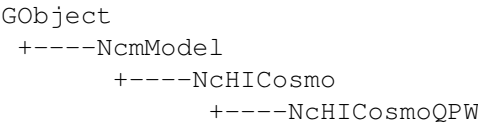
"qp-fit"	gboolean	: Read / Write
----------	----------	----------------

q<sup>prime</sup>:fit.  
Default value: TRUE

---



Object Hierarchy



Properties

"H0"	gdouble	: Read / Write
"H0-fit"	gboolean	: Read / Write
"Omegat"	gdouble	: Read / Write
"Omegat-fit"	gboolean	: Read / Write
"flat"	gboolean	: Read / Write / Construct Only
"q0"	gdouble	: Read / Write
"q0-fit"	gboolean	: Read / Write
"qp"	GVariant*	: Read / Write
"qp-fit"	GVariant*	: Read / Write
"qp-length"	guint	: Read / Write / Construct Only
"zf"	gdouble	: Read / Write / Construct Only

Description

FIXME

Details

enum NcHICosmoQPWSPParams

```
typedef enum {
    NC_HICOSMO_QPW_H0 = 0,
    NC_HICOSMO_QPW_OMEGA_T,
} NcHICosmoQPWSPParams;
```

NC\_HICOSMO\_QPW\_H0 FIXME

NC\_HICOSMO\_QPW\_OMEGA\_T FIXME

NC\_HICOSMO\_QPW\_Q0 FIXME

enum NcHICosmoQPWVParams

```
typedef enum {
} NcHICosmoQPWVParams;
```

NC\_HICOSMO\_QPW\_QP FIXME

NC\_HICOSMO\_QPW\_DEFAULT\_H0

```
#define NC_HICOSMO_QPW_DEFAULT_H0 ncm_c_hubble_cte_wmap ()
```

```
#define NC_HICOSMO_QPW_DEFAULT_OMEGA_T ( 1.0)
```

```
#define NC_HICOSMO_QPW_DEFAULT_Q0      (-0.5)
```

```
#define NC_HICOSMO_QPW_DEFAULT_QP      ( 0.0)
```

```
#define NC_HICOSMO_QPW_DEFAULT_QP_LEN      (4)
```

```
struct NcHICosmoQPWClass {
};
```

```
struct NcHICosmoQPW;
```

```
struct NcHICosmoQPWContPrior {
};
```

```
struct NcHICosmoQPWAsymCDMPrior {
};
```

[illegible]

**nc\_hicosmo\_qpw\_add\_continuity\_prior ()**

```
void          nc_hicosmo_qpw_add_continuity_prior (NcHICosmoQPW *qpw,
                                                    NcmLikelihood *lh,
                                                    gint knot,
                                                    gdouble sigma);
```

FIXME

***qpw*** : FIXME

***lh*** : FIXME

***knot*** : FIXME

***sigma*** : FIXME

**nc\_hicosmo\_qpw\_add\_continuity\_priors ()**

```
void          nc_hicosmo_qpw_add_continuity_priors
                                                    (NcHICosmoQPW *qpw,
                                                    NcmLikelihood *lh,
                                                    gdouble sigma);
```

FIXME

***qpw*** : FIXME

***lh*** : FIXME

***sigma*** : FIXME

**nc\_hicosmo\_qpw\_add\_asymptotic\_cdm\_prior ()**

```
void          nc_hicosmo_qpw_add_asymptotic_cdm_prior
                                                    (NcHICosmoQPW *qpw,
                                                    NcmLikelihood *lh,
                                                    gdouble z,
                                                    gdouble q,
                                                    gdouble sigma);
```

FIXME

***qpw*** : FIXME

***lh*** : FIXME

***z*** : FIXME

***q*** : FIXME

***sigma*** : FIXME

**nc\_hicosmo\_qpw\_change\_params ()**

```
void          nc_hicosmo_qpw_change_params      (NcHICosmoQPW *qpw,
                                                    gdouble z);
```

FIXME

***qpw*** : FIXME

***z*** : FIXME



**nc\_hicosmo\_qpw\_change\_params\_qpp ()**

```
void nc_hicosmo_qpw_change_params_qpp (NcHICosmoQPW *qpw);
```

FIXME

*qpw* : FIXME

**nc\_hicosmo\_qpw\_index ()**

```
guint nc_hicosmo_qpw_index (NcHICosmoQPW *qpw, gdouble z);
```

FIXME

*qpw* : FIXME

*z* : FIXME

*Returns* : FIXME

**Property Details**

**The "H0" property**

"H0"	gdouble	: Read / Write
------	---------	----------------

H\_0.

Allowed values: [10,500]

Default value: 73

**The "H0-fit" property**

"H0-fit"	gboolean	: Read / Write
----------	----------	----------------

H\_0:fit.

Default value: FALSE

**The "Omegat" property**

"Omegat"	gdouble	: Read / Write
----------	---------	----------------

\Omega\_t.

Allowed values: [-5,5]

Default value: 1

**The "Omegat-fit" property**

"Omegat-fit"	gboolean	: Read / Write
--------------	----------	----------------

\Omega\_t:fit.

Default value: FALSE

---

**The "flat" property**

"flat"	gboolean	: Read / Write / Construct Only
--------	----------	---------------------------------

set model flat.

Default value: FALSE

**The "q0" property**

"q0"	gdouble	: Read / Write
------	---------	----------------

q\_0.

Allowed values: [-50,50]

Default value: -0.5

**The "q0-fit" property**

"q0-fit"	gboolean	: Read / Write
----------	----------	----------------

q\_0:fit.

Default value: TRUE

**The "qp" property**

"qp"	GVariant*	: Read / Write
------	-----------	----------------

q<sup>^</sup>\prime.

Allowed values: GVariant<a\*>

Default value: NULL

**The "qp-fit" property**

"qp-fit"	GVariant*	: Read / Write
----------	-----------	----------------

q<sup>^</sup>\prime:fit.

Allowed values: GVariant<a\*>

Default value: NULL

**The "qp-length" property**

"qp-length"	guint	: Read / Write / Construct Only
-------------	-------	---------------------------------

q<sup>^</sup>\prime:length.

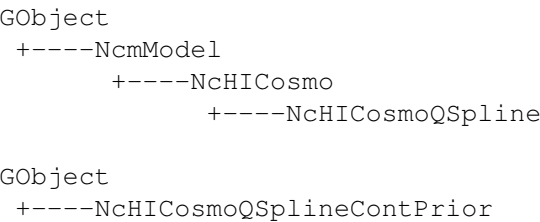
Default value: 4

---



```
guint          nc_hicosmo_qspline_cont_prior_get_nknots
                                                         (NcHICosmoQSplineContPrior *qsplin
void          nc_hicosmo_qspline_cont_prior_set_lnsigma
                                                         (NcHICosmoQSplineContPrior *qsplin
                                                         guint i,
                                                         gdouble ln_sigma);
void          nc_hicosmo_qspline_cont_prior_set_all_lnsigma
                                                         (NcHICosmoQSplineContPrior *qsplin
                                                         gdouble ln_sigma);
gdouble       nc_hicosmo_qspline_cont_prior_get_lnsigma
                                                         (NcHICosmoQSplineContPrior *qsplin
                                                         guint i);
```

Object Hierarchy



Properties

"H0"	gdouble	: Read / Write
"H0-fit"	gboolean	: Read / Write
"Omegat"	gdouble	: Read / Write
"Omegat-fit"	gboolean	: Read / Write
"asdrag"	gdouble	: Read / Write
"asdrag-fit"	gboolean	: Read / Write
"qparam"	GVariant*	: Read / Write
"qparam-fit"	GVariant*	: Read / Write
"qparam-length"	guint	: Read / Write / Construct Only
"spline"	NcmSpline*	: Read / Write / Construct Only
"zf"	gdouble	: Read / Write / Construct Only
"nknots"	guint	: Read / Write / Construct Only

Description

FIXME

Details

enum NcHICosmoQSplineSParams

```
typedef enum {
    NC_HICOSMO_QSPLINE_H0 = 0,
    NC_HICOSMO_QSPLINE_OMEGA_T,
} NcHICosmoQSplineSParams;
```

NC\_HICOSMO\_QSPLINE\_H0 FIXME

NC\_HICOSMO\_QSPLINE\_OMEGA\_T FIXME

NC\_HICOSMO\_QSPLINE\_AS\_DRAG FIXME

**enum NcHICosmoQSplineVParams**

```
typedef enum {
} NcHICosmoQSplineVParams;
```

**NC\_HICOSMO\_QSPLINE\_Q** FIXME**NC\_HICOSMO\_QSPLINE\_DEFAULT\_H0**

```
#define NC_HICOSMO_QSPLINE_DEFAULT_H0      ncm_c_hubble_cte_wmap  ()
```

**NC\_HICOSMO\_QSPLINE\_DEFAULT\_OMEGA\_T**

```
#define NC_HICOSMO_QSPLINE_DEFAULT_OMEGA_T      ( 1.0)
```

**NC\_HICOSMO\_QSPLINE\_DEFAULT\_AS\_DRAG**

```
#define NC_HICOSMO_QSPLINE_DEFAULT_AS_DRAG      ( 0.035)
```

**NC\_HICOSMO\_QSPLINE\_DEFAULT\_Q**

```
#define NC_HICOSMO_QSPLINE_DEFAULT_Q      (-0.5)
```

**NC\_HICOSMO\_QSPLINE\_DEFAULT\_Q\_LEN**

```
#define NC_HICOSMO_QSPLINE_DEFAULT_Q_LEN (3)
```

**struct NcHICosmoQSplineClass**

```
struct NcHICosmoQSplineClass {
};
```

**struct NcHICosmoQSpline**

```
struct NcHICosmoQSpline;
```

**nc\_hicosmo\_qspline\_new ()**

```
NcHICosmoQSpline * nc_hicosmo_qspline_new      (NcmSpline *s,
                                                  gsize np,
                                                  gdouble z_f);
```

FIXME

**s** : FIXME**np** : FIXME**z\_f** : FIXME**Returns** : FIXME

**nc\_hicosmo\_qspline\_add\_continuity\_prior ()**

```
void nc_hicosmo_qspline_add_continuity_prior
(NcHICosmoQSpline *qspline,
 NcmLikelihood *lh,
 gint knot,
 NcHICosmoQSplineContPrior * ↔
 qspline_cp);
```

FIXME

**qspline**: FIXME**lh**: FIXME**knot**: FIXME**qspline\_cp**: FIXME**nc\_hicosmo\_qspline\_add\_continuity\_priors ()**

```
NcHICosmoQSplineContPrior * nc_hicosmo_qspline_add_continuity_priors
(NcHICosmoQSpline *qspline,
 NcmLikelihood *lh,
 gdouble sigma);
```

FIXME

**qspline**: FIXME**lh**: FIXME**sigma**: FIXME**Returns**: FIXME. *[transfer full]***nc\_hicosmo\_qspline\_add\_continuity\_constraint ()**

```
void nc_hicosmo_qspline_add_continuity_constraint
(NcHICosmoQSpline *qspline,
 NcmFit *fit,
 gint knot,
 NcHICosmoQSplineContPrior * ↔
 qspline_cp);
```

FIXME

**qspline**: FIXME**fit**: FIXME**knot**: FIXME**qspline\_cp**: FIXME

**nc\_hicosmo\_qspline\_add\_continuity\_constraints ()**

```
NcHICosmoQSplineContPrior * nc_hicosmo_qspline_add_continuity_constraints
                                (NcHICosmoQSpline *qspline,
                                 NcmFit *fit,
                                 gdouble sigma);
```

FIXME

**qspline**: FIXME

**fit**: FIXME

**sigma**: FIXME

**Returns**: FIXME. *[transfer full]*

**struct NcHICosmoQSplineContPriorClass**

```
struct NcHICosmoQSplineContPriorClass {
};
```

**struct NcHICosmoQSplineContPrior**

```
struct NcHICosmoQSplineContPrior;
```

**nc\_hicosmo\_qspline\_cont\_prior\_new ()**

```
NcHICosmoQSplineContPrior * nc_hicosmo_qspline_cont_prior_new
                                (guint nknots);
```

FIXME

**nknots**: FIXME

**Returns**: FIXME. *[transfer full]*

**nc\_hicosmo\_qspline\_cont\_prior\_ref ()**

```
NcHICosmoQSplineContPrior * nc_hicosmo_qspline_cont_prior_ref
                                (NcHICosmoQSplineContPrior * ↔
                                 qspline_cp);
```

FIXME

**qspline\_cp**: FIXME

**Returns**: FIXME. *[transfer full]*

**nc\_hicosmo\_qspline\_cont\_prior\_free ()**

```
void nc_hicosmo_qspline_cont_prior_free (NcHICosmoQSplineContPrior * ↔
                                           qspline_cp);
```

FIXME

**qspline\_cp**: FIXME

**nc\_hicosmo\_qspline\_cont\_prior\_set\_nknots ()**

```
void                nc_hicosmo_qspline_cont_prior_set_nknots
                    (NcHICosmoQSplineContPrior * ↔
                     qspline_cp,
                     guint nknots);
```

FIXME

**qspline\_cp**: FIXME**nknots**: FIXME**nc\_hicosmo\_qspline\_cont\_prior\_get\_nknots ()**

```
guint              nc_hicosmo_qspline_cont_prior_get_nknots
                    (NcHICosmoQSplineContPrior * ↔
                     qspline_cp);
```

FIXME

**qspline\_cp**: FIXME**Returns**: FIXME**nc\_hicosmo\_qspline\_cont\_prior\_set\_lnsigma ()**

```
void                nc_hicosmo_qspline_cont_prior_set_lnsigma
                    (NcHICosmoQSplineContPrior * ↔
                     qspline_cp,
                     guint i,
                     gdouble ln_sigma);
```

FIXME

**qspline\_cp**: FIXME**i**: FIXME**ln\_sigma**: FIXME**nc\_hicosmo\_qspline\_cont\_prior\_set\_all\_lnsigma ()**

```
void                nc_hicosmo_qspline_cont_prior_set_all_lnsigma
                    (NcHICosmoQSplineContPrior * ↔
                     qspline_cp,
                     gdouble ln_sigma);
```

FIXME

**qspline\_cp**: FIXME**ln\_sigma**: FIXME



**nc\_hicosmo\_qspline\_cont\_prior\_get Insgma ()**

```
gdouble          nc_hicosmo_qspline_cont_prior_get_lnsigma
                                                           (NcHICosmoQSplineContPrior * ↔
                                                           qspline_cp,
                                                           guint i);
```

FIXME

*qspline\_cp*: FIXME

*i*: FIXME

*Returns* : FIXME

**Property Details**

**The "H0" property**

"H0"	gdouble	: Read / Write
------	---------	----------------

H\_0.

Allowed values: [10,500]

Default value: 73

**The "H0-fit" property**

"H0-fit"	gboolean	: Read / Write
----------	----------	----------------

H\_0:fit.

Default value: FALSE

**The "Omegat" property**

"Omegat"	gdouble	: Read / Write
----------	---------	----------------

Omega\_t.

Allowed values: [-5,5]

Default value: 1

**The "Omegat-fit" property**

"Omegat-fit"	gboolean	: Read / Write
--------------	----------	----------------

Omega\_t:fit.

Default value: FALSE

**The "asdrag" property**

"asdrag"	gdouble	: Read / Write
----------	---------	----------------

A\_s drag.

Allowed values: [0,5]

Default value: 0.035

---

**The "asdrag-fit" property**

"asdrag-fit"	gboolean	: Read / Write
--------------	----------	----------------

A\_s drag:fit.

Default value: FALSE

**The "qparam" property**

"qparam"	GVariant*	: Read / Write
----------	-----------	----------------

qparam.

Allowed values: GVariant<ad>

Default value: NULL

**The "qparam-fit" property**

"qparam-fit"	GVariant*	: Read / Write
--------------	-----------	----------------

qparam:fit.

Allowed values: GVariant<a\*>

Default value: NULL

**The "qparam-length" property**

"qparam-length"	guint	: Read / Write / Construct Only
-----------------	-------	---------------------------------

qparam:length.

Default value: 3

**The "spline" property**

"spline"	NcmSpline*	: Read / Write / Construct Only
----------	------------	---------------------------------

Spline object.

**The "zf" property**

"zf"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

final redshift.

Allowed values: [0,100]

Default value: 1

**The "nknots" property**

"nknots"	guint	: Read / Write / Construct Only
----------	-------	---------------------------------

Number of knots.

Allowed values: >= 3

Default value: 3

---

# Chapter 5

# Cosmological Functions

## 5.1 Cosmological Distances and Times

Cosmological Distances and Times — Calculate cosmological distances and related quantities.

## Synopsis

gdouble	(*NcDistanceFunc0)	(NcDistance *dist, NcHICosmo *cosmo);
gdouble	(*NcDistanceFunc1)	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
struct	NcDistanceClass;	
struct	NcDistance;	
NcDistance *	nc_distance_new	(gdouble z_f);
NcDistance *	nc_distance_ref	(NcDistance *dist);
void	nc_distance_prepare	(NcDistance *dist, NcHICosmo *cosmo);
void	nc_distance_prepare_if_needed	(NcDistance *dist, NcHICosmo *cosmo);
void	nc_distance_free	(NcDistance *dist);
void	nc_distance_clear	(NcDistance **dist);
NcmMSetFunc *	nc_distance_func0_new	(NcDistance *dist, NcDistanceFunc0 f0);
NcmMSetFunc *	nc_distance_func1_new	(NcDistance *dist, NcDistanceFunc1 f1);
gdouble	nc_distance_hubble	(NcDistance *dist, NcHICosmo *cosmo);
gdouble	nc_distance_decoupling_redshift	(NcDistance *dist, NcHICosmo *cosmo);
gdouble	nc_distance_drag_redshift	(NcDistance *dist, NcHICosmo *cosmo);
gdouble	nc_distance_shift_parameter_lss	(NcDistance *dist, NcHICosmo *cosmo);
gdouble	nc_distance_comoving_lss	(NcDistance *dist, NcHICosmo *cosmo);
gdouble	nc_distance_acoustic_scale	(NcDistance *dist, NcHICosmo *cosmo);
gdouble	nc_distance_Omega_k	(NcDistance *dist, NcHICosmo *cosmo);

gdouble	nc_distance_angular_diameter_curvature_scale	(NcDistance *dist, NcHICosmo *cosmo);
gdouble	nc_distance_comoving	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_transverse	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_luminosity	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_modulus	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_luminosity_hef	(NcDistance *dist, NcHICosmo *cosmo, gdouble z_he, gdouble z_cmb);
gdouble	nc_distance_modulus_hef	(NcDistance *dist, NcHICosmo *cosmo, gdouble z_he, gdouble z_cmb);
gdouble	nc_distance_shift_parameter	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_dilation_scale	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_bao_A_scale	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_sound_horizon	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_dsound_horizon_dz	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_bao_r_Dv	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_cosmic_time	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_lookback_time	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_conformal_time	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_conformal_lookback_time	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
gdouble	nc_distance_cosmic_time_mks_scale	(NcDistance *dist, NcHICosmo *cosmo);
gdouble	nc_distance_conformal_time_mks_scale	(NcDistance *dist,

---

```
NcHICosmo *cosmo);
```

## Object Hierarchy

```
GObject
+----NcDistance
```

## Properties

```
"zf"                                gdouble                                : Read / Write / Construct
```

## Description

This object implements several distances used in cosmology, here we have the following definitions.

The Hubble scale is simply defined as the inverse of the Hubble function  $H(z)$  [[nc\\_hicosmo\\_H\(\)](#)],  $\begin{equation}\label{eq:def:DH} D_H(z) = \frac{c}{H(z)}, \quad D_{H0} = \frac{c}{H_0}. \end{equation}$  where  $c$  is the speed of light [[ncm\\_c\\_c\(\)](#)],  $z$  is the redshift and  $H_0 \equiv H(0)$  is the Hubble parameter [[nc\\_hicosmo\\_H0\(\)](#)]. The comoving distance  $D_c$  is defined as  $\begin{equation}\label{eq:def:Dc} D_c(z) = \int_0^z \frac{dz'}{E(z')}, \end{equation}$  where  $E(z)$  is the normalized Hubble function [[nc\\_hicosmo\\_E\(\)](#)], i.e.,  $\begin{equation}\label{eq:def:Ez} E(z) \equiv \frac{H(z)}{H_0}. \end{equation}$  Note that both quantities are adimensional, in other words, one can think them as in units of the Hubble scale today.

The transverse comoving distance  $D_t$  and its derivative with respect to  $z$  are given by  $\begin{equation}\label{eq:def:Dt} D_t(z) = \frac{\sinh(\sqrt{\Omega_{k0}} D_c(z))}{\sqrt{\Omega_{k0}}}, \quad \frac{dD_t}{dz}(z) = \frac{\cosh(\sqrt{\Omega_{k0}} D_c(z))}{\sqrt{\Omega_{k0}}} \end{equation}$  where  $\Omega_{k0}$  is the value of the curvature today [[nc\\_hicosmo\\_Omega\\_k\(\)](#)]. Using the definition above we have that the luminosity distance is  $\begin{equation}\label{eq:def:Dl} D_l = (1+z)D_t, \end{equation}$  and the distance modulus is given by  $\begin{equation}\label{eq:def:mu} \mu(z) = 5 \log_{10}(D_l(z)) + 25, \end{equation}$  where  $\log_{10}$  represents the logarithm in the decimal base. Note that the distance modulus is usually defined as  $5 \log_{10}(D_{H0} D_l(z) / \text{pc}) - 5$ , where  $\text{pc}$  is parsec [[ncm\\_c\\_pc\(\)](#)]. Thus, this differs from our definition by a factor of  $5 \log_{10}(D_{H0} / \text{Mpc})$ , where  $\text{Mpc}$  is megaparsec [[ncm\\_c\\_Mpc\(\)](#)].

## Details

### NcDistanceFunc0 ()

```
gdouble (*NcDistanceFunc0) (NcDistance *dist,
                             NcHICosmo *cosmo);
```

### NcDistanceFunc1 ()

```
gdouble (*NcDistanceFunc1) (NcDistance *dist,
                             NcHICosmo *cosmo,
                             gdouble z);
```

### struct NcDistanceClass

```
struct NcDistanceClass {
};
```

**struct NcDistance**

```
struct NcDistance;
```

**nc\_distance\_new ()**

```
NcDistance *      nc_distance_new      (gdouble z_f);
```

Creates a new **NcDistance** object optimized to perform distance calculations to redshift up to  $z_f$ .

**z\_f** : final redshift  $z_f$ .

**Returns** : a new **NcDistance**.

**nc\_distance\_ref ()**

```
NcDistance *      nc_distance_ref      (NcDistance *dist);
```

FIXME

**dist** : a **NcDistance**.

**Returns** : FIXME. *[transfer full]*

**nc\_distance\_prepare ()**

```
void              nc_distance_prepare   (NcDistance *dist,  
                                         NcHICosmo *cosmo);
```

FIXME

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**nc\_distance\_prepare\_if\_needed ()**

```
void              nc_distance_prepare_if_needed (NcDistance *dist,  
                                                NcHICosmo *cosmo);
```

FIXME

**dist** : FIXME,

**cosmo** : FIXME

**nc\_distance\_free ()**

```
void              nc_distance_free      (NcDistance *dist);
```

FIXME

**dist** : a **NcDistance**.

---

**nc\_distance\_clear ()**

```
void nc_distance_clear (NcDistance **dist);
```

FIXME

**dist** : a **NcDistance**.

**nc\_distance\_func0\_new ()**

```
NcmMSetFunc * nc_distance_func0_new (NcDistance *dist,
                                     NcDistanceFunc0 f0);
```

**dist** : FIXME

**f0** : FIXME. *[scope notified]*

**Returns** : FIXME. *[transfer full]*

**nc\_distance\_func1\_new ()**

```
NcmMSetFunc * nc_distance_func1_new (NcDistance *dist,
                                     NcDistanceFunc1 f1);
```

**dist** : FIXME

**f1** : FIXME. *[scope notified]*

**Returns** : FIXME. *[transfer full]*

**nc\_distance\_hubble ()**

```
gdouble nc_distance_hubble (NcDistance *dist,
                             NcHICosmo *cosmo);
```

Calculate the curvature scale today as defined in Eq [\eqref{eq:def:DH}](#) in units of megaparsec (Mpc) [**ncm\_c\_Mpc()**].

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**Returns** :  $D_{\{H0\}}$ .

**nc\_distance\_decoupling\_redshift ()**

```
gdouble nc_distance_decoupling_redshift (NcDistance *dist,
                                         NcHICosmo *cosmo);
```

The decoupling redshift  $z_{\star}$  corresponds to the epoch of the last scattering surface of the cosmic microwave background photons.

This function computes  $z_{\star}$  using [**nc\_hicosmo\_z\_iss()**], if *cosmo* implements it, or using Hu & Sugiyama fitting formula [Hu \(1996\)](#),  $z_{\star} = 1048 \left(1 + 1.24 \times 10^{-3} (\Omega_b h^2)^{-0.738}\right) \left(1 + g_1 (\Omega_m h^2)^{g_2}\right)$ , where  $\Omega_b h^2$  [**nc\_hicosmo\_Omega\_bh2()**] and  $\Omega_m h^2$  [**nc\_hicosmo\_Omega\_mh2()**] are, respectively, the baryonic and matter density parameters times the square of the dimensionless Hubble parameter  $h$ ,  $H_0 = 100 \text{ km/s/Mpc}$ . The parameters  $g_1$  and  $g_2$  are given by  $g_1 = \frac{0.0783 (\Omega_b h^2)^{-0.238}}{(1 + 39.5 (\Omega_b h^2)^{0.763})}$ ; and  $g_2 = \frac{0.56}{(1 + 21.1 (\Omega_b h^2)^{1.81})}$ .

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**Returns** :  $z_{\star}$

### nc\_distance\_drag\_redshift ()

```
gdouble          nc_distance_drag_redshift      (NcDistance *dist,
                                                NcHICosmo *cosmo);
```

Drag redshift is the epoch at which baryons were released from photons.

This function computes  $z_d$  using the fitting formula given in [Eisenstein & Hu \(1998\)](#),  $z_d = \frac{1291}{h^2} \frac{(\Omega_m h^2)^{0.251}}{(1 + 0.659 (\Omega_m h^2)^{0.828})} \left(1 + b_1 (\Omega_b h^2)^{b_2}\right)$ , where  $\Omega_b h^2$  [[nc\\_hicosmo\\_Omega\\_bh2\(\)](#)] and  $\Omega_m h^2$  [[nc\\_hicosmo\\_Omega\\_mh2\(\)](#)] are, respectively, the baryonic and matter density parameters times the square of the dimensionless Hubble parameter  $h$ ,  $H_0 = 100 h$   $\frac{\text{km/s}}{\text{Mpc}}$ . The parameters  $b_1$  and  $b_2$  are given by  $b_1 = 0.313 (\Omega_m h^2)^{-0.419} \left(1 + 0.607 (\Omega_m h^2)^{0.674}\right)$ ; and  $b_2 = 0.238 (\Omega_m h^2)^{0.223}$ .

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**Returns** :  $z_d$ .

### nc\_distance\_shift\_parameter\_lss ()

```
gdouble          nc_distance_shift_parameter_lss  (NcDistance *dist,
                                                NcHICosmo *cosmo);
```

Compute the shift parameter  $R(z)$  [[nc\\_distance\\_shift\\_parameter\(\)](#)] at the decoupling redshift  $z_{\star}$  [[nc\\_distance\\_decoupling\\_redshift\(\)](#)].

**dist** : a **NcDistance**

**cosmo** : a **NcHICosmo**

**Returns** :  $R(z_{\star})$ .

### nc\_distance\_comoving\_lss ()

```
gdouble          nc_distance_comoving_lss        (NcDistance *dist,
                                                NcHICosmo *cosmo);
```

Compute the comoving distance  $D_c(z)$  [Eq. [\eqref{eq:def:Dc}](#)] at the decoupling redshift  $z_{\star}$  [[nc\\_distance\\_decoupling\\_redshift\(\)](#)].

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**Returns** :  $D_c(z_{\star})$ .



**nc\_distance\_acoustic\_scale ()**

```
gdouble          nc_distance_acoustic_scale      (NcDistance *dist,
                                                  NcHICosmo *cosmo);
```

Compute the acoustic scale  $l_A(z_{\star})$  at  $z_{\star}$  [[nc\\_distance\\_decoupling\\_redshift\(\)](#)], 
$$l_A(z_{\star}) = \pi \frac{D_t(z_{\star})}{r_s(z_{\star})},$$
 where  $D_t(z_{\star})$  is the comoving transverse distance [[nc\\_distance\\_transverse\(\)](#)] and  $r_s(z_{\star})$  is the sound horizon [[nc\\_distance\\_sound\\_horizon\(\)](#)] both both computed at  $z_{\star}$ .

**dist** : a [NcDistance](#).

**cosmo** : a [NcHICosmo](#).

**Returns** :  $l_A(z_{\star})$ .

**nc\_distance\_Omega\_k ()**

```
gdouble          nc_distance_Omega_k            (NcDistance *dist,
                                                  NcHICosmo *cosmo);
```

**nc\_distance\_angular\_diameter\_curvature\_scale ()**

```
gdouble          nc_distance_angular_diameter_curvature_scale
                                                  (NcDistance *dist,
                                                  NcHICosmo *cosmo);
```

We define the angular diameter curvature scale  $D_a(z_{\star})$  as  $D_a(z_{\star}) = \frac{E(z_{\star})}{1 + z_{\star}} D_t(z_{\star})$ , where  $z_{\star}$  is the decoupling redshift, given by [[nc\\_distance\\_decoupling\\_redshift\(\)](#)],  $E(z_{\star})$  is the normalized Hubble function [Eq. [\eqref{eq:def:Ez}](#)] and  $D_t(z_{\star})$  is the transverse comoving distance [Eq. [\eqref{eq:def:Dt}](#)] both computed at  $z_{\star}$ .

**dist** : a [NcDistance](#)

**cosmo** : a [NcHICosmo](#)

**Returns** :  $D_a(z_{\star})$ .

**nc\_distance\_comoving ()**

```
gdouble          nc_distance_comoving           (NcDistance *dist,
                                                  NcHICosmo *cosmo,
                                                  gdouble z);
```

Calculate the comoving distance  $D_c(z)$  as defined in Eq. [\eqref{eq:def:Dc}](#).

**dist** : a [NcDistance](#).

**cosmo** : a [NcHICosmo](#).

**z** : redshift  $z$ .

**Returns** :  $D_c(z)$ .

**nc\_distance\_transverse ()**

```
gdouble          nc_distance_transverse      (NcDistance *dist,
                                              NcHICosmo *cosmo,
                                              gdouble z);
```

Compute the transverse comoving distance  $D_t(z)$  defined in Eq. [\eqref{eq:def:Dt}](#).

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z** : redshift  $z$ .

**Returns** :  $D_t(z)$ .

**nc\_distance\_luminosity ()**

```
gdouble          nc_distance_luminosity      (NcDistance *dist,
                                              NcHICosmo *cosmo,
                                              gdouble z);
```

Compute the luminosity distance  $D_l(z)$  defined in Eq. [\eqref{eq:def:Dl}](#).

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z** : redshift  $z$ .

**Returns** :  $D_l(z)$ .

**nc\_distance\_modulus ()**

```
gdouble          nc_distance_modulus         (NcDistance *dist,
                                              NcHICosmo *cosmo,
                                              gdouble z);
```

Compute the distance modulus  $\mu(z)$  defined in Eq. [\eqref{eq:def:mu}](#).

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z** : redshift  $z$ .

**Returns** :  $\mu(z)$ .

**nc\_distance\_luminosity\_hef ()**

```
gdouble          nc_distance_luminosity_hef  (NcDistance *dist,
                                              NcHICosmo *cosmo,
                                              gdouble z_he,
                                              gdouble z_cmb);
```

Calculate the luminosity distance  $D_l$  corrected to our local frame.

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z\_he** : redshift  $z_{\text{he}}$  in our local frame.

**z\_cmb** : redshift  $z_{\text{CMB}}$  in the CMB frame.

**Returns** :  $D_l(z_{\text{he}}, z_{\text{CMB}})$ .

### nc\_distance\_modulus\_hef ()

```
gdouble          nc_distance_modulus_hef      (NcDistance *dist,
                                                NcHICosmo *cosmo,
                                                gdouble z_he,
                                                gdouble z_cmb);
```

Calculate the distance modulus using the frame corrected luminosity distance [**nc\_distance\_luminosity\_hef()**].

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z\_he** : redshift  $z_{\text{he}}$  in our local frame.

**z\_cmb** : redshift  $z_{\text{CMB}}$  in the CMB frame.

**Returns** :  $\mu(z_{\text{he}}, z_{\text{CMB}})$ .

### nc\_distance\_shift\_parameter ()

```
gdouble          nc_distance_shift_parameter  (NcDistance *dist,
                                                NcHICosmo *cosmo,
                                                gdouble z);
```

The shift parameter  $R(z)$  is defined as 
$$R(z) = \frac{\sqrt{\Omega_m H_0^2}}{(1+z) D_A(z)} \sqrt{\Omega_m} D_t(z),$$
 where  $\Omega_m$  is the matter density parameter [**nc\_hicosmo\_Omega\_m()**],  $D_A(z) = D_{H_0} D_t(z) / (1+z)$  is the angular diameter distance and  $D_t(z)$  is the transverse comoving distance [Eq. [\eqref{eq:def:Dt}](#)].

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z** : redshift  $z$ .

**Returns** :  $R(z)$ .

### nc\_distance\_dilation\_scale ()

```
gdouble          nc_distance_dilation_scale   (NcDistance *dist,
                                                NcHICosmo *cosmo,
                                                gdouble z);
```

The dilation scale is the cube root of the product of the radial dilation times the square of the transverse dilation -- (arXiv:astro-ph/0501171)

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z** : redshift  $z$ .

**Returns** :  $D_V(z)$ .

**nc\_distance\_bao\_A\_scale ()**

```
gdouble          nc_distance_bao_A_scale      (NcDistance *dist,
                                                NcHICosmo *cosmo,
                                                gdouble z);
```

Bao 'A' scale  $D_v(z) \sqrt{\Omega_m} / z$  -- (arXiv:astro-ph/0501171)

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z** : the redshift  $z$ .

**Returns** : FIXME

**nc\_distance\_sound\_horizon ()**

```
gdouble          nc_distance_sound_horizon    (NcDistance *dist,
                                                NcHICosmo *cosmo,
                                                gdouble z);
```

Compute the sound horizon  $r_s$ , 
$$r_s(z) = \int_z^\infty \frac{c_s(z')}{E(z')} dz',$$
 where  $c_s(z)$  is the speed of sound wave and  $E(z)$  is the normalized Hubble function [**nc\_hicosmo\_E()**].

The integrand is given by 
$$\frac{c_s(z')}{H(z')} = \frac{1}{\sqrt{E(z')}} \frac{1}{(3 + \frac{9}{4} (1 + 0.2271 n_{\text{eff}}) \frac{\Omega_b}{\Omega_r (1 + z')})},$$
 where  $n_{\text{eff}}$  is the effective number of neutrinos [**ncm\_c\_neutrino\_n\_eff()**],  $\Omega_b$  [**nc\_hicosmo\_Omega\_b()**] and  $\Omega_r$  [**nc\_hicosmo\_Omega\_r()**] are the baryonic and radiation density parameter, respectively. If  $\Omega_r = 0$ , the integrand returns 0.0.

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z** : redshift  $z$ .

**Returns** :  $r_s(z)$ .

**nc\_distance\_dsound\_horizon\_dz ()**

```
gdouble          nc_distance_dsound_horizon_dz (NcDistance *dist,
                                                NcHICosmo *cosmo,
                                                gdouble z);
```

Calculate the sound horizon [**nc\_distance\_sound\_horizon()**] derivative with respect to  $z$ , 
$$\frac{dr_s(z)}{dz} = - \frac{c_s(z)}{E(z)},$$
 where  $c_s(z) / E(z)$  is given by Eq. [\eqref{eq:def:rs:integrand}](#).

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z** : redshift  $z$ .

**Returns** :  $\frac{dr_s(z)}{dz}$ .

**nc\_distance\_bao\_r\_Dv ()**

gdouble	nc_distance_bao_r_Dv	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
---------	----------------------	--

$r(z_d) / D_v(z)$  -- (arXiv:0705.3323).

**dist** : a **NcDistance**.

**cosmo** : a **NcHICosmo**.

**z** : the redshift \$z\$.

**Returns** : FIXME

**nc\_distance\_cosmic\_time ()**

gdouble	nc_distance_cosmic_time	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
---------	-------------------------	--

**nc\_distance\_lookback\_time ()**

gdouble	nc_distance_lookback_time	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
---------	---------------------------	--

**nc\_distance\_conformal\_time ()**

gdouble	nc_distance_conformal_time	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
---------	----------------------------	--

**nc\_distance\_conformal\_lookback\_time ()**

gdouble	nc_distance_conformal_lookback_time	(NcDistance *dist, NcHICosmo *cosmo, gdouble z);
---------	-------------------------------------	--

**nc\_distance\_cosmic\_time\_mks\_scale ()**

gdouble	nc_distance_cosmic_time_mks_scale	(NcDistance *dist, NcHICosmo *cosmo);
---------	-----------------------------------	--

**nc\_distance\_conformal\_time\_mks\_scale ()**

gdouble	nc_distance_conformal_time_mks_scale	(NcDistance *dist, NcHICosmo *cosmo);
---------	--------------------------------------	--

## Property Details

### The "zf" property

"zf"	gdouble	: Read / Write / Construct
------	---------	----------------------------

Final cached redshift.

Allowed values:  $\geq 0$

Default value: 10

## 5.2 Recombination

Recombination — Cosmic recombination abstract object.

### Synopsis

```
#include <numcosmo/nc_recomb.h>
```

```
struct          NcRecombClass;
struct          NcRecomb;
NcRecomb *      nc_recomb_new_from_name      (gchar *recomb_name);
NcRecomb *      nc_recomb_ref                (NcRecomb *recomb);
void            nc_recomb_free               (NcRecomb *recomb);
void            nc_recomb_prepare            (NcRecomb *recomb,
                                             NcHICosmo *cosmo);
void            nc_recomb_prepare_if_needed  (NcRecomb *recomb,
                                             NcHICosmo *cosmo);
gdouble         nc_recomb_HI_ion_saha        (NcHICosmo *cosmo,
                                             const gdouble x);
gdouble         nc_recomb_HeI_ion_saha       (NcHICosmo *cosmo,
                                             const gdouble x);
gdouble         nc_recomb_HeII_ion_saha      (NcHICosmo *cosmo,
                                             const gdouble x);
gdouble         nc_recomb_HeII_ion_saha_x    (NcHICosmo *cosmo,
                                             const gdouble f);
gdouble         nc_recomb_HeII_ion_saha_x_by_HeIII_He
                                             (NcHICosmo *cosmo,
                                             const gdouble f);
gdouble         nc_recomb_He_fully_ionized_Xe
                                             (NcHICosmo *cosmo,
                                             const gdouble x);
gdouble         nc_recomb_equilibrium_Xe     (NcRecomb *recomb,
                                             NcHICosmo *cosmo,
                                             const gdouble x);
gdouble         nc_recomb_Xe                 (NcRecomb *recomb,
                                             NcHICosmo *cosmo,
                                             const gdouble lambda);
gdouble         nc_recomb_dtau_dx            (NcRecomb *recomb,
                                             NcHICosmo *cosmo,
                                             const gdouble lambda);
gdouble         nc_recomb_dtau_dlambda      (NcRecomb *recomb,
                                             NcHICosmo *cosmo,
                                             const gdouble lambda);
gdouble         nc_recomb_d2tau_dlambda2    (NcRecomb *recomb,
```

		NcHICosmo *cosmo, const gdouble lambda);
gdouble	nc_recomb_d3tau_dlambd3	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
gdouble	nc_recomb_tau	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
gdouble	nc_recomb_tau_lambda0_lambda1	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda0, const gdouble lambda1);
gdouble	nc_recomb_log_v_tau	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
gdouble	nc_recomb_v_tau	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
gdouble	nc_recomb_dv_tau_dlambd	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
gdouble	nc_recomb_d2v_tau_dlambd2	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
gdouble	nc_recomb_v_tau_lambda_mode	(NcRecomb *recomb, NcHICosmo *cosmo);
void	nc_recomb_v_tau_lambda_features	(NcRecomb *recomb, NcHICosmo *cosmo, gdouble logref, gdouble *lambda_max, gdouble *lambda_l, gdouble *lambda_u);
gdouble	nc_recomb_tau_zstar	(NcRecomb *recomb, NcHICosmo *cosmo);
gdouble	nc_recomb_tau_cutoff	(NcRecomb *recomb, NcHICosmo *cosmo);
gdouble	nc_recomb_tau_zdrag	(NcRecomb *recomb, NcHICosmo *cosmo);
gdouble	nc_recomb_dtau_dlambd_Xe	(NcHICosmo *cosmo, const gdouble lambda);
gdouble	nc_recomb_He_fully_ionized_dtau_dlambd	(NcHICosmo *cosmo, const gdouble lambda);
gdouble	nc_recomb_pequignot_HI_case_B	(NcHICosmo *cosmo, const gdouble Tm);
gdouble	nc_recomb_pequignot_HI_case_B_dTm	(NcHICosmo *cosmo, const gdouble Tm);
gdouble	nc_recomb_hummer_HeI_case_B	(NcHICosmo *cosmo, const gdouble Tm);
gdouble	nc_recomb_hummer_HeI_case_B_dTm	(NcHICosmo *cosmo, const gdouble Tm);
gdouble	nc_recomb_weinberg_HII_ion_rate	(NcHICosmo *cosmo, gdouble XHII, gdouble Tm, gdouble XHeII, gdouble x);

## Object Hierarchy

```
GObject
+----NcRecomb
      +----NcRecombSeager
```

## Properties

"init-frac"	gdouble	: Read / Write / Construct
"prec"	gdouble	: Read / Write / Construct
"zi"	gdouble	: Read / Write / Construct

## Description

`\newcommand{\He}{\text{He}}` `\newcommand{\HeI}{\text{HeI}}` `\newcommand{\HeII}{\text{HeII}}` `\newcommand{\HeIII}{\text{HeIII}}`  
`\newcommand{\Hy}{\text{H}}` `\newcommand{\HyI}{\text{HI}}` `\newcommand{\HyII}{\text{HII}}` `\newcommand{\e}{\text{e}^-}`  
`}} $`

The **NcRecomb** abstract object describe a general recombination process. To describe the functions we have the following definitions, more details in [Weinberg \(2008\)](#).

We refer to the total number of hydrogen nucleus (ionized or not) as  $n_{\text{Hy}}$ , the neutral hydrogen atoms as  $n_{\text{HyI}}$  and ionized hydrogen as  $n_{\text{HyII}}$  and therefore  $n_{\text{HyI}} + n_{\text{HyII}} = n_{\text{Hy}}$ . In the same way for the helium the number of helium nuclei is  $n_{\text{He}}$  and the neutral, single and double ionized as  $n_{\text{HeI}}$ ,  $n_{\text{HeII}}$  and  $n_{\text{HeIII}}$  respectively.

We also define the helium primordial abundance as the ratio of the helium mass to the total baryonic mass, i.e.,  $Y_p = \frac{n_{\text{He}} m_{\text{He}}}{(n_{\text{He}} m_{\text{He}} + n_{\text{Hy}} m_{\text{Hy}})}$  where  $m_{\text{Hy}}$  and  $m_{\text{He}}$  are the hydrogen and helium mass.

The element abundances are define as the ratio of the element by the total number of free protons  $n_p \equiv n_{\text{Hy}}$ :  $X_f = \frac{n_f}{n_p}$  where  $f$  is any one of the elements describe above and  $e$  represent the number of free electrons.

These fractions have the following properties:  $X_{\text{HyI}} + X_{\text{HyII}} = 1$ ,  $X_{\text{HeI}} + X_{\text{HeII}} + X_{\text{HeIII}} = X_{\text{He}}$ ,  $X_{\text{He}} \equiv \frac{m_p}{m_{\text{He}}} \frac{Y_p}{1 - Y_p}$ . We also define the number of free electrons as  $n_e$ . Assuming a neutral universe we have  $X_e = X_{\text{HyII}} + X_{\text{HeII}} + 2X_{\text{HeIII}}$ .

## Equilibrium fractions

Equilibrium ratio  $X_{\text{HyII}} X_e / X_{\text{Hy}}$  through Saha equation, i.e.,  $\frac{X_{\text{HyII}} X_e}{X_{\text{Hy}}} = \frac{e^{-\text{HyI}_{1s}/(k_{\text{BT}})}}{n_{\text{Hy}} \lambda_e^3}$  where  $\text{HyI}_{1s}$  is the hydrogen  $1s$  binding energy `ncm_c_H_bind_1s()`,  $\lambda_e$  is the electron thermal wavelength, i.e.,  $\lambda_e = \sqrt{\frac{2\pi\hbar^2}{m_e k_{\text{B}} T}}$  where  $k_{\text{B}}$  is the Boltzmann constant `ncm_c_kb()`,  $m_e$  the electron mass and  $\hbar$  is the Planck constant `ncm_c_hbar()`.

This calculation is done using the Saha equation as in [Weinberg \(2008\)](#).

The equilibrium single/non-ionized helium ratio  $X_{\text{HeII}} X_e / X_{\text{HeI}}$  through Saha equation, i.e.,  $\frac{X_{\text{HeII}} X_e}{X_{\text{HeI}}} = \frac{e^{-\text{HeI}_{1s}/(k_{\text{BT}})}}{4n_{\text{Hy}} \lambda_e^3}$  where  $\text{HeI}_{1s}$  is the helium I  $1s$  binding energy `ncm_c_HeI_bind_1s()`. This calculation is done using the Saha equation as in [Seager \(2000\)](#).

The equilibrium double/single-ionized helium ratio  $X_{\text{HeIII}} X_e / X_{\text{HeII}}$  through Saha equation, i.e.,  $\frac{X_{\text{HeIII}} X_e}{X_{\text{HeII}}} = \frac{e^{-\text{HeII}_{1s}/(k_{\text{BT}})}}{4n_{\text{Hy}} \lambda_e^3}$  where  $\text{HeII}_{1s}$  is the helium II  $1s$  binding energy `ncm_c_HeII_bind_1s()`. This calculation is done using the Saha equation as in [Seager \(2000\)](#).

The default value of the helium primordial abundance is given by `ncm_c_prim_He_Yp()`. The primordial helium fraction is define by `ncm_c_prim_XHe()`.



## Optical depth and visibility function

The derivative of the optical depth  $\tau$  with respect to the redshift time  $\lambda \equiv -\log(x) = -\log(1+z)$  is  $\frac{d\tau}{d\lambda} = -\frac{c\sigma_T n_B}{H}$ , where  $c$  is the speed of light [ncm\_c\_c()],  $\sigma_T$  is the Thomson cross section [ncm\_c\_thomson\_cs()],  $n_B$  is the number density of baryons and  $H$  the Hubble function. We define the optical depth  $\tau$  integrating from the present time, i.e.,  $\tau = \int_0^\lambda \frac{d\tau}{d\lambda} d\lambda$ . Using the equations above we define the visibility function  $v_\tau$  as  $v_\tau = \frac{d\tau}{d\lambda} e^{-\tau}$ .

## Details

### struct NcRecombClass

```
struct NcRecombClass {
};
```

### struct NcRecomb

```
struct NcRecomb;
```

### nc\_recomb\_new\_from\_name ()

```
NcRecomb *      nc_recomb_new_from_name      (gchar *recomb_name);
```

FIXME

**recomb\_name** : a string representing a **NcRecomb** object.

**Returns** : a new **NcRecomb**.

### nc\_recomb\_ref ()

```
NcRecomb *      nc_recomb_ref      (NcRecomb *recomb);
```

Increases the reference count of *recomb*.

**recomb** : a **NcRecomb**.

**Returns** : *recomb*. [transfer full]

### nc\_recomb\_free ()

```
void      nc_recomb_free      (NcRecomb *recomb);
```

Decreases the reference count of *recomb*.

**recomb** : a **NcRecomb**.

**nc\_recomb\_prepare ()**

```
void                nc_recomb_prepare                (NcRecomb *recomb,
                                                    NcHICosmo *cosmo);
```

Prepare the object using the model *cosmo*.

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**nc\_recomb\_prepare\_if\_needed ()**

```
void                nc_recomb_prepare_if_needed      (NcRecomb *recomb,
                                                    NcHICosmo *cosmo);
```

**nc\_recomb\_HI\_ion\_saha ()**

```
gdouble            nc_recomb_HI_ion_saha            (NcHICosmo *cosmo,
                                                    const gdouble x);
```

Calculate the equilibrium ionized/non-ionized hydrogen abundance ratio  $X_{\text{HyII}}X_{\text{e}} / X_{\text{HyI}}$ . See Eq. [\eqref{eq:saha:HyI}](#).

**cosmo** : a **NcHICosmo**.

**x** : redshift factor  $x$ .

**Returns** : the abundance ratio  $X_{\text{HyII}}X_{\text{e}} / X_{\text{HyI}}$ .

**nc\_recomb\_HeI\_ion\_saha ()**

```
gdouble            nc_recomb_HeI_ion_saha           (NcHICosmo *cosmo,
                                                    const gdouble x);
```

Calculate the equilibrium single/non-ionized helium ratio  $X_{\text{HeII}}X_{\text{e}}/X_{\text{HeI}}$ . See Eq. [\eqref{eq:saha:HeI}](#).

**cosmo** : a **NcHICosmo**.

**x** : redshift factor  $x$ .

**Returns** : the ratio  $X_{\text{HeII}}X_{\text{e}}/X_{\text{HeI}}$ .

**nc\_recomb\_HeII\_ion\_saha ()**

```
gdouble            nc_recomb_HeII_ion_saha          (NcHICosmo *cosmo,
                                                    const gdouble x);
```

Calculate the equilibrium double/single ionized helium ratio ( $X_{\text{HeIII}}X_{\text{e}}/X_{\text{HeII}}$ ). See Eq. [\eqref{eq:saha:HeII}](#).

**cosmo** : a **NcHICosmo**.

**x** : redshift factor  $x$ .

**Returns** : the ratio  $X_{\text{HeIII}}X_{\text{e}}/X_{\text{HeII}}$

**nc\_recomb\_HeII\_ion\_saha\_x ()**

```
gdouble          nc_recomb_HeII_ion_saha_x          (NcHICosmo *cosmo,
                                                    const gdouble f);
```

Calculate the redshift where the ratio  $X_{\text{HeIII}}/X_{\text{HeII}} = f$ . This calculation is done by finding the value of  $x$  where  $\frac{e^{-\text{HeII}_{1s}}/(k_{\text{BT}})}{4n_{\text{Hy}}\lambda_{\text{e}}^3} = f$ .

**cosmo** : a **NcHICosmo**.

**f** :  $X_{\text{HeIII}}/X_{\text{HeII}}$

**Returns** : the value of  $x$  where the ratio *f* occur.

**nc\_recomb\_HeII\_ion\_saha\_x\_by\_HeIII\_He ()**

```
gdouble          nc_recomb_HeII_ion_saha_x_by_HeIII_He
                                                    (NcHICosmo *cosmo,
                                                    const gdouble f);
```

Calculate the redshift where the ratio  $X_{\text{HeIII}}/X_{\text{He}} = f$ . This calculation is done assuming that hydrogen and helium are fully ionized, i.e.,  $\text{HyI} = 0 = \text{HeI}$ . In this case  $\frac{X_{\text{HeIII}}/X_{\text{He}}}{X_{\text{HeII}}} = \frac{f}{1-f} \left[ 1 + X_{\text{He}}(1+f) \right]$ .

**cosmo** : a **NcHICosmo**.

**f** : value of  $X_{\text{HeIII}}/X_{\text{He}}$

**Returns** : the value of  $x$  where the ratio *f* occur.

**nc\_recomb\_He\_fully\_ionized\_Xe ()**

```
gdouble          nc_recomb_He_fully_ionized_Xe      (NcHICosmo *cosmo,
                                                    const gdouble x);
```

Assuming that all helium is single or double ionized and all hydrogen is ionized we have  $X_{\text{e}} = 1 + X_{\text{HeII}} + 2X_{\text{HeIII}}$ ,  $X_{\text{He}} = X_{\text{HeII}} + X_{\text{HeIII}}$ , thus,  $X_{\text{HeIII}} = X_{\text{e}} - X_{\text{He}} - 1$ ,  $X_{\text{HeII}} = 1 + 2X_{\text{He}} - X_{\text{e}}$ . Using **nc\_recomb\_HeII\_ion\_saha\_x** and **ncm\_c\_prim\_XHe()** we obtain  $X_{\text{e}}$ .

**cosmo** : a **NcHICosmo**.

**x** : redshift factor  $x$ .

**Returns** :  $X_{\text{e}}$ .

**nc\_recomb\_equilibrium\_Xe ()**

```
gdouble          nc_recomb_equilibrium_Xe          (NcRecomb *recomb,
                                                    NcHICosmo *cosmo,
                                                    const gdouble x);
```

Calculates the ionization fraction  $X_{\text{e}}$  assuming equilibrium at all times. It solves the system containing all Saha's equations  $\text{Eq}_{\text{saha:HyI}}$ ,  $\text{Eq}_{\text{saha:HeI}}$  and  $\text{Eq}_{\text{saha:HeII}}$  and the constraints  $\text{Eq}_{\text{Hy:add}}$ ,  $\text{Eq}_{\text{He:add}}$  and  $\text{Eq}_{\text{def:Xe}}$ .

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**x** :  $x$ .

**Returns** :  $X_{\text{e}}$ .

**nc\_recomb\_Xe ()**

gdouble	nc_recomb_Xe	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	--------------	---

Calculates the value of  $X_e$  at  $x$ .

**recomb** : a **NcRecomb**

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $X_e$ .

**nc\_recomb\_dtau\_dx ()**

gdouble	nc_recomb_dtau_dx	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	-------------------	---

FIXME

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $d\tau/dx$ .

**nc\_recomb\_dtau\_dlambda ()**

gdouble	nc_recomb_dtau_dlambda	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	------------------------	---

FIXME

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $d\tau/d\lambda$ .

**nc\_recomb\_d2tau\_dlambda2 ()**

gdouble	nc_recomb_d2tau_dlambda2	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	--------------------------	---

FIXME

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $d^2\tau/d\lambda^2$ .

**nc\_recomb\_d3tau\_dlambda3 ()**

gdouble	nc_recomb_d3tau_dlambda3	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	--------------------------	---

FIXME

**recomb** : a **NcRecomb**.**cosmo** : a **NcHICosmo**.**lambda** :  $\lambda$ .**Returns** :  $d^3\tau/d\lambda^3$ .**nc\_recomb\_tau ()**

gdouble	nc_recomb_tau	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	---------------	---

FIXME

**recomb** : a **NcRecomb**.**cosmo** : a **NcHICosmo**.**lambda** :  $\lambda$ .**Returns** :  $\tau$ .**nc\_recomb\_tau\_lambda0\_lambda1 ()**

gdouble	nc_recomb_tau_lambda0_lambda1	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda0, const gdouble lambda1);
---------	-------------------------------	--

FIXME

**recomb** : a **NcRecomb**.**cosmo** : a **NcHICosmo**.**lambda0** :  $\lambda_0$ .**lambda1** :  $\lambda_1$ .**Returns** :  $\tau(\lambda_1) - \tau(\lambda_0)$ .

**nc\_recomb\_log\_v\_tau ()**

gdouble	nc_recomb_log_v_tau	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	---------------------	---

FIXME

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $\log(v_{\tau})$ .

**nc\_recomb\_v\_tau ()**

gdouble	nc_recomb_v_tau	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	-----------------	---

FIXME

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $v_{\tau}$ .

**nc\_recomb\_dv\_tau\_dlambda ()**

gdouble	nc_recomb_dv_tau_dlambda	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	--------------------------	---

FIXME

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $dv_{\tau}/d\lambda$ .

**nc\_recomb\_d2v\_tau\_dlambda2 ()**

gdouble	nc_recomb_d2v_tau_dlambda2	(NcRecomb *recomb, NcHICosmo *cosmo, const gdouble lambda);
---------	----------------------------	---

FIXME

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $d^2v_{\tau}/d\lambda^2$ .

**nc\_recomb\_v\_tau\_lambda\_mode ()**

```
gdouble          nc_recomb_v_tau_lambda_mode      (NcRecomb *recomb,
                                                    NcHICosmo *cosmo);
```

Calculate the maximum of the visibility function [Eq \eqref{eq:def:vtau}], the value of  $\lambda_{\text{max}}$  where  $dv_{\tau}(\lambda) = 0$ .

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**Returns** :  $\lambda_{\text{max}}$ .

**nc\_recomb\_v\_tau\_lambda\_features ()**

```
void          nc_recomb_v_tau_lambda_features      (NcRecomb *recomb,
                                                    NcHICosmo *cosmo,
                                                    gdouble logref,
                                                    gdouble *lambda_max,
                                                    gdouble *lambda_l,
                                                    gdouble *lambda_u);
```

Calculate the maximum of the visibility function [Eq \eqref{eq:def:vtau}], i.e, the value of  $\lambda_{\text{max}}$  where  $dv_{\tau}(\lambda) = 0$ , and the values where the visibility drop to  $v_{\tau}(\lambda_{\text{max}})e^{-\logref}$  to the left  $\lambda_l$  and to the right  $\lambda_u$  of  $\lambda_{\text{max}}$ .

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**logref** : the logarithm of the reference scale.

**lambda\_max** :  $\lambda_{\text{max}}$ . [out]

**lambda\_l** :  $\lambda_l$ . [out]

**lambda\_u** :  $\lambda_u$ . [out]

**nc\_recomb\_tau\_zstar ()**

```
gdouble          nc_recomb_tau_zstar              (NcRecomb *recomb,
                                                    NcHICosmo *cosmo);
```

Calculate the value of  $\lambda$  where the optical depth [Eq \eqref{eq:def:tau}] is equal to one, i.e.,  $\tau(\lambda^{\star}) = 1$ .

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**Returns** :  $\lambda^{\star}$ .

**nc\_recomb\_tau\_cutoff ()**

```
gdouble          nc_recomb_tau_cutoff          (NcRecomb *recomb,
                                                NcHICosmo *cosmo);
```

Calculate the value of  $\lambda$  where the optical depth [Eq. \eqref{eq:def:tau}] attains a value such that  $e^{-\tau(\lambda_{\text{cutoff}})} = \epsilon_{\text{double}}$ , i.e., is equal to the minimum value of a double which add to one.

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**Returns** :  $\lambda_{\text{cutoff}}$ .

**nc\_recomb\_tau\_zdrag ()**

```
gdouble          nc_recomb_tau_zdrag          (NcRecomb *recomb,
                                                NcHICosmo *cosmo);
```

FIXME

**recomb** : a **NcRecomb**.

**cosmo** : a **NcHICosmo**.

**Returns** : FIXME.

**nc\_recomb\_dtau\_dlambda\_Xe ()**

```
gdouble          nc_recomb_dtau_dlambda_Xe    (NcHICosmo *cosmo,
                                                const gdouble lambda);
```

The derivative of the optical depth [Eq. \eqref{eq:def:dtaudlambda}] over the ionization fraction  $X_e$  [Eq. \eqref{eq:def:Xe}].

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $X_e^{-1} d\tau/d\lambda$ .

**nc\_recomb\_He\_fully\_ionized\_dtau\_dlambda ()**

```
gdouble          nc_recomb_He_fully_ionized_dtau_dlambda
                                                (NcHICosmo *cosmo,
                                                const gdouble lambda);
```

The derivative of the optical depth [Eq. \eqref{eq:def:dtaudlambda}], considering fully ionized helium and hydrogen [**nc\_recomb\_He\_fully\_ionized**].

**cosmo** : a **NcHICosmo**.

**lambda** :  $\lambda$ .

**Returns** :  $d\tau/d\lambda$ .



**nc\_recomb\_pequignot\_HI\_case\_B ()**

```
gdouble          nc_recomb_pequignot_HI_case_B      (NCHICosmo *cosmo,
                                                    const gdouble Tm);
```

The case B  $\text{H\text{II}}$  recombination coefficient.

The fitting formula of the case B recombination coefficient for  $\text{H\text{II}}$  as in [Pequignot \(1991\)](#).

**cosmo** : a [NCHICosmo](#).

**Tm** : the matter (baryons) temperature  $T_{\text{m}}$

**Returns** : the value of the case B recombination coefficient for  $\text{H\text{II}}$ ,  $\alpha_{\text{H}}$ .

**nc\_recomb\_pequignot\_HI\_case\_B\_dTm ()**

```
gdouble          nc_recomb_pequignot_HI_case_B_dTm  (NCHICosmo *cosmo,
                                                    const gdouble Tm);
```

The case B  $\text{H\text{II}}$  recombination coefficient derivative with respect to  $T_{\text{m}}$ .

The derivative of the fitting formula of the case B recombination coefficient for  $\text{H\text{II}}$  [nc\\_recomb\\_pequignot\\_HI\\_case\\_B\(\)](#).

**cosmo** : a [NCHICosmo](#).

**Tm** : the matter (baryons) temperature  $T_{\text{m}}$

**Returns** : the value of the case B recombination coefficient for  $\text{H\text{II}}$ ,  $d\alpha_{\text{H}}/dT_{\text{m}}$ .

**nc\_recomb\_hummer\_HeI\_case\_B ()**

```
gdouble          nc_recomb_hummer_HeI_case_B      (NCHICosmo *cosmo,
                                                    const gdouble Tm);
```

The case B  $\text{HeII}$  recombination coefficient.

The fitting formula of the case B recombination coefficient for  $\text{HeII}$  as in [Hummer \(1998\)](#).

**cosmo** : a [NCHICosmo](#).

**Tm** : the matter (baryons) temperature  $T_{\text{m}}$

**Returns** : the value of the case B recombination coefficient for  $\text{HeII}$ ,  $\alpha_{\text{H}}$ .

**nc\_recomb\_hummer\_HeI\_case\_B\_dTm ()**

```
gdouble          nc_recomb_hummer_HeI_case_B_dTm  (NCHICosmo *cosmo,
                                                    const gdouble Tm);
```

The case B  $\text{HeII}$  recombination coefficient derivative with respect to  $T_{\text{m}}$ .

The derivative of the fitting formula of the case B recombination coefficient for  $\text{HeII}$  [nc\\_recomb\\_hummer\\_HeI\\_case\\_B\(\)](#).

**cosmo** : a [NCHICosmo](#).

**Tm** : the matter (baryons) temperature  $T_{\text{m}}$

**Returns** : the value of the case B recombination coefficient for  $\text{HeII}$ ,  $d\alpha_{\text{H}}/dT_{\text{m}}$ .

**nc\_recomb\_weinberg\_HII\_ion\_rate ()**

```
gdouble          nc_recomb_weinberg_HII_ion_rate      (NcHICosmo *cosmo,
                                                       gdouble XHII,
                                                       gdouble Tm,
                                                       gdouble XHeII,
                                                       gdouble x);
```

\$dX\_{\rm e}/dx\$ implemented using Weinbergs book

*cosmo* : a **NcHICosmo**.

*XHII* : FIXME

*Tm* : FIXME

*XHeII* : FIXME

*x* : normalized scale factor inverse  $x = 1 + z = a_0/a$

*Returns* : FIXME

**Property Details**

**The "init-frac" property**

"init-frac"	gdouble	: Read / Write / Construct
-------------	---------	----------------------------

Initial fraction to start numerical integration.

Allowed values: [0,1]

Default value: 1e-11

**The "prec" property**

"prec"	gdouble	: Read / Write / Construct
--------	---------	----------------------------

The precision used in the calculations.

Allowed values: [0,1]

Default value: 1e-07

**The "zi" property**

"zi"	gdouble	: Read / Write / Construct
------	---------	----------------------------

Initial redshift to prepare the recombination functions.

Allowed values:  $\geq 0$

Default value: 1e+12

**5.3 Recombination Seager 1999.**

Recombination Seager 1999. — Cosmic recombination implementing Seager (1999)

## Synopsis

```
#include <numcosmo/nc_recomb_seager.h>

struct          NcRecombSeagerClass;
struct          NcRecombSeager;
NcRecomb *      nc_recomb_seager_new          (void);
NcRecomb *      nc_recomb_seager_new_full     (gdouble init_frac,
                                                gdouble zi,
                                                gdouble prec);
```

## Object Hierarchy

```
GObject
+----NcRecomb
      +----NcRecombSeager
```

## Description

Cosmic recombination as describe in [Seager \(1999\)](#). It uses `nc_recomb_HeII_ion_saha_x_by_HeIII_He()` to obtain the value of  $\lambda$  where the numerical integration will start.

## Details

### struct NcRecombSeagerClass

```
struct NcRecombSeagerClass {
};
```

### struct NcRecombSeager

```
struct NcRecombSeager;
```

### nc\_recomb\_seager\_new ()

```
NcRecomb *      nc_recomb_seager_new          (void);
```

### nc\_recomb\_seager\_new\_full ()

```
NcRecomb *      nc_recomb_seager_new_full     (gdouble init_frac,
                                                gdouble zi,
                                                gdouble prec);
```

## 5.4 Supernovae Distance Covariance

Supernovae Distance Covariance — Calculates the covariance between distance estimates

## Synopsis

```

enum                NcSNIADistCovParams;
struct              NcSNIADistCovClass;
struct              NcSNIADistCov;
NcSNIADistCov *     nc_snia_dist_cov_new                (NcDistance *dist);
NcSNIADistCov *     nc_snia_dist_cov_ref                (NcSNIADistCov *dcov);
void                nc_snia_dist_cov_free                (NcSNIADistCov *dcov);
void                nc_snia_dist_cov_clear                (NcSNIADistCov **dcov);
void                nc_snia_dist_cov_prepare                (NcSNIADistCov *dcov,
                                                            NcmMSet *mset);
void                nc_snia_dist_cov_prepare_if_needed    (NcSNIADistCov *dcov,
                                                            NcmMSet *mset);
void                nc_snia_dist_cov_calc                (NcSNIADistCov *dcov,
                                                            NcDataSNIACov *snia_cov,
                                                            NcmMatrix *cov);
void                nc_snia_dist_cov_mean                (NcSNIADistCov *dcov,
                                                            NcHICosmo *cosmo,
                                                            NcDataSNIACov *snia_cov,
                                                            NcmVector *y);

#define              NC_SNIA_DIST_COV_DEFAULT_ALPHA
#define              NC_SNIA_DIST_COV_DEFAULT_BETA
#define              NC_SNIA_DIST_COV_DEFAULT_M1
#define              NC_SNIA_DIST_COV_DEFAULT_M2
#define              NC_SNIA_DIST_COV_DEFAULT_PARAMS_ABSTOL
#define              NC_SNIA_DIST_COV_SIGMA_INT
#define              NC_SNIA_DIST_COV_SIGMA_INT_DEFAULT_LEN
#define              NC_SNIA_DIST_COV_DEFAULT_SIGMA_INT
#define              NC_SNIA_DIST_COV_VPARAM_LEN

```

## Object Hierarchy

```

GObject
+----NcmModel
      +----NcSNIADistCov

```

## Properties

"M1"	gdouble	: Read / Write
"M1-fit"	gboolean	: Read / Write
"M2"	gdouble	: Read / Write
"M2-fit"	gboolean	: Read / Write
"alpha"	gdouble	: Read / Write
"alpha-fit"	gboolean	: Read / Write
"beta"	gdouble	: Read / Write
"beta-fit"	gboolean	: Read / Write
"dist"	NcDistance*	: Read / Write / Construct
"sigma-int"	GVariant*	: Read / Write
"sigma-int-fit"	GVariant*	: Read / Write
"sigma-int-length"	guint	: Read / Write / Construct Only

## Description

This object implements the calculation necessary to make a statistical analysis using data from [Conley et al. \(2011\)](#) and [Sullivan et al. \(2011\)](#).

## Details

### enum NcSNIADistCovParams

```
typedef enum {
    NC_SNIA_DIST_COV_ALPHA = 0,
    NC_SNIA_DIST_COV_BETA,
    NC_SNIA_DIST_COV_M1,
} NcSNIADistCovParams;
```

FIXME

**NC\_SNIA\_DIST\_COV\_ALPHA** FIXME

**NC\_SNIA\_DIST\_COV\_BETA** FIXME

**NC\_SNIA\_DIST\_COV\_M1** FIXME

**NC\_SNIA\_DIST\_COV\_M2** FIXME

### struct NcSNIADistCovClass

```
struct NcSNIADistCovClass {
};
```

### struct NcSNIADistCov

```
struct NcSNIADistCov;
```

### nc\_snia\_dist\_cov\_new ()

```
NcSNIADistCov *      nc_snia_dist_cov_new      (NcDistance *dist);
```

FIXME

**dist** : FIXME

**Returns** : FIXME

### nc\_snia\_dist\_cov\_ref ()

```
NcSNIADistCov *      nc_snia_dist_cov_ref      (NcSNIADistCov *dcov);
```

FIXME

**dcov** : FIXME

**Returns** : FIXME. *[transfer full]*

### nc\_snia\_dist\_cov\_free ()

```
void      nc_snia_dist_cov_free      (NcSNIADistCov *dcov);
```

FIXME

**dcov** : FIXME

**nc\_snia\_dist\_cov\_clear ()**

```
void nc_snia_dist_cov_clear (NcSNIADistCov **dcov);
```

FIXME

*dcov* : FIXME

**nc\_snia\_dist\_cov\_prepare ()**

```
void nc_snia_dist_cov_prepare (NcSNIADistCov *dcov,
                               NcmMSet *mset);
```

**nc\_snia\_dist\_cov\_prepare\_if\_needed ()**

```
void nc_snia_dist_cov_prepare_if_needed (NcSNIADistCov *dcov,
                                         NcmMSet *mset);
```

**nc\_snia\_dist\_cov\_calc ()**

```
void nc_snia_dist_cov_calc (NcSNIADistCov *dcov,
                           NcDataSNIACov *snia_cov,
                           NcmMatrix *cov);
```

FIXME

*dcov* : FIXME

*snia\_cov* : FIXME

*cov* : FIXME

**nc\_snia\_dist\_cov\_mean ()**

```
void nc_snia_dist_cov_mean (NcSNIADistCov *dcov,
                           NcHICosmo *cosmo,
                           NcDataSNIACov *snia_cov,
                           NcmVector *y);
```

FIXME

*dcov* : FIXME

*cosmo* : FIXME

*snia\_cov* : FIXME

*y* : FIXME

**NC\_SNIA\_DIST\_COV\_DEFAULT\_ALPHA**

```
#define NC_SNIA_DIST_COV_DEFAULT_ALPHA (1.45)
```

**NC\_SNIA\_DIST\_COV\_DEFAULT\_BETA**

```
#define NC_SNIA_DIST_COV_DEFAULT_BETA (3.16)
```

**NC\_SNIA\_DIST\_COV\_DEFAULT\_M1**

```
#define NC_SNIA_DIST_COV_DEFAULT_M1 (-19.1686133146)
```

**NC\_SNIA\_DIST\_COV\_DEFAULT\_M2**

```
#define NC_SNIA_DIST_COV_DEFAULT_M2 (-19.1856133146)
```

**NC\_SNIA\_DIST\_COV\_DEFAULT\_PARAMS\_ABSTOL**

```
#define NC_SNIA_DIST_COV_DEFAULT_PARAMS_ABSTOL (0.0)
```

**NC\_SNIA\_DIST\_COV\_SIGMA\_INT**

```
#define NC_SNIA_DIST_COV_SIGMA_INT (0)
```

**NC\_SNIA\_DIST\_COV\_SIGMA\_INT\_DEFAULT\_LEN**

```
#define NC_SNIA_DIST_COV_SIGMA_INT_DEFAULT_LEN (4)
```

**NC\_SNIA\_DIST\_COV\_DEFAULT\_SIGMA\_INT**

```
#define NC_SNIA_DIST_COV_DEFAULT_SIGMA_INT (0.0989)
```

**NC\_SNIA\_DIST\_COV\_VPARAM\_LEN**

```
#define NC_SNIA_DIST_COV_VPARAM_LEN (1)
```

**Property Details**

**The "M1" property**

"M1"	gdouble	: Read / Write
------	---------	----------------

Absolute Magnitude 1.  
Allowed values: [-50,10]  
Default value: -19.1686

**The "M1-fit" property**

"M1-fit"	gboolean	: Read / Write
----------	----------	----------------

Absolute Magnitude 1:fit.

Default value: FALSE

**The "M2" property**

"M2"	gdouble	: Read / Write
------	---------	----------------

Absolute Magnitude 2.

Allowed values: [-50,10]

Default value: -19.1856

**The "M2-fit" property**

"M2-fit"	gboolean	: Read / Write
----------	----------	----------------

Absolute Magnitude 2:fit.

Default value: FALSE

**The "alpha" property**

"alpha"	gdouble	: Read / Write
---------	---------	----------------

alpha.

Allowed values: [-10,10]

Default value: 1.45

**The "alpha-fit" property**

"alpha-fit"	gboolean	: Read / Write
-------------	----------	----------------

alpha:fit.

Default value: FALSE

**The "beta" property**

"beta"	gdouble	: Read / Write
--------	---------	----------------

beta.

Allowed values: [-10,10]

Default value: 3.16

---



**The "beta-fit" property**

"beta-fit"	gboolean	: Read / Write
------------	----------	----------------

beta:fit.

Default value: FALSE

**The "dist" property**

"dist"	NcDistance*	: Read / Write / Construct
--------	-------------	----------------------------

Distance object.

**The "sigma-int" property**

"sigma-int"	GVariant*	: Read / Write
-------------	-----------	----------------

Sigma intrinsic.

Allowed values: GVariant<ad>

Default value: NULL

**The "sigma-int-fit" property**

"sigma-int-fit"	GVariant*	: Read / Write
-----------------	-----------	----------------

Sigma intrinsic:fit.

Allowed values: GVariant<a\*>

Default value: NULL

**The "sigma-int-length" property**

"sigma-int-length"	guint	: Read / Write / Construct Only
--------------------	-------	---------------------------------

Sigma intrinsic:length.

Default value: 4

## 5.5 Scale Factor

Scale Factor — FIXME

## Synopsis

```
enum                NcScaleFactorTimeType;
struct              NcScaleFactor;
NcScaleFactor *     nc_scale_factor_new          (NcScaleFactorTimeType ttype,
                                                  gdouble zf);
NcScaleFactor *     nc_scale_factor_copy        (NcScaleFactor *a);
void                nc_scale_factor_free        (NcScaleFactor *a);
void                nc_scale_factor_prepare     (NcScaleFactor *a,
                                                  NCHICosmo *model);
void                nc_scale_factor_prepare_if_needed (NcScaleFactor *a,
                                                  NCHICosmo *model);
gdouble             nc_scale_factor_z_t        (NcScaleFactor *a,
                                                  gdouble t);
gdouble             nc_scale_factor_a_t        (NcScaleFactor *a,
                                                  gdouble t);
gdouble             nc_scale_factor_t_z        (NcScaleFactor *a,
                                                  gdouble z);
gdouble             nc_scale_factor_t_x        (NcScaleFactor *a,
                                                  gdouble x);
```

## Object Hierarchy

```
GBoxed
+----NcScaleFactor
```

## Description

FIXME

## Details

### enum NcScaleFactorTimeType

```
typedef enum {
    NC_TIME_COSMIC = 0,
    NC_TIME_CONFORMAL
} NcScaleFactorTimeType;
```

FIXME

**NC\_TIME\_COSMIC** Cosmic time

**NC\_TIME\_CONFORMAL** Conformal time

### struct NcScaleFactor

```
struct NcScaleFactor {
};
```

FIXME

**nc\_scale\_factor\_new ()**

```
NcScaleFactor *      nc_scale_factor_new      (NcScaleFactorTimeType ttype,  
                                              gdouble zf);
```

FIXME

**ttype** : a **NcScaleFactorTimeType**

**zf** : FIXME

**Returns** : FIXME

**nc\_scale\_factor\_copy ()**

```
NcScaleFactor *      nc_scale_factor_copy      (NcScaleFactor *a);
```

FIXME

**a** : FIXME

**Returns** : FIXME

**nc\_scale\_factor\_free ()**

```
void                 nc_scale_factor_free      (NcScaleFactor *a);
```

FIXME

**a** : FIXME

**nc\_scale\_factor\_prepare ()**

```
void                 nc_scale_factor_prepare    (NcScaleFactor *a,  
                                              NcHICosmo *model);
```

FIXME

**a** : FIXME

**model** : FIXME

**Returns** : FIXME

**nc\_scale\_factor\_prepare\_if\_needed ()**

```
void                 nc_scale_factor_prepare_if_needed    (NcScaleFactor *a,  
                                                          NcHICosmo *model);
```

FIXME

**a** : FIXME

**model** : FIXME

**Returns** : FIXME

---

**nc\_scale\_factor\_z\_t()**

gdouble	nc_scale_factor_z_t	(NcScaleFactor *a, gdouble t);
---------	---------------------	-----------------------------------

FIXME

**a** : FIXME**t** : FIXME**Returns** : FIXME**nc\_scale\_factor\_a\_t()**

gdouble	nc_scale_factor_a_t	(NcScaleFactor *a, gdouble t);
---------	---------------------	-----------------------------------

FIXME

**a** : FIXME**t** : FIXME**Returns** : FIXME**nc\_scale\_factor\_t\_z()**

gdouble	nc_scale_factor_t_z	(NcScaleFactor *a, gdouble z);
---------	---------------------	-----------------------------------

FIXME

**a** : FIXME**z** : FIXME**Returns** : FIXME**nc\_scale\_factor\_t\_x()**

gdouble	nc_scale_factor_t_x	(NcScaleFactor *a, gdouble x);
---------	---------------------	-----------------------------------

FIXME

**a** : FIXME**x** : FIXME**Returns** : FIXME

## Chapter 6

# Perturbations

### 6.1 Linear Perturbations

Linear Perturbations — FIXME

#### Synopsis

```
enum                NcLinearPertVars;
#define             NC_PERTURBATION_BASE_SIZE
#define             NC_PERT_dB0
#define             NC_PERT_V
#define             NC_PERT_T
#define             NC_PERT_dTHETA0
#define             NC_PERT_U
#define             NC_PERT_THETA                (n)
#define             NC_PERT_THETA_P              (n)
struct             NcLinearPert;
#define             NC_PERTURBATIONS_LAMBDA2X    (lambda)
#define             NC_PERTURBATIONS_X2LAMBDA    (x)
gpointer           (*NcLinearPertCreate)        (NcLinearPert *pert);
void               (*NcLinearPertConf)          (NcLinearPert *pert);
gboolean           (*NcLinearPertEvol)          (NcLinearPert *pert,
                                                gdouble g);
gboolean           (*NcLinearPertTest)          (NcLinearPert *pert);
void               (*NcLinearPertSources)       (NcLinearPert *pert,
                                                gdouble *S0,
                                                gdouble *S1,
                                                gdouble *S2);
gdouble            (*NcLinearPertGet)           (NcLinearPert *pert);
gdouble            (*NcLinearPertGetN)         (NcLinearPert *pert,
                                                quint n);

struct             NcLinearPertOdeSolver;
enum               NcLinearPertSplineTypes;
#define             NC_LINEAR_PERTURBATIONS_SPLINE_ALL
struct             NcLinearPertSplines;
#define             NC_LINEAR_PERTURBATIONS_GET_SPLINE (pspline,
                                                         n)
#define             NC_LINEAR_PERTURBATIONS        (p)
#define             NC_PERTURBATION_START_X
struct             NcLinearPertTF;
```

```

NcLinearPert *      nc_pert_linear_new          (NcHICosmo *cosmo,
                                                NcRecomb *recomb,
                                                guint lmax,
                                                gdouble tc_reltol,
                                                gdouble reltol,
                                                gdouble tc_abstol,
                                                gdouble abstol);

NcLinearPertSplines * nc_pert_linear_splines_new (NcLinearPert *pert,
                                                NcLinearPertSplineTypes types,
                                                gulong n_deta,
                                                gulong n_evol,
                                                gdouble k0,
                                                gdouble k1);

void      nc_pert_linear_prepare_splines (NcLinearPertSplines *pspline);
void      nc_pert_linear_free            (NcLinearPert *pert);
void      nc_pert_linear_clear           (NcLinearPert **pert);
void      nc_pert_linear_splines_free    (NcLinearPertSplines *pspline);
void      nc_pert_linear_splines_clear   (NcLinearPertSplines **pspline);
gboolean  nc_pert_linear_spline_set_source_at (NcLinearPertSplines *pspline,
                                                gdouble k);

gboolean  nc_pert_linear_calc_Nc_spline (NcLinearPertSplines *pspline,
                                         NcmSpline *pw_spline,
                                         GArray *los_table,
                                         gulong n_interp);

gdouble   nc_pert_linear_los_integrate (NcLinearPertSplines *pspline,
                                         glong l,
                                         gdouble k);

GArray *   nc_pert_linear_create_los_table (gint lmax_los,
                                             gint *los_ini,
                                             gint *los_step);

#define      nc_pert_get_default_los_table (lmax)
extern NcLinearPertOdeSolver *cvodes_solver;
extern NcLinearPertOdeSolver *ncm_gsl_odeiv2_solver;
NcLinearPertTF * nc_pert_transfer_function_new (NcLinearPert *pert,
                                                gdouble k0,
                                                gdouble k1,
                                                gulong np);

void      nc_pert_transfer_function_prepare (NcLinearPertTF *perttf);
gdouble   nc_pert_transfer_function_get    (NcLinearPertTF *perttf,
                                             gdouble kh);

```

## Description

FIXME

## Details

### enum NcLinearPertVars

```

typedef enum {
    NC_PERT_B0 = 0,
    NC_PERT_THETA0,
    NC_PERT_C0,
    NC_PERT_PHI,
    NC_PERT_B1,
    NC_PERT_THETA1,

```

```
NC_PERT_C1,  
NC_PERT_THETA2,  
NC_PERT_THETA_P0,  
NC_PERT_THETA_P1,  
NC_PERT_THETA_P2  
} NcLinearPertVars;
```

FIXME

**NC\_PERT\_B0** FIXME

**NC\_PERT\_THETA0** FIXME

**NC\_PERT\_C0** FIXME

**NC\_PERT\_PHI** FIXME

**NC\_PERT\_B1** FIXME

**NC\_PERT\_THETA1** FIXME

**NC\_PERT\_C1** FIXME

**NC\_PERT\_THETA2** FIXME

**NC\_PERT\_THETA\_P0** FIXME

**NC\_PERT\_THETA\_P1** FIXME

**NC\_PERT\_THETA\_P2** FIXME

**NC\_PERTURBATION\_BASE\_SIZE**

```
#define NC_PERTURBATION_BASE_SIZE (NC_PERT_THETA2 + 1)
```

**NC\_PERT\_dB0**

```
#define NC_PERT_dB0      NC_PERT_B0
```

**NC\_PERT\_V**

```
#define NC_PERT_V      NC_PERT_C1
```

**NC\_PERT\_T**

```
#define NC_PERT_T      NC_PERT_B1
```

**NC\_PERT\_dTHETA0**

```
#define NC_PERT_dTHETA0 NC_PERT_THETA0
```

**NC\_PERT\_U**

```
#define NC_PERT_U          NC_PERT_THETA1
```

**NC\_PERT\_THETA()**

```
#define NC_PERT_THETA(n)   ((n <= 2) ? (_itheta_table[n])   : (NC_PERT_THETA_P2 + 1) + (2*( ←  
n-3)))
```

**NC\_PERT\_THETA\_P()**

```
#define NC_PERT_THETA_P(n) ((n <= 2) ? (_itheta_p_table[n]) : (NC_PERT_THETA_P2 + 1) + (2*( ←  
n-3)+1))
```

**struct NcLinearPert**

```
struct NcLinearPert {  
};
```

FIXME

**NC\_PERTURBATIONS\_LAMBDA2X()**

```
#define NC_PERTURBATIONS_LAMBDA2X(lambda) (exp (-(lambda)))
```

**NC\_PERTURBATIONS\_X2LAMBDA()**

```
#define NC_PERTURBATIONS_X2LAMBDA(x) (-log (x))
```

**NcLinearPertCreate ()**

```
gpointer          (*NcLinearPertCreate)          (NcLinearPert *pert);
```

**NcLinearPertConf ()**

```
void              (*NcLinearPertConf)            (NcLinearPert *pert);
```

**NcLinearPertEvol ()**

```
gboolean          (*NcLinearPertEvol)            (NcLinearPert *pert,  
gdouble g);
```

**NcLinearPertTest ()**

```
gboolean          (*NcLinearPertTest)            (NcLinearPert *pert);
```



**NcLinearPertSources ()**

```
void (*NcLinearPertSources) (NcLinearPert *pert,
                             gdouble *S0,
                             gdouble *S1,
                             gdouble *S2);
```

**NcLinearPertGet ()**

```
gdouble (*NcLinearPertGet) (NcLinearPert *pert);
```

**NcLinearPertGetN ()**

```
gdouble (*NcLinearPertGetN) (NcLinearPert *pert,
                              guint n);
```

**struct NcLinearPertOdeSolver**

```
struct NcLinearPertOdeSolver {
};
```

FIXME

**enum NcLinearPertSplineTypes**

```
typedef enum {
    NC_LINEAR_PERTURBATIONS_SPLINE_SOURCES = 1 << NC_PERTURBATION_BASE_SIZE,
    NC_LINEAR_PERTURBATIONS_SPLINE_PHI     = 1 << NC_PERT_PHI,
    NC_LINEAR_PERTURBATIONS_SPLINE_THETA0  = 1 << NC_PERT_THETA0,
    NC_LINEAR_PERTURBATIONS_SPLINE_C0     = 1 << NC_PERT_C0,
    NC_LINEAR_PERTURBATIONS_SPLINE_B0     = 1 << NC_PERT_B0,
    NC_LINEAR_PERTURBATIONS_SPLINE_THETA1 = 1 << NC_PERT_THETA1,
    NC_LINEAR_PERTURBATIONS_SPLINE_C1     = 1 << NC_PERT_C1,
    NC_LINEAR_PERTURBATIONS_SPLINE_B1     = 1 << NC_PERT_B1,
    NC_LINEAR_PERTURBATIONS_SPLINE_THETA2 = 1 << NC_PERT_THETA2,
} NcLinearPertSplineTypes;
```

FIXME

**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_SOURCES** FIXME**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_PHI** FIXME**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_THETA0** FIXME**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_C0** FIXME**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_B0** FIXME**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_THETA1** FIXME**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_C1** FIXME**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_B1** FIXME**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_THETA2** FIXME

**NC\_LINEAR\_PERTURBATIONS\_SPLINE\_ALL**

```
#define NC_LINEAR_PERTURBATIONS_SPLINE_ALL (~0)
```

**struct NcLinearPertSplines**

```
struct NcLinearPertSplines {
};
```

**NC\_LINEAR\_PERTURBATIONS\_GET\_SPLINE()**

```
#define NC_LINEAR_PERTURBATIONS_GET_SPLINE(pspline,n) ((pspline)->s[n])
```

**NC\_LINEAR\_PERTURBATIONS()**

```
#define NC_LINEAR_PERTURBATIONS(p) ((NcLinearPert *) (p))
```

**NC\_PERTURBATION\_START\_X**

```
#define NC_PERTURBATION_START_X (1.0e12)
```

**struct NcLinearPertTF**

```
struct NcLinearPertTF {
};
```

FIXME

**nc\_pert\_linear\_new ()**

```
NcLinearPert *      nc_pert_linear_new      (NcHICosmo *cosmo,
                                             NcRecomb *recomb,
                                             guint lmax,
                                             gdouble tc_reltol,
                                             gdouble reltol,
                                             gdouble tc_abstol,
                                             gdouble abstol);
```

FIXME

**cosmo** : a **NcHICosmo**.

**recomb** : FIXME

**lmax** : FIXME

**tc\_reltol** : FIXME

**reltol** : FIXME

**tc\_abstol** : FIXME

**abstol** : FIXME

**Returns** : FIXME

**nc\_pert\_linear\_splines\_new ()**

```
NcLinearPertSplines * nc_pert_linear_splines_new (NcLinearPert *pert,
                                                    NcLinearPertSplineTypes types,
                                                    gulong n_deta,
                                                    gulong n_evol,
                                                    gdouble k0,
                                                    gdouble k1);
```

FIXME

***pert*** : a **NcLinearPert**

***types*** : a **NcLinearPertSplineTypes**

***n\_deta*** : FIXME

***n\_evol*** : FIXME

***k0*** : FIXME

***k1*** : FIXME

***Returns*** : FIXME

**nc\_pert\_linear\_prepare\_splines ()**

```
void nc_pert_linear_prepare_splines (NcLinearPertSplines *pspline);
```

FIXME

***pspline*** : a **NcLinearPertSplines**

**nc\_pert\_linear\_free ()**

```
void nc_pert_linear_free (NcLinearPert *pert);
```

FIXME

***pert*** : a **NcLinearPert**.

**nc\_pert\_linear\_clear ()**

```
void nc_pert_linear_clear (NcLinearPert **pert);
```

FIXME

***pert*** : a **NcLinearPert**.

**nc\_pert\_linear\_splines\_free ()**

```
void nc_pert_linear_splines_free (NcLinearPertSplines *pspline);
```

FIXME

***pspline*** : a **NcLinearPertSplines**.

**nc\_pert\_linear\_splines\_clear ()**

```
void nc_pert_linear_splines_clear (NcLinearPertSplines **pspline);
```

FIXME

**pspline**: a **NcLinearPertSplines**.

**nc\_pert\_linear\_spline\_set\_source\_at ()**

```
gboolean nc_pert_linear_spline_set_source_at (NcLinearPertSplines *pspline,
                                              gdouble k);
```

FIXME

**pspline**: a **NcLinearPertSplines**

**k**: FIXME

**Returns**: FIXME

**nc\_pert\_linear\_calc\_Nc\_spline ()**

```
gboolean nc_pert_linear_calc_Nc_spline (NcLinearPertSplines *pspline,
                                         NcmSpline *pw_spline,
                                         GArray *los_table,
                                         gulong n_interp);
```

FIXME

**pspline**: a **NcLinearPertSplines**

**pw\_spline**: a **NcmSpline**

**los\_table**: FIXME

**n\_interp**: FIXME

**Returns**: FIXME

**nc\_pert\_linear\_los\_integrate ()**

```
gdouble nc_pert_linear_los_integrate (NcLinearPertSplines *pspline,
                                       glong l,
                                       gdouble k);
```

FIXME

**pspline**: a **NcLinearPertSplines**

**l**: FIXME

**k**: FIXME

**Returns**: FIXME

**nc\_pert\_linear\_create\_los\_table ()**

```
GArray *          nc_pert_linear_create_los_table      (gint lmax_los,
                                                         gint *los_ini,
                                                         gint *los_step);
```

FIXME

***lmax\_los*** : FIXME

***los\_ini*** : FIXME

***los\_step*** : FIXME

***Returns*** : FIXME

**nc\_pert\_get\_default\_los\_table()**

```
#define nc_pert_get_default_los_table(lmax) (nc_pert_linear_create_los_table (lmax, ↵
    _nc_default_los_init, _nc_default_los_step))
```

**cvodes\_solver**

```
extern NcLinearPertOdeSolver *cvodes_solver;
```

**ncm\_gsl\_odeiv2\_solver**

```
extern NcLinearPertOdeSolver *ncm_gsl_odeiv2_solver;
```

**nc\_pert\_transfer\_function\_new ()**

```
NcLinearPertTF *    nc_pert_transfer_function_new      (NcLinearPert *pert,
                                                         gdouble k0,
                                                         gdouble k1,
                                                         gulong np);
```

FIXME

***pert*** : a **NcLinearPert**

***k0*** : FIXME

***k1*** : FIXME

***np*** : FIXME

***Returns*** : FIXME

**nc\_pert\_transfer\_function\_prepare ()**

```
void                nc_pert_transfer_function_prepare  (NcLinearPertTF *perttf);
```

FIXME

***perttf*** : a **NcLinearPertTF**

**nc\_pert\_transfer\_function\_get ()**

```
gdouble          nc_pert_transfer_function_get      (NcLinearPertTF *perttf,
                                                    gdouble kh);
```

FIXME

*perttf*: a **NcLinearPertTF**

*kh*: FIXME

*Returns*: FIXME

## 6.2 Perturbation Covariance

Perturbation Covariance — FIXME

### Synopsis

```
void          nc_pert_cov_direct                  (NcLinearPert *pert);
```

### Description

FIXME

### Details

**nc\_pert\_cov\_direct ()**

```
void          nc_pert_cov_direct                  (NcLinearPert *pert);
```

## 6.3 Hydrodynamic Perturbations

Hydrodynamic Perturbations — FIXME

### Synopsis

```
void          nc_hydrodyn_adiabatic_stub          (void);
```

### Description

FIXME

### Details

**nc\_hydrodyn\_adiabatic\_stub ()**

```
void          nc_hydrodyn_adiabatic_stub          (void);
```

## Chapter 7

# Large Scale Structure

### 7.1 Window Function

#### 7.1.1 Window Function Abstract Class

Window Function Abstract Class — Defines the prototype of the **NcWindow** object.

##### Synopsis

```

struct          NcWindowClass;
struct          NcWindow;
NcWindow *      nc_window_new_from_name      (gchar *window_name);
gdouble         nc_window_volume            (NcWindow *wf);
gdouble         nc_window_eval_fourier      (const NcWindow *wf,
                                             const gdouble k,
                                             const gdouble R);
gdouble         nc_window_deriv_fourier     (const NcWindow *wf,
                                             const gdouble k,
                                             const gdouble R);
gdouble         nc_window_eval_realspace    (const NcWindow *wf,
                                             const gdouble r,
                                             const gdouble R);
void            nc_window_free              (NcWindow *wf);
void            nc_window_clear             (NcWindow **wf);

```

##### Object Hierarchy

```

GObject
+----NcWindow
      +----NcWindowGaussian
      +----NcWindowTophat

```

##### Description

This module comprises the set of functions to compute the window function in both real and Fourier spaces as well as its derivative with respect to the scale  $R$  in Fourier space.

In order to study the statistical properties of the density fluctuation field at a certain scale  $R$ , we use the window function. As an example, to compute the variance of the density contrast at scale  $R$ , we convolve the window function in the Fourier space with the power spectrum.

**Details****struct NcWindowClass**

```
struct NcWindowClass {
};
```

**struct NcWindow**

```
struct NcWindow;
```

**nc\_window\_new\_from\_name ()**

```
NcWindow *          nc_window_new_from_name          (gchar *window_name);
```

This function returns a new **NcWindow** whose type is defined by *window\_name* string.

**window\_name** : "NcWindowTophat" or "NcWindowGaussian".

**Returns** : A new **NcWindow**.

**nc\_window\_volume ()**

```
gdouble          nc_window_volume          (NcWindow *wf);
```

This function returns the volume of the region (with radius 1) defined by the window function.

Top-hat volume: **NC\_WINDOW\_VOLUME\_TOPHAT**.

Gaussian volume: **NC\_WINDOW\_VOLUME\_GAUSSIAN**.

**wf** : a **NcWindow**.

**Returns** : The volume (with radius 1) defined by *wf*.

**nc\_window\_eval\_fourier ()**

```
gdouble          nc_window_eval_fourier          (const NcWindow *wf,
                                                  const gdouble k,
                                                  const gdouble R);
```

This function computes the window function in the Fourier space.

**wf** : a **NcWindow**.

**k** : mode.

**R** : scale.

**Returns** : The value of the window function in the Fourier space at scale *R*.



**nc\_window\_deriv\_fourier ()**

```
gdouble          nc_window_deriv_fourier          (const NcWindow *wf,
                                                    const gdouble k,
                                                    const gdouble R);
```

This function returns the derivative with respect to  $R$  of the window function in the Fourier space.

**wf** : a **NcWindow**.

**k** : mode.

**R** : scale.

**Returns** : The value of the first derivative of the window function in the Fourier space at scale  $R$ .

**nc\_window\_eval\_realspace ()**

```
gdouble          nc_window_eval_realspace         (const NcWindow *wf,
                                                    const gdouble r,
                                                    const gdouble R);
```

This function computes the window function in real space.

**wf** : a **NcWindow**.

**r** : distance module to the center point of the filtered region.

**R** : scale.

**Returns** : The value of the window function in the real space at scale  $R$ .

**nc\_window\_free ()**

```
void             nc_window_free                   (NcWindow *wf);
```

Atomically decrements the reference count of **wf** by one. If the reference count drops to 0, all memory allocated by **wf** is released.

**wf** : a **NcWindow**.

**nc\_window\_clear ()**

```
void             nc_window_clear                  (NcWindow **wf);
```

Atomically decrements the reference count of **wf** by one. If the reference count drops to 0, all memory allocated by **wf** is released. Set the pointer to NULL;

**wf** : a **NcWindow**.

**7.1.2 Top-hat Window Function**

Top-hat Window Function — Provides a **NcWindow** of top-hat type filter.

## Synopsis

```
struct          NcWindowTophatClass;
struct          NcWindowTophat;
NcWindow *      nc_window_tophat_new          ();
#define          NC_WINDOW_VOLUME_TOPHAT
```

## Object Hierarchy

```
GObject
+-----NcWindow
+-----NcWindowTophat
```

## Description

This object implements the **NcWindow** class for a top-hat window function.

This object returns the top hat window function in the real space. 
$$W_{\text{TH}}(r, R) = \frac{3}{4\pi R^3} \left[ \begin{array}{l} 1 \text{ if } r \leq R \\ 0 \text{ if } r > R \end{array} \right]$$
 The mass enclosed within the volume selected by this window function is  $M_{\text{TH}}(R) = \frac{4\pi}{3} \bar{\rho} R^3$ , where  $\bar{\rho}(z)$  is the mean density of the universe at redshift  $z$ .

The top-hat window function in the Fourier space is given by

$$W_{\text{th}}(k, R) = \frac{3}{(kR)^3} (\sin kR - (kR)\cos kR) \quad \&= \quad \frac{3}{(kR)} j_1(kR)$$
 where  $j_1(kR)$  is the spherical Bessel function.

The first derivative with respect to  $R$  
$$\frac{dW_{\text{TH}}(k, R)}{dR} = \frac{-9}{k^3 R^4} (\sin kR - (kR)\cos kR) + \frac{3}{k R^2} \sin kR$$

## Details

### struct NcWindowTophatClass

```
struct NcWindowTophatClass {
};
```

### struct NcWindowTophat

```
struct NcWindowTophat;
```

### nc\_window\_tophat\_new ()

```
NcWindow *      nc_window_tophat_new          ();
```

This function returns a **NcWindow** with a **NcWindowTophat** implementation.

**Returns :** A new **NcWindow**.

### NC\_WINDOW\_VOLUME\_TOPHAT

```
#define NC_WINDOW_VOLUME_TOPHAT (4.0 * M_PI / 3.0)
```

### 7.1.3 Gaussian window function

Gaussian window function — Provides a **NcWindow** of Gaussian type filter.

#### Synopsis

```
struct          NcWindowGaussianClass;
struct          NcWindowGaussian;
NcWindow *      nc_window_gaussian_new          ();
#define          NC_WINDOW_VOLUME_GAUSSIAN
```

#### Object Hierarchy

```
GObject
+----NcWindow
      +----NcWindowGaussian
```

#### Description

This object implements the **NcWindow** class for a Gaussian window function.

This function returns the gaussian window function in the real space, 
$$W_G(r, R) = (2 \pi R^2)^{-3/2} \exp \left( -\frac{r^2}{2 R^2} \right).$$
 The mass enclosed within the volume selected by this window function is  $M_G(R) = (2 \pi)^{3/2} \overline{\rho}(z) R^3$ , where  $\overline{\rho}(z)$  is the mean density of the universe at redshift  $z$ .

This function returns the gaussian window function in the Fourier space, 
$$W_G(k, R) = \exp \left( -\frac{k^2 R^2}{2} \right).$$

This function returns the derivative with respect to  $R$  of the gaussian window function in the real space, 
$$\frac{dW_G(k, R)}{dR} = -k^2 R \exp \left( -\frac{k^2 R^2}{2} \right).$$

#### Details

##### struct NcWindowGaussianClass

```
struct NcWindowGaussianClass {
};
```

##### struct NcWindowGaussian

```
struct NcWindowGaussian;
```

##### nc\_window\_gaussian\_new ()

```
NcWindow *      nc_window_gaussian_new          ();
```

This function returns a **NcWindow** with a **NcWindowGaussian** implementation.

**Returns :** A new **NcWindow**.

##### NC\_WINDOW\_VOLUME\_GAUSSIAN

```
#define NC_WINDOW_VOLUME_GAUSSIAN (sqrt(2.0 * M_PI)*sqrt(2.0 * M_PI)*sqrt(2.0 * M_PI)) /* ←
      (2.0 \Pi)^(3/2) */
```

## 7.2 Transfer Function

### 7.2.1 Transfer Function Abstract Class

Transfer Function Abstract Class — Defines the prototype of the `NcTransferFunc` object.

#### Synopsis

```

struct          NcTransferFuncClass;
struct          NcTransferFunc;
NcTransferFunc * nc_transfer_func_new_from_name    (gchar *transfer_name);
void            nc_transfer_func_prepare          (NcTransferFunc *tf,
                                                  NCHICosmo *model);
gdouble         nc_transfer_func_eval             (NcTransferFunc *tf,
                                                  NCHICosmo *model,
                                                  gdouble kh);
gdouble         nc_transfer_func_matter_powerspectrum
                                                  (NcTransferFunc *tf,
                                                  NCHICosmo *model,
                                                  gdouble kh);
void            nc_transfer_func_free             (NcTransferFunc *tf);
void            nc_transfer_func_clear            (NcTransferFunc **tf);

```

#### Object Hierarchy

```

GObject
+----NcTransferFunc
      +----NcTransferFuncBBKS
      +----NcTransferFuncCAMB
      +----NcTransferFuncEH
      +----NcTransferFuncPert

```

#### Description

This module comprises the set of functions to compute the transfer function and derived quantities.

#### Details

##### struct NcTransferFuncClass

```

struct NcTransferFuncClass {
};

```

##### struct NcTransferFunc

```

struct NcTransferFunc;

```

**nc\_transfer\_func\_new\_from\_name ()**

```
NcTransferFunc * nc_transfer_func_new_from_name (gchar *transfer_name);
```

This function returns a new **NcTransferFunc** whose type is defined by *transfer\_name*.

**transfer\_name** : string which specifies the transfer function type.

**Returns** : A new **NcTransferFunc**.

**nc\_transfer\_func\_prepare ()**

```
void nc_transfer_func_prepare (NcTransferFunc *tf,  
NcHICosmo *model);
```

FIXME

**tf** : a **NcTransferFunc**.

**model** : a **NcHICosmo**.

**nc\_transfer\_func\_eval ()**

```
gdouble nc_transfer_func_eval (NcTransferFunc *tf,  
NcHICosmo *model,  
gdouble kh);
```

FIXME

**tf** : a **NcTransferFunc**.

**model** : a **NcHICosmo**.

**kh** : FIXME

**Returns** : FIXME

**nc\_transfer\_func\_matter\_powerspectrum ()**

```
gdouble nc_transfer_func_matter_powerspectrum  
(NcTransferFunc *tf,  
NcHICosmo *model,  
gdouble kh);
```

FIXME

**tf** : a **NcTransferFunc**.

**model** : a **NcHICosmo**.

**kh** : FIXME

**Returns** : FIXME

**nc\_transfer\_func\_free ()**

```
void nc_transfer_func_free (NcTransferFunc *tf);
```

Atomically decrements the reference count of *tf* by one. If the reference count drops to 0, all memory allocated by *tf* is released.

**tf**: a **NcTransferFunc**.

**nc\_transfer\_func\_clear ()**

```
void nc_transfer_func_clear (NcTransferFunc **tf);
```

Atomically decrements the reference count of *tf* by one. If the reference count drops to 0, all memory allocated by *tf* is released. Set the pointer to NULL.

**tf**: a **NcTransferFunc**.

**7.2.2 BBKS Transfer Function**

BBKS Transfer Function — FIXME

**Synopsis**

```
struct NcTransferFuncBBKSClass;
struct NcTransferFuncBBKS;
NcTransferFunc * nc_transfer_func_bbks_new ();
```

**Object Hierarchy**

```
GObject
+----NcTransferFunc
      +----NcTransferFuncBBKS
```

**Description**

FIXME

**Details****struct NcTransferFuncBBKSClass**

```
struct NcTransferFuncBBKSClass {
};
```

**struct NcTransferFuncBBKS**

```
struct NcTransferFuncBBKS;
```

**nc\_transfer\_func\_bbks\_new ()**

```
NcTransferFunc * nc_transfer_func_bbks_new ();
```

FIXME

**Returns :** A new **NcTransferFunc**.

**7.2.3 EH Transfer Function**

EH Transfer Function — FIXME

**Synopsis**

```
struct NcTransferFuncEHClass;
struct NcTransferFuncEH;
NcTransferFunc * nc_transfer_func_eh_new ();
```

**Object Hierarchy**

```
GObject
+----NcTransferFunc
      +----NcTransferFuncEH
```

**Description**

FIXME

**Details****struct NcTransferFuncEHClass**

```
struct NcTransferFuncEHClass {
};
```

**struct NcTransferFuncEH**

```
struct NcTransferFuncEH;
```

**nc\_transfer\_func\_eh\_new ()**

```
NcTransferFunc * nc_transfer_func_eh_new ();
```

FIXME

**Returns :** A new **NcTransferFunc**.

**7.2.4 CAMB Transfer Function**

CAMB Transfer Function — FIXME

**Synopsis**

```

struct          NcTransferFuncCAMBClass;
struct          NcTransferFuncCAMB;
NcTransferFunc * nc_transfer_func_camb_new          ();
extern gchar *   camb_filename;

```

**Object Hierarchy**

```

GObject
+----NcTransferFunc
      +----NcTransferFuncCAMB

```

**Description**

FIXME

**Details****struct NcTransferFuncCAMBClass**

```

struct NcTransferFuncCAMBClass {
};

```

**struct NcTransferFuncCAMB**

```

struct NcTransferFuncCAMB;

```

**nc\_transfer\_func\_camb\_new ()**

```

NcTransferFunc *   nc_transfer_func_camb_new          ();

```

FIXME

**Returns :** A new **NcTransferFunc**.

**camb\_filename**

```

extern gchar *camb_filename;

```

**7.2.5 Pert Transfer Function**

Pert Transfer Function — FIXME

**Synopsis**

```

struct          NcTransferFuncPertClass;
struct          NcTransferFuncPert;

```



**Object Hierarchy**

```

GObject
+----NcTransferFunc
      +----NcTransferFuncPert

```

**Description**

FIXME

**Details****struct NcTransferFuncPertClass**

```

struct NcTransferFuncPertClass {
};

```

**struct NcTransferFuncPert**

```

struct NcTransferFuncPert;

```

**7.3 Perturbations Growth Function**

Perturbations Growth Function — FIXME

**Synopsis**

```

struct          NcGrowthFuncClass;
struct          NcGrowthFunc;
NcGrowthFunc *  nc_growth_func_new          (void);
NcGrowthFunc *  nc_growth_func_copy        (NcGrowthFunc *gf);
void            nc_growth_func_free        (NcGrowthFunc *gf);
void            nc_growth_func_clear       (NcGrowthFunc **gf);
void            nc_growth_func_prepare     (NcGrowthFunc *gf,
                                           NcHICosmo *model);
gdouble         nc_growth_func_eval        (NcGrowthFunc *gf,
                                           NcHICosmo *model,
                                           gdouble z);
gdouble         nc_growth_func_eval_deriv (NcGrowthFunc *gf,
                                           NcHICosmo *model,
                                           gdouble z);
void            nc_growth_func_eval_both   (NcGrowthFunc *gf,
                                           NcHICosmo *model,
                                           gdouble z,
                                           gdouble *d,
                                           gdouble *f);

```

**Object Hierarchy**

```

GObject
+----NcGrowthFunc

```

## Description

FIXME

## Details

### struct NcGrowthFuncClass

```
struct NcGrowthFuncClass {  
};
```

### struct NcGrowthFunc

```
struct NcGrowthFunc;
```

### nc\_growth\_func\_new ()

```
NcGrowthFunc *      nc_growth_func_new      (void);
```

This function allocates memory for a new **NcGrowthFunc** object.

**Returns :** A new **NcGrowthFunc**.

### nc\_growth\_func\_copy ()

```
NcGrowthFunc *      nc_growth_func_copy      (NcGrowthFunc *gf);
```

This function duplicates *gf*.

**gf :** a **NcGrowthFunc**.

**Returns :** A **NcGrowthFunc**. *[transfer full]*

### nc\_growth\_func\_free ()

```
void                nc_growth_func_free      (NcGrowthFunc *gf);
```

Atomically decrements the reference count of *gf* by one. If the reference count drops to 0, all memory allocated by *gf* is released.

**gf :** a **NcGrowthFunc**.

### nc\_growth\_func\_clear ()

```
void                nc_growth_func_clear      (NcGrowthFunc **gf);
```

Atomically decrements the reference count of *gf* by one. If the reference count drops to 0, all memory allocated by *gf* is released. Set pointer to NULL.

**gf :** a **NcGrowthFunc**.

---

**nc\_growth\_func\_prepare ()**

```
void                nc_growth_func_prepare      (NcGrowthFunc *gf,
                                                NcHICosmo *model);
```

FIXME

**gf** : a **NcGrowthFunc**.

**model** : a **NcHICosmo**.

**nc\_growth\_func\_eval ()**

```
gdouble            nc_growth_func_eval        (NcGrowthFunc *gf,
                                                NcHICosmo *model,
                                                gdouble z);
```

FIXME

**gf** : a **NcGrowthFunc**.

**model** : a **NcHICosmo**.

**z** : redshift.

**Returns** : The normalized growth function at *z*.

**nc\_growth\_func\_eval\_deriv ()**

```
gdouble            nc_growth_func_eval_deriv  (NcGrowthFunc *gf,
                                                NcHICosmo *model,
                                                gdouble z);
```

FIXME

**gf** : a **NcGrowthFunc**.

**model** : a **NcHICosmo**.

**z** : redshift.

**Returns** : FIXME

**nc\_growth\_func\_eval\_both ()**

```
void                nc_growth_func_eval_both  (NcGrowthFunc *gf,
                                                NcHICosmo *model,
                                                gdouble z,
                                                gdouble *d,
                                                gdouble *f);
```

FIXME

**gf** : a **NcGrowthFunc**.

**model** : a **NcHICosmo**.

**z** : redshift.

**d** : Growth function.

**f** : Growth function derivative.

**Returns** : FIXME

## 7.4 Matter Fluctuation Variance

Matter Fluctuation Variance — FIXME

### Synopsis

```
enum                NcMatterVarStrategy;
struct              NcMatterVarClass;
struct              NcMatterVar;
NcMatterVar *       nc_matter_var_new                (NcMatterVarStrategy vs,
                                                    NcWindow *wp,
                                                    NcTransferFunc *tf);

NcMatterVar *       nc_matter_var_copy              (NcMatterVar *vp);
void                nc_matter_var_free              (NcMatterVar *vp);
void                nc_matter_var_clear             (NcMatterVar **vp);
void                nc_matter_var_prepare           (NcMatterVar *vp,
                                                    NcHICosmo *model);

gdouble             nc_matter_var_var0              (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gdouble lnR);

gdouble             nc_matter_var_dlnvar0_dR         (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gdouble lnR);

gdouble             nc_matter_var_dlnvar0_dlnR       (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gdouble lnR);

gdouble             nc_matter_var_mass_to_R          (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gdouble M);

gdouble             nc_matter_var_R_to_mass          (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gdouble R);

gdouble             nc_matter_var_lnM_to_lnR         (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gdouble lnM);

gdouble             nc_matter_var_lnR_to_lnM         (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gdouble lnR);

gdouble             nc_matter_var_integrand_over_window2
                                                    (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gdouble k);

gdouble             nc_matter_var_spectral_moment_over_growth2
                                                    (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gint n);

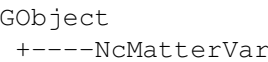
gdouble             nc_matter_var_spectral_moment_over_growth2_tophat
                                                    (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gint n);

gdouble             nc_matter_var_spectral_moment_over_growth2_gaussian
                                                    (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gint n);

gdouble             nc_matter_var_dsigma0_dR        (NcMatterVar *vp,
```

```
gdouble          nc_matter_var_sigma8_sqrtvar0
NcHICosmo *model,
gdouble lnR);
(NcMatterVar *vp,
NcHICosmo *model);
```

Object Hierarchy



Properties

"strategy"	NcMatterVarStrategy	: Read / Write / Construct Only
"transfer"	NcTransferFunc*	: Read / Write / Construct Only
"window"	NcWindow*	: Read / Write / Construct Only

Description

FIXME

Details

enum NcMatterVarStrategy

```
typedef enum {
    NC_MATTER_VAR_NUMINT,
    NC_MATTER_VAR_SPLINEINT,
    NC_MATTER_VAR_FFT,
} NcMatterVarStrategy;
```

FIXME

**NC\_MATTER\_VAR\_NUMINT** Compute variance with numerical integration.

**NC\_MATTER\_VAR\_SPLINEINT** Compute variance with spline.

**NC\_MATTER\_VAR\_FFT** Compute using fft.

struct NcMatterVarClass

```
struct NcMatterVarClass {
};
```

struct NcMatterVar

```
struct NcMatterVar;
```

**nc\_matter\_var\_new ()**

```
NcMatterVar *      nc_matter_var_new      (NcMatterVarStrategy vs,
                                           NcWindow *wp,
                                           NcTransferFunc *tf);
```

This function allocates memory for a new **NcMatterVar** object and sets its properties to the values from the input arguments.

**vs** : a **NcMatterVarStrategy**.

**wp** : a **NcWindow**.

**tf** : a **NcTransferFunc**.

**Returns** : A new **NcMatterVar**.

**nc\_matter\_var\_copy ()**

```
NcMatterVar *      nc_matter_var_copy      (NcMatterVar *vp);
```

This function duplicates the **NcMatterVar** object setting the same values of the original properties.

**vp** : a **NcMatterVar**.

**Returns** : A new **NcMatterVar**. *[transfer full]*

**nc\_matter\_var\_free ()**

```
void              nc_matter_var_free      (NcMatterVar *vp);
```

Atomically decrements the reference count of **vp** by one. If the reference count drops to 0, all memory allocated by **vp** is released.

**vp** : a **NcMatterVar**.

**nc\_matter\_var\_clear ()**

```
void              nc_matter_var_clear      (NcMatterVar **vp);
```

Atomically decrements the reference count of **vp** by one. If the reference count drops to 0, all memory allocated by **vp** is released. Set pointer to NULL.

**vp** : a **NcMatterVar**.

**nc\_matter\_var\_prepare ()**

```
void              nc_matter_var_prepare      (NcMatterVar *vp,
                                           NcHICosmo *model);
```

FIXME

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**nc\_matter\_var\_var0 ()**

gdouble	nc_matter_var_var0	(NcMatterVar *vp, NcHICosmo *model, gdouble lnR);
---------	--------------------	---

This function returns the variance of the density contrast at redshift  $z = 0$  computed at scale R

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**lnR** : logarithm base e of the radius.

**Returns** : a gdouble which is the variance  $\sigma^2(R, z = 0)$ .

**nc\_matter\_var\_dlnvar0\_dR ()**

gdouble	nc_matter_var_dlnvar0_dR	(NcMatterVar *vp, NcHICosmo *model, gdouble lnR);
---------	--------------------------	---

FIXME

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**lnR** : logarithm base e of the radius.

**Returns** : FIXME

**nc\_matter\_var\_dlnvar0\_dlnR ()**

gdouble	nc_matter_var_dlnvar0_dlnR	(NcMatterVar *vp, NcHICosmo *model, gdouble lnR);
---------	----------------------------	---

FIXME

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**lnR** : logarithm base e of the radius.

**Returns** : FIXME

**nc\_matter\_var\_mass\_to\_R ()**

gdouble	nc_matter_var_mass_to_R	(NcMatterVar *vp, NcHICosmo *model, gdouble M);
---------	-------------------------	---

FIXME

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**M** : mass enclosed in the volume specified by the window function.

**Returns** : FIXME

**nc\_matter\_var\_R\_to\_mass ()**

gdouble	nc_matter_var_R_to_mass	(NcMatterVar *vp, NcHICosmo *model, gdouble R);
---------	-------------------------	---

FIXME mass enclosed in the volume specified by the window function

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**R** : radius.

**Returns** : FIXME

**nc\_matter\_var\_lnM\_to\_lnR ()**

gdouble	nc_matter_var_lnM_to_lnR	(NcMatterVar *vp, NcHICosmo *model, gdouble lnM);
---------	--------------------------	---

FIXME

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**lnM** : logarithm base e of the mass enclosed in the volume specified by the window function.

**Returns** : FIXME

**nc\_matter\_var\_lnR\_to\_lnM ()**

gdouble	nc_matter_var_lnR_to_lnM	(NcMatterVar *vp, NcHICosmo *model, gdouble lnR);
---------	--------------------------	---

FIXME mass enclosed in the volume specified by the window function

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**lnR** : logarithm base e of the radius.

**Returns** : FIXME

**nc\_matter\_var\_integrand\_over\_window2 ()**

gdouble	nc_matter_var_integrand_over_window2	(NcMatterVar *vp, NcHICosmo *model, gdouble k);
---------	--------------------------------------	---

FIXME

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**k** : FIXME

**Returns** : FIXME



**nc\_matter\_var\_spectral\_moment\_over\_growth2 ()**

```
gdouble          nc_matter_var_spectral_moment_over_growth2
                  (NcMatterVar *vp,
                   NcHICosmo *model,
                   gint n);
```

FIXME  $\frac{\{\sigma^2\}}{\{D^2\}}$

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**n** : FIXME

**Returns** : FIXME

**nc\_matter\_var\_spectral\_moment\_over\_growth2\_tophat ()**

```
gdouble          nc_matter_var_spectral_moment_over_growth2_tophat
                  (NcMatterVar *vp,
                   NcHICosmo *model,
                   gint n);
```

FIXME

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**n** : FIXME

**Returns** : FIXME

**nc\_matter\_var\_spectral\_moment\_over\_growth2\_gaussian ()**

```
gdouble          nc_matter_var_spectral_moment_over_growth2_gaussian
                  (NcMatterVar *vp,
                   NcHICosmo *model,
                   gint n);
```

FIXME

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**n** : FIXME

**Returns** : FIXME

**nc\_matter\_var\_dsigma0\_dR ()**

```
gdouble          nc_matter_var_dsigma0_dR
                  (NcMatterVar *vp,
                   NcHICosmo *model,
                   gdouble lnR);
```

**nc\_matter\_var\_sigma8\_sqrtvar0 ()**

```
gdouble          nc_matter_var_sigma8_sqrtvar0      (NcMatterVar *vp,
                                                    NcHICosmo *model);
```

**Property Details****The "strategy" property**

"strategy"	NcMatterVarStrategy	: Read / Write / Construct Only
------------	---------------------	---------------------------------

Strategy.

Default value: NC\_MATTER\_VAR\_FFT

**The "transfer" property**

"transfer"	NcTransferFunc*	: Read / Write / Construct Only
------------	-----------------	---------------------------------

This property keeps the transferfunc object.

**The "window" property**

"window"	NcWindow*	: Read / Write / Construct Only
----------	-----------	---------------------------------

This property keeps the window object.

**7.5 Multiplicity Function****7.5.1 Multiplicity Function**

Multiplicity Function — Dark Matter Halo FIXME

**Synopsis**

```
struct          NcMultiplicityFuncClass;
struct          NcMultiplicityFunc;
NcMultiplicityFunc * nc_multiplicity_func_new_from_name (gchar *multiplicity_name);
gdouble          nc_multiplicity_func_eval              (NcMultiplicityFunc *mulf,
                                                         NcHICosmo *model,
                                                         gdouble sigma,
                                                         gdouble z);

void            nc_multiplicity_func_free              (NcMultiplicityFunc *mulf);
void            nc_multiplicity_func_clear             (NcMultiplicityFunc **mulf);
```

## Object Hierarchy

```
GObject
+----NcMultiplicityFunc
      +----NcMultiplicityFuncJenkins
      +----NcMultiplicityFuncPS
      +----NcMultiplicityFuncST
      +----NcMultiplicityFuncTinkerCrit
      +----NcMultiplicityFuncTinker
      +----NcMultiplicityFuncTinkerMean
      +----NcMultiplicityFuncWarren
```

## Description

FIXME

## Details

### struct NcMultiplicityFuncClass

```
struct NcMultiplicityFuncClass {
};
```

### struct NcMultiplicityFunc

```
struct NcMultiplicityFunc;
```

### nc\_multiplicity\_func\_new\_from\_name ()

```
NcMultiplicityFunc * nc_multiplicity_func_new_from_name (gchar *multiplicity_name);
```

This function returns a new **NcMultiplicityFunc** whose type is defined by *multiplicity\_name*.

***multiplicity\_name***: string which specifies the multiplicity function type.

**Returns**: A new **NcMultiplicityFunc**.

### nc\_multiplicity\_func\_eval ()

```
gdouble          nc_multiplicity_func_eval          (NcMultiplicityFunc *mul_f,
                                                    NcHICosmo *model,
                                                    gdouble sigma,
                                                    gdouble z);
```

FIXME

***mul\_f***: a **NcMultiplicityFunc**.

***model***: a **NcHICosmo**.

***sigma***: FIXME

***z***: redshift.

**Returns**: FIXME

**nc\_multiplicity\_func\_free ()**

```
void nc_multiplicity_func_free (NcMultiplicityFunc *mulf);
```

Atomically decrements the reference count of *mulf* by one. If the reference count drops to 0, all memory allocated by *mulf* is released.

*mulf*: a **NcMultiplicityFunc**.

**nc\_multiplicity\_func\_clear ()**

```
void nc_multiplicity_func_clear (NcMultiplicityFunc **mulf);
```

Atomically decrements the reference count of *mulf* by one. If the reference count drops to 0, all memory allocated by *mulf* is released. Set pointer to NULL.

*mulf*: a **NcMultiplicityFunc**.

**7.5.2 Press-Schechter Multiplicity Function**

Press-Schechter Multiplicity Function — Dark Matter Halo FIXME

**Synopsis**

```
struct NcMultiplicityFuncPSClass;
struct NcMultiplicityFuncPS;
NcMultiplicityFunc * nc_multiplicity_func_ps_new (gdouble delta_c);
void nc_multiplicity_func_ps_set_delta_c (NcMultiplicityFuncPS *mulf_ps,
                                          gdouble delta_c);
gdouble nc_multiplicity_func_ps_get_delta_c (const NcMultiplicityFuncPS *mulf_
```

**Object Hierarchy**

```
GObject
+----NcMultiplicityFunc
      +----NcMultiplicityFuncPS
```

**Properties**

```
"critical-delta"          gdouble          : Read / Write / Construct Only
```

**Description**

FIXME

**Details****struct NcMultiplicityFuncPSClass**

```
struct NcMultiplicityFuncPSClass {
};
```

**struct NcMultiplicityFuncPS**

```
struct NcMultiplicityFuncPS;
```

**nc\_multiplicity\_func\_ps\_new ()**

```
NcMultiplicityFunc * nc_multiplicity_func_ps_new (gdouble delta_c);
```

FIXME

**delta\_c**: FIXME

**Returns**: A new **NcMultiplicityFunc**.

**nc\_multiplicity\_func\_ps\_set\_delta\_c ()**

```
void nc_multiplicity_func_ps_set_delta_c (NcMultiplicityFuncPS *mulf_ps,
                                           gdouble delta_c);
```

Sets the value *delta\_c* to the "critical-delta" property.

**mulf\_ps**: a **NcMultiplicityFuncPS**.

**delta\_c**: value of "critical-delta".

**nc\_multiplicity\_func\_ps\_get\_delta\_c ()**

```
gdouble nc_multiplicity_func_ps_get_delta_c (const NcMultiplicityFuncPS * ←
      mulf_ps);
```

**mulf\_ps**: a **NcMultiplicityFuncPS**.

**Returns**: the value of "critical\_delta" property.

**Property Details****The "critical-delta" property**

"critical-delta"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Critical delta.

Default value: 1.686

**7.5.3 Sheth-Tormen Multiplicity Function**

Sheth-Tormen Multiplicity Function — Dark Matter Halo FIXME

Synopsis

```
struct          NcMultiplicityFuncSTClass;
struct          NcMultiplicityFuncST;
NcMultiplicityFunc * nc_multiplicity_func_st_new      (gdouble A,
                                                    gdouble b,
                                                    gdouble p,
                                                    gdouble delta_c);

void            nc_multiplicity_func_st_set_A        (NcMultiplicityFuncST *mulf_st,
                                                    gdouble A);

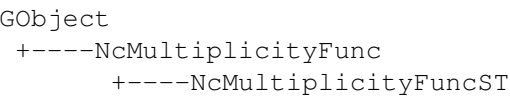
gdouble         nc_multiplicity_func_st_get_A        (const NcMultiplicityFuncST *mulf_
void           nc_multiplicity_func_st_set_b        (NcMultiplicityFuncST *mulf_st,
                                                    gdouble b);

gdouble         nc_multiplicity_func_st_get_b        (const NcMultiplicityFuncST *mulf_
void           nc_multiplicity_func_st_set_p        (NcMultiplicityFuncST *mulf_st,
                                                    gdouble p);

gdouble         nc_multiplicity_func_st_get_p        (const NcMultiplicityFuncST *mulf_
void           nc_multiplicity_func_st_set_delta_c  (NcMultiplicityFuncST *mulf_st,
                                                    gdouble delta_c);

gdouble         nc_multiplicity_func_st_get_delta_c (const NcMultiplicityFuncST *mulf_
```

Object Hierarchy



Properties

"A"	gdouble	: Read / Write / Construct Only
"b"	gdouble	: Read / Write / Construct Only
"critical-delta"	gdouble	: Read / Write / Construct Only
"p"	gdouble	: Read / Write / Construct Only

Description

FIXME

Details

struct NcMultiplicityFuncSTClass

```
struct NcMultiplicityFuncSTClass {
};
```

struct NcMultiplicityFuncST

```
struct NcMultiplicityFuncST;
```

---

**nc\_multiplicity\_func\_st\_new ()**

```
NcMultiplicityFunc * nc_multiplicity_func_st_new (gdouble A,
                                                  gdouble b,
                                                  gdouble p,
                                                  gdouble delta_c);
```

FIXME

**A** : FIXME

**b** : FIXME

**p** : FIXME

**delta\_c** : FIXME

**Returns** : A new **NcMultiplicityFunc**.

**nc\_multiplicity\_func\_st\_set\_A ()**

```
void nc_multiplicity_func_st_set_A (NcMultiplicityFuncST *mulf_st,
                                     gdouble A);
```

Sets the value *A* to the "A" property.

**mulf\_st** : a **NcMultiplicityFuncST**.

**A** : value of "A".

**nc\_multiplicity\_func\_st\_get\_A ()**

```
gdouble nc_multiplicity_func_st_get_A (const NcMultiplicityFuncST * ←
    mulf_st);
```

**mulf\_st** : a **NcMultiplicityFuncST**.

**Returns** : the value of "A" property.

**nc\_multiplicity\_func\_st\_set\_b ()**

```
void nc_multiplicity_func_st_set_b (NcMultiplicityFuncST *mulf_st,
                                     gdouble b);
```

Sets the value *b* to the "b" property.

**mulf\_st** : a **NcMultiplicityFuncST**.

**b** : value of "b".

**nc\_multiplicity\_func\_st\_get\_b ()**

```
gdouble nc_multiplicity_func_st_get_b (const NcMultiplicityFuncST * ←
    mulf_st);
```

**mulf\_st** : a **NcMultiplicityFuncST**.

**Returns** : the value of "b" property.

**nc\_multiplicity\_func\_st\_set\_p ()**

```
void                nc_multiplicity_func_st_set_p      (NcMultiplicityFuncST *mulf_st,
                                                         gdouble p);
```

Sets the value  $p$  to the "p" property.

*mulf\_st* : a NcMultiplicityFuncST.

*p* : value of "p".

**nc\_multiplicity\_func\_st\_get\_p ()**

```
gdouble            nc_multiplicity_func_st_get_p      (const NcMultiplicityFuncST * ←
    mulf_st);
```

*mulf\_st* : a NcMultiplicityFuncST.

*Returns* : the value of "p" property.

**nc\_multiplicity\_func\_st\_set\_delta\_c ()**

```
void                nc_multiplicity_func_st_set_delta_c (NcMultiplicityFuncST *mulf_st,
                                                         gdouble delta_c);
```

Sets the value  $\delta_c$  to the "critical-delta" property.

*mulf\_st* : a NcMultiplicityFuncST.

*delta\_c* : value of "critical-delta".

**nc\_multiplicity\_func\_st\_get\_delta\_c ()**

```
gdouble            nc_multiplicity_func_st_get_delta_c (const NcMultiplicityFuncST * ←
    mulf_st);
```

*mulf\_st* : a NcMultiplicityFuncST.

*Returns* : the value of "critical\_delta" property.

**Property Details**

**The "A" property**

"A"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 0.3222

**The "b" property**

"b"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 0.707



**The "critical-delta" property**

"critical-delta"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Critical delta.

Default value: 1.686

**The "p" property**

"p"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 0.3

**7.5.4 Jenkins Multiplicity Function**

Jenkins Multiplicity Function — Dark Matter Halo FIXME

**Synopsis**

```

struct          NcMultiplicityFuncJenkinsClass;
struct          NcMultiplicityFuncJenkins;
NcMultiplicityFunc * nc_multiplicity_func_jenkins_new (gdouble A,
                                                       gdouble A_tCDM,
                                                       gdouble B,
                                                       gdouble B_tCDM,
                                                       gdouble epsilon,
                                                       gdouble epsilon_tCDM);

void            nc_multiplicity_func_jenkins_set_A (NcMultiplicityFuncJenkins *mulf_j,
                                                    gdouble A);

gdouble         nc_multiplicity_func_jenkins_get_A (const NcMultiplicityFuncJenkins *mulf_j);
void            nc_multiplicity_func_jenkins_set_A_tCDM (NcMultiplicityFuncJenkins *mulf_j,
                                                         gdouble A_tCDM);

gdouble         nc_multiplicity_func_jenkins_get_A_tCDM (const NcMultiplicityFuncJenkins *mulf_j);

void            nc_multiplicity_func_jenkins_set_B (NcMultiplicityFuncJenkins *mulf_j,
                                                    gdouble B);

gdouble         nc_multiplicity_func_jenkins_get_B (const NcMultiplicityFuncJenkins *mulf_j);
void            nc_multiplicity_func_jenkins_set_B_tCDM (NcMultiplicityFuncJenkins *mulf_j,
                                                         gdouble B_tCDM);

gdouble         nc_multiplicity_func_jenkins_get_B_tCDM (const NcMultiplicityFuncJenkins *mulf_j);

void            nc_multiplicity_func_jenkins_set_epsilon (NcMultiplicityFuncJenkins *mulf_j,
                                                         gdouble epsilon);

gdouble         nc_multiplicity_func_jenkins_get_epsilon (const NcMultiplicityFuncJenkins *mulf_j);

void            nc_multiplicity_func_jenkins_set_epsilon_tCDM (NcMultiplicityFuncJenkins *mulf_j,
                                                                gdouble epsilon_tCDM);

gdouble         nc_multiplicity_func_jenkins_get_epsilon_tCDM (const NcMultiplicityFuncJenkins *mulf_j);

```

## Object Hierarchy

```
GObject
+----NcMultiplicityFunc
      +----NcMultiplicityFuncJenkins
```

## Properties

"A"	gdouble	: Read / Write / Construct Only
"A-tCDM"	gdouble	: Read / Write / Construct Only
"B"	gdouble	: Read / Write / Construct Only
"B-tCDM"	gdouble	: Read / Write / Construct Only
"epsilon"	gdouble	: Read / Write / Construct Only
"epsilon-tCDM"	gdouble	: Read / Write / Construct Only

## Description

FIXME

## Details

### struct NcMultiplicityFuncJenkinsClass

```
struct NcMultiplicityFuncJenkinsClass {
};
```

### struct NcMultiplicityFuncJenkins

```
struct NcMultiplicityFuncJenkins;
```

### nc\_multiplicity\_func\_jenkins\_new ()

```
NcMultiplicityFunc * nc_multiplicity_func_jenkins_new (gdouble A,
                                                       gdouble A_tCDM,
                                                       gdouble B,
                                                       gdouble B_tCDM,
                                                       gdouble epsilon,
                                                       gdouble epsilon_tCDM);
```

FIXME

**A**: FIXME

**A\_tCDM**: FIXME

**B**: FIXME

**B\_tCDM**: FIXME

**epsilon**: FIXME

**epsilon\_tCDM**: FIXME

**Returns**: A new **NcMultiplicityFunc**.

**nc\_multiplicity\_func\_jenkins\_set\_A ()**

```
void          nc_multiplicity_func_jenkins_set_A (NcMultiplicityFuncJenkins * ←
    mulf_jenkins,
                                                    gdouble A);
```

Sets the value  $A$  to the "A" property.

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

***A*** : value of "A".

**nc\_multiplicity\_func\_jenkins\_get\_A ()**

```
gdouble          nc_multiplicity_func_jenkins_get_A (const NcMultiplicityFuncJenkins * ←
    mulf_jenkins);
```

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

**Returns** : the value of "A" property.

**nc\_multiplicity\_func\_jenkins\_set\_A\_tCDM ()**

```
void          nc_multiplicity_func_jenkins_set_A_tCDM
                                                    (NcMultiplicityFuncJenkins * ←
    mulf_jenkins,
    gdouble A_tCDM);
```

Sets the value  $A_{tCDM}$  to the "A-tCDM" property.

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

***A\_tCDM*** : value of "A-tCDM".

**nc\_multiplicity\_func\_jenkins\_get\_A\_tCDM ()**

```
gdouble          nc_multiplicity_func_jenkins_get_A_tCDM
                                                    (const NcMultiplicityFuncJenkins * ←
    mulf_jenkins);
```

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

**Returns** : the value of "A-tCDM" property.

**nc\_multiplicity\_func\_jenkins\_set\_B ()**

```
void          nc_multiplicity_func_jenkins_set_B (NcMultiplicityFuncJenkins * ←
    mulf_jenkins,
                                                    gdouble B);
```

Sets the value  $B$  to the "B" property.

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

***B*** : value of "B".

**nc\_multiplicity\_func\_jenkins\_get\_B ()**

```
gdouble          nc_multiplicity_func_jenkins_get_B  (const NcMultiplicityFuncJenkins * ←
    mulf_jenkins);
```

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

**Returns** : the value of "B" property.

**nc\_multiplicity\_func\_jenkins\_set\_B\_tCDM ()**

```
void              nc_multiplicity_func_jenkins_set_B_tCDM
                  (NcMultiplicityFuncJenkins * ←
                  mulf_jenkins,
                  gdouble B_tCDM);
```

Sets the value *B\_tCDM* to the "B-tCDM" property.

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

***B\_tCDM*** : value of "B-tCDM".

**nc\_multiplicity\_func\_jenkins\_get\_B\_tCDM ()**

```
gdouble          nc_multiplicity_func_jenkins_get_B_tCDM
                  (const NcMultiplicityFuncJenkins * ←
                  mulf_jenkins);
```

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

**Returns** : the value of "B-tCDM" property.

**nc\_multiplicity\_func\_jenkins\_set\_epsilon ()**

```
void              nc_multiplicity_func_jenkins_set_epsilon
                  (NcMultiplicityFuncJenkins * ←
                  mulf_jenkins,
                  gdouble epsilon);
```

Sets the value *epsilon* to the "epsilon" property.

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

***epsilon*** : value of "epsilon".

**nc\_multiplicity\_func\_jenkins\_get\_epsilon ()**

```
gdouble          nc_multiplicity_func_jenkins_get_epsilon
                  (const NcMultiplicityFuncJenkins * ←
                  mulf_jenkins);
```

***mulf\_jenkins*** : a **NcMultiplicityFuncJenkins**.

**Returns** : the value of "epsilon" property.

**nc\_multiplicity\_func\_jenkins\_set\_epsilon\_tCDM ()**

```
void                                nc_multiplicity_func_jenkins_set_epsilon_tCDM
                                   (NcMultiplicityFuncJenkins * ↔
                                   mulf_jenkins,
                                   gdouble epsilon_tCDM);
```

Sets the value *epsilon\_tCDM* to the "epsilon-tCDM" property.

*mulf\_jenkins* : a **NcMultiplicityFuncJenkins**.

*epsilon\_tCDM* : value of "epsilon-tCDM".

**nc\_multiplicity\_func\_jenkins\_get\_epsilon\_tCDM ()**

```
gdouble                            nc_multiplicity_func_jenkins_get_epsilon_tCDM
                                   (const NcMultiplicityFuncJenkins * ↔
                                   mulf_jenkins);
```

*mulf\_jenkins* : a **NcMultiplicityFuncJenkins**.

**Returns** : the value of "epsilon-tCDM" property.

**Property Details**

**The "A" property**

"A"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 0.315

**The "A-tCDM" property**

"A-tCDM"	gdouble	: Read / Write / Construct Only
----------	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 0

**The "B" property**

"B"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 0.61

**The "B-tCDM" property**

"B-tCDM"	gdouble	: Read / Write / Construct Only
----------	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 0

---

The "epsilon" property

"epsilon"	gdouble	: Read / Write / Construct Only
-----------	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 3.8

The "epsilon-tCDM" property

"epsilon-tCDM"	gdouble	: Read / Write / Construct Only
----------------	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 0

7.5.5 Warren Multiplicity Function

Warren Multiplicity Function — Dark Matter Halo FIXME

Synopsis

```
struct          NcMultiplicityFuncWarrenClass;
struct          NcMultiplicityFuncWarren;
NcMultiplicityFunc * nc_multiplicity_func_warren_new (gdouble A,
                                                    gdouble a,
                                                    gdouble b,
                                                    gdouble c);

void            nc_multiplicity_func_warren_set_A (NcMultiplicityFuncWarren *mulf_warren,
                                                    gdouble A);

gdouble         nc_multiplicity_func_warren_get_A (const NcMultiplicityFuncWarren *mulf_warren);
void            nc_multiplicity_func_warren_set_a (NcMultiplicityFuncWarren *mulf_warren,
                                                    gdouble a);

gdouble         nc_multiplicity_func_warren_get_a (const NcMultiplicityFuncWarren *mulf_warren);
void            nc_multiplicity_func_warren_set_b (NcMultiplicityFuncWarren *mulf_warren,
                                                    gdouble b);

gdouble         nc_multiplicity_func_warren_get_b (const NcMultiplicityFuncWarren *mulf_warren);
void            nc_multiplicity_func_warren_set_c (NcMultiplicityFuncWarren *mulf_warren,
                                                    gdouble c);

gdouble         nc_multiplicity_func_warren_get_c (const NcMultiplicityFuncWarren *mulf_warren);
```

Object Hierarchy

```
GObject
+----NcMultiplicityFunc
      +----NcMultiplicityFuncWarren
```

Properties

"A"	gdouble	: Read / Write / Construct Only
"a"	gdouble	: Read / Write / Construct Only
"b"	gdouble	: Read / Write / Construct Only
"c"	gdouble	: Read / Write / Construct Only

**Description**

FIXME

**Details****struct NcMultiplicityFuncWarrenClass**

```
struct NcMultiplicityFuncWarrenClass {
};
```

**struct NcMultiplicityFuncWarren**

```
struct NcMultiplicityFuncWarren;
```

**nc\_multiplicity\_func\_warren\_new ()**

```
NcMultiplicityFunc * nc_multiplicity_func_warren_new    (gdouble A,
                                                         gdouble a,
                                                         gdouble b,
                                                         gdouble c);
```

FIXME

**A** : FIXME

**a** : FIXME

**b** : FIXME

**c** : FIXME

**Returns** : A new **NcMultiplicityFunc**.

**nc\_multiplicity\_func\_warren\_set\_A ()**

```
void          nc_multiplicity_func_warren_set_A    (NcMultiplicityFuncWarren * ↔
mulf_warren,                                     gdouble A);
```

Sets the value *A* to the "**A**" property.

**mulf\_warren** : a **NcMultiplicityFuncWarren**.

**A** : value of "**A**".

**nc\_multiplicity\_func\_warren\_get\_A ()**

```
gdouble          nc_multiplicity_func_warren_get_A    (const NcMultiplicityFuncWarren * ↔
mulf_warren);
```

**mulf\_warren** : a **NcMultiplicityFuncWarren**.

**Returns** : the value of "**A**" property.

**nc\_multiplicity\_func\_warren\_set\_a ()**

```
void          nc_multiplicity_func_warren_set_a  (NcMultiplicityFuncWarren * ↔  
    mulf_warren,  
                                                    gdouble a);
```

Sets the value *a* to the "a" property.

***mulf\_warren*** : a **NcMultiplicityFuncWarren**.

***a*** : value of "a".

**nc\_multiplicity\_func\_warren\_get\_a ()**

```
gdouble       nc_multiplicity_func_warren_get_a  (const NcMultiplicityFuncWarren * ↔  
    mulf_warren);
```

***mulf\_warren*** : a **NcMultiplicityFuncWarren**.

**Returns** : the value of "a" property.

**nc\_multiplicity\_func\_warren\_set\_b ()**

```
void          nc_multiplicity_func_warren_set_b  (NcMultiplicityFuncWarren * ↔  
    mulf_warren,  
                                                    gdouble b);
```

Sets the value *b* to the "b" property.

***mulf\_warren*** : a **NcMultiplicityFuncWarren**.

***b*** : value of "b".

**nc\_multiplicity\_func\_warren\_get\_b ()**

```
gdouble       nc_multiplicity_func_warren_get_b  (const NcMultiplicityFuncWarren * ↔  
    mulf_warren);
```

***mulf\_warren*** : a **NcMultiplicityFuncWarren**.

**Returns** : the value of "b" property.

**nc\_multiplicity\_func\_warren\_set\_c ()**

```
void          nc_multiplicity_func_warren_set_c  (NcMultiplicityFuncWarren * ↔  
    mulf_warren,  
                                                    gdouble c);
```

Sets the value *c* to the "c" property.

***mulf\_warren*** : a **NcMultiplicityFuncWarren**.

***c*** : value of "c".

---



**nc\_multiplicity\_func\_warren\_get\_c()**

```
gdouble      nc_multiplicity_func_warren_get_c      (const NcMultiplicityFuncWarren * ↵  
    mul_f_warren);
```

*mul\_f\_warren* : a **NcMultiplicityFuncWarren**.

*Returns* : the value of "c" property.

**Property Details**

**The "A" property**

"A"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)  
Default value: 0.7234

**The "a" property**

"a"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)  
Default value: 1.625

**The "b" property**

"b"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)  
Default value: 0.2538

**The "c" property**

"c"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)  
Default value: 1.1982

**7.5.6 Tinker Multiplicity Function**

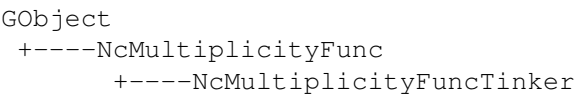
Tinker Multiplicity Function — Dark Matter Halo FIXME

Synopsis

```
struct          NcMultiplicityFuncTinkerClass;
struct          NcMultiplicityFuncTinker;
NcMultiplicityFunc * nc_multiplicity_func_tinker_new      (gdouble A0,
                                                         gdouble a0,
                                                         gdouble b0,
                                                         gdouble c,
                                                         gdouble Delta);

void            nc_multiplicity_func_tinker_set_A0        (NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble A0);
gdouble         nc_multiplicity_func_tinker_get_A0        (const NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble A0);
void            nc_multiplicity_func_tinker_set_a0        (NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble a0);
gdouble         nc_multiplicity_func_tinker_get_a0        (const NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble a0);
void            nc_multiplicity_func_tinker_set_b0        (NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble b0);
gdouble         nc_multiplicity_func_tinker_get_b0        (const NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble b0);
void            nc_multiplicity_func_tinker_set_c          (NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble c);
gdouble         nc_multiplicity_func_tinker_get_c          (const NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble c);
void            nc_multiplicity_func_tinker_set_Delta      (NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble Delta);
gdouble         nc_multiplicity_func_tinker_get_Delta      (const NcMultiplicityFuncTinker *mulf_tinker,
                                                         gdouble Delta);
```

Object Hierarchy



Properties

"A0"	gdouble	: Read / Write / Construct Only
"Delta"	gdouble	: Read / Write / Construct Only
"a0"	gdouble	: Read / Write / Construct Only
"b0"	gdouble	: Read / Write / Construct Only
"c"	gdouble	: Read / Write / Construct Only

Description

FIXME

Details

struct NcMultiplicityFuncTinkerClass

```
struct NcMultiplicityFuncTinkerClass {
};
```

**struct NcMultiplicityFuncTinker**

```
struct NcMultiplicityFuncTinker;
```

**nc\_multiplicity\_func\_tinker\_new ()**

```
NcMultiplicityFunc * nc_multiplicity_func_tinker_new (gdouble A0,
                                                    gdouble a0,
                                                    gdouble b0,
                                                    gdouble c,
                                                    gdouble Delta);
```

FIXME

**A0**: FIXME

**a0**: FIXME

**b0**: FIXME

**c**: FIXME

**Delta**: FIXME

**Returns**: A new **NcMultiplicityFunc**.

**nc\_multiplicity\_func\_tinker\_set\_A0 ()**

```
void nc_multiplicity_func_tinker_set_A0 (NcMultiplicityFuncTinker * ↔
mulf_tinker, gdouble A0);
```

Sets the value *A0* to the "A0" property.

**mulf\_tinker**: a **NcMultiplicityFuncTinker**.

**A0**: value of "A0".

**nc\_multiplicity\_func\_tinker\_get\_A0 ()**

```
gdouble nc_multiplicity_func_tinker_get_A0 (const NcMultiplicityFuncTinker * ↔
mulf_tinker);
```

**mulf\_tinker**: a **NcMultiplicityFuncTinker**.

**Returns**: the value of "A0" property.

**nc\_multiplicity\_func\_tinker\_set\_a0 ()**

```
void nc_multiplicity_func_tinker_set_a0 (NcMultiplicityFuncTinker * ↔
mulf_tinker, gdouble a0);
```

Sets the value *a0* to the "a0" property.

**mulf\_tinker**: a **NcMultiplicityFuncTinker**.

**a0**: value of "a0".

**nc\_multiplicity\_func\_tinker\_get\_a0 ()**

```
gdouble          nc_multiplicity_func_tinker_get_a0  (const NcMultiplicityFuncTinker * ↔  
mulf_tinker);
```

**mulf\_tinker** : a **NcMultiplicityFuncTinker**.

**Returns** : the value of "a0" property.

**nc\_multiplicity\_func\_tinker\_set\_b0 ()**

```
void          nc_multiplicity_func_tinker_set_b0  (NcMultiplicityFuncTinker * ↔  
mulf_tinker,  
gdouble b0);
```

Sets the value *b0* to the "b0" property.

**mulf\_tinker** : a **NcMultiplicityFuncTinker**.

**b0** : value of "b0".

**nc\_multiplicity\_func\_tinker\_get\_b0 ()**

```
gdouble          nc_multiplicity_func_tinker_get_b0  (const NcMultiplicityFuncTinker * ↔  
mulf_tinker);
```

**mulf\_tinker** : a **NcMultiplicityFuncTinker**.

**Returns** : the value of "b0" property.

**nc\_multiplicity\_func\_tinker\_set\_c ()**

```
void          nc_multiplicity_func_tinker_set_c  (NcMultiplicityFuncTinker * ↔  
mulf_tinker,  
gdouble c);
```

Sets the value *c* to the "c" property.

**mulf\_tinker** : a **NcMultiplicityFuncTinker**.

**c** : value of "c".

**nc\_multiplicity\_func\_tinker\_get\_c ()**

```
gdouble          nc_multiplicity_func_tinker_get_c  (const NcMultiplicityFuncTinker * ↔  
mulf_tinker);
```

**mulf\_tinker** : a **NcMultiplicityFuncTinker**.

**Returns** : the value of "c" property.

---

**nc\_multiplicity\_func\_tinker\_set\_Delta ()**

```
void                                nc_multiplicity_func_tinker_set_Delta
                                   (NcMultiplicityFuncTinker * ↔
                                   mulf_tinker,
                                   gdouble Delta);
```

Sets the value *Delta* to the "Delta" property.

*mulf\_tinker* : a NcMultiplicityFuncTinker.

*Delta* : value of "Delta".

**nc\_multiplicity\_func\_tinker\_get\_Delta ()**

```
gdouble                            nc_multiplicity_func_tinker_get_Delta
                                   (const NcMultiplicityFuncTinker * ↔
                                   mulf_tinker);
```

*mulf\_tinker* : a NcMultiplicityFuncTinker.

*Returns* : the value of "Delta" property.

**Property Details**

**The "A0" property**

"A0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 0.186

**The "Delta" property**

"Delta"	gdouble	: Read / Write / Construct Only
---------	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 200

**The "a0" property**

"a0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 1.47

**The "b0" property**

"b0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

FIXME Set correct values (limits)

Default value: 2.57

---

The "c" property

"c"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME Set correct values (limits)  
Default value: 1.19

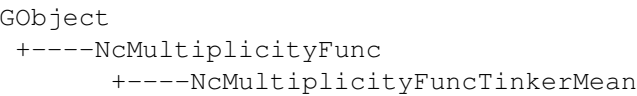
7.5.7 Mean Tinker Multiplicity Function

Mean Tinker Multiplicity Function — Dark Matter Halo FIXME

Synopsis

```
struct          NcMultiplicityFuncTinkerMeanClass;
struct          NcMultiplicityFuncTinkerMean;
NcMultiplicityFunc * nc_multiplicity_func_tinker_mean_new
                                (gdouble Delta);
void            nc_multiplicity_func_tinker_mean_set_Delta
                                (NcMultiplicityFuncTinkerMean *mul
                                gdouble Delta);
gdouble        nc_multiplicity_func_tinker_mean_get_Delta
                                (const NcMultiplicityFuncTinkerMea
```

Object Hierarchy



Properties

"Delta"	gdouble	: Read / Write / Construct Only
---------	---------	---------------------------------

Description

FIXME

Details

struct NcMultiplicityFuncTinkerMeanClass

```
struct NcMultiplicityFuncTinkerMeanClass {
};
```

struct NcMultiplicityFuncTinkerMean

```
struct NcMultiplicityFuncTinkerMean;
```

**nc\_multiplicity\_func\_tinker\_mean\_new ()**

```
NcMultiplicityFunc * nc_multiplicity_func_tinker_mean_new
                                                                (gdouble Delta);
```

FIXME

**Delta** : FIXME

**Returns** : A new **NcMultiplicityFunc**.

**nc\_multiplicity\_func\_tinker\_mean\_set\_Delta ()**

```
void nc_multiplicity_func_tinker_mean_set_Delta
                                           (NcMultiplicityFuncTinkerMean * ←
                                           mulf_tinker_mean,
                                           gdouble Delta);
```

Sets the value *Delta* to the **"Delta"** property.

**mulf\_tinker\_mean** : a **NcMultiplicityFuncTinkerMean**.

**Delta** : value of **"Delta"**.

**nc\_multiplicity\_func\_tinker\_mean\_get\_Delta ()**

```
gdouble nc_multiplicity_func_tinker_mean_get_Delta
                                           (const NcMultiplicityFuncTinkerMean ←
                                           *mulf_tinker_mean);
```

**mulf\_tinker\_mean** : a **NcMultiplicityFuncTinkerMean**.

**Returns** : the value of **"Delta"** property.

**Property Details**

**The "Delta" property**

"Delta"	gdouble	: Read / Write / Construct Only
---------	---------	---------------------------------

FIXME Set correct values (limits)

Allowed values: [200,3200]

Default value: 200

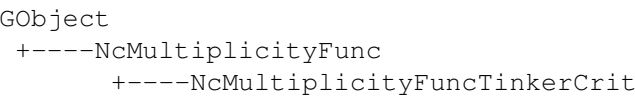
**7.5.8 Critical Tinker Multiplicity Function**

Critical Tinker Multiplicity Function — Dark Matter Halo FIXME

Synopsis

```
struct          NcMultiplicityFuncTinkerCritClass;
struct          NcMultiplicityFuncTinkerCrit;
NcMultiplicityFunc * nc_multiplicity_func_tinker_crit_new
                                   (gdouble Delta);
void            nc_multiplicity_func_tinker_crit_set_Delta
                                   (NcMultiplicityFuncTinkerCrit *mul
                                   gdouble Delta);
gdouble         nc_multiplicity_func_tinker_crit_get_Delta
                                   (const NcMultiplicityFuncTinkerCri
```

Object Hierarchy



Properties

"Delta"	gdouble	: Read / Write / Construct Only
---------	---------	---------------------------------

Description

FIXME

Details

struct NcMultiplicityFuncTinkerCritClass

```
struct NcMultiplicityFuncTinkerCritClass {
};
```

struct NcMultiplicityFuncTinkerCrit

```
struct NcMultiplicityFuncTinkerCrit;
```

nc\_multiplicity\_func\_tinker\_crit\_new ()

```
NcMultiplicityFunc * nc_multiplicity_func_tinker_crit_new
                                   (gdouble Delta);
```

FIXME

*Delta*: FIXME

*Returns*: A new **NcMultiplicityFunc**.



### nc\_multiplicity\_func\_tinker\_crit\_set\_Delta ()

```
void nc_multiplicity_func_tinker_crit_set_Delta
                                (NcMultiplicityFuncTinkerCrit *  $\leftrightarrow$ 
                                mul_f_tinker_crit,
                                gdouble Delta);
```

Sets the value *Delta* to the "Delta" property.

***mult\_tinker\_crit***: a **NcMultiplicityFuncTinkerCrit**.

*Delta*: value of "Delta".

**nc\_multiplicity\_func\_tinker\_crit\_get\_Delta ()**

[illegible]

***mult\_tinker\_crit***: a **NcMultiplicityFuncTinkerCrit**.

**Returns** : the value of "Delta" property.

### Property Details

## The "Delta" property

"Delta"	gdouble	: Read / Write / Construct Only
---------	---------	---------------------------------

FIXME Set correct values (limits)

Allowed values: [200,3200]

Default value: 200

## 7.6 Mass Function

## Mass Function — FIXME

## Synopsis

```

struct      NcMassFunctionClass;
struct      NcMassFunction;
enum        NcMassFunctionSplineOptimize;
NcMassFunction * nc_mass_function_new      (NcDistance *dist,
                                           NcMatterVar *vp,
                                           NcGrowthFunc *gf,
                                           NcMultiplicityFunc *mul);
NcMassFunction * nc_mass_function_copy    (NcMassFunction *mfp);
void          nc_mass_function_free      (NcMassFunction *mfp);
void          nc_mass_function_clear     (NcMassFunction **mfp);
void          nc_mass_function_set_area   (NcMassFunction *mfp,
                                           gdouble area);
void          nc_mass_function_set_area_sd (NcMassFunction *mfp,

```

void	nc_mass_function_set_eval_limits	gdoube area_sd); (NcMassFunction *mfp, NCHICosmo *model, gdoube lnMi, gdoube lnMf, gdoube zi, gdoube zf);
void	nc_mass_function_prepare	(NcMassFunction *mfp, NCHICosmo *model);
void	nc_mass_function_prepare_if_needed	(NcMassFunction *mfp, NCHICosmo *model);
gdoube	nc_mass_function_dn_dlnm	(NcMassFunction *mfp, NCHICosmo *model, gdoube lnM, gdoube z);
gdoube	nc_mass_function_dv_dzdomaga	(NcMassFunction *mfp, NCHICosmo *model, gdoube z);
gdoube	nc_mass_function_d2n_dzdlnm	(NcMassFunction *mfp, NCHICosmo *model, gdoube lnM, gdoube z);
gdoube	nc_mass_function_dn_dz	(NcMassFunction *mfp, NCHICosmo *model, gdoube lnMl, gdoube lnMu, gdoube z, gboolean spline);
gdoube	nc_mass_function_n	(NcMassFunction *mfp, NCHICosmo *model, gdoube lnMl, gdoube lnMu, gdoube zl, gdoube zu, NcMassFunctionSplineOptimize spline);
gdoube	nc_mass_function_dn_M_to_inf_dv	(NcMassFunction *mfp, NCHICosmo *model, gdoube M, gdoube z);
gdoube	nc_mass_function_dn_Ml_to_M2_dv	(NcMassFunction *mfp, NCHICosmo *model, gdoube Ml, gdoube M2, gdoube z);
void	nc_mass_function_sigma	(NcMassFunction *mfp, NCHICosmo *model, gdoube lnM, gdoube z, gdoube *dn_dlnM_ptr, gdoube *sigma_ptr);
void	nc_mass_function_alpha_eff	(NcMatterVar *vp, NCHICosmo *model, gdoube lnM, gdoube *a_eff_ptr);

## Object Hierarchy

```
GObject
+----NcMassFunction
```

## Properties

"area"	gdouble	: Read / Write / Construct
"distance"	NcDistance*	: Read / Write / Construct Only
"growth"	NcGrowthFunc*	: Read / Write / Construct Only
"multiplicity"	NcMultiplicityFunc*	: Read / Write / Construct Only
"variance"	NcMatterVar*	: Read / Write / Construct Only

## Description

FIXME

## Details

### struct NcMassFunctionClass

```
struct NcMassFunctionClass {
};
```

### struct NcMassFunction

```
struct NcMassFunction;
```

### enum NcMassFunctionSplineOptimize

```
typedef enum {
    NC_MASS_FUNCTION_SPLINE_NONE = 0,
    NC_MASS_FUNCTION_SPLINE_LNM,
    NC_MASS_FUNCTION_SPLINE_Z,
} NcMassFunctionSplineOptimize;
```

FIXME

**NC\_MASS\_FUNCTION\_SPLINE\_NONE** FIXME

**NC\_MASS\_FUNCTION\_SPLINE\_LNM** FIXME

**NC\_MASS\_FUNCTION\_SPLINE\_Z** FIXME

### nc\_mass\_function\_new ()

```
NcMassFunction *    nc_mass_function_new          (NcDistance *dist,
                                                    NcMatterVar *vp,
                                                    NcGrowthFunc *gf,
                                                    NcMultiplicityFunc *mulf);
```

This function allocates memory for a new **NcMassFunction** object and sets its properties to the values from the input arguments.

**dist** : a **NcDistance** sets to "distance".

**vp** : a **NcMatterVar** sets to "variance".

**gf** : a **NcGrowthFunc** sets to "growth".

**mul** : a **NcMultiplicityFunc** sets to "multiplicity".

**Returns** : A new **NcMassFunction**.

### nc\_mass\_function\_copy ()

```
NcMassFunction *   nc_mass_function_copy           (NcMassFunction *mfp);
```

This function duplicates the **NcMassFunction** object setting the same values of the original properties.

**mfp** : a **NcMassFunction**.

**Returns** : A new **NcMassFunction**. [*transfer full*]

### nc\_mass\_function\_free ()

```
void               nc_mass_function_free           (NcMassFunction *mfp);
```

Atomically decrements the reference count of *mfp* by one. If the reference count drops to 0, all memory allocated by *mfp* is released.

**mfp** : a **NcMassFunction**.

### nc\_mass\_function\_clear ()

```
void               nc_mass_function_clear          (NcMassFunction **mfp);
```

Atomically decrements the reference count of *mfp* by one. If the reference count drops to 0, all memory allocated by *mfp* is released. Set pointer to NULL.

**mfp** : a **NcMassFunction**.

### nc\_mass\_function\_set\_area ()

```
void               nc_mass_function_set_area       (NcMassFunction *mfp,  
                                                    gdouble area);
```

FIXME

**mfp** : a **NcMassFunction**.

**area** : area in steradian.

### nc\_mass\_function\_set\_area\_sd ()

```
void               nc_mass_function_set_area_sd    (NcMassFunction *mfp,  
                                                    gdouble area_sd);
```

FIXME

**mfp** : a **NcMassFunction**.

**area\_sd** : area in square degree.

**nc\_mass\_function\_set\_eval\_limits ()**

```
void                nc_mass_function_set_eval_limits  (NcMassFunction *mfp,
                                                    NcHICosmo *model,
                                                    gdouble lnMi,
                                                    gdouble lnMf,
                                                    gdouble zi,
                                                    gdouble zf);
```

FIXME

**mfp** : a **NcMassFunction**.

**model** : a **NcHICosmo**.

**lnMi** : minimum logarithm base e of mass.

**lnMf** : maximum logarithm base e of mass.

**zi** : minimum redshift.

**zf** : maximum redshift.

**nc\_mass\_function\_prepare ()**

```
void                nc_mass_function_prepare        (NcMassFunction *mfp,
                                                    NcHICosmo *model);
```

FIXME

**mfp** : a **NcMassFunction**.

**model** : a **NcHICosmo**.

**nc\_mass\_function\_prepare\_if\_needed ()**

```
void                nc_mass_function_prepare_if_needed (NcMassFunction *mfp,
                                                    NcHICosmo *model);
```

FIXME

**mfp** : a **NcMassFunction**.

**model** : a **NcHICosmo**.

**nc\_mass\_function\_dn\_dlnm ()**

```
gdouble            nc_mass_function_dn_dlnm        (NcMassFunction *mfp,
                                                    NcHICosmo *model,
                                                    gdouble lnM,
                                                    gdouble z);
```

This function computes the comoving number density of dark matter halos at redshift  $z$  and mass  $M$ .

**mfp** : a **NcMassFunction**.

**model** : a **NcHICosmo**.

**lnM** : logarithm base e of mass.

**z** : redshift.

**Returns** :  $dn/dlnM$ .

**nc\_mass\_function\_dv\_dzdomega ()**

gdouble	nc_mass_function_dv_dzdomega	(NcMassFunction *mfp, NcHICosmo *model, gdouble z);
---------	------------------------------	---

FIXME

**mfp** : a **NcMassFunction**.

**model** : a **NcHICosmo**.

**z** : redshift.

**Returns** : FIXME

**nc\_mass\_function\_d2n\_dzdlnm ()**

gdouble	nc_mass_function_d2n_dzdlnm	(NcMassFunction *mfp, NcHICosmo *model, gdouble lnM, gdouble z);
---------	-----------------------------	---

FIXME

**mfp** : a **NcMassFunction**.

**model** : a **NcHICosmo**.

**lnM** : logarithm base e of mass.

**z** : redshift.

**Returns** : FIXME

**nc\_mass\_function\_dn\_dz ()**

gdouble	nc_mass_function_dn_dz	(NcMassFunction *mfp, NcHICosmo *model, gdouble lnMl, gdouble lnMu, gdouble z, gboolean spline);
---------	------------------------	---

FIXME

**mfp** : a **NcMassFunction**.

**model** : a **NcHICosmo**.

**lnMl** : logarithm base e of mass, lower threshold.

**lnMu** : logarithm base e of mass, upper threshold.

**z** : redshift.

**spline** : Whenever to create an intermediary spline of the mass integration.

**Returns** : FIXME

**nc\_mass\_function\_n ()**

```

gdouble          nc_mass_function_n          (NcMassFunction *mfp,
                                              NcHICosmo *model,
                                              gdouble lnMl,
                                              gdouble lnMu,
                                              gdouble zl,
                                              gdouble zu,
                                              NcMassFunctionSplineOptimize ←
                                              spline);

```

FIXME

**mfp** : a **NcMassFunction**.

**model** : a **NcHICosmo**.

**lnMl** : logarithm base e of mass, lower threshold.

**lnMu** : logarithm base e of mass, upper threshold.

**zl** : minimum redshift.

**zu** : maximum redshift.

**spline** : Whenever to create an intermediary spline of the integration.

**Returns** : FIXME

**nc\_mass\_function\_dn\_M\_to\_inf\_dv ()**

```

gdouble          nc_mass_function_dn_M_to_inf_dv  (NcMassFunction *mfp,
                                                  NcHICosmo *model,
                                                  gdouble M,
                                                  gdouble z);

```

**nc\_mass\_function\_dn\_M1\_to\_M2\_dv ()**

```

gdouble          nc_mass_function_dn_M1_to_M2_dv  (NcMassFunction *mfp,
                                                  NcHICosmo *model,
                                                  gdouble M1,
                                                  gdouble M2,
                                                  gdouble z);

```

**nc\_mass\_function\_sigma ()**

```

void          nc_mass_function_sigma  (NcMassFunction *mfp,
                                       NcHICosmo *model,
                                       gdouble lnM,
                                       gdouble z,
                                       gdouble *dn_dlnM_ptr,
                                       gdouble *sigma_ptr);

```

This function computes the standard deviation of density contrast of the matter fluctuations and the the comoving number density of dark matter halos at redshift  $z$  and mass  $M$ . These values are stored in *sigma\_ptr* and *dn\_dlnM\_ptr*, respectively.

**mfp** : a **NcMassFunction**.

**model** : a **NcHICosmo**.

**lnM** : logarithm base e of mass.

**z** : redshift.

**dn\_dlnM\_ptr** : pointer to comoving number density of halos.

**sigma\_ptr** : pointer to the standard deviation of the density contrast.

### nc\_mass\_function\_alpha\_eff ()

```
void                nc_mass_function_alpha_eff      (NcMatterVar *vp,
                                                    NcHICosmo *model,
                                                    gdouble lnM,
                                                    gdouble *a_eff_ptr);
```

FIXME

**vp** : a **NcMatterVar**.

**model** : a **NcHICosmo**.

**lnM** : logarithm base e of mass.

**a\_eff\_ptr** : FIXME

## Property Details

### The "area" property

"area"	gdouble	: Read / Write / Construct
--------	---------	----------------------------

This property sets the angular area in steradian.

Allowed values: [0,12.5664]

Default value: 0.0609235

### The "distance" property

"distance"	NcDistance*	: Read / Write / Construct Only
------------	-------------	---------------------------------

This property keeps the distance object.

### The "growth" property

"growth"	NcGrowthFunc*	: Read / Write / Construct Only
----------	---------------	---------------------------------

This property keeps the growth function object.

### The "multiplicity" property

"multiplicity"	NcMultiplicityFunc*	: Read / Write / Construct Only
----------------	---------------------	---------------------------------

This property keeps the multiplicity function object.



**The "variance" property**

"variance"	NcMatterVar*	: Read / Write / Construct Only
------------	--------------	---------------------------------

This property keeps the matter variance object.

## 7.7 Halo Bias Function Type

### 7.7.1 Halo Bias Function Type

Halo Bias Function Type — FIXME

**Synopsis**

```

struct          NcHaloBiasTypeClass;
struct          NcHaloBiasType;
NcHaloBiasType * nc_halo_bias_type_new_from_name (gchar *bias_name);
gdouble         nc_halo_bias_type_eval          (NcHaloBiasType *biasf,
                                                gdouble sigma,
                                                gdouble z);

void            nc_halo_bias_type_free          (NcHaloBiasType *biasf);
void            nc_halo_bias_type_clear        (NcHaloBiasType **biasf);

```

**Object Hierarchy**

```

GObject
+----NcHaloBiasType
      +----NcHaloBiasTypePS
      +----NcHaloBiasTypeSTEllip
      +----NcHaloBiasTypeSTSpher
      +----NcHaloBiasTypeTinker

```

**Description**

FIXME

**Details****struct NcHaloBiasTypeClass**

```

struct NcHaloBiasTypeClass {
};

```

**struct NcHaloBiasType**

```

struct NcHaloBiasType;

```

**nc\_halo\_bias\_type\_new\_from\_name ()**

```
NcHaloBiasType *      nc_halo_bias_type_new_from_name      (gchar *bias_name);
```

This function returns a new **NcMultiplicityFunc** whose type is defined by *multiplicity\_name*.

**bias\_name** : string which specifies the multiplicity function type.

**Returns** : A new **NcHaloBiasType**.

**nc\_halo\_bias\_type\_eval ()**

```
gdouble              nc_halo_bias_type_eval              (NcHaloBiasType *biasf,
                                                         gdouble sigma,
                                                         gdouble z);
```

FIXME

**biasf** : a **NcHaloBiasType**.

**sigma** : FIXME

**z** : redshift.

**Returns** : FIXME

**nc\_halo\_bias\_type\_free ()**

```
void                  nc_halo_bias_type_free              (NcHaloBiasType *biasf);
```

Atomically decrements the reference count of *biasf* by one. If the reference count drops to 0, all memory allocated by *biasf* is released.

**biasf** : a **NcHaloBiasType**.

**nc\_halo\_bias\_type\_clear ()**

```
void                  nc_halo_bias_type_clear              (NcHaloBiasType **biasf);
```

Atomically decrements the reference count of *biasf* by one. If the reference count drops to 0, all memory allocated by *biasf* is released. Set pointer to NULL.

**biasf** : a **NcHaloBiasType**.

**7.7.2 PS Halo Bias Function Type**

PS Halo Bias Function Type — Press-Schechter FIXME

**Synopsis**

```
struct                NcHaloBiasTypePSClass;
struct                NcHaloBiasTypePS;
NcHaloBiasType *      nc_halo_bias_type_ps_new            (gdouble delta_c);
void                  nc_halo_bias_type_ps_set_delta_c    (NcHaloBiasTypePS *biasf_ps,
                                                         gdouble delta_c);
gdouble               nc_halo_bias_type_ps_get_delta_c    (const NcHaloBiasTypePS *biasf_ps)
```

Object Hierarchy

```
GObject
+----NcHaloBiasType
      +----NcHaloBiasTypePS
```

Properties

"critical-delta"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Description

FIXME

Details

struct NcHaloBiasTypePSClass

```
struct NcHaloBiasTypePSClass {
};
```

struct NcHaloBiasTypePS

```
struct NcHaloBiasTypePS;
```

nc\_halo\_bias\_type\_ps\_new ()

```
NcHaloBiasType * nc_halo_bias_type_ps_new (gdouble delta_c);
```

FIXME

*delta\_c*: FIXME

*Returns*: A new **NcHaloBiasType**.

nc\_halo\_bias\_type\_ps\_set\_delta\_c ()

```
void nc_halo_bias_type_ps_set_delta_c (NcHaloBiasTypePS *biasf_ps,
gdouble delta_c);
```

Sets the value *delta\_c* to the "critical-delta" property.

*biasf\_ps*: a **NcHaloBiasTypePS**.

*delta\_c*: value of "critical-delta".

nc\_halo\_bias\_type\_ps\_get\_delta\_c ()

```
gdouble nc_halo_bias_type_ps_get_delta_c (const NcHaloBiasTypePS *biasf_ps);
```

*biasf\_ps*: a **NcHaloBiasTypePS**.

*Returns*: the value of "critical\_delta" property.

---

Property Details

The "critical-delta" property

"critical-delta"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Critical delta.  
Default value: 1.686

7.7.3 ST Halo Bias Function Type

ST Halo Bias Function Type — Sheth-Tormen Spherical FIXME

Synopsis

```
struct      NcHaloBiasTypeSTSpherClass;
struct      NcHaloBiasTypeSTSpher;
NcHaloBiasType * nc_halo_bias_type_st_spher_new      (gdouble delta_c,
                                                    gdouble a,
                                                    gdouble p);

void        nc_halo_bias_type_st_spher_set_delta_c  (NcHaloBiasTypeSTSpher *biasf_st_s,
                                                    gdouble delta_c);

gdouble     nc_halo_bias_type_st_spher_get_delta_c  (const NcHaloBiasTypeSTSpher *biasf_st_s);

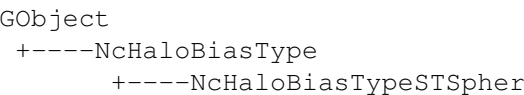
void        nc_halo_bias_type_st_spher_set_a        (NcHaloBiasTypeSTSpher *biasf_st_s,
                                                    gdouble a);

gdouble     nc_halo_bias_type_st_spher_get_a        (const NcHaloBiasTypeSTSpher *biasf_st_s);

void        nc_halo_bias_type_st_spher_set_p        (NcHaloBiasTypeSTSpher *biasf_st_s,
                                                    gdouble p);

gdouble     nc_halo_bias_type_st_spher_get_p        (const NcHaloBiasTypeSTSpher *biasf_st_s);
```

Object Hierarchy



Properties

"a"	gdouble	: Read / Write / Construct Only
"critical-delta"	gdouble	: Read / Write / Construct Only
"p"	gdouble	: Read / Write / Construct Only

Description

FIXME

Details

struct NcHaloBiasTypeSTSpherClass

```
struct NcHaloBiasTypeSTSpherClass {
};
```

**struct NcHaloBiasTypeSTSpher**

```
struct NcHaloBiasTypeSTSpher;
```

**nc\_halo\_bias\_type\_st\_spher\_new ()**

```
NcHaloBiasType * nc_halo_bias_type_st_spher_new (gdouble delta_c,
                                                  gdouble a,
                                                  gdouble p);
```

FIXME

**delta\_c** : FIXME

**a** : FIXME

**p** : FIXME

**Returns** : A new **NcHaloBiasType**.

**nc\_halo\_bias\_type\_st\_spher\_set\_delta\_c ()**

```
void nc_halo_bias_type_st_spher_set_delta_c (NcHaloBiasTypeSTSpher * ↔
                                              biasf_st_spher,
                                              gdouble delta_c);
```

Sets the value *delta\_c* to the "critical-delta" property.

**biasf\_st\_spher** : a **NcHaloBiasTypeSTSpher**.

**delta\_c** : value of "critical-delta".

**nc\_halo\_bias\_type\_st\_spher\_get\_delta\_c ()**

```
gdouble nc_halo_bias_type_st_spher_get_delta_c (const NcHaloBiasTypeSTSpher * ↔
                                                  biasf_st_spher);
```

**biasf\_st\_spher** : a **NcHaloBiasTypeSTSpher**.

**Returns** : the value of "critical\_delta" property.

**nc\_halo\_bias\_type\_st\_spher\_set\_a ()**

```
void nc_halo_bias_type_st_spher_set_a (NcHaloBiasTypeSTSpher * ↔
    biasf_st_spher,
    gdouble a);
```

Sets the value *a* to the "a" property.

**biasf\_st\_spher** : a **NcHaloBiasTypeSTSpher**.

**a** : value of "a".

**nc\_halo\_bias\_type\_st\_spher\_get\_a ()**

```
gdouble nc_halo_bias_type_st_spher_get_a (const NcHaloBiasTypeSTSpher * ←  
    biasf_st_spher);
```

*biasf\_st\_spher* : a **NcHaloBiasTypeSTSpher**.

*Returns* : the value of "a" property.

**nc\_halo\_bias\_type\_st\_spher\_set\_p ()**

```
void nc_halo_bias_type_st_spher_set_p (NcHaloBiasTypeSTSpher * ←  
    biasf_st_spher,  
                                        gdouble p);
```

Sets the value *p* to the "p" property.

*biasf\_st\_spher* : a **NcHaloBiasTypeSTSpher**.

*p* : value of "p".

**nc\_halo\_bias\_type\_st\_spher\_get\_p ()**

```
gdouble nc_halo_bias_type_st_spher_get_p (const NcHaloBiasTypeSTSpher * ←  
    biasf_st_spher);
```

*biasf\_st\_spher* : a **NcHaloBiasTypeSTSpher**.

*Returns* : the value of "p" property.

**Property Details**

**The "a" property**

"a"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME (check limits values)

Default value: 0.75

**The "critical-delta" property**

"critical-delta"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Critical delta.

Default value: 1.686

**The "p" property**

"p"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME (check limits values)

Default value: 0.75

---

### 7.7.4 ST Halo Bias Function Type

ST Halo Bias Function Type — Sheth-Tormen Elliptical FIXME

#### Synopsis

```
struct          NcHaloBiasTypeSEllipClass;
struct          NcHaloBiasTypeSEllip;
NcHaloBiasType * nc_halo_bias_type_st_ellip_new      (gdouble delta_c,
                                                    gdouble a,
                                                    gdouble b,
                                                    gdouble c);

void            nc_halo_bias_type_st_ellip_set_delta_c (NcHaloBiasTypeSEllip *biasf_st_e
                                                    gdouble delta_c);

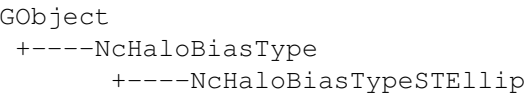
gdouble         nc_halo_bias_type_st_ellip_get_delta_c (const NcHaloBiasTypeSEllip *bias
void            nc_halo_bias_type_st_ellip_set_a      (NcHaloBiasTypeSEllip *biasf_st_e
                                                    gdouble a);

gdouble         nc_halo_bias_type_st_ellip_get_a      (const NcHaloBiasTypeSEllip *bias
void            nc_halo_bias_type_st_ellip_set_b      (NcHaloBiasTypeSEllip *biasf_st_e
                                                    gdouble b);

gdouble         nc_halo_bias_type_st_ellip_get_b      (const NcHaloBiasTypeSEllip *bias
void            nc_halo_bias_type_st_ellip_set_c      (NcHaloBiasTypeSEllip *biasf_st_e
                                                    gdouble c);

gdouble         nc_halo_bias_type_st_ellip_get_c      (const NcHaloBiasTypeSEllip *bias
```

#### Object Hierarchy



#### Properties

"a"	gdouble	: Read / Write / Construct Only
"b"	gdouble	: Read / Write / Construct Only
"c"	gdouble	: Read / Write / Construct Only
"critical-delta"	gdouble	: Read / Write / Construct Only

#### Description

FIXME

#### Details

##### struct NcHaloBiasTypeSEllipClass

```
struct NcHaloBiasTypeSEllipClass {
};
```

**struct NcHaloBiasTypeSEllip**

```
struct NcHaloBiasTypeSEllip;
```

**nc\_halo\_bias\_type\_st\_ellip\_new ()**

```
NcHaloBiasType *      nc_halo_bias_type_st_ellip_new      (gdouble delta_c,
                                                           gdouble a,
                                                           gdouble b,
                                                           gdouble c);
```

FIXME

**delta\_c**: FIXME

**a**: FIXME

**b**: FIXME

**c**: FIXME

**Returns**: A new **NcHaloBiasType**.

**nc\_halo\_bias\_type\_st\_ellip\_set\_delta\_c ()**

```
void                  nc_halo_bias_type_st_ellip_set_delta_c
                                                           (NcHaloBiasTypeSEllip * ←
                                                           biasf_st_ellip,
                                                           gdouble delta_c);
```

Sets the value *delta\_c* to the "critical-delta" property.

**biasf\_st\_ellip**: a **NcHaloBiasTypeSEllip**.

**delta\_c**: value of "critical-delta".

**nc\_halo\_bias\_type\_st\_ellip\_get\_delta\_c ()**

```
gdouble              nc_halo_bias_type_st_ellip_get_delta_c
                                                           (const NcHaloBiasTypeSEllip * ←
                                                           biasf_st_ellip);
```

**biasf\_st\_ellip**: a **NcHaloBiasTypeSEllip**.

**Returns**: the value of "critical\_delta" property.

**nc\_halo\_bias\_type\_st\_ellip\_set\_a ()**

```
void                  nc_halo_bias_type_st_ellip_set_a      (NcHaloBiasTypeSEllip * ←
                                                           biasf_st_ellip,
                                                           gdouble a);
```

Sets the value *a* to the "a" property.

**biasf\_st\_ellip**: a **NcHaloBiasTypeSEllip**.

**a**: value of "a".



**nc\_halo\_bias\_type\_st\_ellip\_get\_a ()**

```
gdouble          nc_halo_bias_type_st_ellip_get_a      (const NcHaloBiasTypeSEllip * ↵
    biasf_st_ellip);
```

**biasf\_st\_ellip**: a **NcHaloBiasTypeSEllip**.

**Returns** : the value of "a" property.

**nc\_halo\_bias\_type\_st\_ellip\_set\_b ()**

```
void          nc_halo_bias_type_st_ellip_set_b      (NcHaloBiasTypeSEllip * ↵
    biasf_st_ellip,
                                                    gdouble b);
```

Sets the value *b* to the "b" property.

**biasf\_st\_ellip**: a **NcHaloBiasTypeSEllip**.

**b**: value of "b".

**nc\_halo\_bias\_type\_st\_ellip\_get\_b ()**

```
gdouble          nc_halo_bias_type_st_ellip_get_b      (const NcHaloBiasTypeSEllip * ↵
    biasf_st_ellip);
```

**biasf\_st\_ellip**: a **NcHaloBiasTypeSEllip**.

**Returns** : the value of "b" property.

**nc\_halo\_bias\_type\_st\_ellip\_set\_c ()**

```
void          nc_halo_bias_type_st_ellip_set_c      (NcHaloBiasTypeSEllip * ↵
    biasf_st_ellip,
                                                    gdouble c);
```

Sets the value *c* to the "c" property.

**biasf\_st\_ellip**: a **NcHaloBiasTypeSEllip**.

**c**: value of "c".

**nc\_halo\_bias\_type\_st\_ellip\_get\_c ()**

```
gdouble          nc_halo_bias_type_st_ellip_get_c      (const NcHaloBiasTypeSEllip * ↵
    biasf_st_ellip);
```

**biasf\_st\_ellip**: a **NcHaloBiasTypeSEllip**.

**Returns** : the value of "c" property.

Property Details

The "a" property

"a"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME (check limits values)  
Default value: 0.707

The "b" property

"b"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME (check limits values)  
Default value: 0.5

The "c" property

"c"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME (check limits values)  
Default value: 0.6

The "critical-delta" property

"critical-delta"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Critical delta.  
Default value: 1.686

7.7.5 Tinker Halo Bias Function Type

Tinker Halo Bias Function Type — Tinker et al. FIXME

Synopsis

```
struct      NcHaloBiasTypeTinkerClass;
struct      NcHaloBiasTypeTinker;
NcHaloBiasType * nc_halo_bias_type_tinker_new      (gdouble delta_c,
                                                    gdouble B,
                                                    gdouble b,
                                                    gdouble c,
                                                    gdouble Delta);

void        nc_halo_bias_type_tinker_set_delta_c  (NcHaloBiasTypeTinker *biasf_tinker,
                                                    gdouble delta_c);

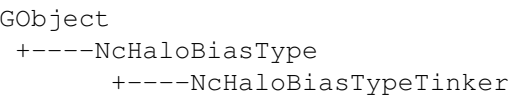
gdouble     nc_halo_bias_type_tinker_get_delta_c  (const NcHaloBiasTypeTinker *biasf_tinker);

void        nc_halo_bias_type_tinker_set_B        (NcHaloBiasTypeTinker *biasf_tinker,
                                                    gdouble B);

gdouble     nc_halo_bias_type_tinker_get_B        (const NcHaloBiasTypeTinker *biasf_tinker);
```

void	nc_halo_bias_type_tinker_set_b	(NcHaloBiasTypeTinker *biasf_tinker, gdouble b);
gdouble	nc_halo_bias_type_tinker_get_b	(const NcHaloBiasTypeTinker *biasf_tinker, gdouble *b);
void	nc_halo_bias_type_tinker_set_c	(NcHaloBiasTypeTinker *biasf_tinker, gdouble c);
gdouble	nc_halo_bias_type_tinker_get_c	(const NcHaloBiasTypeTinker *biasf_tinker, gdouble *c);
void	nc_halo_bias_type_tinker_set_Delta	(NcHaloBiasTypeTinker *biasf_tinker, gdouble Delta);
gdouble	nc_halo_bias_type_tinker_get_Delta	(const NcHaloBiasTypeTinker *biasf_tinker, gdouble *Delta);

Object Hierarchy



Properties

"B"	gdouble	: Read / Write / Construct Only
"Delta"	gdouble	: Read / Write / Construct Only
"b"	gdouble	: Read / Write / Construct Only
"c"	gdouble	: Read / Write / Construct Only
"critical-delta"	gdouble	: Read / Write / Construct Only

Description

FIXME

Details

struct NcHaloBiasTypeTinkerClass

```
struct NcHaloBiasTypeTinkerClass {
};
```

struct NcHaloBiasTypeTinker

```
struct NcHaloBiasTypeTinker;
```

nc\_halo\_bias\_type\_tinker\_new ()

```
NcHaloBiasType * nc_halo_bias_type_tinker_new (gdouble delta_c,
                                                gdouble B,
                                                gdouble b,
                                                gdouble c,
                                                gdouble Delta);
```

FIXME

*delta\_c*: FIXME

*B*: FIXME

***b*** : FIXME

***c*** : FIXME

***Delta*** : FIXME

**Returns** : A new **NcHaloBiasType**.

#### **nc\_halo\_bias\_type\_tinker\_set\_delta\_c ()**

```
void                nc_halo_bias_type_tinker_set_delta_c
                    (NcHaloBiasTypeTinker *biasf_tinker ←
                    ,
                    gdouble delta_c);
```

Sets the value *delta\_c* to the "critical-delta" property.

***biasf\_tinker*** : a **NcHaloBiasTypeTinker**.

***delta\_c*** : value of "critical-delta".

#### **nc\_halo\_bias\_type\_tinker\_get\_delta\_c ()**

```
gdouble            nc_halo_bias_type_tinker_get_delta_c
                    (const NcHaloBiasTypeTinker * ←
                    biasf_tinker);
```

***biasf\_tinker*** : a **NcHaloBiasTypeTinker**.

**Returns** : the value of "critical\_delta" property.

#### **nc\_halo\_bias\_type\_tinker\_set\_B ()**

```
void                nc_halo_bias_type_tinker_set_B      (NcHaloBiasTypeTinker *biasf_tinker ←
                    ,
                    gdouble B);
```

Sets the value *B* to the "B" property.

***biasf\_tinker*** : a **NcHaloBiasTypeTinker**.

***B*** : value of "B".

#### **nc\_halo\_bias\_type\_tinker\_get\_B ()**

```
gdouble            nc_halo_bias_type_tinker_get_B      (const NcHaloBiasTypeTinker * ←
                    biasf_tinker);
```

***biasf\_tinker*** : a **NcHaloBiasTypeTinker**.

**Returns** : the value of "B" property.

**nc\_halo\_bias\_type\_tinker\_set\_b ()**

```
void          nc_halo_bias_type_tinker_set_b      (NcHaloBiasTypeTinker *biasf_tinker ←
    ,
                                                    gdouble b);
```

Sets the value  $b$  to the "b" property.

**biasf\_tinker** : a **NcHaloBiasTypeTinker**.

**b** : value of "b".

**nc\_halo\_bias\_type\_tinker\_get\_b ()**

```
gdouble          nc_halo_bias_type_tinker_get_b      (const NcHaloBiasTypeTinker * ←
    biasf_tinker);
```

**biasf\_tinker** : a **NcHaloBiasTypeTinker**.

**Returns** : the value of "b" property.

**nc\_halo\_bias\_type\_tinker\_set\_c ()**

```
void          nc_halo_bias_type_tinker_set_c      (NcHaloBiasTypeTinker *biasf_tinker ←
    ,
                                                    gdouble c);
```

Sets the value  $c$  to the "c" property.

**biasf\_tinker** : a **NcHaloBiasTypeTinker**.

**c** : value of "c".

**nc\_halo\_bias\_type\_tinker\_get\_c ()**

```
gdouble          nc_halo_bias_type_tinker_get_c      (const NcHaloBiasTypeTinker * ←
    biasf_tinker);
```

**biasf\_tinker** : a **NcHaloBiasTypeTinker**.

**Returns** : the value of "c" property.

**nc\_halo\_bias\_type\_tinker\_set\_Delta ()**

```
void          nc_halo_bias_type_tinker_set_Delta   (NcHaloBiasTypeTinker *biasf_tinker ←
    ,
                                                    gdouble Delta);
```

Sets the value  $\Delta$  to the "Delta" property.

**biasf\_tinker** : a **NcHaloBiasTypeTinker**.

**Delta** : value of "Delta".

**nc\_halo\_bias\_type\_tinker\_get\_Delta ()**

```
gdouble nc_halo_bias_type_tinker_get_Delta (const NcHaloBiasTypeTinker * ←
    biasf_tinker);
```

***biasf\_tinker*** : a **NcHaloBiasTypeTinker**.

***Returns*** : the value of **"Delta"** property.

**Property Details**

**The "B" property**

"B"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME (check limits values)

Default value: 0.183

**The "Delta" property**

"Delta"	gdouble	: Read / Write / Construct Only
---------	---------	---------------------------------

FIXME (check limits values)

Default value: 200

**The "b" property**

"b"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME (check limits values)

Default value: 1.5

**The "c" property**

"c"	gdouble	: Read / Write / Construct Only
-----	---------	---------------------------------

FIXME (check limits values)

Default value: 2.4

**The "critical-delta" property**

"critical-delta"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Critical delta.

Default value: 1.686

**7.8 Mean Halo Bias Function**

Mean Halo Bias Function — FIXME

## Synopsis

```

struct          NcHaloBiasFuncClass;
struct          NcHaloBiasFunc;
NcHaloBiasFunc * nc_halo_bias_func_new          (NcMassFunction *mfp,
                                                NcHaloBiasType *biasf);
NcHaloBiasFunc * nc_halo_bias_func_copy        (NcHaloBiasFunc *mbiasf);
void            nc_halo_bias_func_free         (NcHaloBiasFunc *mbiasf);
void            nc_halo_bias_func_clear        (NcHaloBiasFunc **mbiasf);
gdouble         nc_halo_bias_func_integrand    (NcHaloBiasFunc *mbiasf,
                                                NcHICosmo *model,
                                                gdouble lnM,
                                                gdouble z);

```

## Object Hierarchy

```

GObject
+----NcHaloBiasFunc

```

## Properties

"bias-type"	NcHaloBiasType*	: Read / Write / Construct Only
"mass-function"	NcMassFunction*	: Read / Write / Construct Only

## Description

FIXME

## Details

### struct NcHaloBiasFuncClass

```

struct NcHaloBiasFuncClass {
};

```

### struct NcHaloBiasFunc

```

struct NcHaloBiasFunc;

```

### nc\_halo\_bias\_func\_new ()

```

NcHaloBiasFunc * nc_halo_bias_func_new          (NcMassFunction *mfp,
                                                NcHaloBiasType *biasf);

```

This function allocates memory for a new **NcHaloBiasFunc** object and sets its properties to the values from the input arguments.

**mfp** : a **NcMassFunction**.

**biasf** : a **NcHaloBiasType**. *[allow-none]*

**Returns** : A new **NcHaloBiasFunc**.

**nc\_halo\_bias\_func\_copy ()**

```
NcHaloBiasFunc * nc_halo_bias_func_copy (NcHaloBiasFunc *mbiasf);
```

Duplicates the **NcHaloBiasFunc** object setting the same values of the original properties.

**mbiasf** : a **NcHaloBiasFunc**.

**Returns** : A new **NcHaloBiasFunc**. *[transfer full]*

**nc\_halo\_bias\_func\_free ()**

```
void nc_halo_bias_func_free (NcHaloBiasFunc *mbiasf);
```

Atomically decrements the reference count of *mbiasf* by one. If the reference count drops to 0, all memory allocated by *mbiasf* is released.

**mbiasf** : a **NcHaloBiasFunc**.

**nc\_halo\_bias\_func\_clear ()**

```
void nc_halo_bias_func_clear (NcHaloBiasFunc **mbiasf);
```

Atomically decrements the reference count of *mbiasf* by one. If the reference count drops to 0, all memory allocated by *mbiasf* is released. Set pointer to NULL.

**mbiasf** : a **NcHaloBiasFunc**.

**nc\_halo\_bias\_func\_integrand ()**

```
gdouble nc_halo_bias_func_integrand (NcHaloBiasFunc *mbiasf,
                                     NcHICosmo *model,
                                     gdouble lnM,
                                     gdouble z);
```

This function is the integrand of the mean bias, i.e., the product of the mass function with the bias function. As both functions depend on the standard deviation of the matter density contrast, we implement this function to compute  $\sigma(M, z)$  just once.

It is worth noting that the multiplicity function must be compatible with the bias function.

**mbiasf** : a **NcHaloBiasFunc**.

**model** : a **NcHICosmo**.

**lnM** : logarithm base e of the mass.

**z** : redshift.

**Returns** : a double which corresponds to the mean bias integrand for lnM and at redshift z.

**Property Details****The "bias-type" property**

"bias-type"	NcHaloBiasType*	: Read / Write / Construct Only
-------------	-----------------	---------------------------------

Bias Function Type.



The "mass-function" property

"mass-function"	NcMassFunction*	: Read / Write / Construct Only
-----------------	-----------------	---------------------------------

This property keeps the mass function object.

7.9 Galaxy Angular Corelation Function

Galaxy Angular Corelation Function — FIXME

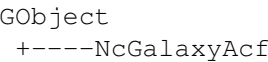
Synopsis

```
struct          NcGalaxyAcfClass;
struct          NcGalaxyAcf;
NcGalaxyAcf *   nc_galaxy_acf_new          (NcGrowthFunc *gf,
                                             NcDistance *dist,
                                             NcTransferFunc *tf);

gdouble         nc_galaxy_acf_psi          (NcGalaxyAcf *acf,
                                             NcHICosmo *model,
                                             gdouble k,
                                             guint l);

void            nc_galaxy_acf_prepare_psi  (NcGalaxyAcf *acf,
                                             NcHICosmo *model,
                                             guint l);
```

Object Hierarchy



Description

FIXME

Details

struct NcGalaxyAcfClass

```
struct NcGalaxyAcfClass {
};
```

struct NcGalaxyAcf

```
struct NcGalaxyAcf;
```

nc\_galaxy\_acf\_new ()

```
NcGalaxyAcf *   nc_galaxy_acf_new          (NcGrowthFunc *gf,
                                             NcDistance *dist,
                                             NcTransferFunc *tf);
```

## nc\_galaxy\_acf\_psi ()

```

gdouble          nc_galaxy_acf_psi          (NcGalaxyAcf *acf,
                                              NcHICosmo *model,
                                              gdouble k,
                                              quint l);

```

### nc\_galaxy\_acf\_prepare\_psi ()

```
void nc_galaxy_acf_prepare_psi(NcGalaxyAcf *acf,
                               NcHICosmo *model,
                               quint l);
```

## 7.10 Cluster Redshift

### 7.10.1 Abstract Cluster Redshift Object

Abstract Cluster Redshift Object — Observed redshift distribution

## Synopsis

[illegible]

## Object Hierarchy

```
GObject
+----NcClusterRedshift
      +----NcClusterPhotozGauss
      +----NcClusterPhotozGaussGlobal
      +----NcClusterRedshiftNodist
```

## Description

FIXME

## Details

### enum NcClusterRedshiftImpl

```
typedef enum {
    NC_CLUSTER_REDSHIFT_P          = 1 << 0,
    NC_CLUSTER_REDSHIFT_INTP       = 1 << 1,
    NC_CLUSTER_REDSHIFT_RESAMPLE   = 1 << 2,
    NC_CLUSTER_REDSHIFT_P_LIMITS   = 1 << 3,
    NC_CLUSTER_REDSHIFT_N_LIMITS   = 1 << 4,
} NcClusterRedshiftImpl;
```

**NC\_CLUSTER\_REDSHIFT\_P** FIXME

**NC\_CLUSTER\_REDSHIFT\_INTP** FIXME

**NC\_CLUSTER\_REDSHIFT\_RESAMPLE** FIXME

**NC\_CLUSTER\_REDSHIFT\_P\_LIMITS** FIXME

**NC\_CLUSTER\_REDSHIFT\_N\_LIMITS** FIXME

### NC\_CLUSTER\_REDSHIFT\_IMPL\_ALL

```
#define NC_CLUSTER_REDSHIFT_IMPL_ALL (~0)
```

### struct NcClusterRedshiftClass

```
struct NcClusterRedshiftClass {
};
```

### struct NcClusterRedshift

```
struct NcClusterRedshift;
```

### nc\_cluster\_redshift\_new\_from\_name ()

```
NcClusterRedshift * nc_cluster_redshift_new_from_name (gchar *redshift_name);
```

This function returns a new **NcClusterRedshift** whose type is defined by *redshift\_name*.

**redshift\_name** : string which specifies the type of the redshift distribution.

**Returns** : A new **NcClusterRedshift**.

**nc\_cluster\_redshift\_ref ()**

```
NcClusterRedshift * nc_cluster_redshift_ref (NcClusterRedshift *clusterz);
```

FIXME

**clusterz** : FIXME.

**Returns** : *clusterz*. *[transfer full]*

**nc\_cluster\_redshift\_free ()**

```
void nc_cluster_redshift_free (NcClusterRedshift *clusterz);
```

Atomically decrements the reference count of *clusterz* by one. If the reference count drops to 0, all memory allocated by *clusterz* is released.

**clusterz** : a [NcClusterRedshift](#).

**nc\_cluster\_redshift\_clear ()**

```
void nc_cluster_redshift_clear (NcClusterRedshift **clusterz);
```

Atomically decrements the reference count of *clusterz* by one. If the reference count drops to 0, all memory allocated by *clusterz* is released. Set pointer to NULL.

**clusterz** : a [NcClusterRedshift](#).

**nc\_cluster\_redshift\_impl ()**

```
NcClusterRedshiftImpl nc_cluster_redshift_impl (NcClusterRedshift *clusterz);
```

FIXME

**clusterz** : FIXME.

**Returns** : FIXME

**nc\_cluster\_redshift\_obs\_len ()**

```
quint nc_cluster_redshift_obs_len (NcClusterRedshift *clusterz);
```

FIXME

**clusterz** : FIXME.

**Returns** : FIXME

**nc\_cluster\_redshift\_obs\_params\_len ()**

```
quint nc_cluster_redshift_obs_params_len (NcClusterRedshift *clusterz);
```

FIXME

**clusterz** : FIXME.

**Returns** : FIXME

**nc\_cluster\_redshift\_p ()**

gdouble	nc_cluster_redshift_p	(NcClusterRedshift *clusterz, gdouble lnM, gdouble z, gdouble *z_obs, gdouble *z_obs_params);
---------	-----------------------	---

FIXME

**clusterz** : a **NcClusterRedshift**.

**z** : true redshift.

**lnM** : true mass.

**z\_obs** : observed redshift.

**z\_obs\_params** : observed redshift params.

**Returns** : FIXME

**nc\_cluster\_redshift\_intp ()**

gdouble	nc_cluster_redshift_intp	(NcClusterRedshift *clusterz, gdouble lnM, gdouble z);
---------	--------------------------	--

FIXME

**clusterz** : a **NcClusterRedshift**.

**z** : true redshift.

**lnM** : true mass.

**Returns** : FIXME

**nc\_cluster\_redshift\_resample ()**

gboolean	nc_cluster_redshift_resample	(NcClusterRedshift *clusterz, gdouble lnM, gdouble z, gdouble *z_obs, gdouble *z_obs_params);
----------	------------------------------	---

FIXME The function which will call this one is responsible to allocate enough memory for *z\_lower* and *z\_upper*.

**clusterz** : a **NcClusterRedshift**.

**z** : true redshift.

**lnM** : true mass.

**z\_obs** : observed redshift. *[out]*

**z\_obs\_params** : observed redshift params. *[out]*

**Returns** : FIXME

**nc\_cluster\_redshift\_p\_limits ()**

```
void                nc_cluster_redshift_p_limits      (NcClusterRedshift *clusterz,
                                                       gdouble *z_obs,
                                                       gdouble *z_obs_params,
                                                       gdouble *z_lower,
                                                       gdouble *z_upper);
```

FIXME The function which will call this one is responsible to allocate memory for *z\_lower* and *z\_upper*.

**clusterz** : a **NcClusterRedshift**.

**z\_obs** : observed redshift.

**z\_obs\_params** : observed redshift params.

**z\_lower** : pointer to the lower limit of the true redshift integration. *[out]*

**z\_upper** : pointer to the upper limit of the true redshift integration. *[out]*

**nc\_cluster\_redshift\_n\_limits ()**

```
void                nc_cluster_redshift_n_limits      (NcClusterRedshift *clusterz,
                                                       gdouble *z_lower,
                                                       gdouble *z_upper);
```

FIXME The function which will call this one is responsible to allocate memory for *z\_lower* and *z\_upper*.

**clusterz** : a **NcClusterRedshift**.

**z\_lower** : pointer to the lower limit of the true redshift. *[out]*

**z\_upper** : pointer to the upper limit of the true redshift. *[out]*

**7.10.2 Cluster Abundance Redshift No Distribution**

Cluster Abundance Redshift No Distribution — FIXME

**Synopsis**

```
struct              NcClusterRedshiftNodistClass;
struct              NcClusterRedshiftNodist;
```

**Object Hierarchy**

```
GObject
+----NcClusterRedshift
      +----NcClusterRedshiftNodist
```

**Properties**

"z-max"	gdouble	: Read / Write / Construct Only
"z-min"	gdouble	: Read / Write / Construct Only

Description

FIXME

Details

struct NcClusterRedshiftNodistClass

```
struct NcClusterRedshiftNodistClass {  
};
```

struct NcClusterRedshiftNodist

```
struct NcClusterRedshiftNodist;
```

Property Details

The "z-max" property

"z-max"	gdouble	: Read / Write / Construct Only
---------	---------	---------------------------------

Maximum z.

Allowed values:  $\geq 0$

Default value: 1

The "z-min" property

"z-min"	gdouble	: Read / Write / Construct Only
---------	---------	---------------------------------

Minimum z.

Allowed values:  $\geq 0$

Default value: 0

7.10.3 Individual Gaussian Photoz Cluster

Individual Gaussian Photoz Cluster — Gaussian photometric redshift

Synopsis

```
struct NcClusterPhotozGaussClass;  
struct NcClusterPhotozGauss;  
NcClusterRedshift * nc_cluster_photoz_gauss_new      ();  
#define NC_CLUSTER_PHOTOZ_GAUSS_BIAS  
#define NC_CLUSTER_PHOTOZ_GAUSS_SIGMA
```

Object Hierarchy

```
GObject  
+----NcClusterRedshift  
      +----NcClusterPhotozGauss
```

Properties

"pz-max"	gdouble	: Read / Write / Construct Only
"pz-min"	gdouble	: Read / Write / Construct Only

Description

FIXME

Details

struct NcClusterPhotozGaussClass

```
struct NcClusterPhotozGaussClass {  
};
```

struct NcClusterPhotozGauss

```
struct NcClusterPhotozGauss;
```

nc\_cluster\_photoz\_gauss\_new ()

```
NcClusterRedshift * nc_cluster_photoz_gauss_new      ();
```

FIXME

*Returns* : A new **NcClusterRedshift**.

NC\_CLUSTER\_PHOTOZ\_GAUSS\_BIAS

```
#define NC_CLUSTER_PHOTOZ_GAUSS_BIAS  (0)
```

NC\_CLUSTER\_PHOTOZ\_GAUSS\_SIGMA

```
#define NC_CLUSTER_PHOTOZ_GAUSS_SIGMA  (1)
```

Property Details

The "pz-max" property

"pz-max"	gdouble	: Read / Write / Construct Only
----------	---------	---------------------------------

Maximum photoz.

Allowed values: >= 0

Default value: 0



The "pz-min" property

"pz-min"	gdouble	: Read / Write / Construct Only
----------	---------	---------------------------------

Minimum photoz.

Allowed values:  $\geq 0$

Default value: 0

7.10.4 Global Gaussian Photoz Cluster

Global Gaussian Photoz Cluster — Gaussian photometric redshift

Synopsis

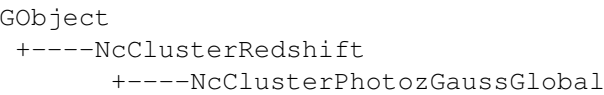
```
struct          NcClusterPhotozGaussGlobalClass;
struct          NcClusterPhotozGaussGlobal;
NcClusterRedshift * nc_cluster_photoz_gauss_global_new  (gdouble pz_min,
                                                         gdouble pz_max,
                                                         gdouble z_bias,
                                                         gdouble sigma0);

void            nc_cluster_photoz_gauss_global_set_z_bias
                                                         (NcClusterPhotozGaussGlobal *pzg_g
                                                         gdouble z_bias);

gdouble         nc_cluster_photoz_gauss_global_get_z_bias
                                                         (const NcClusterPhotozGaussGlobal
void           nc_cluster_photoz_gauss_global_set_sigma0
                                                         (NcClusterPhotozGaussGlobal *pzg_g
                                                         gdouble sigma0);

gdouble         nc_cluster_photoz_gauss_global_get_sigma0
                                                         (const NcClusterPhotozGaussGlobal
```

Object Hierarchy



Properties

"pz-max"	gdouble	: Read / Write / Construct Only
"pz-min"	gdouble	: Read / Write / Construct Only
"sigma0"	gdouble	: Read / Write / Construct Only
"z-bias"	gdouble	: Read / Write / Construct Only

Description

FIXME

Details

struct NcClusterPhotozGaussGlobalClass

```
struct NcClusterPhotozGaussGlobalClass {
};
```

**struct NcClusterPhotozGaussGlobal**

```
struct NcClusterPhotozGaussGlobal;
```

**nc\_cluster\_photoz\_gauss\_global\_new ()**

```
NcClusterRedshift * nc_cluster_photoz_gauss_global_new (gdouble pz_min,
                                                         gdouble pz_max,
                                                         gdouble z_bias,
                                                         gdouble sigma0);
```

FIXME

**pz\_min**: FIXME

**pz\_max**: FIXME

**z\_bias**: FIXME

**sigma0**: FIXME

**Returns**: A new **NcClusterRedshift**.

**nc\_cluster\_photoz\_gauss\_global\_set\_z\_bias ()**

```
void nc_cluster_photoz_gauss_global_set_z_bias
(NcClusterPhotozGaussGlobal * ←
 pzg_global,
 gdouble z_bias);
```

Sets the value *z\_bias* to the "z-bias" property.

**pzg\_global**: a **NcClusterPhotozGaussGlobal**.

**z\_bias**: value of "z-bias".

**nc\_cluster\_photoz\_gauss\_global\_get\_z\_bias ()**

```
gdouble nc_cluster_photoz_gauss_global_get_z_bias
(const NcClusterPhotozGaussGlobal * ←
 pzg_global);
```

**pzg\_global**: a **NcClusterPhotozGaussGlobal**.

**Returns**: the value of "z-bias" property.

**nc\_cluster\_photoz\_gauss\_global\_set\_sigma0 ()**

```
void nc_cluster_photoz_gauss_global_set_sigma0
(NcClusterPhotozGaussGlobal * ←
 pzg_global,
 gdouble sigma0);
```

Sets the value *sigma0* to the "sigma0" property.

**pzg\_global**: a **NcClusterPhotozGaussGlobal**.

**sigma0**: value of "sigma0".

**nc\_cluster\_photoz\_gauss\_global\_get\_sigma0 ()**

```
gdouble          nc_cluster_photoz_gauss_global_get_sigma0
                                     (const NcClusterPhotozGaussGlobal * ←
                                     pzg_global);
```

*pzg\_global* : a **NcClusterPhotozGaussGlobal**.

**Returns** : the value of **"sigma0"** property.

**Property Details**

**The "pz-max" property**

"pz-max"	gdouble	: Read / Write / Construct Only
----------	---------	---------------------------------

Maximum photoz.

Allowed values:  $\geq 0$

Default value: 0

**The "pz-min" property**

"pz-min"	gdouble	: Read / Write / Construct Only
----------	---------	---------------------------------

Minimum photoz.

Allowed values:  $\geq 0$

Default value: 0

**The "sigma0" property**

"sigma0"	gdouble	: Read / Write / Construct Only
----------	---------	---------------------------------

FIXME Set correct values (limits)

Allowed values:  $\geq 0$

Default value: 0.03

**The "z-bias" property**

"z-bias"	gdouble	: Read / Write / Construct Only
----------	---------	---------------------------------

z-bias.

Default value: 0

**7.11 Cluster Mass**

**7.11.1 Cluster Mass Distribution**

Cluster Mass Distribution — FIXME

**Synopsis**

```

enum                NcClusterMassImpl;
#define              NC_CLUSTER_MASS_IMPL_ALL
struct               NcClusterMassClass;
struct               NcClusterMass;
NcClusterMass *      nc_cluster_mass_new_from_name      (gchar *mass_name);
NcClusterMass *      nc_cluster_mass_ref               (NcClusterMass *clusterm);
void                 nc_cluster_mass_free              (NcClusterMass *clusterm);
void                 nc_cluster_mass_clear             (NcClusterMass **clusterm);
NcClusterMassImpl    nc_cluster_mass_impl              (NcClusterMass *clusterm);
guint                nc_cluster_mass_obs_len           (NcClusterMass *clusterm);
guint                nc_cluster_mass_obs_params_len    (NcClusterMass *clusterm);
gdouble              nc_cluster_mass_p                (NcClusterMass *clusterm,
                                                       NCHICosmo *model,
                                                       gdouble lnM,
                                                       gdouble z,
                                                       gdouble *lnM_obs,
                                                       gdouble *lnM_obs_params);
gdouble              nc_cluster_mass_intp              (NcClusterMass *clusterm,
                                                       NCHICosmo *model,
                                                       gdouble lnM,
                                                       gdouble z);
gboolean             nc_cluster_mass_resample          (NcClusterMass *clusterm,
                                                       NCHICosmo *model,
                                                       gdouble lnM,
                                                       gdouble z,
                                                       gdouble *lnM_obs,
                                                       gdouble *lnM_obs_params);
void                 nc_cluster_mass_p_limits          (NcClusterMass *clusterm,
                                                       NCHICosmo *model,
                                                       gdouble *lnM_obs,
                                                       gdouble *lnM_obs_params,
                                                       gdouble *lnM_lower,
                                                       gdouble *lnM_upper);
void                 nc_cluster_mass_n_limits          (NcClusterMass *clusterm,
                                                       NCHICosmo *model,
                                                       gdouble *lnM_lower,
                                                       gdouble *lnM_upper);
void                 nc_cluster_mass_log_all_models    ();

```

**Object Hierarchy**

```

GObject
+----NcmModel
      +----NcClusterMass
            +----NcClusterMassBenson
            +----NcClusterMassLnnormal
            +----NcClusterMassNodist
            +----NcClusterMassVanderlinde

```

**Description**

FIXME

**Details****enum NcClusterMassImpl**

```
typedef enum {
    NC_CLUSTER_MASS_P          = 1 << 0,
    NC_CLUSTER_MASS_INTP       = 1 << 1,
    NC_CLUSTER_MASS_RESAMPLE    = 1 << 2,
    NC_CLUSTER_MASS_P_LIMITS   = 1 << 3,
    NC_CLUSTER_MASS_N_LIMITS   = 1 << 4,
} NcClusterMassImpl;
```

**NC\_CLUSTER\_MASS\_P** FIXME**NC\_CLUSTER\_MASS\_INTP** FIXME**NC\_CLUSTER\_MASS\_RESAMPLE** FIXME**NC\_CLUSTER\_MASS\_P\_LIMITS** FIXME**NC\_CLUSTER\_MASS\_N\_LIMITS** FIXME**NC\_CLUSTER\_MASS\_IMPL\_ALL**

```
#define NC_CLUSTER_MASS_IMPL_ALL (~0)
```

**struct NcClusterMassClass**

```
struct NcClusterMassClass {
};
```

**struct NcClusterMass**

```
struct NcClusterMass;
```

**nc\_cluster\_mass\_new\_from\_name ()**

```
NcClusterMass *      nc_cluster_mass_new_from_name      (gchar *mass_name);
```

This function returns a new **NcClusterMass** whose type is defined by *mass\_name*.

**mass\_name** : string which specifies the type of the mass distribution.

**Returns** : A new **NcClusterMass**.

**nc\_cluster\_mass\_ref ()**

```
NcClusterMass *      nc_cluster_mass_ref                (NcClusterMass *clusterm);
```

FIXME

**clusterm** : a **NcClusterMass**.

**Returns** : *clusterm*. [transfer full]

**nc\_cluster\_mass\_free ()**

```
void nc_cluster_mass_free (NcClusterMass *clusterterm);
```

FIXME

**clusterterm** : a **NcClusterMass**.**nc\_cluster\_mass\_clear ()**

```
void nc_cluster_mass_clear (NcClusterMass **clusterterm);
```

FIXME

**clusterterm** : a **NcClusterMass**.**nc\_cluster\_mass\_impl ()**

```
NcClusterMassImpl nc_cluster_mass_impl (NcClusterMass *clusterterm);
```

FIXME

**clusterterm** : a **NcClusterMass**.**Returns** : FIXME**nc\_cluster\_mass\_obs\_len ()**

```
quint nc_cluster_mass_obs_len (NcClusterMass *clusterterm);
```

FIXME

**clusterterm** : a **NcClusterMass**.**Returns** : FIXME**nc\_cluster\_mass\_obs\_params\_len ()**

```
quint nc_cluster_mass_obs_params_len (NcClusterMass *clusterterm);
```

FIXME

**clusterterm** : a **NcClusterMass**.**Returns** : FIXME

**nc\_cluster\_mass\_p ()**

gdouble	nc_cluster_mass_p	(NcClusterMass *clusterm, NcHICosmo *model, gdouble lnM, gdouble z, gdouble *lnM_obs, gdouble *lnM_obs_params);
---------	-------------------	--

FIXME

**clusterm** : a **NcClusterMass**.

**model** : a **NcHICosmo**.

**z** : true redshift.

**lnM** : logarithm base e of the true mass.

**lnM\_obs** : logarithm base e of the observed mass.

**lnM\_obs\_params** : observed mass params.

**Returns** : FIXME

**nc\_cluster\_mass\_intp ()**

gdouble	nc_cluster_mass_intp	(NcClusterMass *clusterm, NcHICosmo *model, gdouble lnM, gdouble z);
---------	----------------------	---

FIXME

**clusterm** : a **NcClusterMass**.

**model** : a **NcHICosmo**.

**z** : true redshift.

**lnM** : logarithm base e of the true mass.

**Returns** : FIXME

**nc\_cluster\_mass\_resample ()**

gboolean	nc_cluster_mass_resample	(NcClusterMass *clusterm, NcHICosmo *model, gdouble lnM, gdouble z, gdouble *lnM_obs, gdouble *lnM_obs_params);
----------	--------------------------	--

FIXME

**clusterm** : a **NcClusterMass**.

**model** : a **NcHICosmo**.

**z** : true redshift.

***lnM*** : logarithm base e of the true mass.

***lnM\_obs*** : logarithm base e of the observed mass. *[out]*

***lnM\_obs\_params*** : observed mass params. *[out]*

**Returns** : FIXME

#### **nc\_cluster\_mass\_p\_limits ()**

```
void                nc_cluster_mass_p_limits      (NcClusterMass *clusterm,
                                                  NcHICosmo *model,
                                                  gdouble *lnM_obs,
                                                  gdouble *lnM_obs_params,
                                                  gdouble *lnM_lower,
                                                  gdouble *lnM_upper);
```

FIXME The function which will call this one is responsible to allocate memory for *lnM\_lower* and *lnM\_upper*.

***clusterm*** : a **NcClusterMass**.

***model*** : a **NcHICosmo**.

***lnM\_obs*** : observed mass.

***lnM\_obs\_params*** : observed mass params.

***lnM\_lower*** : pointer to the lower limit of the real mass integration. *[out]*

***lnM\_upper*** : pointer to the upper limit of the real mass integration. *[out]*

#### **nc\_cluster\_mass\_n\_limits ()**

```
void                nc_cluster_mass_n_limits      (NcClusterMass *clusterm,
                                                  NcHICosmo *model,
                                                  gdouble *lnM_lower,
                                                  gdouble *lnM_upper);
```

FIXME The function which will call this one is responsible to allocate memory for *lnM\_lower* and *lnM\_upper*.

***clusterm*** : a **NcClusterMass**.

***model*** : a **NcHICosmo**.

***lnM\_lower*** : lower limit of the logarithm base e of the true mass. *[out]*

***lnM\_upper*** : upper limit of the logarithm base e of the true mass. *[out]*

#### **nc\_cluster\_mass\_log\_all\_models ()**

```
void                nc_cluster_mass_log_all_models      ();
```

FIXME

### **7.11.2 Cluster Mass No Distribution**

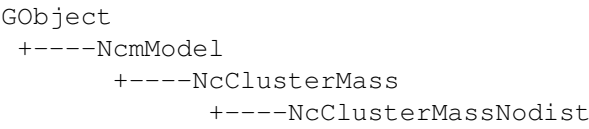
Cluster Mass No Distribution — FIXME



Synopsis

```
struct          NcClusterMassNodistClass;
struct          NcClusterMassNodist;
```

Object Hierarchy



Properties

"lnM-max"	gdouble	: Read / Write / Construct Only
"lnM-min"	gdouble	: Read / Write / Construct Only

Description

FIXME

Details

struct NcClusterMassNodistClass

```
struct NcClusterMassNodistClass {
};
```

struct NcClusterMassNodist

```
struct NcClusterMassNodist;
```

Property Details

The "lnM-max" property

"lnM-max"	gdouble	: Read / Write / Construct Only
-----------	---------	---------------------------------

Maximum mass.  
Allowed values: >= 25.3284  
Default value: 36.8414

The "lnM-min" property

"lnM-min"	gdouble	: Read / Write / Construct Only
-----------	---------	---------------------------------

Minimum mass.  
Allowed values: >= 25.3284  
Default value: 31.543

---

### 7.11.3 Cluster Mass Ln Normal Distribution

Cluster Mass Ln Normal Distribution — FIXME

#### Synopsis

```
struct          NcClusterMassLnnormalClass;
struct          NcClusterMassLnnormal;
#define         NC_CLUSTER_MASS_LNNORMAL_BIAS
#define         NC_CLUSTER_MASS_LNNORMAL_SIGMA
```

#### Object Hierarchy

```
GObject
+----NcmModel
      +----NcClusterMass
            +----NcClusterMassLnnormal
```

#### Properties

"lnMobs-max"	gdouble	: Read / Write / Construct Only
"lnMobs-min"	gdouble	: Read / Write / Construct Only

#### Description

FIXME

#### Details

##### struct NcClusterMassLnnormalClass

```
struct NcClusterMassLnnormalClass {
};
```

##### struct NcClusterMassLnnormal

```
struct NcClusterMassLnnormal;
```

##### NC\_CLUSTER\_MASS\_LNNORMAL\_BIAS

```
#define NC_CLUSTER_MASS_LNNORMAL_BIAS (0)
```

##### NC\_CLUSTER\_MASS\_LNNORMAL\_SIGMA

```
#define NC_CLUSTER_MASS_LNNORMAL_SIGMA (1)
```

## Property Details

### The "lnMobs-max" property

"lnMobs-max"	gdouble	: Read / Write / Construct Only
--------------	---------	---------------------------------

Maximum LnMobs.

Allowed values:  $\geq 25.3284$

Default value: 36.8414

### The "lnMobs-min" property

"lnMobs-min"	gdouble	: Read / Write / Construct Only
--------------	---------	---------------------------------

Minimum LnMobs.

Allowed values:  $\geq 25.3284$

Default value: 31.543

## 7.11.4 SZ Cluster Mass Distribution

SZ Cluster Mass Distribution — Sunyaev-Zel'dovich FIXME

### Synopsis

```
enum                NcClusterMassVanderlindeParams;
#define              NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_A_SZ
#define              NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_B_SZ
#define              NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_C_SZ
#define              NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_D_SZ
#define              NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_PARAMS_ABSTOL
struct               NcClusterMassVanderlindeClass;
struct               NcClusterMassVanderlinde;
```

### Object Hierarchy

```
GObject
+----NcmModel
      +----NcClusterMass
            +----NcClusterMassVanderlinde
```

### Properties

"Asz"	gdouble	: Read / Write
"Asz-fit"	gboolean	: Read / Write
"Bsz"	gdouble	: Read / Write
"Bsz-fit"	gboolean	: Read / Write
"Csz"	gdouble	: Read / Write
"Csz-fit"	gboolean	: Read / Write
"Dsz"	gdouble	: Read / Write
"Dsz-fit"	gboolean	: Read / Write
"M0"	gdouble	: Read / Write / Construct Only
"signif-obs-max"	gdouble	: Read / Write / Construct Only
"signif-obs-min"	gdouble	: Read / Write / Construct Only
"z0"	gdouble	: Read / Write / Construct Only

## Description

FIXME

## Details

### enum NcClusterMassVanderlindeParams

```
typedef enum {  
    NC_CLUSTER_MASS_VANDERLINDE_A_SZ = 0,  
    NC_CLUSTER_MASS_VANDERLINDE_B_SZ,  
    NC_CLUSTER_MASS_VANDERLINDE_C_SZ,  
} NcClusterMassVanderlindeParams;
```

FIXME

**NC\_CLUSTER\_MASS\_VANDERLINDE\_A\_SZ** normalization of the mass-observable relation

**NC\_CLUSTER\_MASS\_VANDERLINDE\_B\_SZ** FIXME

**NC\_CLUSTER\_MASS\_VANDERLINDE\_C\_SZ** FIXME

**NC\_CLUSTER\_MASS\_VANDERLINDE\_D\_SZ** standard deviation of the mass-observable relation

### NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_A\_SZ

```
#define NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_A_SZ (6.01)
```

### NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_B\_SZ

```
#define NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_B_SZ (1.31)
```

### NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_C\_SZ

```
#define NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_C_SZ (1.6)
```

### NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_D\_SZ

```
#define NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_D_SZ (0.21)
```

### NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_PARAMS\_ABSTOL

```
#define NC_CLUSTER_MASS_VANDERLINDE_DEFAULT_PARAMS_ABSTOL (0.0)
```

### struct NcClusterMassVanderlindeClass

```
struct NcClusterMassVanderlindeClass {  
};
```

**struct NcClusterMassVanderlinde**

```
struct NcClusterMassVanderlinde;
```

**Property Details**

**The "Asz" property**

"Asz"	gdouble	: Read / Write
-------	---------	----------------

A\_{SZ}.  
Allowed values: [1e-08,10]  
Default value: 6.01

**The "Asz-fit" property**

"Asz-fit"	gboolean	: Read / Write
-----------	----------	----------------

A\_{SZ}:fit.  
Default value: FALSE

**The "Bsz" property**

"Bsz"	gdouble	: Read / Write
-------	---------	----------------

B\_{SZ}.  
Allowed values: [1e-08,10]  
Default value: 1.31

**The "Bsz-fit" property**

"Bsz-fit"	gboolean	: Read / Write
-----------	----------	----------------

B\_{SZ}:fit.  
Default value: FALSE

**The "Csz" property**

"Csz"	gdouble	: Read / Write
-------	---------	----------------

C\_{SZ}.  
Allowed values: [1e-08,10]  
Default value: 1.6

**The "Csz-fit" property**

"Csz-fit"	gboolean	: Read / Write
-----------	----------	----------------

C\_{SZ}:fit.  
Default value: FALSE

---

**The "Dsz" property**

"Dsz"	gdouble	: Read / Write
-------	---------	----------------

D\_{SZ}.

Allowed values: [1e-08,10]

Default value: 0.21

**The "Dsz-fit" property**

"Dsz-fit"	gboolean	: Read / Write
-----------	----------	----------------

D\_{SZ}:fit.

Default value: FALSE

**The "M0" property**

"M0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

Reference mass (in  $h^{(-1)} * M_{\text{sun}}$  unit) in the SZ signal-mass scaling relation. FIXME Set correct values (limits)

Allowed values:  $\geq 1e+13$

Default value:  $5e+14$

**The "signif-obs-max" property**

"signif-obs-max"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Maximum obsevational significance.

Allowed values:  $\geq 2$

Default value: 40

**The "signif-obs-min" property**

"signif-obs-min"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Minimum obsevational significance.

Allowed values:  $\geq 2$

Default value: 5

**The "z0" property**

"z0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

Reference redshift in the SZ signal-mass scaling relation. FIXME Set correct values (limits)

Allowed values:  $\geq 0$

Default value: 0.6

---

## 7.11.5 SZ Cluster Mass Distribution

SZ Cluster Mass Distribution — Sunyaev-Zel'dovich FIXME

### Synopsis

```
enum                NcClusterMassBensonParams;
#define             NC_CLUSTER_MASS_BENSON_DEFAULT_A_SZ
#define             NC_CLUSTER_MASS_BENSON_DEFAULT_B_SZ
#define             NC_CLUSTER_MASS_BENSON_DEFAULT_C_SZ
#define             NC_CLUSTER_MASS_BENSON_DEFAULT_D_SZ
#define             NC_CLUSTER_MASS_BENSON_DEFAULT_PARAMS_ABSTOL
#define             NC_CLUSTER_MASS_BENSON_M_LOWER_BOUND
#define             NC_CLUSTER_MASS_BENSON_XI_ZETA_DIST_CUT
struct              NcClusterMassBensonClass;
struct              NcClusterMassBenson;
```

### Object Hierarchy

```
GObject
+----NcmModel
      +----NcClusterMass
            +----NcClusterMassBenson
                  +----NcClusterMassBensonXRay
```

### Properties

"Asz"	gdouble	: Read / Write
"Asz-fit"	gboolean	: Read / Write
"Bsz"	gdouble	: Read / Write
"Bsz-fit"	gboolean	: Read / Write
"Csz"	gdouble	: Read / Write
"Csz-fit"	gboolean	: Read / Write
"Dsz"	gdouble	: Read / Write
"Dsz-fit"	gboolean	: Read / Write
"M0"	gdouble	: Read / Write / Construct Only
"signif-obs-max"	gdouble	: Read / Write / Construct Only
"signif-obs-min"	gdouble	: Read / Write / Construct Only
"z0"	gdouble	: Read / Write / Construct Only

### Description

FIXME

### Details

#### enum NcClusterMassBensonParams

```
typedef enum {
    NC_CLUSTER_MASS_BENSON_A_SZ = 0,
    NC_CLUSTER_MASS_BENSON_B_SZ,
    NC_CLUSTER_MASS_BENSON_C_SZ,
} NcClusterMassBensonParams;
```

FIXME

**NC\_CLUSTER\_MASS\_BENSON\_A\_SZ** normalization of the mass-observable relation

**NC\_CLUSTER\_MASS\_BENSON\_B\_SZ** FIXME

**NC\_CLUSTER\_MASS\_BENSON\_C\_SZ** FIXME

**NC\_CLUSTER\_MASS\_BENSON\_D\_SZ** standard deviation of the mass-observable relation

**NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_A\_SZ**

```
#define NC_CLUSTER_MASS_BENSON_DEFAULT_A_SZ (5.58)
```

**NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_B\_SZ**

```
#define NC_CLUSTER_MASS_BENSON_DEFAULT_B_SZ (1.32)
```

**NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_C\_SZ**

```
#define NC_CLUSTER_MASS_BENSON_DEFAULT_C_SZ (0.87)
```

**NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_D\_SZ**

```
#define NC_CLUSTER_MASS_BENSON_DEFAULT_D_SZ (0.24)
```

**NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_PARAMS\_ABSTOL**

```
#define NC_CLUSTER_MASS_BENSON_DEFAULT_PARAMS_ABSTOL (0.0)
```

**NC\_CLUSTER\_MASS\_BENSON\_M\_LOWER\_BOUND**

```
#define NC_CLUSTER_MASS_BENSON_M_LOWER_BOUND (1.0e13)
```

**NC\_CLUSTER\_MASS\_BENSON\_XI\_ZETA\_DIST\_CUT**

```
#define NC_CLUSTER_MASS_BENSON_XI_ZETA_DIST_CUT (2.0)
```

**struct NcClusterMassBensonClass**

```
struct NcClusterMassBensonClass {  
};
```

**struct NcClusterMassBenson**

```
struct NcClusterMassBenson;
```

---



Property Details

The "Asz" property

"Asz"	gdouble	: Read / Write
-------	---------	----------------

A\_{SZ}.

Allowed values: [1e-08,10]

Default value: 5.58

The "Asz-fit" property

"Asz-fit"	gboolean	: Read / Write
-----------	----------	----------------

A\_{SZ}:fit.

Default value: FALSE

The "Bsz" property

"Bsz"	gdouble	: Read / Write
-------	---------	----------------

B\_{SZ}.

Allowed values: [1e-08,10]

Default value: 1.32

The "Bsz-fit" property

"Bsz-fit"	gboolean	: Read / Write
-----------	----------	----------------

B\_{SZ}:fit.

Default value: FALSE

The "Csz" property

"Csz"	gdouble	: Read / Write
-------	---------	----------------

C\_{SZ}.

Allowed values: [1e-08,10]

Default value: 0.87

The "Csz-fit" property

"Csz-fit"	gboolean	: Read / Write
-----------	----------	----------------

C\_{SZ}:fit.

Default value: FALSE

---

**The "Dsz" property**

"Dsz"	gdouble	: Read / Write
-------	---------	----------------

D\_{SZ}.

Allowed values: [1e-08,10]

Default value: 0.24

**The "Dsz-fit" property**

"Dsz-fit"	gboolean	: Read / Write
-----------	----------	----------------

D\_{SZ}:fit.

Default value: FALSE

**The "M0" property**

"M0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

Reference mass (in  $h^{(-1)} * M_{\text{sun}}$  unit) in the SZ signal-mass scaling relation. FIXME Set correct values (limits)

Allowed values:  $\geq 1e+13$

Default value:  $3e+14$

**The "signif-obs-max" property**

"signif-obs-max"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Maximum obsevational significance.

Allowed values:  $\geq 2$

Default value: 40

**The "signif-obs-min" property**

"signif-obs-min"	gdouble	: Read / Write / Construct Only
------------------	---------	---------------------------------

Minimum obsevational significance.

Allowed values:  $\geq 2$

Default value: 5

**The "z0" property**

"z0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

Reference redshift in the SZ signal-mass scaling relation. FIXME Set correct values (limits)

Allowed values:  $\geq 0$

Default value: 0.6

---

### 7.11.6 SZ and X ray Cluster Abundance Mass Distributions

SZ and X ray Cluster Abundance Mass Distributions — Sunyaev-Zel'dovich FIXME

#### Synopsis

```
enum                NcClusterMassBensonXRayParams;
#define             NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_A_X
#define             NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_B_X
#define             NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_C_X
#define             NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_D_X
#define             NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_PARAMS_ABSTOL
struct              NcClusterMassBensonXRayClass;
struct              NcClusterMassBensonXRay;
```

#### Object Hierarchy

```
GObject
+----NcmModel
      +----NcClusterMass
            +----NcClusterMassBenson
                  +----NcClusterMassBensonXRay
```

#### Properties

"Ax"	gdouble	: Read / Write
"Ax-fit"	gboolean	: Read / Write
"Bx"	gdouble	: Read / Write
"Bx-fit"	gboolean	: Read / Write
"Cx"	gdouble	: Read / Write
"Cx-fit"	gboolean	: Read / Write
"Dx"	gdouble	: Read / Write
"Dx-fit"	gboolean	: Read / Write
"M0x"	gdouble	: Read / Write / Construct Only
"Y0"	gdouble	: Read / Write / Construct Only
"Yx-obs-max"	gdouble	: Read / Write / Construct Only
"Yx-obs-min"	gdouble	: Read / Write / Construct Only

#### Description

FIXME

#### Details

##### enum NcClusterMassBensonXRayParams

```
typedef enum {
    NC_CLUSTER_MASS_BENSON_XRAY_A_X = NC_CLUSTER_MASS_BENSON_SPARAM_LEN,
    NC_CLUSTER_MASS_BENSON_XRAY_B_X,
    NC_CLUSTER_MASS_BENSON_XRAY_C_X,
} NcClusterMassBensonXRayParams;
```

FIXME

**NC\_CLUSTER\_MASS\_BENSON\_XRAY\_A\_X** normalization of the X-ray mass-observable relation

**NC\_CLUSTER\_MASS\_BENSON\_XRAY\_B\_X** FIXME

**NC\_CLUSTER\_MASS\_BENSON\_XRAY\_C\_X** FIXME

**NC\_CLUSTER\_MASS\_BENSON\_XRAY\_D\_X** standard deviation of the X-ray mass-observable relation

**NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_A\_X**

```
#define NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_A_X (5.77)
```

**NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_B\_X**

```
#define NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_B_X (0.57)
```

**NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_C\_X**

```
#define NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_C_X (-0.4)
```

**NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_D\_X**

```
#define NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_D_X (0.12)
```

**NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_PARAMS\_ABSTOL**

```
#define NC_CLUSTER_MASS_BENSON_XRAY_DEFAULT_PARAMS_ABSTOL (0.0)
```

**struct NcClusterMassBensonXRayClass**

```
struct NcClusterMassBensonXRayClass {  
};
```

**struct NcClusterMassBensonXRay**

```
struct NcClusterMassBensonXRay;
```

**Property Details**

**The "Ax" property**

"Ax"	gdouble	: Read / Write
------	---------	----------------

A\_{X}.

Allowed values: [1e-08,10]

Default value: 5.77

**The "Ax-fit" property**

"Ax-fit"	gboolean	: Read / Write
----------	----------	----------------

A\_{X}:fit.

Default value: TRUE

**The "Bx" property**

"Bx"	gdouble	: Read / Write
------	---------	----------------

B\_{X}.

Allowed values: [1e-08,10]

Default value: 0.57

**The "Bx-fit" property**

"Bx-fit"	gboolean	: Read / Write
----------	----------	----------------

B\_{X}:fit.

Default value: FALSE

**The "Cx" property**

"Cx"	gdouble	: Read / Write
------	---------	----------------

C\_{X}.

Allowed values: [-2,8]

Default value: -0.4

**The "Cx-fit" property**

"Cx-fit"	gboolean	: Read / Write
----------	----------	----------------

C\_{X}:fit.

Default value: FALSE

**The "Dx" property**

"Dx"	gdouble	: Read / Write
------	---------	----------------

D\_{X}.

Allowed values: [1e-08,10]

Default value: 0.12

**The "Dx-fit" property**

"Dx-fit"	gboolean	: Read / Write
----------	----------	----------------

D\_{X}:fit.

Default value: FALSE

---

The "M0x" property

"M0x"	gdouble	: Read / Write / Construct Only
-------	---------	---------------------------------

X Ray Reference mass.

Allowed values:  $\geq 1e+13$

Default value:  $1e+14$

The "Y0" property

"Y0"	gdouble	: Read / Write / Construct Only
------	---------	---------------------------------

Yx reference (in  $10^{14} * M_{\text{sun}} * \text{keV}$  unit) in the X-Ray proxy-mass scaling relation. FIXME Set correct values (limits)

Allowed values:  $\geq 1$

Default value: 3

The "Yx-obs-max" property

"Yx-obs-max"	gdouble	: Read / Write / Construct Only
--------------	---------	---------------------------------

Maximum obsevational Yx.

Allowed values:  $\geq 2$

Default value: 50

The "Yx-obs-min" property

"Yx-obs-min"	gdouble	: Read / Write / Construct Only
--------------	---------	---------------------------------

Minimum obsevational Yx.

Allowed values:  $\geq 1e-20$

Default value: 0.1

## 7.12 Cluster Abundance Distribution

Cluster Abundance Distribution — FIXME

### Synopsis

```
typedef      NcClusterAbundanceDataBinZ;
gdouble      (*NcClusterAbundanceN)      (NcClusterAbundance *cad,
                                           NcHICosmo *model);

gdouble      (*NcClusterAbundanceIntPd2N) (NcClusterAbundance *cad,
                                           NcHICosmo *model,
                                           gdouble lnM,
                                           gdouble z);

#define      nc_cluster_abundance_d2NdzdlnM_val (cad,
                                                  cp,
                                                  lnM,
```

```

z)
#define nc_cluster_abundance_dNdz_val (cad,
    cp,
    lnMl,
    lnMu,
    z)
#define nc_cluster_abundance_dNdlnM_val (cad,
    cp,
    lnM,
    zl,
    zu)
#define nc_cluster_abundance_N_val (cad,
    cp,
    lnMl,
    lnMu,
    zl,
    zu)

struct NcClusterAbundanceClass;
struct NcClusterAbundance;
NcClusterAbundance * nc_cluster_abundance_new (NcMassFunction *mfp,
    NcHaloBiasFunc *mbiasf,
    NcClusterRedshift *clusterz,
    NcClusterMass *clusterm);

NcClusterAbundance * nc_cluster_abundance_nodist_new (NcMassFunction *mfp,
    NcHaloBiasFunc *mbiasf);

NcClusterAbundance * nc_cluster_abundance_copy (NcClusterAbundance *cad);
NcClusterAbundance * nc_cluster_abundance_ref (NcClusterAbundance *cad);
void nc_cluster_abundance_free (NcClusterAbundance *cad);
void nc_cluster_abundance_clear (NcClusterAbundance **cad);
void nc_cluster_abundance_prepare (NcClusterAbundance *cad,
    NcHICosmo *model);

void nc_cluster_abundance_prepare_inv_dNdz (NcClusterAbundance *cad,
    NcHICosmo *model);

void nc_cluster_abundance_prepare_inv_dNdlnM_z (NcClusterAbundance *cad,
    NcHICosmo *model,
    gdouble z);

gdouble nc_cluster_abundance_z_p_lnm_p_d2n (NcClusterAbundance *cad,
    NcHICosmo *model,
    gdouble *lnM_obs,
    gdouble *lnM_obs_params,
    gdouble *z_obs,
    gdouble *z_obs_params);

gdouble nc_cluster_abundance_z_p_d2n (NcClusterAbundance *cad,
    NcHICosmo *model,
    gdouble lnM,
    gdouble *z_obs,
    gdouble *z_obs_params);

gdouble nc_cluster_abundance_lnm_p_d2n (NcClusterAbundance *cad,
    NcHICosmo *model,
    gdouble *lnM_obs,
    gdouble *lnM_obs_params,
    gdouble z);

gdouble nc_cluster_abundance_d2n (NcClusterAbundance *cad,
    NcHICosmo *model,
    gdouble lnM,

```

---

gdouble	nc_cluster_abundance_true_n	gdouble z); (NcClusterAbundance *cad, NcHICosmo *model);
gdouble	nc_cluster_abundance_n	(NcClusterAbundance *cad, NcHICosmo *model);
gdouble	nc_cluster_abundance_intp_d2n	(NcClusterAbundance *cad, NcHICosmo *model, gdouble lnM, gdouble z);
void	nc_cluster_abundance_set_redshift	(NcClusterAbundance *cad, NcClusterRedshift *clusterz);
void	nc_cluster_abundance_set_mass	(NcClusterAbundance *cad, NcClusterMass *clusterm);
NcClusterRedshift *	nc_cluster_abundance_get_redshift	(NcClusterAbundance *cad);
NcClusterMass *	nc_cluster_abundance_get_mass	(NcClusterAbundance *cad);
NcClusterRedshift *	nc_cluster_abundance_peek_redshift	(NcClusterAbundance *cad);
NcClusterMass *	nc_cluster_abundance_peek_mass	(NcClusterAbundance *cad);
void	nc_bias_mean_prepare	(NcClusterAbundance *cad, NcHICosmo *model);
gdouble	nc_bias_mean_val	(NcClusterAbundance *cad, NcHICosmo *model, gdouble lnMl, gdouble lnMu, gdouble z);
gdouble	nc_ca_mean_bias_numerator	(NcClusterAbundance *cad, NcHICosmo *model, gdouble lnM, gdouble z);
gdouble	nc_ca_mean_bias_denominator	(NcClusterAbundance *cad, NcHICosmo *model, gdouble lnM, gdouble z);
gdouble	nc_ca_mean_bias	(NcClusterAbundance *cad, NcHICosmo *model, gdouble lnM, gdouble z);
gdouble	nc_ca_mean_bias_Mobs_numerator	(NcClusterAbundance *cad, NcHICosmo *model, gdouble lnMobs, gdouble z);
gdouble	nc_ca_mean_bias_Mobs_denominator	(NcClusterAbundance *cad, NcHICosmo *model, gdouble lnMobs, gdouble z);

NcClusterAbundanceDataBinM;  
 NcClusterAbundanceDataP;  
 NcClusterAbundanceDataBin;

## Object Hierarchy

```

GObject
+----NcClusterAbundance

```

## Properties

"mass"	NcClusterMass*	: Read / Write / Construct
--------	----------------	----------------------------



"mass-function"	NcMassFunction*	: Read / Write / Construct Only
"mean-bias"	NcHaloBiasFunc*	: Read / Write / Construct Only
"redshift"	NcClusterRedshift*	: Read / Write / Construct

## Description

FIXME

## Details

### NcClusterAbundanceDataBinZ

```
typedef struct _NcClusterAbundanceDataBinZ NcClusterAbundanceDataBinZ;
```

### NcClusterAbundanceN ()

```
gdouble (*NcClusterAbundanceN) (NcClusterAbundance *cad,  
NcHICosmo *model);
```

### NcClusterAbundanceIntPd2N ()

```
gdouble (*NcClusterAbundanceIntPd2N) (NcClusterAbundance *cad,  
NcHICosmo *model,  
gdouble lnM,  
gdouble z);
```

### nc\_cluster\_abundance\_d2NdzdlnM\_val()

```
#define nc_cluster_abundance_d2NdzdlnM_val(cad, cp, lnM, z) (cad)->d2NdzdlnM_val(cad, cp, lnM, z)
```

### nc\_cluster\_abundance\_dNdz\_val()

```
#define nc_cluster_abundance_dNdz_val(cad, cp, lnMl, lnMu, z) (cad)->dNdz_val(cad, cp, lnMl, lnMu, ↵  
z)
```

### nc\_cluster\_abundance\_dNdlnM\_val()

```
#define nc_cluster_abundance_dNdlnM_val(cad, cp, lnM, zl, zu) (cad)->dNdlnM_val(cad, cp, lnM, zl, ↵  
zu)
```

### nc\_cluster\_abundance\_N\_val()

```
#define nc_cluster_abundance_N_val(cad, cp, lnMl, lnMu, zl, zu) (cad)->N_val(cad, cp, lnMl, lnMu, zl ↵  
, zu)
```

**struct NcClusterAbundanceClass**

```
struct NcClusterAbundanceClass {
};
```

**struct NcClusterAbundance**

```
struct NcClusterAbundance;
```

**nc\_cluster\_abundance\_new ()**

```
NcClusterAbundance * nc_cluster_abundance_new (NcMassFunction *mfp,
                                                NcHaloBiasFunc *mbiasf,
                                                NcClusterRedshift *clusterz,
                                                NcClusterMass *clusterm);
```

This function allocates memory for a new **NcClusterAbundance** object and sets its properties to the values from the input arguments.

**mfp** : a **NcMassFunction**.

**mbiasf** : a **NcHaloBiasFunc**. *[allow-none]*

**clusterz** : a **NcClusterRedshift**.

**clusterm** : a **NcClusterMass**.

**Returns** : A new **NcClusterAbundance**.

**nc\_cluster\_abundance\_nodist\_new ()**

```
NcClusterAbundance * nc_cluster_abundance_nodist_new (NcMassFunction *mfp,
                                                       NcHaloBiasFunc *mbiasf);
```

This function allocates memory for a new **NcClusterAbundance** object and sets its properties to the values from the input arguments.

**mfp** : a **NcMassFunction**.

**mbiasf** : a **NcHaloBiasFunc**. *[allow-none]*

**Returns** : A new **NcClusterAbundance**.

**nc\_cluster\_abundance\_copy ()**

```
NcClusterAbundance * nc_cluster_abundance_copy (NcClusterAbundance *cad);
```

Duplicates the **NcClusterAbundance** object setting the same values of the original properties.

**cad** : a **NcClusterAbundance**.

**Returns** : A new **NcClusterAbundance**. *[transfer full]*

**nc\_cluster\_abundance\_ref ()**

```
NcClusterAbundance * nc_cluster_abundance_ref (NcClusterAbundance *cad);
```

Increases the reference count of *cad* by one.

**cad** : a **NcClusterAbundance**.

**Returns** : *cad*. [*transfer full*]

**nc\_cluster\_abundance\_free ()**

```
void nc_cluster_abundance_free (NcClusterAbundance *cad);
```

Atomically decrements the reference count of *cad* by one. If the reference count drops to 0, all memory allocated by *cad* is released.

**cad** : a **NcClusterAbundance**.

**nc\_cluster\_abundance\_clear ()**

```
void nc_cluster_abundance_clear (NcClusterAbundance **cad);
```

Atomically decrements the reference count of *cad* by one. If the reference count drops to 0, all memory allocated by *cad* is released.

**cad** : a **NcClusterAbundance**.

**nc\_cluster\_abundance\_prepare ()**

```
void nc_cluster_abundance_prepare (NcClusterAbundance *cad,  
                                   NcHICosmo *model);
```

This function prepares ...

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

**nc\_cluster\_abundance\_prepare\_inv\_dNdz ()**

```
void nc_cluster_abundance_prepare_inv_dNdz  
      (NcClusterAbundance *cad,  
       NcHICosmo *model);
```

This function prepares a bidimensional spline...

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

---

**nc\_cluster\_abundance\_prepare\_inv\_dNdlN\_z ()**

```
void                                nc_cluster_abundance_prepare_inv_dNdlN_z
                                   (NcClusterAbundance *cad,
                                   NcHICosmo *model,
                                   gdouble z);
```

This function prepares a spline where the x array corresponds to the value of  $\int_{\ln M_0}^{\ln M_1} d^2N/dz d\ln M dM / \int_{\ln M_i}^{\ln M_f} dN/dz dM$  given a redshift  $z$  and the y array contains the values of logarithms base e of the mass. It is used to generate a sample of  $\ln M$  values.

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

**z** : redshift.

**nc\_cluster\_abundance\_z\_p\_lnm\_p\_d2n ()**

```
gdouble                            nc_cluster_abundance_z_p_lnm_p_d2n
                                   (NcClusterAbundance *cad,
                                   NcHICosmo *model,
                                   gdouble *lnM_obs,
                                   gdouble *lnM_obs_params,
                                   gdouble *z_obs,
                                   gdouble *z_obs_params);
```

This function computes  $\int_0^\infty dz \int_0^\infty d\ln M \frac{d^2N(\ln M, z)}{dz d\ln M} * P(z^{\text{phot}}|z) * P(\ln M^{\text{obs}}|\ln M, z)$ . We studied the convergence of this integral to optimize this function. We verified that it converges to 5 decimal places at the redshift interval  $[z^{\text{phot}} - 10\sigma_{\text{phot}}, z^{\text{phot}} + 10\sigma_{\text{phot}}]$  and the mass interval  $[\ln M^{\text{obs}} - 7\sigma_{\ln M}, \ln M^{\text{obs}} + 7\sigma_{\ln M}]$ .

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

**lnM\_obs** : FIXME.

**lnM\_obs\_params** : FIXME.

**z\_obs** : FIXME.

**z\_obs\_params** : FIXME.

**Returns** : a gdouble which represents  $\frac{d^2N(\ln M^{\text{obs}}, z^{\text{phot}})}{dz d\ln M}$ .

**nc\_cluster\_abundance\_z\_p\_d2n ()**

```
gdouble                            nc_cluster_abundance_z_p_d2n
                                   (NcClusterAbundance *cad,
                                   NcHICosmo *model,
                                   gdouble lnM,
                                   gdouble *z_obs,
                                   gdouble *z_obs_params);
```

This function computes  $\int_{z^{\text{phot}} - 10\sigma_{\text{phot}}}^{z^{\text{phot}} + 10\sigma_{\text{phot}}} dz \frac{d^2N}{dz d\ln M} * P(z^{\text{photo}}|z)$ . The integral limits were determined requiring a precision to five decimal places.

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

**lnM** : the logarithm base e of the mass (gdouble).

**z\_obs** : FIXME.

**z\_obs\_params** : FIXME.

**Returns** : a gdouble which corresponds to  $\int_{z_{\text{phot}} - 10\sigma_{\text{phot}}}^{z_{\text{phot}} + 10\sigma_{\text{phot}}} dz \, \frac{d^2N}{dz} P(z^{\text{photo}}|z) / f$.$

#### nc\_cluster\_abundance\_lnm\_p\_d2n ()

```
gdouble          nc_cluster_abundance_lnm_p_d2n      (NcClusterAbundance *cad,
                                                       NcHICosmo *model,
                                                       gdouble *lnM_obs,
                                                       gdouble *lnM_obs_params,
                                                       gdouble z);
```

This function computes  $\int_{\ln M^{\text{obs}} - 7\sigma_{\ln M}}^{\ln M^{\text{obs}} + 7\sigma_{\ln M}} d\ln M \, \frac{d^2N}{d\ln M} P(\ln M^{\text{obs}}|\ln M) / f$.$  The integral limits were determined requiring a precision to five decimal places.

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

**lnM\_obs** : FIXME.

**lnM\_obs\_params** : FIXME.

**z** : redshift (gdouble).

**Returns** : a gdouble which corresponds to  $\int_{\ln M^{\text{obs}} - 7\sigma_{\ln M}}^{\ln M^{\text{obs}} + 7\sigma_{\ln M}} d\ln M \, \frac{d^2N}{d\ln M} P(\ln M^{\text{obs}}|\ln M) / f$.$

#### nc\_cluster\_abundance\_d2n ()

```
gdouble          nc_cluster_abundance_d2n      (NcClusterAbundance *cad,
                                                       NcHICosmo *model,
                                                       gdouble lnM,
                                                       gdouble z);
```

This function computes  $\int_{\ln M^{\text{obs}} - 7\sigma_{\ln M}}^{\ln M^{\text{obs}} + 7\sigma_{\ln M}} d\ln M \, \frac{d^2N}{d\ln M} P(\ln M^{\text{obs}}|\ln M) / f$.$  The integral limits were determined requiring a precision to five decimal places.

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

**lnM** : true mass.

**z** : true redshift.

**Returns** : a gdouble which corresponds to  $\int_{\ln M^{\text{obs}} - 7\sigma_{\ln M}}^{\ln M^{\text{obs}} + 7\sigma_{\ln M}} d\ln M \, \frac{d^2N}{d\ln M} P(\ln M^{\text{obs}}|\ln M) / f$.$

**nc\_cluster\_abundance\_true\_n ()**

gdouble	nc_cluster_abundance_true_n	(NcClusterAbundance *cad, NcHICosmo *model);
---------	-----------------------------	---

FIXME

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

**Returns** : FIXME

**nc\_cluster\_abundance\_n ()**

gdouble	nc_cluster_abundance_n	(NcClusterAbundance *cad, NcHICosmo *model);
---------	------------------------	---

FIXME

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

**Returns** : FIXME

**nc\_cluster\_abundance\_intp\_d2n ()**

gdouble	nc_cluster_abundance_intp_d2n	(NcClusterAbundance *cad, NcHICosmo *model, gdouble lnM, gdouble z);
---------	-------------------------------	---

FIXME

**cad** : a **NcClusterAbundance**.

**model** : a **NcHICosmo**.

**lnM** : FIXME

**z** : redshift.

**Returns** : FIXME

**nc\_cluster\_abundance\_set\_redshift ()**

void	nc_cluster_abundance_set_redshift	(NcClusterAbundance *cad, NcClusterRedshift *clusterz);
------	-----------------------------------	--

Sets the value *clusterz* to the "redshift" property.

**cad** : a **NcClusterAbundance**.

**clusterz** : value of "redshift".

**nc\_cluster\_abundance\_set\_mass ()**

```
void nc_cluster_abundance_set_mass(NcClusterAbundance *cad,  
                                   NcClusterMass *clusterm);
```

Sets the value `clusterm` to the "mass" property.

**cad**: a **NcClusterAbundance**.

*clusterm*: value of "mass".

**nc\_cluster\_abundance\_get\_redshift ()**

```
NcClusterRedshift * nc_cluster_abundance_get_redshift (NcClusterAbundance *cad);
```

Gets the value of the "redshift" property.

**cad**: a **NcClusterAbundance**.

**Returns :** the value of "redshift" property. *[transfer full]*

**nc\_cluster\_abundance\_get\_mass ()**

```
NcClusterMass * nc_cluster_abundance_get_mass (NcClusterAbundance *cad);
```

Gets the value of the "mass" property.

**cad**: a **NcClusterAbundance**.

**Returns :** the value of "mass" property. *[transfer full]*

**nc\_cluster\_abundance\_peek\_redshift ()**

```
NcClusterRedshift * nc_cluster_abundance_peek_redshift (NcClusterAbundance *cad);
```

Gets the value of the "redshift" property.

**cad**: a **NcClusterAbundance**.

**Returns :** the value of "redshift" property. *[transfer none]*

**nc\_cluster\_abundance\_peek\_mass ()**

```
NcClusterMass * nc_cluster_abundance_peek_mass (NcClusterAbundance *cad);
```

Gets the value of the "mass" property.

**cad**: a **NcClusterAbundance**.

**Returns :** the value of "mass" property. *[transfer none]*

**nc\_bias\_mean\_prepare ()**

```
void nc_bias_mean_prepare(NcClusterAbundance *cad,
                          NcHICosmo *model);
```

**nc\_bias\_mean\_val ()**

```
gdouble          nc_bias_mean_val          (NcClusterAbundance *cad,  
                                           NcHICosmo *model,  
                                           gdouble lnMl,  
                                           gdouble lnMu,  
                                           gdouble z);
```

**nc\_ca\_mean\_bias\_numerator ()**

```
gdouble          nc_ca_mean_bias_numerator (NcClusterAbundance *cad,  
                                           NcHICosmo *model,  
                                           gdouble lnM,  
                                           gdouble z);
```

**nc\_ca\_mean\_bias\_denominator ()**

```
gdouble          nc_ca_mean_bias_denominator (NcClusterAbundance *cad,  
                                              NcHICosmo *model,  
                                              gdouble lnM,  
                                              gdouble z);
```

**nc\_ca\_mean\_bias ()**

```
gdouble          nc_ca_mean_bias          (NcClusterAbundance *cad,  
                                           NcHICosmo *model,  
                                           gdouble lnM,  
                                           gdouble z);
```

**nc\_ca\_mean\_bias\_Mobs\_numerator ()**

```
gdouble          nc_ca_mean_bias_Mobs_numerator (NcClusterAbundance *cad,  
                                                  NcHICosmo *model,  
                                                  gdouble lnMobs,  
                                                  gdouble z);
```

**nc\_ca\_mean\_bias\_Mobs\_denominator ()**

```
gdouble          nc_ca_mean_bias_Mobs_denominator (NcClusterAbundance *cad,  
                                                    NcHICosmo *model,  
                                                    gdouble lnMobs,  
                                                    gdouble z);
```

**NcClusterAbundanceDataBinM**

```
typedef struct _NcClusterAbundanceDataBinM NcClusterAbundanceDataBinM;
```



**NcClusterAbundanceDataP**

```
typedef struct _NcClusterAbundanceDataP NcClusterAbundanceDataP;
```

**NcClusterAbundanceDataBin**

```
typedef struct _NcClusterAbundanceDataBin NcClusterAbundanceDataBin;
```

**Property Details**

**The "mass" property**

"mass"	NcClusterMass*	: Read / Write / Construct
--------	----------------	----------------------------

Mass object.

**The "mass-function" property**

"mass-function"	NcMassFunction*	: Read / Write / Construct Only
-----------------	-----------------	---------------------------------

Mass Function.

**The "mean-bias" property**

"mean-bias"	NcHaloBiasFunc*	: Read / Write / Construct Only
-------------	-----------------	---------------------------------

Mean Halo Bias Function.

**The "redshift" property**

"redshift"	NcClusterRedshift*	: Read / Write / Construct
------------	--------------------	----------------------------

Redshift object.

---

## Chapter 8

# Cosmological Data

### 8.1 CMB Data

CMB Data — Helper function for obtaining CMB data

#### Synopsis

```
enum                                NcDataCMBId;
NcmData *                          nc_data_cmb_create      (NcDistance *dist,
                                                            NcDataCMBId id);
```

#### Description

FIXME

#### Details

**enum NcDataCMBId**

```
typedef enum {
    NC_DATA_CMB_SHIFT_PARAM_WMAP3 = 0,
    NC_DATA_CMB_SHIFT_PARAM_WMAP5,
    NC_DATA_CMB_SHIFT_PARAM_WMAP7,
    NC_DATA_CMB_DIST_PRIORS_WMAP5,
    NC_DATA_CMB_DIST_PRIORS_WMAP7,
} NcDataCMBId;
```

FIXME

**NC\_DATA\_CMB\_SHIFT\_PARAM\_WMAP3** FIXME

**NC\_DATA\_CMB\_SHIFT\_PARAM\_WMAP5** FIXME

**NC\_DATA\_CMB\_SHIFT\_PARAM\_WMAP7** FIXME

**NC\_DATA\_CMB\_DIST\_PRIORS\_WMAP5** FIXME

**NC\_DATA\_CMB\_DIST\_PRIORS\_WMAP7** FIXME

**NC\_DATA\_CMB\_DIST\_PRIORS\_WMAP9** FIXME

**nc\_data\_cmb\_create ()**

```
NcmData *          nc_data_cmb_create          (NcDistance *dist,
                                                NcDataCMBId id);
```

FIXME

*dist* : FIXME

*id* : FIXME

*Returns* : FIXME. *[transfer full]*

**8.2 Cosmic Microwave Background Data -- Shift Parameter**

Cosmic Microwave Background Data -- Shift Parameter — CMB shift parameter implementation

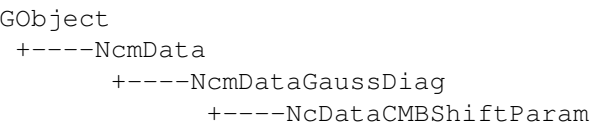
**Synopsis**

```
struct          NcDataCMBShiftParamClass;
struct          NcDataCMBShiftParam;
NcmData *       nc_data_cmb_shift_param_new      (NcDistance *dist,
                                                NcDataCMBId id);

void           nc_data_cmb_shift_param_set_sample (NcDataCMBShiftParam *cmb_shift_pa
                                                NcDataCMBId id);

NcDataCMBId     nc_data_cmb_shift_param_get_sample (NcDataCMBShiftParam *cmb_shift_pa
```

**Object Hierarchy**



**Properties**

"dist"	NcDistance*	: Read / Write / Construct
"sample-id"	NcDataCMBId	: Read / Write

**Description**

FIXME

**Details**

**struct NcDataCMBShiftParamClass**

```
struct NcDataCMBShiftParamClass {
};
```

**struct NcDataCMBShiftParam**

```
struct NcDataCMBShiftParam;
```

**nc\_data\_cmb\_shift\_param\_new ()**

```
NcmData * nc_data_cmb_shift_param_new (NcDistance *dist,
                                       NcDataCMBId id);
```

FIXME

*dist* : FIXME

*id* : FIXME

*Returns* : FIXME

**nc\_data\_cmb\_shift\_param\_set\_sample ()**

```
void nc_data_cmb_shift_param_set_sample (NcDataCMBShiftParam * ↵
    cmb_shift_param,
                                       NcDataCMBId id);
```

FIXME

*cmb\_shift\_param* : a **NcDataCMBShiftParam**.

*id* : FIXME

**nc\_data\_cmb\_shift\_param\_get\_sample ()**

```
NcDataCMBId nc_data_cmb_shift_param_get_sample (NcDataCMBShiftParam * ↵
    cmb_shift_param);
```

FIXME

*cmb\_shift\_param* : a **NcDataCMBShiftParam**

*Returns* : FIXME

**Property Details**

**The "dist" property**

"dist"	NcDistance*	: Read / Write / Construct
--------	-------------	----------------------------

Distance object.

**The "sample-id" property**

"sample-id"	NcDataCMBId	: Read / Write
-------------	-------------	----------------

Sample id.

Default value: NC\_DATA\_CMB\_SHIFT\_PARAM\_WMAP7

---

## 8.3 Cosmic Microwave Background Data -- Distance priors

Cosmic Microwave Background Data -- Distance priors — CMB distance priors implementation

### Synopsis

```
struct          NcDataCMBDistPriorsClass;
struct          NcDataCMBDistPriors;
NcmData *       nc_data_cmb_dist_priors_new      (NcDistance *dist,
                                                  NcDataCMBId id);
void            nc_data_cmb_dist_priors_set_sample (NcDataCMBDistPriors *cmb_dist_pri
                                                  NcDataCMBId id);
NcDataCMBId     nc_data_cmb_dist_priors_get_sample (NcDataCMBDistPriors *cmb_dist_pri
```

### Object Hierarchy

```
GObject
+-----NcmData
      +-----NcmDataGauss
            +-----NcDataCMBDistPriors
```

### Properties

"dist"	NcDistance*	: Read / Write / Construct
"sample-id"	NcDataCMBId	: Read / Write

### Description

FIXME

### Details

#### struct NcDataCMBDistPriorsClass

```
struct NcDataCMBDistPriorsClass {
};
```

#### struct NcDataCMBDistPriors

```
struct NcDataCMBDistPriors;
```

#### nc\_data\_cmb\_dist\_priors\_new ()

```
NcmData *       nc_data_cmb_dist_priors_new      (NcDistance *dist,
                                                  NcDataCMBId id);
```

This function allocates memory for a new **NcmData** object and sets its properties to the values from the input arguments.

**dist** : a **NcDistance**.

**id** : a **NcDataCMBId**.

**Returns** : A **NcmData**.

**nc\_data\_cmb\_dist\_priors\_set\_sample ()**

```
void nc_data_cmb_dist_priors_set_sample (NcDataCMBDistPriors * ←
    cmb_dist_priors,
                                         NcDataCMBId id);
```

This function sets the elements of both a vector and a matrix to the best-fit and the inverse covariance matrix values, respectively, of the CMB distance priors sample specified by *id*.

**cmb\_dist\_priors**: a **NcDataCMBDistPriors**.  
**id**: a **NcDataCMBId**.

**nc\_data\_cmb\_dist\_priors\_get\_sample ()**

```
NcDataCMBId nc_data_cmb_dist_priors_get_sample (NcDataCMBDistPriors * ←
    cmb_dist_priors);
```

This function returns the id of the CMB distance priors sample.

**cmb\_dist\_priors**: a **NcDataCMBDistPriors**  
**Returns**: a **NcDataCMBId**.

**Property Details**

**The "dist" property**

"dist"	NcDistance*	: Read / Write / Construct
--------	-------------	----------------------------

Distance object.

**The "sample-id" property**

"sample-id"	NcDataCMBId	: Read / Write
-------------	-------------	----------------

Sample id.

Default value: NC\_DATA\_CMB\_DIST\_PRIORS\_WMAP7

**8.4 Hubble Function Data**

Hubble Function Data — Object representing Hubble Function data

**Synopsis**

```
enum NcDataHubbleId;
struct NcDataHubbleClass;
struct NcDataHubble;
NcmData * nc_data_hubble_new (NcDataHubbleId id);
void nc_data_hubble_set_size (NcDataHubble *hubble,
                              quint np);
quint nc_data_hubble_get_size (NcDataHubble *hubble);
void nc_data_hubble_set_sample (NcDataHubble *hubble,
                                NcDataHubbleId id);
NcDataHubbleId nc_data_hubble_get_sample (NcDataHubble *hubble);
```

## Object Hierarchy

```
GObject
+-----NcmData
      +-----NcmDataGaussDiag
            +-----NcDataHubble
```

## Properties

"sample-id"                      NcDataHubbleId                      : Read / Write

## Description

FIXME

## Details

### enum NcDataHubbleId

```
typedef enum {
    NC_DATA_HUBBLE_SIMON2005 = 0,
    NC_DATA_HUBBLE_CABRE,
    NC_DATA_HUBBLE_STERN2009,
    NC_DATA_HUBBLE_MORESCO2012_BC03,
    NC_DATA_HUBBLE_MORESCO2012_MASTRO,
} NcDataHubbleId;
```

FIXME

**NC\_DATA\_HUBBLE\_SIMON2005** FIXME

**NC\_DATA\_HUBBLE\_CABRE** FIXME

**NC\_DATA\_HUBBLE\_STERN2009** FIXME

**NC\_DATA\_HUBBLE\_MORESCO2012\_BC03** FIXME

**NC\_DATA\_HUBBLE\_MORESCO2012\_MASTRO** FIXME

**NC\_DATA\_HUBBLE\_BUSCA2013\_BAO\_WMAP** FIXME

### struct NcDataHubbleClass

```
struct NcDataHubbleClass {
};
```

### struct NcDataHubble

```
struct NcDataHubble;
```

**nc\_data\_hubble\_new ()**

```
NcmData *          nc_data_hubble_new          (NcDataHubbleId id);
```

FIXME

*id*: FIXME

**Returns**: FIXME

**nc\_data\_hubble\_set\_size ()**

```
void              nc_data_hubble_set_size      (NcDataHubble *hubble,  
                                               quint np);
```

FIXME

*hubble*: a **NcDataHubble**

*np*: FIXME

**Returns**: FIXME

**nc\_data\_hubble\_get\_size ()**

```
quint            nc_data_hubble_get_size      (NcDataHubble *hubble);
```

FIXME

*hubble*: a **NcDataHubble**

**Returns**: FIXME

**nc\_data\_hubble\_set\_sample ()**

```
void              nc_data_hubble_set_sample    (NcDataHubble *hubble,  
                                               NcDataHubbleId id);
```

FIXME

*hubble*: a **NcDataHubble**.

*id*: FIXME

**nc\_data\_hubble\_get\_sample ()**

```
NcDataHubbleId    nc_data_hubble_get_sample    (NcDataHubble *hubble);
```

FIXME

*hubble*: a **NcDataHubble**

**Returns**: FIXME

---



Property Details

The "sample-id" property

"sample-id"	NcDataHubbleId	: Read / Write
-------------	----------------	----------------

Sample id.

Default value: NC\_DATA\_HUBBLE\_CABRE

8.5 Hubble Function Data from BAO

Hubble Function Data from BAO — Object representing Hubble Function BAO data

Synopsis

```
enum          NcDataHubbleBaoId;
struct        NcDataHubbleBaoClass;
struct        NcDataHubbleBao;
NcmData *     nc_data_hubble_bao_new          (NcDistance *dist,
                                              NcDataHubbleBaoId id);
void          nc_data_hubble_bao_set_size     (NcDataHubbleBao *hubble_bao,
                                              guint np);
guint         nc_data_hubble_bao_get_size     (NcDataHubbleBao *hubble_bao);
void          nc_data_hubble_bao_set_sample   (NcDataHubbleBao *hubble_bao,
                                              NcDataHubbleBaoId id);
NcDataHubbleBaoId nc_data_hubble_bao_get_sample (NcDataHubbleBao *hubble_bao);
```

Object Hierarchy

```
GObject
+-----NcmData
      +-----NcmDataGaussDiag
            +-----NcDataHubbleBao
```

Properties

"dist"	NcDistance*	: Read / Write / Construct
"sample-id"	NcDataHubbleBaoId	: Read / Write

Description

FIXME

Details

enum NcDataHubbleBaold

```
typedef enum {
} NcDataHubbleBaoId;
```

FIXME

NC\_DATA\_HUBBLE\_BAO\_BUSCA2013 FIXME

**struct NcDataHubbleBaoClass**

```
struct NcDataHubbleBaoClass {  
};
```

**struct NcDataHubbleBao**

```
struct NcDataHubbleBao;
```

**nc\_data\_hubble\_bao\_new ()**

```
NcmData *          nc_data_hubble_bao_new          (NcDistance *dist,  
                                                    NcDataHubbleBaoId id);
```

FIXME

**dist** : FIXME

**id** : FIXME

**Returns** : FIXME

**nc\_data\_hubble\_bao\_set\_size ()**

```
void              nc_data_hubble_bao_set_size      (NcDataHubbleBao *hubble_bao,  
                                                    guint np);
```

FIXME

**hubble\_bao** : a **NcDataHubbleBao**

**np** : FIXME

**Returns** : FIXME

**nc\_data\_hubble\_bao\_get\_size ()**

```
guint            nc_data_hubble_bao_get_size      (NcDataHubbleBao *hubble_bao);
```

FIXME

**hubble\_bao** : a **NcDataHubbleBao**

**Returns** : FIXME

**nc\_data\_hubble\_bao\_set\_sample ()**

```
void              nc_data_hubble_bao_set_sample   (NcDataHubbleBao *hubble_bao,  
                                                    NcDataHubbleBaoId id);
```

FIXME

**hubble\_bao** : a **NcDataHubbleBao**.

**id** : FIXME

---

**nc\_data\_hubble\_bao\_get\_sample ()**

```
NcDataHubbleBaoId  nc_data_hubble_bao_get_sample      (NcDataHubbleBao *hubble_bao);
```

FIXME

*hubble\_bao* : a **NcDataHubbleBao**

*Returns* : FIXME

**Property Details**

**The "dist" property**

"dist"	NcDistance*	: Read / Write / Construct
--------	-------------	----------------------------

Distance object.

**The "sample-id" property**

"sample-id"	NcDataHubbleBaoId	: Read / Write
-------------	-------------------	----------------

Sample id.

Default value: NC\_DATA\_HUBBLE\_BAO\_BUSCA2013

**8.6 BAO Data**

BAO Data — Helper function for obtaining BAO data

**Synopsis**

```
enum                                NcDataBaoId;
NcmData *                          nc_data_bao_create      (NcDistance *dist,
                                                            NcDataBaoId id);
```

**Description**

FIXME

**Details**

**enum NcDataBaoid**

```
typedef enum {
    NC_DATA_BAO_A_EISENSTEIN2005 = 0,
    NC_DATA_BAO_DV_EISENSTEIN2005,
    NC_DATA_BAO_DVDV_PERCIVAL2007,
    NC_DATA_BAO_DVDV_PERCIVAL2010,
    NC_DATA_BAO_RDV_PERCIVAL2007,
    NC_DATA_BAO_RDV_PERCIVAL2010,
    NC_DATA_BAO_RDV_BEUTLER2011,
    NC_DATA_BAO_RDV_PADMANABHAN2012,
    NC_DATA_BAO_RDV_ANDERSON2012,
} NcDataBaoId;
```

FIXME

**NC\_DATA\_BAO\_A\_EISENSTEIN2005** FIXME

**NC\_DATA\_BAO\_DV\_EISENSTEIN2005** FIXME

**NC\_DATA\_BAO\_DVDV\_PERCIVAL2007** FIXME

**NC\_DATA\_BAO\_DVDV\_PERCIVAL2010** FIXME

**NC\_DATA\_BAO\_RDV\_PERCIVAL2007** FIXME

**NC\_DATA\_BAO\_RDV\_PERCIVAL2010** FIXME

**NC\_DATA\_BAO\_RDV\_BEUTLER2011** FIXME

**NC\_DATA\_BAO\_RDV\_PADMANABHAN2012** FIXME

**NC\_DATA\_BAO\_RDV\_ANDERSON2012** FIXME

**NC\_DATA\_BAO\_RDV\_BLAKE2012** FIXME

**nc\_data\_bao\_create ()**

```
NcmData *          nc_data_bao_create          (NcDistance *dist,
                                                NcDataBaoId id);
```

FIXME

**dist** : FIXME

**id** : FIXME

**Returns** : FIXME. *[transfer full]*

## 8.7 Baryonic Oscillation Data -- Acoustic Scale

Baryonic Oscillation Data -- Acoustic Scale — BAO acoustic scale estimator

### Synopsis

```
struct          NcDataBaoAClass;
struct          NcDataBaoA;
NcmData *      nc_data_bao_a_new          (NcDistance *dist,
                                           NcDataBaoId id);

void           nc_data_bao_a_set_size     (NcDataBaoA *bao_a,
                                           guint np);

guint          nc_data_bao_a_get_size     (NcDataBaoA *bao_a);
void           nc_data_bao_a_set_sample   (NcDataBaoA *bao_a,
                                           NcDataBaoId id);

NcDataBaoId    nc_data_bao_a_get_sample   (NcDataBaoA *bao_a);
```

### Object Hierarchy

```
GObject
+----NcmData
      +----NcmDataGaussDiag
            +----NcDataBaoA
```

Properties

"dist"	NcDistance*	: Read / Write / Construct
"sample-id"	NcDataBaoId	: Read / Write

Description

The acoustic scale is defined as  $A \equiv D_V(z) \frac{1}{\sqrt{\Omega_m H_0^2}} \{z\}$  See Section 4.5 from [Eisenstein et al. \(2005\)](#).

Details

struct NcDataBaoAClass

```
struct NcDataBaoAClass {  
};
```

struct NcDataBaoA

```
struct NcDataBaoA;
```

nc\_data\_bao\_a\_new ()

```
NcmData *      nc_data_bao_a_new      (NcDistance *dist,  
                                       NcDataBaoId id);
```

FIXME

*dist* : FIXME

*id* : FIXME

*Returns* : FIXME

nc\_data\_bao\_a\_set\_size ()

```
void           nc_data_bao_a_set_size (NcDataBaoA *bao_a,  
                                       quint np);
```

FIXME

*bao\_a* : a [NcDataBaoA](#)

*np* : FIXME

*Returns* : FIXME

nc\_data\_bao\_a\_get\_size ()

```
quint          nc_data_bao_a_get_size (NcDataBaoA *bao_a);
```

FIXME

*bao\_a* : a [NcDataBaoA](#)

*Returns* : FIXME

**nc\_data\_bao\_a\_set\_sample ()**

```
void nc_data_bao_a_set_sample (NcDataBaoA *bao_a, NcDataBaoId id);
```

FIXME

**bao\_a** : a **NcDataBaoA**.

**id** : FIXME

**nc\_data\_bao\_a\_get\_sample ()**

```
NcDataBaoId nc_data_bao_a_get_sample (NcDataBaoA *bao_a);
```

FIXME

**bao\_a** : a **NcDataBaoA**

**Returns** : FIXME

**Property Details**

**The "dist" property**

"dist"	NcDistance*	: Read / Write / Construct
--------	-------------	----------------------------

Distance object.

**The "sample-id" property**

"sample-id"	NcDataBaoId	: Read / Write
-------------	-------------	----------------

Sample id.

Default value: NC\_DATA\_BAO\_A\_EISENSTEIN2005

**8.8 Baryonic Oscillation Data -- Volume Mean**

Baryonic Oscillation Data -- Volume Mean — BAO averaged volume \$D\_V\$ estimator

**Synopsis**

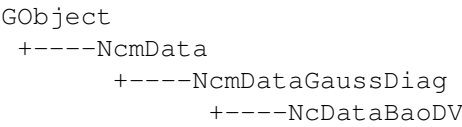
```
struct NcDataBaoDVClass;
struct NcDataBaoDV;
NcmData * nc_data_bao_dv_new (NcDistance *dist, NcDataBaoId id);

void nc_data_bao_dv_set_size (NcDataBaoDV *bao_dv, quint np);

quint nc_data_bao_dv_get_size (NcDataBaoDV *bao_dv);
void nc_data_bao_dv_set_sample (NcDataBaoDV *bao_dv, NcDataBaoId id);

NcDataBaoId nc_data_bao_dv_get_sample (NcDataBaoDV *bao_dv);
```

Object Hierarchy



Properties

"dist"	NcDistance*	: Read / Write / Construct
"sample-id"	NcDataBaoId	: Read / Write

Description

See [Eisenstein et al. \(2005\)](#).

Details

struct NcDataBaoDVClass

```
struct NcDataBaoDVClass {
};
```

struct NcDataBaoDV

```
struct NcDataBaoDV;
```

nc\_data\_bao\_dv\_new ()

```
NcmData *          nc_data_bao_dv_new          (NcDistance *dist,
                                                NcDataBaoId id);
```

FIXME

*dist* : FIXME

*id* : FIXME

*Returns* : FIXME

nc\_data\_bao\_dv\_set\_size ()

```
void          nc_data_bao_dv_set_size          (NcDataBaoDV *bao_dv,
                                                quint np);
```

FIXME

*bao\_dv* : a [NcDataBaoDV](#)

*np* : FIXME

*Returns* : FIXME

**nc\_data\_bao\_dv\_get\_size ()**

```
quint      nc_data_bao_dv_get_size      (NcDataBaoDV *bao_dv);
```

FIXME

*bao\_dv* : a **NcDataBaoDV**

*Returns* : FIXME

**nc\_data\_bao\_dv\_set\_sample ()**

```
void      nc_data_bao_dv_set_sample      (NcDataBaoDV *bao_dv,  
                                           NcDataBaoId id);
```

FIXME

*bao\_dv* : a **NcDataBaoDV**.

*id* : FIXME

**nc\_data\_bao\_dv\_get\_sample ()**

```
NcDataBaoId      nc_data_bao_dv_get_sample      (NcDataBaoDV *bao_dv);
```

FIXME

*bao\_dv* : a **NcDataBaoDV**

*Returns* : FIXME

**Property Details**

**The "dist" property**

"dist"	NcDistance*	: Read / Write / Construct
--------	-------------	----------------------------

Distance object.

**The "sample-id" property**

"sample-id"	NcDataBaoId	: Read / Write
-------------	-------------	----------------

Sample id.

Default value: NC\_DATA\_BAO\_DV\_EISENSTEIN2005

**8.9 Baryonic Oscillation Data -- rDv**

Baryonic Oscillation Data -- rDv — BAO  $r/D_V$  ratio estimator



## Synopsis

```

struct          NcDataBaoRDVClass;
struct          NcDataBaoRDV;
NcmData *       nc_data_bao_rdv_new          (NcDistance *dist,
                                              NcDataBaoId id);

void            nc_data_bao_rdv_set_size     (NcDataBaoRDV *bao_rdv,
                                              guint np);

guint           nc_data_bao_rdv_get_size     (NcDataBaoRDV *bao_rdv);
void            nc_data_bao_rdv_set_sample   (NcDataBaoRDV *bao_rdv,
                                              NcDataBaoId id);

NcDataBaoId     nc_data_bao_rdv_get_sample   (NcDataBaoRDV *bao_rdv);

```

## Object Hierarchy

```

GObject
+-----NcmData
      +-----NcmDataGauss
            +-----NcDataBaoRDV

```

## Properties

"dist"	NcDistance*	: Read / Write / Construct
"sample-id"	NcDataBaoId	: Read / Write

## Description

See [Percival et al. \(2007\)](#).

## Details

### struct NcDataBaoRDVClass

```

struct NcDataBaoRDVClass {
};

```

### struct NcDataBaoRDV

```

struct NcDataBaoRDV;

```

### nc\_data\_bao\_rdv\_new ()

```

NcmData *       nc_data_bao_rdv_new          (NcDistance *dist,
                                              NcDataBaoId id);

```

FIXME

**dist** : FIXME

**id** : FIXME

**Returns** : FIXME

**nc\_data\_bao\_rdv\_set\_size ()**

```
void                nc_data_bao_rdv_set_size                (NcDataBaoRDV *bao_rdv,
                                                            quint np);
```

FIXME

***bao\_rdv*** : a **NcDataBaoRDV**

***np*** : FIXME

***Returns*** : FIXME

**nc\_data\_bao\_rdv\_get\_size ()**

```
quint                nc_data_bao_rdv_get_size                (NcDataBaoRDV *bao_rdv);
```

FIXME

***bao\_rdv*** : a **NcDataBaoRDV**

***Returns*** : FIXME

**nc\_data\_bao\_rdv\_set\_sample ()**

```
void                nc_data_bao_rdv_set_sample                (NcDataBaoRDV *bao_rdv,
                                                            NcDataBaoId id);
```

FIXME

***bao\_rdv*** : a **NcDataBaoRDV**.

***id*** : FIXME

**nc\_data\_bao\_rdv\_get\_sample ()**

```
NcDataBaoId          nc_data_bao_rdv_get_sample                (NcDataBaoRDV *bao_rdv);
```

FIXME

***bao\_rdv*** : a **NcDataBaoRDV**

***Returns*** : FIXME

**Property Details**

**The "dist" property**

```
"dist"                NcDistance*                : Read / Write / Construct
```

Distance object.

**The "sample-id" property**

"sample-id"	NcDataBaoId	: Read / Write
-------------	-------------	----------------

Sample id.

Default value: NC\_DATA\_BAO\_RDV\_PERCIVAL2010

**8.10 Baryonic Oscillation Data -- DVDV**

Baryonic Oscillation Data -- DVDV — BAO  $\$D\_V/D\_V\$$  ratio estimator

**Synopsis**

```

struct          NcDataBaoDVDVClass;
struct          NcDataBaoDVDV;
NcmData *      nc_data_bao_dvdv_new          (NcDistance *dist,
                                              NcDataBaoId id);
void          nc_data_bao_dvdv_set_sample    (NcDataBaoDVDV *bao_dvdv,
                                              NcDataBaoId id);
NcDataBaoId    nc_data_bao_dvdv_get_sample   (NcDataBaoDVDV *bao_dvdv);

```

**Object Hierarchy**

```

GObject
+----NcmData
      +----NcmDataGaussDiag
            +----NcDataBaoDVDV

```

**Properties**

"dist"	NcDistance*	: Read / Write / Construct
"sample-id"	NcDataBaoId	: Read / Write

**Description**

See [Percival et al. \(2007\)](#).

**Details****struct NcDataBaoDVDVClass**

```

struct NcDataBaoDVDVClass {
};

```

**struct NcDataBaoDVDV**

```

struct NcDataBaoDVDV;

```

**nc\_data\_bao\_dvdv\_new ()**

```
NcmData *          nc_data_bao_dvdv_new          (NcDistance *dist,
                                                    NcDataBaoId id);
```

FIXME

*dist* : FIXME

*id* : FIXME

*Returns* : FIXME

**nc\_data\_bao\_dvdv\_set\_sample ()**

```
void              nc_data_bao_dvdv_set_sample    (NcDataBaoDVDV *bao_dvdv,
                                                    NcDataBaoId id);
```

**nc\_data\_bao\_dvdv\_get\_sample ()**

```
NcDataBaoId       nc_data_bao_dvdv_get_sample    (NcDataBaoDVDV *bao_dvdv);
```

FIXME

*bao\_dvdv* : a **NcDataBaoDVDV**

*Returns* : FIXME

**Property Details**

**The "dist" property**

"dist"	NcDistance*	: Read / Write / Construct
--------	-------------	----------------------------

Distance object.

**The "sample-id" property**

"sample-id"	NcDataBaoId	: Read / Write
-------------	-------------	----------------

Sample id.

Default value: NC\_DATA\_BAO\_DVDV\_PERCIVAL2007

**8.11 SN Ia Data**

SN Ia Data — Helper function for obtaining Supernovae Ia data

## Synopsis

```
enum                NcDataSNIAId;
#define             NC_DATA_SNIA_SIMPLE_START
#define             NC_DATA_SNIA_SIMPLE_END
#define             NC_DATA_SNIA_SIMPLE_LEN
#define             NC_DATA_SNIA_COV_START
#define             NC_DATA_SNIA_COV_END
#define             NC_DATA_SNIA_COV_LEN
void               nc_data_snia_load_cat      (NcDataSNIAcov *snia_cov,
                                              NcDataSNIAId id);
gchar *            nc_data_snia_get_fits      (const gchar *filename,
                                              gboolean check_size);
gchar *            nc_data_snia_get_catalog   (gchar *id);
gchar *            nc_data_snia_get_catalog_by_id (NcDataSNIAId id);
```

## Description

FIXME

## Details

### enum NcDataSNIAId

```
typedef enum {
    NC_DATA_SNIA_SIMPLE_GOLD_157 = 0,
    NC_DATA_SNIA_SIMPLE_GOLD_182,
    NC_DATA_SNIA_SIMPLE_GOLD_182_FULL,
    NC_DATA_SNIA_SIMPLE_ESSENCE,
    NC_DATA_SNIA_SIMPLE_LEGACY,
    NC_DATA_SNIA_SIMPLE_UNION,
    NC_DATA_SNIA_SIMPLE_CfA3,
    NC_DATA_SNIA_SIMPLE_UNION2,
    NC_DATA_SNIA_SIMPLE_UNION2_1,
    NC_DATA_SNIA_COV_SNLS3_SYS_STAT,
} NcDataSNIAId;
```

FIXME

**NC\_DATA\_SNIA\_SIMPLE\_GOLD\_157** FIXME

**NC\_DATA\_SNIA\_SIMPLE\_GOLD\_182** FIXME

**NC\_DATA\_SNIA\_SIMPLE\_GOLD\_182\_FULL** FIXME

**NC\_DATA\_SNIA\_SIMPLE\_ESSENCE** FIXME

**NC\_DATA\_SNIA\_SIMPLE\_LEGACY** FIXME

**NC\_DATA\_SNIA\_SIMPLE\_UNION** FIXME

**NC\_DATA\_SNIA\_SIMPLE\_CfA3** FIXME

**NC\_DATA\_SNIA\_SIMPLE\_UNION2** FIXME

**NC\_DATA\_SNIA\_SIMPLE\_UNION2\_1** FIXME

**NC\_DATA\_SNIA\_COV\_SNLS3\_SYS\_STAT** FIXME

**NC\_DATA\_SNIA\_COV\_SNLS3\_STAT\_ONLY** FIXME

**NC\_DATA\_SNIA\_SIMPLE\_START**

```
#define NC_DATA_SNIA_SIMPLE_START NC_DATA_SNIA_SIMPLE_GOLD_157
```

**NC\_DATA\_SNIA\_SIMPLE\_END**

```
#define NC_DATA_SNIA_SIMPLE_END NC_DATA_SNIA_SIMPLE_UNION2_1
```

**NC\_DATA\_SNIA\_SIMPLE\_LEN**

```
#define NC_DATA_SNIA_SIMPLE_LEN ((NC_DATA_SNIA_SIMPLE_END) - (NC_DATA_SNIA_SIMPLE_START) + 1) ↵
```

**NC\_DATA\_SNIA\_COV\_START**

```
#define NC_DATA_SNIA_COV_START NC_DATA_SNIA_COV_SNLS3_SYS_STAT
```

**NC\_DATA\_SNIA\_COV\_END**

```
#define NC_DATA_SNIA_COV_END NC_DATA_SNIA_COV_SNLS3_STAT_ONLY
```

**NC\_DATA\_SNIA\_COV\_LEN**

```
#define NC_DATA_SNIA_COV_LEN ((NC_DATA_SNIA_COV_END) - (NC_DATA_SNIA_COV_START) + 1)
```

**nc\_data\_snia\_load\_cat ()**

```
void nc_data_snia_load_cat (NcDataSNIAcov *snia_cov,  
                             NcDataSNIAid id);
```

**nc\_data\_snia\_get\_fits ()**

```
gchar * nc_data_snia_get_fits (const gchar *filename,  
                                gboolean check_size);
```

**nc\_data\_snia\_get\_catalog ()**

```
gchar * nc_data_snia_get_catalog (gchar *id);
```

FIXME

*id*: FIXME

**Returns**: FIXME. *[transfer full]*

**nc\_data\_snia\_get\_catalog\_by\_id ()**

```
gchar *          nc_data_snia_get_catalog_by_id      (NcDataSNIAId id);
```

FIXME

**id**: FIXME

**Returns**: FIXME. *[transfer full]*

## 8.12 Distance Modulus Data

Distance Modulus Data — Data samples of distance modulus

### Synopsis

```
struct          NcDataDistMuClass;
struct          NcDataDistMu;
NcmData *      nc_data_dist_mu_new          (NcDistance *dist,
                                             NcDataSNIAId id);
void           nc_data_dist_mu_set_size    (NcDataDistMu *dist_mu,
                                             guint np);
guint          nc_data_dist_mu_get_size    (NcDataDistMu *dist_mu);
void           nc_data_dist_mu_set_sample  (NcDataDistMu *dist_mu,
                                             NcDataSNIAId id);
NcDataSNIAId   nc_data_dist_mu_get_sample  (NcDataDistMu *dist_mu);
```

### Object Hierarchy

```
GObject
+----NcmData
      +----NcmDataGaussDiag
            +----NcDataDistMu
```

### Properties

"dist"	NcDistance*	: Read / Write / Construct
"sample-id"	NcDataSNIAId	: Read / Write

### Description

FIXME

### Details

#### struct NcDataDistMuClass

```
struct NcDataDistMuClass {
};
```

**struct NcDataDistMu**

```
struct NcDataDistMu;
```

**nc\_data\_dist\_mu\_new ()**

```
NcmData *          nc_data_dist_mu_new          (NcDistance *dist,  
                                                NcDataSNIAId id);
```

FIXME

**dist** : FIXME

**id** : FIXME

**Returns** : FIXME

**nc\_data\_dist\_mu\_set\_size ()**

```
void              nc_data_dist_mu_set_size      (NcDataDistMu *dist_mu,  
                                                quint np);
```

FIXME

**dist\_mu** : a **NcDataDistMu**

**np** : FIXME

**Returns** : FIXME

**nc\_data\_dist\_mu\_get\_size ()**

```
quint            nc_data_dist_mu_get_size      (NcDataDistMu *dist_mu);
```

FIXME

**dist\_mu** : a **NcDataDistMu**

**Returns** : FIXME

**nc\_data\_dist\_mu\_set\_sample ()**

```
void              nc_data_dist_mu_set_sample   (NcDataDistMu *dist_mu,  
                                                NcDataSNIAId id);
```

FIXME

**dist\_mu** : a **NcDataDistMu**.

**id** : FIXME

---



**nc\_data\_dist\_mu\_get\_sample ()**

NcDataSNIAId	nc_data_dist_mu_get_sample	(NcDataDistMu *dist_mu);
--------------	----------------------------	--------------------------

FIXME

*dist\_mu* : a **NcDataDistMu**

*Returns* : FIXME

**Property Details**

**The "dist" property**

"dist"	NcDistance*	: Read / Write / Construct
--------	-------------	----------------------------

Distance object.

**The "sample-id" property**

"sample-id"	NcDataSNIAId	: Read / Write
-------------	--------------	----------------

Sample id.

Default value: NC\_DATA\_SNIA\_SIMPLE\_UNION2\_1

**8.13 Supernovae Ia Data -- Covariance**

Supernovae Ia Data -- Covariance — SNIa data with covariance error matrix

**Synopsis**

```
enum          NcDataSNIAcovData;
#define       NC_DATA_SNIA_COV_LENGTH
struct        NcDataSNIAcovClass;
struct        NcDataSNIAcov;
NcmData *     nc_data_snia_cov_new          (gboolean use_det);
NcmData *     nc_data_snia_cov_new_full    (gchar *filename,
                                           gboolean use_det);

void          nc_data_snia_cov_set_size    (NcDataSNIAcov *snia_cov,
                                           guint mu_len);

void          nc_data_snia_cov_load_txt    (NcDataSNIAcov *snia_cov,
                                           const gchar *filename);

void          nc_data_snia_cov_load       (NcDataSNIAcov *snia_cov,
                                           const gchar *filename);

void          nc_data_snia_cov_save       (NcDataSNIAcov *snia_cov,
                                           const gchar *filename,
                                           gboolean overwrite);

#define       NC_DATA_SNIA_COV_DATA_GROUP
#define       NC_DATA_SNIA_COV_DATA_LEN_KEY
#define       NC_DATA_SNIA_COV_DATA_KEY
#define       NC_DATA_SNIA_COV_MAG_KEY
#define       NC_DATA_SNIA_COV_WIDTH_KEY
```

```
#define NC_DATA_SNIA_COV_COLOUR_KEY
#define NC_DATA_SNIA_COV_MAG_WIDTH_KEY
#define NC_DATA_SNIA_COV_MAG_COLOUR_KEY
#define NC_DATA_SNIA_COV_WIDTH_COLOUR_KEY
```

## Object Hierarchy

```
GObject
+----NcmData
      +----NcmDataGaussCov
            +----NcDataSNIACov
```

## Properties

"data-filename"	gchar*	: Read / Write / Construct
"sigma-pecz"	gdouble	: Read / Write / Construct

## Description

See [NcSNIADistCov](#).

## Details

### enum NcDataSNIACovData

```
typedef enum {
    NC_DATA_SNIA_COV_ZCMB = 0,
    NC_DATA_SNIA_COV_ZHE,
    NC_DATA_SNIA_COV_SIGMA_Z,
    NC_DATA_SNIA_COV_MAG,
    NC_DATA_SNIA_COV_SIGMA_MAG,
    NC_DATA_SNIA_COV_WIDTH,
    NC_DATA_SNIA_COV_SIGMA_WIDTH,
    NC_DATA_SNIA_COV_COLOUR,
    NC_DATA_SNIA_COV_SIGMA_COLOUR,
    NC_DATA_SNIA_COV_THIRDPAR,
    NC_DATA_SNIA_COV_SIGMA_THIRDPAR,
    NC_DATA_SNIA_COV_DIAG_MAG_WIDTH,
    NC_DATA_SNIA_COV_DIAG_MAG_COLOUR,
    NC_DATA_SNIA_COV_DIAG_WIDTH_COLOUR,
    NC_DATA_SNIA_COV_ABSMAG_SET,
    NC_DATA_SNIA_COV_VAR_MAG,
    NC_DATA_SNIA_COV_VAR_WIDTH,
    NC_DATA_SNIA_COV_VAR_COLOUR,
    NC_DATA_SNIA_COV_VAR_MAG_WIDTH,
    NC_DATA_SNIA_COV_VAR_MAG_COLOUR,
} NcDataSNIACovData;
```

FIXME

**NC\_DATA\_SNIA\_COV\_ZCMB** FIXME

**NC\_DATA\_SNIA\_COV\_ZHE** FIXME

**NC\_DATA\_SNIA\_COV\_SIGMA\_Z** FIXME

```

NC_DATA_SNIA_COV_MAG FIXME
NC_DATA_SNIA_COV_SIGMA_MAG FIXME
NC_DATA_SNIA_COV_WIDTH FIXME
NC_DATA_SNIA_COV_SIGMA_WIDTH FIXME
NC_DATA_SNIA_COV_COLOUR FIXME
NC_DATA_SNIA_COV_SIGMA_COLOUR FIXME
NC_DATA_SNIA_COV_THIRDPAR FIXME
NC_DATA_SNIA_COV_SIGMA_THIRDPAR FIXME
NC_DATA_SNIA_COV_DIAG_MAG_WIDTH FIXME
NC_DATA_SNIA_COV_DIAG_MAG_COLOUR FIXME
NC_DATA_SNIA_COV_DIAG_WIDTH_COLOUR FIXME
NC_DATA_SNIA_COV_ABSMAG_SET FIXME
NC_DATA_SNIA_COV_VAR_MAG FIXME
NC_DATA_SNIA_COV_VAR_WIDTH FIXME
NC_DATA_SNIA_COV_VAR_COLOUR FIXME
NC_DATA_SNIA_COV_VAR_MAG_WIDTH FIXME
NC_DATA_SNIA_COV_VAR_MAG_COLOUR FIXME
NC_DATA_SNIA_COV_VAR_WIDTH_COLOUR FIXME

```

```
NC_DATA_SNIA_COV_LENGTH
```

```
#define NC_DATA_SNIA_COV_LENGTH NC_DATA_SNIA_COV_ABSMAG_SET
```

```
struct NcDataSNIAcCovClass
```

```
struct NcDataSNIAcCovClass {
};
```

```
struct NcDataSNIAcCov
```

```
struct NcDataSNIAcCov;
```

```
nc_data_snia_cov_new ()
```

```
NcmData * nc_data_snia_cov_new (gboolean use_det);
```

```
FIXME
```

```
use_det : FIXME
```

```
Returns : FIXME
```

**nc\_data\_snia\_cov\_new\_full ()**

```
NcmData *          nc_data_snia_cov_new_full      (gchar *filename,  
                                                  gboolean use_det);
```

FIXME

**filename** : FIXME**use\_det** : FIXME**Returns** : FIXME**nc\_data\_snia\_cov\_set\_size ()**

```
void              nc_data_snia_cov_set_size      (NcDataSNIACov *snia_cov,  
                                                  guint mu_len);
```

FIXME

**snia\_cov** : FIXME**mu\_len** : FIXME**nc\_data\_snia\_cov\_load\_txt ()**

```
void              nc_data_snia_cov_load_txt      (NcDataSNIACov *snia_cov,  
                                                  const gchar *filename);
```

FIXME

**snia\_cov** : FIXME**filename** : FIXME**nc\_data\_snia\_cov\_load ()**

```
void              nc_data_snia_cov_load          (NcDataSNIACov *snia_cov,  
                                                  const gchar *filename);
```

FIXME

**snia\_cov** : FIXME**filename** : FIXME**nc\_data\_snia\_cov\_save ()**

```
void              nc_data_snia_cov_save          (NcDataSNIACov *snia_cov,  
                                                  const gchar *filename,  
                                                  gboolean overwrite);
```

FIXME

**snia\_cov** : FIXME**filename** : FIXME**overwrite** : FIXME

**NC\_DATA\_SNIA\_COV\_DATA\_GROUP**

```
#define NC_DATA_SNIA_COV_DATA_GROUP "Supernovae Ia Data"
```

**NC\_DATA\_SNIA\_COV\_DATA\_LEN\_KEY**

```
#define NC_DATA_SNIA_COV_DATA_LEN_KEY "data-length"
```

**NC\_DATA\_SNIA\_COV\_DATA\_KEY**

```
#define NC_DATA_SNIA_COV_DATA_KEY "snia-data"
```

**NC\_DATA\_SNIA\_COV\_MAG\_KEY**

```
#define NC_DATA_SNIA_COV_MAG_KEY "magnitude"
```

**NC\_DATA\_SNIA\_COV\_WIDTH\_KEY**

```
#define NC_DATA_SNIA_COV_WIDTH_KEY "width"
```

**NC\_DATA\_SNIA\_COV\_COLOUR\_KEY**

```
#define NC_DATA_SNIA_COV_COLOUR_KEY "colour"
```

**NC\_DATA\_SNIA\_COV\_MAG\_WIDTH\_KEY**

```
#define NC_DATA_SNIA_COV_MAG_WIDTH_KEY "magnitude-width"
```

**NC\_DATA\_SNIA\_COV\_MAG\_COLOUR\_KEY**

```
#define NC_DATA_SNIA_COV_MAG_COLOUR_KEY "magnitude-colour"
```

**NC\_DATA\_SNIA\_COV\_WIDTH\_COLOUR\_KEY**

```
#define NC_DATA_SNIA_COV_WIDTH_COLOUR_KEY "width-colour"
```

**Property Details**

**The "data-filename" property**

"data-filename"	gchar*	: Read / Write / Construct
-----------------	--------	----------------------------

Data filename.  
Default value: NULL

## The "sigma-pecz" property

```
"sigma-pecz"      gdouble      : Read / Write / Construct
```

Error from SN Ia peculiar velocity.

Allowed values: [0,10]

Default value: 0.0005

## 8.14 Cluster number count data

Cluster number count data — FIXME

## Synopsis

[illegible]

```

gsl_histogram2d *   nc_data_cluster_ncount_hist_lnM_z   (NcmData *data,
                                                         gsl_vector *lnM_nodes,
                                                         gsl_vector *z_nodes);
void               nc_data_cluster_ncount_print         (NcmData *data,
                                                         NcHICosmo *cosmo,
                                                         FILE *out,
                                                         gchar *header);
void               nc_data_cluster_ncount_catalog_save  (NcmData *data,
                                                         gchar *filename,
                                                         gboolean overwrite);
void               nc_data_cluster_ncount_catalog_load (NcmData *data,
                                                         gchar *filename);

```

## Object Hierarchy

```

GObject
+----NcmData
      +----NcDataClusterNCount

```

## Properties

```

"cluster-abundance"      NcClusterAbundance*      : Read / Write / Construct

```

## Description

FIXME

## Details

### enum NcDataClusterAbundanceId

```

typedef enum {
    NC_DATA_CLUSTER_ABUNDANCE_FIT,
    NC_DATA_CLUSTER_ABUNDANCE_TXT,
} NcDataClusterAbundanceId;

```

**NC\_DATA\_CLUSTER\_ABUNDANCE\_FIT** FIXME

**NC\_DATA\_CLUSTER\_ABUNDANCE\_TXT** FIXME

**NC\_DATA\_CLUSTER\_ABUNDANCE\_SAMPLING** FIXME

### struct NcDataClusterNCountClass

```

struct NcDataClusterNCountClass {
};

```

### struct NcDataClusterNCount

```

struct NcDataClusterNCount;

```

```
NcmData * nc_data_cluster_ncount_new (NcClusterAbundance *cad);
```

***cad* : FIXME**

**Returns :** FIXME

```
NcDataClusterNCount * nc_data_cluster_ncount_ref (NcDataClusterNCount *ncount);
```

ncount : FIXME

**Returns :** FIXME. *[transfer full]*

```
void nc_data_cluster_ncount_free (NcDataClusterNCount *ncount);
```

*ncount* : FIXME

```
NcmData * nc_data_cluster_ncount_binned_new (NcClusterAbundance *cad);
```

[illegible]

```
void nc_data_cluster_ncount_binned_init_from_sampling
(
    NcmData *data,
    NcmMSet *mset,
    NcmVector *nodes,
    gboolean obs,
    gdouble area_survey,
    gdouble lnMi,
    gdouble lnMf,
    gdouble photoz_sigma0,
    gdouble photoz_bias,
    gdouble lnM_sigma0,
    gdouble lnM_bias);
```



**nc\_data\_cluster\_ncount\_binned\_save ()**

```
void                nc_data_cluster_ncount_binned_save (NcmData *data,
                                                         gchar *filename);
```

**nc\_data\_cluster\_ncount\_binned\_create\_func ()**

```
NcmMSetFunc *      nc_data_cluster_ncount_binned_create_func
                                                         (NcClusterAbundance *cad);
```

FIXME

**cad** : a **NcClusterAbundance**

**Returns** : FIXME

**nc\_data\_cluster\_ncount\_binned\_lnM\_z\_new ()**

```
NcmData *          nc_data_cluster_ncount_binned_lnM_z_new
                                                         (NcClusterAbundance *cad);
```

FIXME

**cad** : a **NcClusterAbundance**

**Returns** : FIXME

**nc\_data\_cluster\_ncount\_true\_data ()**

```
void                nc_data_cluster_ncount_true_data (NcmData *data,
                                                         gboolean use_true_data);
```

FIXME

**data** : a **NcmData**.

**use\_true\_data** : FIXME

**nc\_data\_cluster\_ncount\_init\_from\_sampling ()**

```
void                nc_data_cluster_ncount_init_from_sampling
                                                         (NcmData *data,
                                                         NcmMSet *mset,
                                                         NcClusterRedshift *clusterz,
                                                         NcClusterMass *clusterm,
                                                         gdouble area_survey);
```

FIXME

**data** : a **NcmData**.

**mset** : a **NcmMSet**.

**clusterz** : a **NcClusterRedshift**.

**clusterm** : a **NcClusterMass**.

**area\_survey** : area in units of square degrees.

**nc\_data\_cluster\_ncount\_bin\_data ()**

```
NcmData *      nc_data_cluster_ncount_bin_data (NcmData *data,
                                                gsl_vector *nodes);
```

FIXME

**data** : a **NcmData**

**nodes** : FIXME

**Returns** : FIXME

**nc\_data\_cluster\_ncount\_hist\_lnM\_z ()**

```
gsl_histogram2d * nc_data_cluster_ncount_hist_lnM_z (NcmData *data,
                                                    gsl_vector *lnM_nodes,
                                                    gsl_vector *z_nodes);
```

FIXME

**data** : a **NcmData**.

**lnM\_nodes** : FIXME

**z\_nodes** : FIXME

**Returns** : FIXME

**nc\_data\_cluster\_ncount\_print ()**

```
void      nc_data_cluster_ncount_print (NcmData *data,
                                         NcHICosmo *cosmo,
                                         FILE *out,
                                         gchar *header);
```

FIXME

**data** : FIXME

**cosmo** : a NcHICosmo

**out** : FIXME

**header** : FIXME

**nc\_data\_cluster\_ncount\_catalog\_save ()**

```
void      nc_data_cluster_ncount_catalog_save (NcmData *data,
                                              gchar *filename,
                                              gboolean overwrite);
```

FIXME

**data** : a **NcmData**

**filename** : name of the file

**overwrite** : FIXME

**nc\_data\_cluster\_ncount\_catalog\_load ()**

```
void nc_data_cluster_ncount_catalog_load (NcmData *data,
                                           gchar *filename);
```

FIXME

**data** : a **NcmData**.  
**filename** : name of the file

**Property Details**

**The "cluster-abundance" property**

"cluster-abundance"	NcClusterAbundance*	: Read / Write / Construct
---------------------	---------------------	----------------------------

Cluster Abundance.

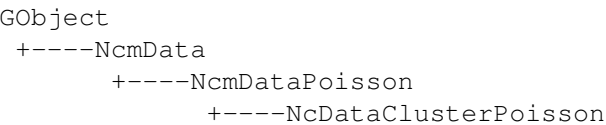
**8.15 Cluster number count data**

Cluster number count data — FIXME

**Synopsis**

```
struct NcDataClusterPoissonClass;
struct NcDataClusterPoisson;
NcmData * nc_data_cluster_poisson_new_cad (NcClusterAbundance *cad);
NcmData * nc_data_cluster_poisson_new (NcDataClusterNCount *ncount);
```

**Object Hierarchy**



**Description**

FIXME

**Details**

**struct NcDataClusterPoissonClass**

```
struct NcDataClusterPoissonClass {
};
```

**struct NcDataClusterPoisson**

```
struct NcDataClusterPoisson;
```

**nc\_data\_cluster\_poisson\_new\_cad ()**

```
NcmData * nc_data_cluster_poisson_new_cad (NcClusterAbundance *cad);
```

FIXME

**cad** : a **NcClusterAbundance**.

**Returns** : FIXME

**nc\_data\_cluster\_poisson\_new ()**

```
NcmData * nc_data_cluster_poisson_new (NcDataClusterNCount *ncount);
```

FIXME

**ncount** : a **NcClusterAbundance**.

**Returns** : FIXME

## Chapter 9

# Mathematical Utilities

### 9.1 Divided Difference

Divided Difference — Divided difference methods for function interpolation with derivatives

#### Synopsis

void	<code>ncm_interp_dd_init</code>	<code>(const gdouble *vx, gdouble *dd, const gint np, const gint nf);</code>
gdouble	<code>ncm_interp_dd_eval</code>	<code>(const gdouble *vx, const gdouble *dd, const gdouble x, const gint np, const gint nf);</code>
void	<code>ncm_interp_dd_init_2_4</code>	<code>(const gdouble *vx, gdouble *dd);</code>
gdouble	<code>ncm_interp_dd_eval_2_4</code>	<code>(const gdouble *vx, const gdouble *dd, const gdouble x);</code>

#### Description

FIXME

#### Details

##### `ncm_interp_dd_init()`

void	<code>ncm_interp_dd_init</code>	<code>(const gdouble *vx, gdouble *dd, const gint np, const gint nf);</code>
------	---------------------------------	--

**ncm\_interp\_dd\_eval ()**

gdouble	ncm_interp_dd_eval	(const gdouble *vx, const gdouble *dd, const gdouble x, const gint np, const gint nf);
---------	--------------------	--

**ncm\_interp\_dd\_init\_2\_4 ()**

void	ncm_interp_dd_init_2_4	(const gdouble *vx, gdouble *dd);
------	------------------------	--------------------------------------

**ncm\_interp\_dd\_eval\_2\_4 ()**

gdouble	ncm_interp_dd_eval_2_4	(const gdouble *vx, const gdouble *dd, const gdouble x);
---------	------------------------	--

## 9.2 Binnary Splitting

Binnary Splitting — Binnary splitting algorithms used to evaluate sums fast and with arbitrary precision

### Synopsis

struct	NcmBinSplit;	
extern mpz_t	NCM_BINSPLIT_ONE;	
void	(*NcmBinSplitEval)	(NcmBinSplit *bs, gulong n1, gulong n2);
NcmBinSplit *	ncm_binsplit_alloc	(gpointer userdata);
glong	ncm_binsplit_test_next	(NcmBinSplit *bs, NcmBinSplitEval bs_eval, gulong nt);
void	ncm_binsplit_join	(NcmBinSplit *bs, NcmBinSplit *bs_l, NcmBinSplit *bs_r);
void	ncm_binsplit_eval_join	(NcmBinSplit *bs, NcmBinSplitEval bs_eval, gulong nt);
gulong	ncm_binsplit_eval_prec	(NcmBinSplit *bs, NcmBinSplitEval bs_eval, gulong step, glong prec);
void	ncm_binsplit_get	(NcmBinSplit *bs, mpfr_t res);
void	ncm_binsplit_get_q	(NcmBinSplit *bs, mpq_t q);
gdouble	ncm_binsplit_get_d	(NcmBinSplit *bs, mp_rnd_t rnd);
#define	NCM_BINSPLIT_DECL	(name,

```

v,
u,
n,
data)
#define          NCM_BINSPLIT_DENC_NULL (a,
b,
c,
d)
```

Description

FIXME

Details

struct NcmBinSplit

```
struct NcmBinSplit {
};
```

FIXME

NCM\_BINSPLIT\_ONE

```
extern mpz_t NCM_BINSPLIT_ONE;
```

NcmBinSplitEval ()

```
void          (*NcmBinSplitEval)          (NcmBinSplit *bs,
gulong n1,
gulong n2);
```

ncm\_binsplit\_alloc ()

```
NcmBinSplit *      ncm_binsplit_alloc          (gpointer userdata);
```

FIXME

**userdata** : FIXME

**Returns** : FIXME

ncm\_binsplit\_test\_next ()

```
glong          ncm_binsplit_test_next          (NcmBinSplit *bs,
NcmBinSplitEval bs_eval,
gulong nt);
```

FIXME

**bs** : a **NcmBinSplit**

**bs\_eval** : a **NcmBinSplitEval**

**nt** : FIXME

**Returns** : FIXME

**ncm\_binsplit\_join ()**

void	ncm_binsplit_join	(NcmBinSplit *bs, NcmBinSplit *bs_l, NcmBinSplit *bs_r);
------	-------------------	--

FIXME

**bs** : a **NcmBinSplit****bs\_l** : a **NcmBinSplit****bs\_r** : a **NcmBinSplit****ncm\_binsplit\_eval\_join ()**

void	ncm_binsplit_eval_join	(NcmBinSplit *bs, NcmBinSplitEval bs_eval, gulong nt);
------	------------------------	--

FIXME

**bs** : a **NcmBinSplit****bs\_eval** : a **NcmBinSplitEval****nt** : FIXME**ncm\_binsplit\_eval\_prec ()**

gulong	ncm_binsplit_eval_prec	(NcmBinSplit *bs, NcmBinSplitEval bs_eval, gulong step, glong prec);
--------	------------------------	---

FIXME

**bs** : a **NcmBinSplit****bs\_eval** : a **NcmBinSplitEval****step** : FIXME**prec** : FIXME**Returns** : FIXME**ncm\_binsplit\_get ()**

void	ncm_binsplit_get	(NcmBinSplit *bs, mpfr_t res);
------	------------------	-----------------------------------

FIXME

**bs** : a **NcmBinSplit****res** : FIXME



```
void ncm_binsplit_get_q(NcmBinSplit *bs, mpq_t q);
```

**bs** : a **NcmBinSplit**

### ncm\_binsplit\_get\_d()

```
gdouble          ncm_binsplit_get_d      (NcmBinSplit *bs,
                                           mp_rnd_t rnd);
```

**bs** : a **NcmBinSplit**

**Returns :** FIXME

```
#define NCM_BINSPLIT_DECL(name,v,u,n,data) static inline void name (mpz_t v, mpz_t u, ←
    gulong n, gpointer data)
```

```
#define NCM_BINSPLIT_DENC_NULL(a,b,c,d)
```

## Polynomials — FIXME

```
gsl_vector *      ncm_poly_new          (gint degree);
gdouble           ncm_poly_eval         (gsl_vector *poly,
                                         gdouble x);

gdouble           ncm_poly_eval_diff    (gsl_vector *poly,
                                         gint n,
                                         gdouble x);

gdouble           ncm_poly_eval_int     (gsl_vector *poly,
                                         gdouble x);

gdouble           ncm_poly_eval_int_P_over_x (gsl_vector *poly,
                                         gdouble x);
```

Description

FIXME

Details

**ncm\_poly\_new ()**

gsl_vector *	ncm_poly_new	(gint degree);
--------------	--------------	----------------

FIXME

***degree*** : FIXME

***Returns*** : FIXME

**ncm\_poly\_eval ()**

gdouble	ncm_poly_eval	(gsl_vector *poly, gdouble x);
---------	---------------	-----------------------------------

FIXME

***poly*** : FIXME

***x*** : FIXME

***Returns*** : FIXME

**ncm\_poly\_eval\_diff ()**

gdouble	ncm_poly_eval_diff	(gsl_vector *poly, gint n, gdouble x);
---------	--------------------	--

FIXME

***poly*** : FIXME

***n*** : FIXME

***x*** : FIXME

***Returns*** : FIXME

**ncm\_poly\_eval\_int ()**

gdouble	ncm_poly_eval_int	(gsl_vector *poly, gdouble x);
---------	-------------------	-----------------------------------

FIXME

***poly*** : FIXME

***x*** : FIXME

***Returns*** : FIXME

---

**ncm\_poly\_eval\_int\_P\_over\_x()**

```
gdouble          ncm_poly_eval_int_P_over_x      (gsl_vector *poly,
                                                  gdouble x);
```

FIXME

*poly* : FIXME

*x* : FIXME

*Returns* : FIXME

## 9.4 Quaternions

Quaternions — Quaternions algebra and mapping to matrix

### Synopsis

```
struct          NcmTriVector;
#define         NC_TRIVEC_SCALE          (a,
                                          b)

#define         NC_TRIVEC_NEW
#define         NC_TRIVEC_SET_0          (v)
#define         NC_TRIVEC_MEMCPY        (a,
                                          b)

#define         NC_TRIVEC_NORM          (a)
#define         NC_TRIVEC_NORMALIZE     (a)
#define         NC_TRIVEC_DOT           (a,
                                          b)

struct          NcmQ;
#define         NC_QUATERNION_NEW
#define         NC_QUATERNION_NEW_I
#define         NC_QUATERNION_SET_I     (q)
#define         NC_QUATERNION_SET_0    (q)
#define         NC_QUATERNION_NORM     (q)
#define         NC_QUATERNION_MEMCPY   (a,
                                          b)

NcmQ *          ncm_quaternion_new      ();
NcmQ *          ncm_quaternion_new_from_vector (NcmTriVector v);
NcmQ *          ncm_quaternion_new_from_data (gdouble x,
                                              gdouble y,
                                              gdouble z,
                                              gdouble theta);

void           ncm_quaternion_set_from_data (NcmQ *q,
                                              gdouble x,
                                              gdouble y,
                                              gdouble z,
                                              gdouble theta);

void           ncm_quaternion_set_random (NcmQ *q);
void           ncm_quaternion_free      (NcmQ *q);
void           ncm_quaternion_normalize (NcmQ *q);
void           ncm_quaternion_conjugate (NcmQ *q);
void           ncm_quaternion_mul       (NcmQ *q,
                                          NcmQ *u,
```

```

void          ncm_quaternion_lmul          NcmQ *res);
void          ncm_quaternion_rmul          (NcmQ *q,
void          ncm_quaternion_conjugate_u_mul (NcmQ *q,
void          ncm_quaternion_conjugate_q_mul (NcmQ *q,
void          ncm_quaternion_rotate          (NcmQ *q,
void          ncm_quaternion_inv_rotate      (NcmQ *q,

```

## Description

FIXME

## Details

### struct NcmTriVector

```

struct NcmTriVector {
};

```

FIXME

### NC\_TRIVEC\_SCALE()

```

#define NC_TRIVEC_SCALE(a, b)

```

### NC\_TRIVEC\_NEW

```

#define NC_TRIVEC_NEW {{0.0, 0.0, 0.0}}

```

### NC\_TRIVEC\_SET\_0()

```

#define NC_TRIVEC_SET_0(v) memset(&(v), 0, sizeof(NcmTriVector))

```

### NC\_TRIVEC\_MEMCPY()

```

#define NC_TRIVEC_MEMCPY(a, b) memcpy (&(a), &(b), sizeof (NcmTriVector))

```

### NC\_TRIVEC\_NORM()

```

#define NC_TRIVEC_NORM(a) sqrt((a).c[0]*(a).c[0] + (a).c[1]*(a).c[1] + (a).c[2]*(a).c[2])

```

**NC\_TRIVEC\_NORMALIZE()**

```
#define NC_TRIVEC_NORMALIZE(a) NC_TRIVEC_SCALE(a, 1.0/NC_TRIVEC_NORM(a))
```

**NC\_TRIVEC\_DOT()**

```
#define NC_TRIVEC_DOT(a, b) ((a).c[0]*(b).c[0] + (a).c[1]*(b).c[1] + (a).c[2]*(b).c[2])
```

**struct NcmQ**

```
struct NcmQ {  
};
```

FIXME

**NC\_QUATERNION\_NEW**

```
#define NC_QUATERNION_NEW {0.0, {{0.0, 0.0, 0.0}}}
```

**NC\_QUATERNION\_NEW\_I**

```
#define NC_QUATERNION_NEW_I {1.0, {{0.0, 0.0, 0.0}}}
```

**NC\_QUATERNION\_SET\_I()**

```
#define NC_QUATERNION_SET_I(q)
```

FIXME

$q$ : FIXME

**NC\_QUATERNION\_SET\_0()**

```
#define NC_QUATERNION_SET_0(q)
```

FIXME

$q$ : FIXME

**NC\_QUATERNION\_NORM()**

```
#define NC_QUATERNION_NORM(q)
```

FIXME

$q$ : FIXME

---

**NC\_QUATERNION\_MEMCPY()**

```
#define NC_QUATERNION_MEMCPY(a,b) memcpy (a, b, sizeof(NcmQ))
```

FIXME

**a** : FIXME

**b** : FIXME

**ncm\_quaternion\_new ()**

```
NcmQ * ncm_quaternion_new ();
```

FIXME

**Returns** : FIXME

**ncm\_quaternion\_new\_from\_vector ()**

```
NcmQ * ncm_quaternion_new_from_vector (NcmTriVector v);
```

FIXME

**v** : a **NcmTriVector**

**Returns** : FIXME

**ncm\_quaternion\_new\_from\_data ()**

```
NcmQ * ncm_quaternion_new_from_data (gdouble x,
                                     gdouble y,
                                     gdouble z,
                                     gdouble theta);
```

FIXME

**x** : FIXME

**y** : FIXME

**z** : FIXME

**theta** : FIXME

**Returns** : FIXME

**ncm\_quaternion\_set\_from\_data ()**

```
void                ncm_quaternion_set_from_data      (NcmQ *q,  
                                                       gdouble x,  
                                                       gdouble y,  
                                                       gdouble z,  
                                                       gdouble theta);
```

FIXME

 $q$ : FIXME $x$ : FIXME $y$ : FIXME $z$ : FIXME $\theta$ : FIXME**ncm\_quaternion\_set\_random ()**

```
void                ncm_quaternion_set_random        (NcmQ *q);
```

FIXME

 $q$ : FIXME**ncm\_quaternion\_free ()**

```
void                ncm_quaternion_free              (NcmQ *q);
```

FIXME

 $q$ : a **NcmQ****ncm\_quaternion\_normalize ()**

```
void                ncm_quaternion_normalize         (NcmQ *q);
```

FIXME

 $q$ : FIXME**ncm\_quaternion\_conjugate ()**

```
void                ncm_quaternion_conjugate         (NcmQ *q);
```

FIXME

 $q$ : FIXME

**ncm\_quaternion\_mul ()**

```
void          ncm_quaternion_mul          (NcmQ *q,  
                                           NcmQ *u,  
                                           NcmQ *res);
```

FIXME

***q*** : FIXME***u*** : FIXME***res*** : FIXME**ncm\_quaternion\_lmul ()**

```
void          ncm_quaternion_lmul        (NcmQ *q,  
                                           NcmQ *u);
```

FIXME

***q*** : FIXME***u*** : FIXME**ncm\_quaternion\_rmul ()**

```
void          ncm_quaternion_rmul        (NcmQ *q,  
                                           NcmQ *u);
```

FIXME

***q*** : FIXME***u*** : FIXME**ncm\_quaternion\_conjugate\_u\_mul ()**

```
void          ncm_quaternion_conjugate_u_mul (NcmQ *q,  
                                              NcmQ *u,  
                                              NcmQ *res);
```

FIXME

***q*** : FIXME***u*** : FIXME***res*** : FIXME



**ncm\_quaternion\_conjugate\_q\_mul ()**

```
void ncm_quaternion_conjugate_q_mul(NcmQ *q,
                                     NcmQ *u,
                                     NcmQ *res);
```

FIXME

 $q: \text{FIXME}$ 

***u* : FIXME**

```
res : FIXME
```

### ncm quaternion rotate ()

```
void ncm_quaternion_rotate (NcmQ *q,
                             NcmTriVector v);
```

FIXME

 $q : \text{FIXME}$ 

**v** : FIXME

**ncm quaternion inv\_rotate ()**

```
void ncm_quaternion_inv_rotate(NcmQ *q, NcmTriVector v);
```

FIXME

 $q: \text{FIXME}$ 

**v** : FIXME

## 9.5 Miscellaneous Utilities

## Miscellaneous Utilities — FIXME

## Synopsis

```

gulong      ncm_random_seed      (void);
void         ncm_finite_diff_calc_J(NcmModel *model,
                                     NcmData *data,
                                     NcmMatrix *jac);
gdouble *   ncm_smoothd          (gdouble *in,
                                   size_t N,
                                   size_t points,
                                   size_t pass);
gboolean     ncm_get_uniform_sample(NcmMSet *mset,
                                    NcmMSetFunc *func,
                                    gdouble x0,
                                    gdouble x1,

```

gboolean	ncm_get_smoothed_uniform_sample	NcmVector *sample); (NcmMSet *mset, NcmMSetFunc *func, gdouble x0, gdouble x1, gdouble delta, NcmVector *sample);
gdouble	ncm_topology_comoving_a0_lss	(guint n, gdouble alpha);
gdouble	ncm_topology_sigma_comoving_a0_lss	(guint n, gdouble alpha, gdouble sigma_alpha);
void	ncm_rational_coarce_double	(gdouble x, mpq_t q);
gdouble	ncm_sphPlm_x	(gint l, gint m, gint order);
gdouble	ncm_sphPlm_test_theta	(gdouble theta, gint lmax, gint *lmin_data);
gsize	ncm_mpfr_out_raw	(FILE *stream, mpfr_t op);
gsize	ncm_mpfr_inp_raw	(mpfr_t rop, FILE *stream);
gsize	ncm_mpq_out_raw	(FILE *f, mpq_t q);
gsize	ncm_mpq_inp_raw	(mpq_t q, FILE *f);
void	ncm_mpz_inits	(mpz_t z, ...);
void	ncm_mpz_clears	(mpz_t z, ...);
gdouble	ncm_sum	(gdouble *d, gulong n);
gdouble	ncm_numdiff_1	(gsl_function *F, const gdouble x, const gdouble ho, gdouble *err);
gdouble	ncm_numdiff_2	(gsl_function *F, gdouble *ofx, const gdouble x, const gdouble ho, gdouble *err);
gdouble	ncm_numdiff_2_err	(gsl_function *F, gdouble *ofx, const gdouble x, const gdouble ho, gdouble err, gdouble *ferr);
gdouble	ncm_sqrtlpx_m1	(gdouble x);
gint	ncm_cmp	(gdouble x, gdouble y, gdouble reltol);
#define	ncm_assert_cmpdouble	(n1, cmp, n2)
#define	ncm_assert_cmpdouble_e	(n1,

```
cmp,  
n2,  
epsilon)
```

Description

FIXME

Details

ncm\_random\_seed ()

```
gulong          ncm_random_seed          (void);
```

FIXME

*Returns* : FIXME

ncm\_finite\_diff\_calc\_J ()

```
void          ncm_finite_diff_calc_J          (NcmModel *model,  
                                                NcmData *data,  
                                                NcmMatrix *jac);
```

FIXME

*model* : FIXME

*data* : FIXME

*jac* : FIXME

ncm\_smoothd ()

```
gdouble *      ncm_smoothd          (gdouble *in,  
                                      size_t N,  
                                      size_t points,  
                                      size_t pass);
```

FIXME

*in* : FIXME

*N* : FIXME

*points* : FIXME

*pass* : FIXME

*Returns* : FIXME

**ncm\_get\_uniform\_sample ()**

```
gboolean          ncm_get_uniform_sample      (NcmMSet *mset,  
                                              NcmMSetFunc *func,  
                                              gdouble x0,  
                                              gdouble x1,  
                                              NcmVector *sample);
```

FIXME

***mset*** : FIXME***func*** : FIXME***x0*** : FIXME***x1*** : FIXME***sample*** : FIXME***Returns*** : FIXME**ncm\_get\_smoothed\_uniform\_sample ()**

```
gboolean          ncm_get_smoothed_uniform_sample (NcmMSet *mset,  
                                                  NcmMSetFunc *func,  
                                                  gdouble x0,  
                                                  gdouble x1,  
                                                  gdouble delta,  
                                                  NcmVector *sample);
```

FIXME

***mset*** : FIXME***func*** : FIXME***x0*** : FIXME***x1*** : FIXME***delta*** : FIXME***sample*** : FIXME***Returns*** : FIXME**ncm\_topology\_comoving\_a0\_lss ()**

```
gdouble          ncm_topology_comoving_a0_lss (guint n,  
                                              gdouble alpha);
```

FIXME

***n*** : FIXME***alpha*** : FIXME***Returns*** : FIXME

**ncm\_topology\_sigma\_comoving\_a0\_lss ()**

```
gdouble          ncm_topology_sigma_comoving_a0_lss  (guint n,
                                                       gdouble alpha,
                                                       gdouble sigma_alpha);
```

FIXME

*n* : FIXME

*alpha* : FIXME

*sigma\_alpha* : FIXME

*Returns* : FIXME

**ncm\_rational\_coarce\_double ()**

```
void          ncm_rational_coarce_double  (gdouble x,
                                           mpq_t q);
```

FIXME

*x* : FIXME

*q* : FIXME

**ncm\_sphPlm\_x ()**

```
gdouble          ncm_sphPlm_x  (gint l,
                                gint m,
                                gint order);
```

FIXME

*l* : FIXME

*m* : FIXME

*order* : FIXME

*Returns* : FIXME

**ncm\_sphPlm\_test\_theta ()**

```
gdouble          ncm_sphPlm_test_theta  (gdouble theta,
                                           gint lmax,
                                           gint *lmin_data);
```

FIXME

*theta* : FIXME

*lmax* : FIXME

*lmin\_data* : FIXME

*Returns* : FIXME

**ncm\_mpfr\_out\_raw ()**

gsize	ncm_mpfr_out_raw	(FILE *stream, mpfr_t op);
-------	------------------	-------------------------------

FIXME

*stream* : FIXME

*op* : FIXME

*Returns* : FIXME

**ncm\_mpfr\_inp\_raw ()**

gsize	ncm_mpfr_inp_raw	(mpfr_t rop, FILE *stream);
-------	------------------	--------------------------------

FIXME

*rop* : FIXME

*stream* : FIXME

*Returns* : FIXME

**ncm\_mpq\_out\_raw ()**

gsize	ncm_mpq_out_raw	(FILE *f, mpq_t q);
-------	-----------------	------------------------

FIXME

*f* : FIXME

*q* : FIXME

*Returns* : FIXME

**ncm\_mpq\_inp\_raw ()**

gsize	ncm_mpq_inp_raw	(mpq_t q, FILE *f);
-------	-----------------	------------------------

FIXME

*q* : FIXME

*f* : FIXME

*Returns* : FIXME

**ncm\_mpz\_inits ()**

```
void                ncm_mpz_inits                (mpz_t z,  
                                                  ...);
```

FIXME

**z** : FIXME

... : FIXME

**ncm\_mpz\_clears ()**

```
void                ncm_mpz_clears              (mpz_t z,  
                                                  ...);
```

FIXME

**z** : FIXME

... : FIXME

**ncm\_sum ()**

```
gdouble            ncm_sum                      (gdouble *d,  
                                                  gulong n);
```

FIXME

**d** : FIXME

**n** : FIXME

**Returns** : FIXME

**ncm\_numdiff\_1 ()**

```
gdouble            ncm_numdiff_1               (gsl_function *F,  
                                                  const gdouble x,  
                                                  const gdouble ho,  
                                                  gdouble *err);
```

FIXME

**F** : FIXME

**x** : FIXME

**ho** : FIXME

**err** : FIXME

**Returns** : FIXME

---

**ncm\_numdiff\_2 ()**

gdouble	ncm_numdiff_2	(gsl_function *F, gdouble *ofx, const gdouble x, const gdouble ho, gdouble *err);
---------	---------------	---

FIXME

***F*** : FIXME

***ofx*** : FIXME

***x*** : FIXME

***ho*** : FIXME

***err*** : FIXME

**Returns** : FIXME

**ncm\_numdiff\_2\_err ()**

gdouble	ncm_numdiff_2_err	(gsl_function *F, gdouble *ofx, const gdouble x, const gdouble ho, gdouble err, gdouble *ferr);
---------	-------------------	--

FIXME

***F*** : FIXME

***ofx*** : FIXME

***x*** : FIXME

***ho*** : FIXME

***err*** : FIXME

***ferr*** : FIXME

**Returns** : FIXME

**ncm\_sqrt1px\_m1 ()**

gdouble	ncm_sqrt1px_m1	(gdouble x);
---------	----------------	--------------

Calculates  $\sqrt{1+x}-1$  using the appropriated taylor series when  $x \approx 1$ .

***x*** : a real number  $>-1$

**Returns** :  $\sqrt{1+x}-1$ .



**ncm\_cmp ()**

```
gint          ncm_cmp          (gdouble x,
                                gdouble y,
                                gdouble reltol);
```

Compare  $x$  and  $y$  and return -1 if  $x < y$ , 0 if  $x == y$  and 1 if  $x > y$ , all comparisons are done with precision  $prec$ .

**$x$**  : a double.

**$y$**  : a double.

**$reltol$**  : relative precision.

**Returns** : -1, 0, 1.

**ncm\_assert\_cmpdouble()**

```
#define          ncm_assert_cmpdouble (n1, cmp, n2)
```

**ncm\_assert\_cmpdouble\_e()**

```
#define          ncm_assert_cmpdouble_e (n1, cmp, n2, epsilon)
```

## 9.6 Matrix Exponential

Matrix Exponential — Simple functions to calculate matrix exponential (only 2x2 to date)

### Synopsis

```
void          ncm_matrix_exp_2x2          (gsl_matrix *B,
                                            gsl_matrix *exp_B);
```

### Description

FIXME

### Details

**ncm\_matrix\_exp\_2x2 ()**

```
void          ncm_matrix_exp_2x2          (gsl_matrix *B,
                                            gsl_matrix *exp_B);
```

FIXME

**$B$**  : FIXME

**$exp\_B$**  : FIXME

## 9.7 MPQ Tree

MPQ Tree — FIXME

### Synopsis

```
GTree *          ncm_mpq_tree_new          (void);
GHashTable *      ncm_mpq_hash_new          (void);
```

### Description

FIXME

### Details

#### ncm\_mpq\_tree\_new ()

```
GTree *          ncm_mpq_tree_new          (void);
```

FIXME

**Returns :** FIXME

#### ncm\_mpq\_hash\_new ()

```
GHashTable *      ncm_mpq_hash_new          (void);
```

FIXME

**Returns :** FIXME. *[transfer full]*

## 9.8 Sundials CVODE interface

Sundials CVODE interface — FIXME

### Synopsis

```
N_Vector          ncm_cvode_util_nvector_new      (guint n);
gboolean          ncm_cvode_util_check_flag      (gpointer flagvalue,
gchar *funcname,
gint opt);
gboolean          ncm_cvode_util_print_stats      (gpointer cvode);
#define           CVODE_CHECK                     (chk,
name,
val,
ret)
```

### Description

FIXME

---

Details

**ncm\_cvode\_util\_nvector\_new ()**

```
N_Vector          ncm_cvode_util_nvector_new      (guint n);
```

FIXME

*n* : FIXME

**Returns** : FIXME

**ncm\_cvode\_util\_check\_flag ()**

```
gboolean          ncm_cvode_util_check_flag      (gpointer flagvalue,  
gchar *funcname,  
gint opt);
```

FIXME

**flagvalue** : FIXME

**funcname** : FIXME

**opt** : FIXME

**Returns** : FIXME

**ncm\_cvode\_util\_print\_stats ()**

```
gboolean          ncm_cvode_util_print_stats      (gpointer cvode);
```

FIXME

**cvode** : FIXME

**Returns** : FIXME

**CVODE\_CHECK()**

```
#define          CVODE_CHECK(chk,name,val,ret)
```

9.9 Magnus Iserles Ode Method

Magnus Iserles Ode Method — Ode solver for fast oscillating systems

## Synopsis

```

struct          NcmMIODEFunction;
struct          NcmMIODE;
NcmMIODE *      ncm_magnus_iserles_ode_new      (long double x0,
                                                NcmMIODEFunction *g);
gboolean        ncm_magnus_iserles_ode_step     (NcmMIODE *mi_ode,
                                                long double h);
gboolean        ncm_magnus_iserles_ode_step_frac (NcmMIODE *mi_ode,
                                                long double frac);
void            ncm_magnus_iserles_ode_eval_vec (NcmMIODE *mi_ode,
                                                long double u_i,
                                                long double up_i,
                                                long double *u,
                                                long double *up);

```

## Description

FIXME

## Details

### struct NcmMIODEFunction

```

struct NcmMIODEFunction {
};

```

FIXME

### struct NcmMIODE

```

struct NcmMIODE {
};

```

FIXME

### ncm\_magnus\_iserles\_ode\_new ()

```

NcmMIODE *      ncm_magnus_iserles_ode_new      (long double x0,
                                                NcmMIODEFunction *g);

```

FIXME

**x0** : FIXME

**g** : a **NcmMIODEFunction**

**Returns** : FIXME

**ncm\_magnus\_iserles\_ode\_step ()**

```
gboolean          ncm_magnus_iserles_ode_step      (NcmMIODE *mi_ode,
                                                    long double h);
```

FIXME

*mi\_ode* : a **NcmMIODE**

*h* : FIXME

**Returns** : FIXME

**ncm\_magnus\_iserles\_ode\_step\_frac ()**

```
gboolean          ncm_magnus_iserles_ode_step_frac (NcmMIODE *mi_ode,
                                                    long double frac);
```

FIXME

*mi\_ode* : a **NcmMIODE**

*frac* : FIXME

**Returns** : FIXME

**ncm\_magnus\_iserles\_ode\_eval\_vec ()**

```
void              ncm_magnus_iserles_ode_eval_vec (NcmMIODE *mi_ode,
                                                    long double u_i,
                                                    long double up_i,
                                                    long double *u,
                                                    long double *up);
```

FIXME

*mi\_ode* : a **NcmMIODE**

*u\_i* : FIXME

*up\_i* : FIXME

*u* : FIXME

*up* : FIXME

## 9.10 Unidimensional Grid

Unidimensional Grid — FIXME

## Synopsis

```

enum                NcmGridNodesEndPoints;
struct              NcmGridSection;
struct              NcmGrid;
#define              NCM_GRID_GET_SEC              (grid)
NcmGrid *            ncm_grid_new                  (gulong nnodes);
NcmGrid *            ncm_grid_new_from_sections    (NcmGridSection *secs);
void                 ncm_grid_free                 (NcmGrid *grid,
                                                    gboolean free_data);
gchar *              ncm_grid_get_name             (NcmGridSection *secs);
void                 ncm_grid_set_sections         (NcmGrid *grid,
                                                    NcmGridSection *secs);
void                 ncm_grid_set_nodes_d          (NcmGrid *grid,
                                                    NcmGridNodesEndPoints incl,
                                                    guint32 start,
                                                    guint32 end,
                                                    gdouble start_val,
                                                    gdouble end_val);
void                 ncm_grid_set_nodes_si         (NcmGrid *grid,
                                                    NcmGridNodesEndPoints incl,
                                                    guint32 start,
                                                    guint32 end,
                                                    glong start_val_num,
                                                    glong start_val_den,
                                                    glong end_val_num,
                                                    glong end_val_den);
gdouble              ncm_grid_get_node_d           (NcmGrid *grid,
                                                    gulong i);
gdouble *            ncm_grid_get_double_array    (NcmGrid *grid);
void                 ncm_grid_write                (NcmGrid *grid,
                                                    FILE *f);
NcmGrid *            ncm_grid_read                 (FILE *f);

```

## Description

FIXME

## Details

### enum NcmGridNodesEndPoints

```

typedef enum {
    NCM_GRID_NODES_START = 1,
    NCM_GRID_NODES_END,
    NCM_GRID_NODES_BOTH,
    NCM_GRID_NODES_NONE,
} NcmGridNodesEndPoints;

```

FIXME

**NCM\_GRID\_NODES\_START** FIXME

**NCM\_GRID\_NODES\_END** FIXME

**NCM\_GRID\_NODES\_BOTH** FIXME

**NCM\_GRID\_NODES\_NONE** FIXME

**struct NcmGridSection**

```
struct NcmGridSection {  
};
```

FIXME

**struct NcmGrid**

```
struct NcmGrid {  
};
```

FIXME

**NCM\_GRID\_GET\_SEC()**

```
#define NCM_GRID_GET_SEC(grid) (&(g_array_index((grid)->sections, NcmGridSection, 0)))
```

**ncm\_grid\_new ()**

```
NcmGrid *          ncm_grid_new                (gulong nnodes);
```

FIXME

***nnodes*** : FIXME***Returns*** : FIXME**ncm\_grid\_new\_from\_sections ()**

```
NcmGrid *          ncm_grid_new_from_sections  (NcmGridSection *secs);
```

FIXME

***secs*** : a **NcmGridSection*****Returns*** : FIXME**ncm\_grid\_free ()**

```
void              ncm_grid_free                (NcmGrid *grid,  
                                                gboolean free_data);
```

FIXME

***grid*** : a **NcmGrid*****free\_data*** : FIXME

**ncm\_grid\_get\_name ()**

```
gchar *          ncm_grid_get_name          (NcmGridSection *secs);
```

FIXME

**secs** : a **NcmGridSection**

**Returns** : FIXME

**ncm\_grid\_set\_sections ()**

```
void          ncm_grid_set_sections          (NcmGrid *grid,  
                                             NcmGridSection *secs);
```

FIXME

**grid** : a **NcmGrid**

**secs** : a **NcmGridSection**

**ncm\_grid\_set\_nodes\_d ()**

```
void          ncm_grid_set_nodes_d          (NcmGrid *grid,  
                                             NcmGridNodesEndpoints incl,  
                                             guint32 start,  
                                             guint32 end,  
                                             gdouble start_val,  
                                             gdouble end_val);
```

FIXME

**grid** : a **NcmGrid**

**incl** : a **NcmGridNodesEndpoints**

**start** : FIXME

**end** : FIXME

**start\_val** : FIXME

**end\_val** : FIXME

**ncm\_grid\_set\_nodes\_si ()**

```
void          ncm_grid_set_nodes_si          (NcmGrid *grid,  
                                             NcmGridNodesEndpoints incl,  
                                             guint32 start,  
                                             guint32 end,  
                                             glong start_val_num,  
                                             glong start_val_den,  
                                             glong end_val_num,  
                                             glong end_val_den);
```

FIXME

---



**grid**: a **NcmGrid**

**incl**: a **NcmGridNodesEndPoints**

**start**: FIXME

**end**: FIXME

**start\_val\_num**: FIXME

**start\_val\_den**: FIXME

**end\_val\_num**: FIXME

**end\_val\_den**: FIXME

### **ncm\_grid\_get\_node\_d ()**

gdouble	ncm_grid_get_node_d	(NcmGrid *grid, gulong i);
---------	---------------------	-------------------------------

FIXME

**grid**: a **NcmGrid**

**i**: FIXME

**Returns**: FIXME

### **ncm\_grid\_get\_double\_array ()**

gdouble *	ncm_grid_get_double_array	(NcmGrid *grid);
-----------	---------------------------	------------------

FIXME

**grid**: a **NcmGrid**

**Returns**: FIXME

### **ncm\_grid\_write ()**

void	ncm_grid_write	(NcmGrid *grid, FILE *f);
------	----------------	------------------------------

FIXME

**grid**: a **NcmGrid**

**f**: FIXME

### **ncm\_grid\_read ()**

NcmGrid *	ncm_grid_read	(FILE *f);
-----------	---------------	------------

FIXME

**f**: FIXME

**Returns**: FIXME

## 9.11 Quadrature Algorithms

Quadrature Algorithms — FIXME

## Synopsis

[illegible]

### Description

FIXME

## Details

**struct NcmQuadFilonError**

```
struct NcmQuadFilonError {
};
```

FIXME

**struct NcmQuadFilon**

```
struct NcmQuadFilon {
};
```

FIXME

```
ncm quadrature filon new ()
```

[illegible]

FIXME

*omega* : FIXME

*inter\_n* : FIXME

*order* : FIXME

*range* : FIXME

*Returns* : FIXME

**ncm\_quadrature\_filon\_calc\_mu\_dxnm ()**

```
gboolean          ncm_quadrature_filon_calc_mu_dxnm      (NcmQuadFilon *quadf);
```

FIXME

*quadf* : a **NcmQuadFilon**

*Returns* : FIXME

**ncm\_quadrature\_filon\_calc\_inter\_point ()**

```
gboolean          ncm_quadrature_filon_calc_inter_point  
                  (NcmQuadFilon *quadf,  
                  gdouble g);
```

FIXME

*quadf* : a **NcmQuadFilon**

*g* : FIXME

*Returns* : FIXME

**ncm\_quadrature\_filon\_calc\_vandermonde ()**

```
gboolean          ncm_quadrature_filon_calc_vandermonde  
                  (NcmQuadFilon *quadf);
```

FIXME

*quadf* : a **NcmQuadFilon**

*Returns* : FIXME

**ncm\_quadrature\_filon\_solve\_vandermonde ()**

```
gboolean          ncm_quadrature_filon_solve_vandermonde  
                  (NcmQuadFilon *quadf);
```

FIXME

*quadf* : a **NcmQuadFilon**

*Returns* : FIXME

---

**ncm\_quadrature\_filon\_eval ()**

```

gboolean          ncm_quadrature_filon_eval          (NcmQuadFilon *quadf,
                                                       gsl_function *F,
                                                       gdouble xi,
                                                       gsl_complex *res,
                                                       gdouble *err);

```

FIXME

**quadf** : a **NcmQuadFilon**

**F** : FIXME

**xi** : FIXME

**res** : FIXME

**err** : FIXME

**Returns** : FIXME

## 9.12 Function Cache

Function Cache — A generic cache for functions values

### Synopsis

```

enum              NcmFunctionCacheSearchType;
struct            NcmFunctionCache;
NcmFunctionCache * ncm_function_cache_new          (guint n,
                                                       gdouble abstol,
                                                       gdouble reltol);

void              ncm_function_cache_free          (NcmFunctionCache *cache);
void              ncm_function_cache_clear         (NcmFunctionCache **cache);
void              ncm_function_cache_insert        (NcmFunctionCache *cache,
                                                       gdouble x,
                                                       ...);

void              ncm_function_cache_insert_vector (NcmFunctionCache *cache,
                                                       gdouble x,
                                                       gsl_vector *p);

gboolean          ncm_function_cache_get           (NcmFunctionCache *cache,
                                                       gdouble *x_ptr,
                                                       gsl_vector **v);

gboolean          ncm_function_cache_get_near      (NcmFunctionCache *cache,
                                                       gdouble x,
                                                       gdouble *x_found_ptr,
                                                       gsl_vector **v,
                                                       NcmFunctionCacheSearchType type);

#define            NC_FUNCTION_CACHE                (p)

```

### Description

FIXME

## Details

### enum NcmFunctionCacheSearchType

```
typedef enum {
    NC_FUNCTION_CACHE_SEARCH_BOTH = 0,
    NC_FUNCTION_CACHE_SEARCH_GT,
    NC_FUNCTION_CACHE_SEARCH_LT,
} NcmFunctionCacheSearchType;
```

FIXME

**NC\_FUNCTION\_CACHE\_SEARCH\_BOTH** FIXME

**NC\_FUNCTION\_CACHE\_SEARCH\_GT** FIXME

**NC\_FUNCTION\_CACHE\_SEARCH\_LT** FIXME

### struct NcmFunctionCache

```
struct NcmFunctionCache {
};
```

### ncm\_function\_cache\_new ()

```
NcmFunctionCache * ncm_function_cache_new (guint n,
                                           gdouble abstol,
                                           gdouble reltol);
```

FIXME

***n***: FIXME

***abstol***: FIXME

***reltol***: FIXME

***Returns***: FIXME

### ncm\_function\_cache\_free ()

```
void ncm_function_cache_free (NcmFunctionCache *cache);
```

FIXME

***cache***: a **NcmFunctionCache**

### ncm\_function\_cache\_clear ()

```
void ncm_function_cache_clear (NcmFunctionCache **cache);
```

FIXME

***cache***: a **NcmFunctionCache**

---

**ncm\_function\_cache\_insert ()**

```
void                ncm_function_cache_insert      (NcmFunctionCache *cache,
                                                    gdouble x,
                                                    ...);
```

**ncm\_function\_cache\_insert\_vector ()**

```
void                ncm_function_cache_insert_vector (NcmFunctionCache *cache,
                                                    gdouble x,
                                                    gsl_vector *p);
```

FIXME

**cache** : a **NcmFunctionCache**

**x** : FIXME

**p** : FIXME

**ncm\_function\_cache\_get ()**

```
gboolean           ncm_function_cache_get        (NcmFunctionCache *cache,
                                                    gdouble *x_ptr,
                                                    gsl_vector **v);
```

FIXME

**cache** : a **NcmFunctionCache**

**x\_ptr** : FIXME

**v** : FIXME

**Returns** : FIXME

**ncm\_function\_cache\_get\_near ()**

```
gboolean           ncm_function_cache_get_near   (NcmFunctionCache *cache,
                                                    gdouble x,
                                                    gdouble *x_found_ptr,
                                                    gsl_vector **v,
                                                    NcmFunctionCacheSearchType type);
```

**cache** : a **NcmFunctionCache**

**x** : FIXME

**x\_found\_ptr** : FIXME

**v** : FIXME

**type** : a **NcmFunctionCacheSearchType**

**NC\_FUNCTION\_CACHE()**

```
#define NC_FUNCTION_CACHE(p) ((NcmFunctionCache *) (p))
```

## 9.13 Memory Pool

Memory Pool — FIXME

### Synopsis

```
gpointer          (*NcmMemoryPoolAlloc)          (void);
struct            NcmMemoryPool;
struct            NcmMemoryPoolSlice;
NcmMemoryPool *   ncm_memory_pool_new            (NcmMemoryPoolAlloc mp_alloc,
                                                  GDestroyNotify mp_free);
void              ncm_memory_pool_free           (NcmMemoryPool *mp,
                                                  gboolean free_slices);
void              ncm_memory_pool_set_min_size   (NcmMemoryPool *mp,
                                                  gsize n);
gpointer          ncm_memory_pool_get            (NcmMemoryPool *mp);
void              ncm_memory_pool_return        (gpointer p);
```

### Description

FIXME

### Details

#### NcmMemoryPoolAlloc ()

```
gpointer          (*NcmMemoryPoolAlloc)          (void);
```

#### struct NcmMemoryPool

```
struct NcmMemoryPool {
};
```

FIXME

#### struct NcmMemoryPoolSlice

```
struct NcmMemoryPoolSlice {
    gpointer p;
    _NCM_MUTEX_TYPE lock;
    NcmMemoryPool *mp;
};
```

**gpointer *p***; Pointer to the actual slice

**\_NCM\_MUTEX\_TYPE *lock***; Mutex lock used by the pool

**NcmMemoryPool \**mp***; A back pointer to the pool

**ncm\_memory\_pool\_new ()**

```
NcmMemoryPool *      ncm_memory_pool_new      (NcmMemoryPoolAlloc mp_alloc,
                                                GDestroyNotify mp_free);
```

This function prepare a memory pool which allocate memory using `mp_alloc` and save it for future use, the memory must be returned to the pool using `ncm_memory_pool_return`. These functions are thread safe.

**`mp_alloc`** : a `NcmMemoryPoolAlloc`, function used to alloc memory

**`mp_free`** : function used to free memory allocated by `mp_alloc`

**Returns** : the memory pool `NcmMemoryPool`

**ncm\_memory\_pool\_free ()**

```
void                ncm_memory_pool_free      (NcmMemoryPool *mp,
                                                gboolean free_slices);
```

This function free the memory pool and also the slices if `free_slices == TRUE` and the pool was built with a free function

**`mp`** : a `NcmMemoryPool`, memory pool to be freed

**`free_slices`** : if true and the pool was built with a free function, free the slices

**ncm\_memory\_pool\_set\_min\_size ()**

```
void                ncm_memory_pool_set_min_size (NcmMemoryPool *mp,
                                                    gsize n);
```

if `n` grater than number of slices then allocate new slices until `n == slices`.

**`mp`** : a `NcmMemoryPool`

**`n`** : minimun number of slices contained in `mp`

**ncm\_memory\_pool\_get ()**

```
gpointer            ncm_memory_pool_get      (NcmMemoryPool *mp);
```

Search in the pool for a non used slice and return the first finded. If none allocate a new one add to the pool and return it.

**`mp`** : a `NcmMemoryPool`

**Returns** : a pointer to an unused `NcmMemoryPoolSlice`. *[transfer full]*

**ncm\_memory\_pool\_return ()**

```
void                ncm_memory_pool_return   (gpointer p);
```

**`p`** : slice to be returned to the pool

**Returns** : the slice pointed by slice to the pool



## 9.14 Numerical Integration

Numerical Integration — FIXME

### Synopsis

```

gsl_integration_workspace ** ncm_integral_get_workspace ();
gdouble (*_NcmIntegrand2dimFunc) (gdouble x,
                                   gdouble y,
                                   gpointer userdata);

struct NcmIntegrand2dim;
struct NcmIntegralFixed;
gint ncm_integral_locked_a_b (gsl_function *F,
                              gdouble a,
                              gdouble b,
                              gdouble abstol,
                              gdouble reltol,
                              gdouble *result,
                              gdouble *error);

gint ncm_integral_locked_a_inf (gsl_function *F,
                                gdouble a,
                                gdouble abstol,
                                gdouble reltol,
                                gdouble *result,
                                gdouble *error);

gint ncm_integral_cached_0_x (NcmFunctionCache *cache,
                              gsl_function *F,
                              gdouble x,
                              gdouble *result,
                              gdouble *error);

gint ncm_integral_cached_x_inf (NcmFunctionCache *cache,
                                gsl_function *F,
                                gdouble x,
                                gdouble *result,
                                gdouble *error);

gboolean ncm_integrate_2dim (NcmIntegrand2dim *integ,
                              gdouble xi,
                              gdouble yi,
                              gdouble xf,
                              gdouble yf,
                              gdouble epsrel,
                              gdouble epsabs,
                              gdouble *result,
                              gdouble *error);

NcmIntegralFixed * ncm_integral_fixed_new (gulong n_nodes,
                                           gulong rule_n,
                                           gdouble xl,
                                           gdouble xu);

void ncm_integral_fixed_free (NcmIntegralFixed *intf);
void ncm_integral_fixed_calc_nodes (NcmIntegralFixed *intf,
                                    gsl_function *F);

gdouble ncm_integral_fixed_nodes_eval (NcmIntegralFixed *intf);
gdouble ncm_integral_fixed_integ_mult (NcmIntegralFixed *intf,
                                       gsl_function *F);

gdouble ncm_integral_fixed_integ_posdef_mult

```

---

```

(NcmIntegralFixed *intf,
 gsl_function *F,
 gdouble max,
 gdouble reltol);

#define NCM_INTEGRAL_PARTITION
#define NCM_INTEGRAL_ALG
#define NCM_INTEGRAL_ERROR
#define NCM_INTEGRAL_ABS_ERROR

```

## Description

FIXME

## Details

### ncm\_integral\_get\_workspace ()

```
gsl_integration_workspace ** ncm_integral_get_workspace ();
```

This function provides a workspace to be used by numerical integration functions of GSL. It keeps a internal pool of workspaces and allocate a new one if the function is called and the pool is empty. It is designed to be used in a multithread enviroment. The workspace must be unlocked in order to return to the pool. This must be done using the [ncm\\_memory\\_pool\\_return](#).

**Returns :** a pointer to [gsl\\_integration\\_workspace](#) structure.

### \_NcmIntegrand2dimFunc ()

```

gdouble (*_NcmIntegrand2dimFunc) (gdouble x,
                                   gdouble y,
                                   gpointer userdata);

```

### struct NcmIntegrand2dim

```

struct NcmIntegrand2dim {
};

```

FIXME

### struct NcmIntegralFixed

```

struct NcmIntegralFixed {
};

```

FIXME

**ncm\_integral\_locked\_a\_b ()**

```

gint          ncm_integral_locked_a_b          (gsl_function *F,
                                                gdouble a,
                                                gdouble b,
                                                gdouble abstol,
                                                gdouble reltol,
                                                gdouble *result,
                                                gdouble *error);

```

This function uses a workspace from the pool and `gsl_integration_qag` function to perform the numerical integration in the  $[a, b]$  interval.

***F*** : a `gsl_function` wich is the integrand.

***a*** : lower integration limit.

***b*** : upper integration limit.

***abstol*** : absolute tolerance.

***reltol*** : relative tolerance.

***result*** : a pointer to a `gdouble` in which the function stores the result.

***error*** : a pointer to a `gdouble` in which the function stores the estimated error.

***Returns*** : the error code returned by `gsl_integration_qag`.

**ncm\_integral\_locked\_a\_inf ()**

```

gint          ncm_integral_locked_a_inf        (gsl_function *F,
                                                gdouble a,
                                                gdouble abstol,
                                                gdouble reltol,
                                                gdouble *result,
                                                gdouble *error);

```

This function uses a workspace from the pool and `gsl_integration_qagiu` function to perform the numerical integration in the  $[a, \infty]$  interval.

***F*** : a `gsl_function` wich is the integrand.

***a*** : lower integration limit.

***abstol*** : absolute tolerance.

***reltol*** : relative tolerance.

***result*** : a pointer to a `gdouble` in which the function stores the result.

***error*** : a pointer to a `gdouble` in which the function stores the estimated error.

***Returns*** : the error code returned by `gsl_integration_qagiu`.

**ncm\_integral\_cached\_0\_x ()**

```
gint          ncm_integral_cached_0_x          (NcmFunctionCache *cache,
                                                gsl_function *F,
                                                gdouble x,
                                                gdouble *result,
                                                gdouble *error);
```

This function searches for the nearest  $x_{\text{near}}$  value previously chosed as the upper integration limit and perform the integration at  $[x_{\text{near}}, x]$  interval. This result is summed to that obtained at  $[0, x_{\text{near}}]$  and then it is saved in the cache.

**cache** : a pointer to **NcmFunctionCache**.

**F** : a gsl\_function wich is the integrand.

**x** : upper integration limit.

**result** : a pointer to a gdouble in which the function stores the result.

**error** : a pointer to a gdouble in which the function stores the estimated error.

**Returns** : the error code returned by gsl\_integration\_qag.

**ncm\_integral\_cached\_x\_inf ()**

```
gint          ncm_integral_cached_x_inf        (NcmFunctionCache *cache,
                                                gsl_function *F,
                                                gdouble x,
                                                gdouble *result,
                                                gdouble *error);
```

This function searches for the nearest  $x_{\text{near}}$  value previously chosed as the lower integration limit and perform the integration at  $[x, x_{\text{near}}]$  interval. This result is summed to that obtained at  $[x_{\text{near}}, \infty]$  and then it is saved in the cache.

**cache** : a pointer to **NcmFunctionCache**.

**F** : a gsl\_function wich is the integrand.

**x** : lower integration limit.

**result** : a pointer to a gdouble in which the function stores the result.

**error** : a pointer to a gdouble in which the function stores the estimated error.

**Returns** : the error code returned by gsl\_integration\_qagi.

**ncm\_integrate\_2dim ()**

```
gboolean      ncm_integrate_2dim              (NcmIntegrand2dim *integ,
                                                gdouble xi,
                                                gdouble yi,
                                                gdouble xf,
                                                gdouble yf,
                                                gdouble epsrel,
                                                gdouble epsabs,
                                                gdouble *result,
                                                gdouble *error);
```

This function FIXME

**integ** : a pointer to **NcmIntegrand2dim**.

**xi** : gdouble which is the lower integration limit of variable x.

**yi** : gdouble which is the lower integration limit of variable y.

**xf** : gdouble which is the upper integration limit of variable x.

**yf** : gdouble which is the upper integration limit of variable y.

**epsrel** : relative error

**epsabs** : absolute error

**result** : a pointer to a gdouble in which the function stores the result.

**error** : a pointer to a gdouble in which the function stores the estimated error.

**Returns** : a gboolean

### ncm\_integral\_fixed\_new ()

```
NcmIntegralFixed * ncm_integral_fixed_new      (gulong n_nodes,
                                              gulong rule_n,
                                              gdouble xl,
                                              gdouble xu);
```

This function prepares the **NcmIntegralFixed** with a grid with `n_nodes - 1` intervals between `xl` and `xu`. In each interval it uses a fixed order (`rule_n`) Gauss-Legendre integration rule to determine the interval inner points. This results in a grid with  $(n\_nodes - 1) * rule\_n$  points.

**n\_nodes** : number of nodes in the full interval.

**rule\_n** : order of the Gauss-Legendre integration rule to be applied in each interval.

**xl** : the interval lower limit.

**xu** : the interval upper limit.

**Returns** : a pointer to the newly created **NcmIntegralFixed** structure.

### ncm\_integral\_fixed\_free ()

```
void ncm_integral_fixed_free (NcmIntegralFixed *intf);
```

This function frees the memory associated to **NcmIntegralFixed**.

**intf** : a pointer to **NcmIntegralFixed**.

### ncm\_integral\_fixed\_calc\_nodes ()

```
void ncm_integral_fixed_calc_nodes (NcmIntegralFixed *intf,
                                   gsl_function *F);
```

This function FIXME

**intf** : a pointer to **NcmIntegralFixed**.

**F** : a pointer to a `gsl_function`.

**ncm\_integral\_fixed\_nodes\_eval ()**

```
gdouble          ncm_integral_fixed_nodes_eval      (NcmIntegralFixed *intf);
```

This function

**intf** : a pointer to **NcmIntegralFixed**.

**Returns** : FIXME

**ncm\_integral\_fixed\_integ\_mult ()**

```
gdouble          ncm_integral_fixed_integ_mult      (NcmIntegralFixed *intf,
                                                    gsl_function *F);
```

This function

**intf** : a pointer to **NcmIntegralFixed**.

**F** : a pointer to **gsl\_function**.

**Returns** : FIXME

**ncm\_integral\_fixed\_integ\_posdef\_mult ()**

```
gdouble          ncm_integral_fixed_integ_posdef_mult
                                                    (NcmIntegralFixed *intf,
                                                    gsl_function *F,
                                                    gdouble max,
                                                    gdouble reltol);
```

This function

**intf** : a pointer to **NcmIntegralFixed**.

**F** : a pointer to **gsl\_function**.

**max** : FIXME

**reltol** : FIXME

**Returns** : FIXME

**NCM\_INTEGRAL\_PARTITION**

```
#define NCM_INTEGRAL_PARTITION 100000
```

**NCM\_INTEGRAL\_ALG**

```
#define NCM_INTEGRAL_ALG 6
```

**NCM\_INTEGRAL\_ERROR**

```
#define NCM_INTEGRAL_ERROR 1e-13
```

**NCM\_INTEGRAL\_ABS\_ERROR**

```
#define NCM_INTEGRAL_ABS_ERROR 1e-13
```

## Chapter 10

# Object Hierarchy

```
GObject
  NcClusterAbundance
  NcmModel
    NcClusterMass
      NcClusterMassBenson
        NcClusterMassBensonXRay
      NcClusterMassLnnormal
      NcClusterMassNodist
      NcClusterMassVanderlinde
    NcHICosmo
      NcHICosmoDE
        NcHICosmoDELinder
        NcHICosmoDEPad
        NcHICosmoDEQe
        NcHICosmoDEXcdm
      NcHICosmoLCDM
      NcHICosmoQConst
      NcHICosmoQLinear
      NcHICosmoQPW
      NcHICosmoQSpline
    NcSNIADistCov
  NcClusterRedshift
    NcClusterPhotozGauss
    NcClusterPhotozGaussGlobal
    NcClusterRedshiftNodist
  NcmData
    NcmDataGaussDiag
      NcDataBaoA
      NcDataBaoDV
      NcDataBaoDVDV
      NcDataCMBSHiftParam
      NcDataDistMu
      NcDataHubbleBao
      NcDataHubble
    NcmDataGauss
      NcDataBaoRDV
      NcDataCMBDistPriors
    NcDataClusterNCount
    NcmDataPoisson
      NcDataClusterPoisson
    NcmDataGaussCov
      NcDataSNIACov
    NcmDataDist1d
  NcDistance
```



```
NcGalaxyAcf
NcGrowthFunc
NcHaloBiasFunc
NcHaloBiasType
    NcHaloBiasTypePS
    NcHaloBiasTypeSTEllip
    NcHaloBiasTypeSTSpher
    NcHaloBiasTypeTinker
NcHICosmoQSplineContPrior
NcMassFunction
NcMatterVar
NcMultiplicityFunc
    NcMultiplicityFuncJenkins
    NcMultiplicityFuncPS
    NcMultiplicityFuncST
    NcMultiplicityFuncTinkerCrit
    NcMultiplicityFuncTinker
    NcMultiplicityFuncTinkerMean
    NcMultiplicityFuncWarren
NcRecomb
    NcRecombSeager
NcTransferFunc
    NcTransferFuncBBKS
    NcTransferFuncCAMB
    NcTransferFuncEH
    NcTransferFuncPert
NcWindow
    NcWindowGaussian
    NcWindowTophat
NcmC
NcmDataset
NcmFftlog
NcmFit
    NcmFitGSLLS
    NcmFitGSLMM
    NcmFitGSLMMS
    NcmFitLevmar
    NcmFitNLOpt
NcmFitMC
NcmFitState
NcmLHRatio1d
NcmLHRatio2d
NcmLikelihood
NcmMatrix
NcmModelCtrl
NcmMSetFunc
NcmMSet
NcmReparam
    NcmReparamLinear
NcmSParam
NcmSpline2d
    NcmSpline2dBicubic
    NcmSpline2dGsl
    NcmSpline2dSpline
NcmSpline
    NcmSplineCubic
        NcmSplineCubicNotaknot
    NcmSplineGsl
NcmVector
NcmVParam
GBoxed
    NcScaleFactor
```

NcmFitConstraint  
NcmLHRatio2dRegion  
NcmMSetPIndex

GBoxed	GObject	NcClusterAbundance
NcClusterMass	NcClusterMassBenson	NcClusterMassBensonXRay
NcClusterMassLnnormal	NcClusterMassNodist	NcClusterMassVanderlinde
NcClusterPhotozGauss	NcClusterPhotozGaussGlobal	NcClusterRedshift
NcClusterRedshiftNodist	NcDataBaoA	NcDataBaoDV
NcDataBaoDVDV	NcDataBaoRDV	NcDataCMBDistPriors
NcDataCMBShiftParam	NcDataClusterNCount	NcDataClusterPoisson
NcDataDistMu	NcDataHubble	NcDataHubbleBao
NcDataSNIACov	NcDistance	NcGalaxyAcf
NcGrowthFunc	NcHICosmo	NcHICosmoDE
NcHICosmoDELinder	NcHICosmoDEPad	NcHICosmoDEQe
NcHICosmoDEXcdm	NcHICosmoLCDM	NcHICosmoQConst
NcHICosmoQLinear	NcHICosmoQPW	NcHICosmoQSpline
NcHICosmoQSplineContPrior	NcHaloBiasFunc	NcHaloBiasType
NcHaloBiasTypePS	NcHaloBiasTypeSTEllip	NcHaloBiasTypeSTSpher
NcHaloBiasTypeTinker	NcMassFunction	NcMatterVar
NcMultiplicityFunc	NcMultiplicityFuncJenkins	NcMultiplicityFuncPS
NcMultiplicityFuncST	NcMultiplicityFuncTinker	NcMultiplicityFuncTinkerCrit
NcMultiplicityFuncTinkerMean	NcMultiplicityFuncWarren	NcRecomb
NcRecombSeager	NcSNIADistCov	NcScaleFactor
NcTransferFunc	NcTransferFuncBBKS	NcTransferFuncCAMB
NcTransferFuncEH	NcTransferFuncPert	NcWindow
NcWindowGaussian	NcWindowTophat	NcmC
NcmData	NcmDataDist1d	NcmDataGauss
NcmDataGaussCov	NcmDataGaussDiag	NcmDataPoisson
NcmDataset	NcmFftlog	NcmFit
NcmFitConstraint	NcmFitGSLLS	NcmFitGSLMM
NcmFitGSLMMS	NcmFitLevmar	NcmFitMC
NcmFitNLOpt	NcmFitState	NcmLHRatio1d
NcmLHRatio2d	NcmLHRatio2dRegion	NcmLikelihood
NcmMSet	NcmMSetFunc	NcmMSetPIndex
NcmMatrix	NcmModel	NcmModelCtrl
NcmReparam	NcmReparamLinear	NcmSParam
NcmSpline	NcmSpline2d	NcmSpline2dBicubic
NcmSpline2dGsl	NcmSpline2dSpline	NcmSplineCubic
NcmSplineCubicNotaknot	NcmSplineGsl	NcmVParam
NcmVector		

## Chapter 11

# Annotation Glossary

### E

**element-type**

Generics and defining elements of containers and arrays.

### A

**allow-none**

NULL is ok, both for passing and for returning.

### O

**out**

Parameter for returning results. Default is transfer full.

### T

**transfer full**

Free data after the code is done.

**transfer container**

Free data container after the code is done.

**transfer none**

Don't free data after the code is done.

### A

**array**

Parameter points to an array of items.

---

## T

### **type**

Override the parsed C type with given type.

## S

### **scope notified**

The callback is valid until the GDestroyNotify argument is called.

## Chapter 12

## Index

- [\\_NcmFitLSFJ](#), 223
- [\\_NcmFitLSJ](#), 223
- [\\_NcmFitM2lnLGrad](#), 223
- [\\_NcmFitM2lnLValGrad](#), 223
- [\\_NcmIntegrand2dimFunc](#), 563
- [\\_NcmSpline2dBicubicOptimizeInt](#), 158
- A**
  - [ncm\\_assert\\_cmpdouble](#), 546
  - [ncm\\_assert\\_cmpdouble\\_e](#), 546
  - [NcMassFunction](#), 428
  - [NcMassFunction:area](#), 433
  - [NcMassFunction:distance](#), 433
  - [NcMassFunction:growth](#), 433
  - [NcMassFunction:multiplicity](#), 433
  - [NcMassFunction:variance](#), 434
  - [NcMassFunctionClass](#), 428
  - [NcMassFunctionSplineOptimize](#), 428
  - [NcMatterVar](#), 398
  - [NcMatterVar:strategy](#), 403
  - [NcMatterVar:transfer](#), 403
  - [NcMatterVar>window](#), 403
  - [NcMatterVarClass](#), 398
  - [NcMatterVarStrategy](#), 398
- B**
  - [NcmBinSplit](#), 528
  - [ncm\\_binsplit\\_alloc](#), 528
  - [NCM\\_BINSPLIT\\_DECL](#), 530
  - [NCM\\_BINSPLIT\\_DENC\\_NULL](#), 530
  - [ncm\\_binsplit\\_eval\\_join](#), 529
  - [ncm\\_binsplit\\_eval\\_prec](#), 529
  - [ncm\\_binsplit\\_get](#), 529
  - [ncm\\_binsplit\\_get\\_d](#), 530
  - [ncm\\_binsplit\\_get\\_q](#), 530
  - [ncm\\_binsplit\\_join](#), 529
  - [NCM\\_BINSPLIT\\_ONE](#), 528
  - [ncm\\_binsplit\\_test\\_next](#), 528
  - [NcmBinSplitEval](#), 528
  - [bterm](#), 140
- C**
  - [NcmC](#), 115
  - [ncm\\_c\\_AR](#), 119
  - [ncm\\_c\\_AU](#), 126
  - [ncm\\_c\\_bao\\_eisenstein\\_A](#), 129
  - [ncm\\_c\\_bao\\_eisenstein\\_DV](#), 129
  - [ncm\\_c\\_bao\\_eisenstein\\_sigma\\_A](#), 129
  - [ncm\\_c\\_bao\\_eisenstein\\_sigma\\_DV](#), 129
  - [ncm\\_c\\_bao\\_eisenstein\\_z](#), 129
  - [ncm\\_c\\_bao\\_percival2007\\_DV\\_DV](#), 130
  - [ncm\\_c\\_bao\\_percival2007\\_sigma\\_DV\\_DV](#), 130
  - [ncm\\_c\\_bao\\_percival2010\\_DV\\_DV](#), 130
  - [ncm\\_c\\_bao\\_percival2010\\_sigma\\_DV\\_DV](#), 130
  - [ncm\\_c\\_blackbody\\_energy\\_density](#), 132
  - [ncm\\_c\\_boltzmann\\_factor\\_H\\_1s](#), 125
  - [ncm\\_c\\_boltzmann\\_factor\\_H\\_2p](#), 126
  - [ncm\\_c\\_boltzmann\\_factor\\_H\\_2s](#), 126
  - [ncm\\_c\\_boltzmann\\_factor\\_HeI\\_1s](#), 126
  - [ncm\\_c\\_boltzmann\\_factor\\_HeI\\_2p](#), 126
  - [ncm\\_c\\_boltzmann\\_factor\\_HeI\\_2s](#), 126
  - [ncm\\_c\\_c](#), 117
  - [ncm\\_c\\_c2](#), 119
  - [ncm\\_c\\_crit\\_density](#), 131
  - [ncm\\_c\\_crit\\_mass\\_density](#), 132
  - [ncm\\_c\\_crit\\_mass\\_density\\_solar\\_Mpc](#), 132
  - [ncm\\_c\\_crit\\_number\\_density\\_n](#), 132
  - [ncm\\_c\\_crit\\_number\\_density\\_p](#), 132
  - [ncm\\_c\\_decay\\_H\\_rate\\_2s\\_1s](#), 120
  - [ncm\\_c\\_decay\\_He\\_rate\\_2s\\_1s](#), 120
  - [ncm\\_c\\_degree\\_to\\_radian](#), 116
  - [ncm\\_c\\_fine\\_struct](#), 117
  - [ncm\\_c\\_fine\\_struct\\_square](#), 119
  - [ncm\\_c\\_G](#), 118
  - [ncm\\_c\\_h](#), 117
  - [ncm\\_c\\_H\\_bind](#), 122
  - [ncm\\_c\\_H\\_bind\\_1s](#), 123
  - [ncm\\_c\\_H\\_bind\\_2p](#), 123
  - [ncm\\_c\\_H\\_bind\\_2s](#), 123
  - [ncm\\_c\\_H\\_Lyman\\_2p](#), 123
  - [ncm\\_c\\_H\\_Lyman\\_2p\\_wl](#), 124
  - [ncm\\_c\\_H\\_Lyman\\_2p\\_wl3\\_8pi](#), 124
  - [ncm\\_c\\_H\\_Lyman\\_2s](#), 123
  - [ncm\\_c\\_H\\_Lyman\\_2s\\_wl](#), 124
  - [ncm\\_c\\_H\\_Lyman\\_2s\\_wl3\\_8pi](#), 124

ncm\_c\_H\_Lyman\_series, 123  
 ncm\_c\_H\_Lyman\_series\_wl, 124  
 ncm\_c\_H\_reduced\_energy, 122  
 ncm\_c\_H\_reduced\_mass, 122  
 ncm\_c\_hbar, 117  
 ncm\_c\_hc, 119  
 ncm\_c\_HeI\_2s\_m\_2p, 121  
 ncm\_c\_HeI\_2s\_m\_2p\_kb, 122  
 ncm\_c\_HeI\_bind\_1s, 120  
 ncm\_c\_HeI\_bind\_2p, 121  
 ncm\_c\_HeI\_bind\_2s, 121  
 ncm\_c\_HeI\_Lyman\_2p, 121  
 ncm\_c\_HeI\_Lyman\_2p\_wl, 121  
 ncm\_c\_HeI\_Lyman\_2p\_wl3\_8pi, 122  
 ncm\_c\_HeI\_Lyman\_2s, 120  
 ncm\_c\_HeI\_Lyman\_2s\_wl, 121  
 ncm\_c\_HeI\_Lyman\_2s\_wl3\_8pi, 122  
 ncm\_c\_HeII\_bind\_1s, 120  
 ncm\_c\_hubble\_cte\_hst, 130  
 ncm\_c\_hubble\_cte\_msa, 130  
 ncm\_c\_hubble\_cte\_wmap, 130  
 ncm\_c\_hubble\_radius, 131  
 ncm\_c\_hubble\_radius\_planck, 131  
 ncm\_c\_kb, 117  
 ncm\_c\_kpc, 119  
 ncm\_c\_ln2pi, 116  
 ncm\_c\_lnpi\_4, 116  
 ncm\_c\_mass\_e, 118  
 ncm\_c\_mass\_n, 118  
 ncm\_c\_mass\_p, 118  
 ncm\_c\_mass\_ratio\_alpha\_p, 118  
 ncm\_c\_mass\_solar, 127  
 ncm\_c\_Mpc, 119  
 ncm\_c\_neutrino\_n\_eff, 130  
 ncm\_c\_pc, 127  
 ncm\_c\_pi, 116  
 ncm\_c\_planck\_length, 118  
 ncm\_c\_planck\_length2, 119  
 ncm\_c\_prim\_H\_Yp, 131  
 ncm\_c\_prim\_He\_Yp, 131  
 ncm\_c\_prim\_XHe, 131  
 ncm\_c\_radian\_0\_2pi, 117  
 ncm\_c\_radian\_to\_degree, 116  
 ncm\_c\_radiation\_h2Omega\_r\_to\_temp, 132  
 ncm\_c\_radiation\_temp\_to\_h2omega\_r, 132  
 ncm\_c\_rest\_energy\_e, 119  
 ncm\_c\_rest\_energy\_n, 120  
 ncm\_c\_rest\_energy\_p, 120  
 ncm\_c\_sign\_sin, 117  
 ncm\_c\_sqrt\_1\_4pi, 115  
 ncm\_c\_sqrt\_2pi, 115  
 ncm\_c\_sqrt\_3\_4pi, 116  
 ncm\_c\_stats\_1sigma, 127  
 ncm\_c\_stats\_2sigma, 127  
 ncm\_c\_stats\_3sigma, 127  
 ncm\_c\_stefan\_boltzmann, 118  
 ncm\_c\_tan\_1arcsec, 116  
 ncm\_c\_thermal\_wl\_e, 124  
 ncm\_c\_thermal\_wl\_n, 125  
 ncm\_c\_thermal\_wl\_p, 125  
 ncm\_c\_thermal\_wn\_e, 125  
 ncm\_c\_thermal\_wn\_n, 125  
 ncm\_c\_thermal\_wn\_p, 125  
 ncm\_c\_thomson\_cs, 118  
 ncm\_c\_wmap3\_cmb\_R, 127  
 ncm\_c\_wmap3\_cmb\_sigma\_R, 128  
 ncm\_c\_wmap3\_cmb\_z, 127  
 ncm\_c\_wmap5\_cmb\_R, 128  
 ncm\_c\_wmap5\_cmb\_sigma\_R, 128  
 ncm\_c\_wmap5\_cmb\_z, 128  
 ncm\_c\_wmap5\_coadded\_I\_K, 128  
 ncm\_c\_wmap5\_coadded\_I\_Ka, 128  
 ncm\_c\_wmap5\_coadded\_I\_Q, 129  
 ncm\_c\_wmap5\_coadded\_I\_V, 129  
 ncm\_c\_wmap5\_coadded\_I\_W, 129  
 ncm\_c\_wmap7\_cmb\_R, 128  
 ncm\_c\_wmap7\_cmb\_sigma\_R, 128  
 ncm\_c\_wmap7\_cmb\_z, 128  
 camb\_filename, 393  
 NcmCClass, 115  
 NCM\_CEIL\_TRUNC, 109  
 ncm\_cfg\_command\_line, 107  
 ncm\_cfg\_count\_named\_instance, 104  
 ncm\_cfg\_create\_from\_name\_params, 107  
 ncm\_cfg\_create\_from\_string, 108  
 NCM\_CFG\_DEFAULT\_SQLITE3\_FILENAME, 111  
 ncm\_cfg\_enable\_gsl\_err\_handler, 99  
 ncm\_cfg\_entries\_to\_keyfile, 100  
 ncm\_cfg\_enum\_print\_all, 101  
 ncm\_cfg\_exist\_named\_instance, 103  
 ncm\_cfg\_exists, 101  
 ncm\_cfg\_fopen, 105  
 ncm\_cfg\_free\_named\_instances, 104  
 ncm\_cfg\_get\_default\_sqlite3, 107  
 ncm\_cfg\_get\_enum\_by\_id\_name\_nick, 100  
 ncm\_cfg\_get\_fullpath, 99  
 ncm\_cfg\_get\_named\_instance, 104  
 ncm\_cfg\_gvalue\_to\_gvariant, 108  
 ncm\_cfg\_init, 99  
 ncm\_cfg\_is\_named\_instance, 103  
 ncm\_cfg\_keyfile\_to\_arg, 100  
 ncm\_cfg\_load\_fftw\_wisdom, 101  
 ncm\_cfg\_load\_matrix, 106  
 ncm\_cfg\_load\_spline, 105  
 ncm\_cfg\_load\_vector, 106  
 ncm\_cfg\_logfile, 102  
 ncm\_cfg\_logfile\_flush, 102  
 ncm\_cfg\_msg\_sepa, 103  
 ncm\_cfg\_named\_instance\_peek\_name, 104  
 ncm\_cfg\_object\_is\_named, 104  
 ncm\_cfg\_object\_set\_property, 107  
 ncm\_cfg\_register\_obj, 99  
 ncm\_cfg\_rng\_get, 103  
 ncm\_cfg\_rng\_get\_state, 103

ncm\_cfg\_rng\_set\_state, 103  
 ncm\_cfg\_save\_fftw\_wisdom, 101  
 ncm\_cfg\_save\_matrix, 107  
 ncm\_cfg\_save\_spline, 106  
 ncm\_cfg\_save\_vector, 106  
 ncm\_cfg\_serialize\_to\_string, 108  
 ncm\_cfg\_serialize\_to\_variant, 108  
 ncm\_cfg\_set\_logfile, 101  
 ncm\_cfg\_set\_named\_instance, 104  
 ncm\_cfg\_string\_to\_comment, 100  
 ncm\_cfg\_vfopen, 105  
 ncm\_cmp, 546  
 NCM\_COMPLEX\_ADD, 111  
 NCM\_COMPLEX\_INC\_MUL, 110  
 NCM\_COMPLEX\_INC\_MUL\_MUL\_REAL, 111  
 NCM\_COMPLEX\_INC\_MUL\_REAL, 110  
 NCM\_COMPLEX\_INC\_MUL\_REAL\_TEST, 110  
 NCM\_COMPLEX\_MUL, 111  
 NCM\_COMPLEX\_MUL\_CONJUGATE, 111  
 NCM\_COMPLEX\_MUL\_REAL, 111  
 cterm, 140  
 CVDlsDenseJacFn, 263  
 CVOICE\_CHECK, 548  
 ncm\_cvoice\_util\_check\_flag, 548  
 ncm\_cvoice\_util\_nvector\_new, 548  
 ncm\_cvoice\_util\_print\_stats, 548  
 cvodes\_solver, 382  
 CVRhsFn, 263

## D

NcmData, 187  
 NcmData:name, 190  
 ncm\_data\_clear, 187  
 ncm\_data\_distld\_get\_size, 205  
 ncm\_data\_distld\_set\_size, 204  
 ncm\_data\_dup, 188  
 ncm\_data\_free, 187  
 ncm\_data\_gauss\_cov\_get\_size, 201  
 ncm\_data\_gauss\_cov\_set\_size, 201  
 ncm\_data\_gauss\_diag\_get\_size, 200  
 ncm\_data\_gauss\_diag\_set\_size, 199  
 ncm\_data\_gauss\_get\_size, 198  
 ncm\_data\_gauss\_set\_size, 198  
 ncm\_data\_get\_desc, 190  
 ncm\_data\_get\_dof, 188  
 ncm\_data\_get\_length, 188  
 ncm\_data\_least\_squares\_f, 189  
 ncm\_data\_least\_squares\_f\_J, 189  
 ncm\_data\_least\_squares\_J, 189  
 ncm\_data\_m2lnL\_grad, 190  
 ncm\_data\_m2lnL\_val, 189  
 ncm\_data\_m2lnL\_val\_grad, 190  
 ncm\_data\_poisson\_init\_from\_histogram, 203  
 ncm\_data\_poisson\_init\_from\_vector, 203  
 ncm\_data\_poisson\_init\_zero, 203  
 ncm\_data\_prepare, 188  
 ncm\_data\_ref, 187

ncm\_data\_resample, 188  
 ncm\_data\_set\_init, 188  
 NcmDataClass, 187  
 NcmDataDistld, 204  
 NcmDataDistld:n-points, 205  
 NcmDataDistldClass, 204  
 NcmDataGauss, 198  
 NcmDataGauss:n-points, 198  
 NcmDataGaussClass, 198  
 NcmDataGaussCov, 201  
 NcmDataGaussCov:n-points, 201  
 NcmDataGaussCov:use-det, 201  
 NcmDataGaussCovClass, 201  
 NcmDataGaussDiag, 199  
 NcmDataGaussDiag:n-points, 200  
 NcmDataGaussDiag:w-mean, 200  
 NcmDataGaussDiagClass, 199  
 NcmDataPoisson, 203  
 NcmDataPoisson:n-points, 203  
 NcmDataPoissonClass, 202  
 NcmDataPoissonType, 202  
 NcmDataset, 192  
 ncm\_dataset\_all\_init, 193  
 ncm\_dataset\_append\_data, 194  
 ncm\_dataset\_clear, 193  
 ncm\_dataset\_copy, 192  
 ncm\_dataset\_dup, 192  
 ncm\_dataset\_free, 193  
 ncm\_dataset\_get\_data, 194  
 ncm\_dataset\_get\_dof, 193  
 ncm\_dataset\_get\_length, 193  
 ncm\_dataset\_get\_n, 193  
 ncm\_dataset\_get\_ndata, 194  
 ncm\_dataset\_has\_least\_squares\_f, 195  
 ncm\_dataset\_has\_least\_squares\_f\_J, 195  
 ncm\_dataset\_has\_least\_squares\_J, 195  
 ncm\_dataset\_has\_m2lnL\_grad, 195  
 ncm\_dataset\_has\_m2lnL\_val, 195  
 ncm\_dataset\_has\_m2lnL\_val\_grad, 196  
 ncm\_dataset\_least\_squares\_f, 196  
 ncm\_dataset\_least\_squares\_f\_J, 196  
 ncm\_dataset\_least\_squares\_J, 196  
 ncm\_dataset\_log\_info, 195  
 ncm\_dataset\_m2lnL\_grad, 197  
 ncm\_dataset\_m2lnL\_val, 196  
 ncm\_dataset\_m2lnL\_val\_grad, 197  
 ncm\_dataset\_new, 192  
 ncm\_dataset\_peek\_data, 194  
 ncm\_dataset\_ref, 192  
 ncm\_dataset\_resample, 194  
 NcmDatasetClass, 192  
 NCM\_DEFAULT\_PRECISION, 109  
 dptsv\_, 27  
 dterm, 140

## F

NcmFftlog, 94

NcmFftlog:k0, 95  
NcmFftlog:L, 95  
NcmFftlog:N, 95  
NcmFftlog:name, 95  
NcmFftlog:r0, 96  
ncm\_fftlog\_get\_length, 95  
ncm\_fftlog\_get\_size, 94  
ncm\_fftlog\_peek\_name, 94  
ncm\_fftlog\_set\_length, 95  
ncm\_fftlog\_set\_name, 94  
ncm\_fftlog\_set\_size, 94  
NcmFftlogClass, 94  
fftw\_complex, 263  
fftw\_plan, 263  
ncm\_finite\_diff\_calc\_J, 540  
NcmFit, 224  
NcmFit:grad-type, 240  
NcmFit:likelihood, 240  
NcmFit:maxiter, 241  
NcmFit:mset, 241  
ncm\_fit\_add\_equality\_constraint, 227  
ncm\_fit\_add\_inequality\_constraint, 227  
ncm\_fit\_chisq\_test, 238  
ncm\_fit\_clear, 226  
ncm\_fit\_constraint\_dup, 224  
ncm\_fit\_constraint\_free, 225  
ncm\_fit\_constraint\_new, 224  
ncm\_fit\_copy\_new, 225  
ncm\_fit\_covar\_cor, 236  
ncm\_fit\_covar\_cov, 235  
ncm\_fit\_covar\_fparam\_cor, 237  
ncm\_fit\_covar\_fparam\_cov, 236  
ncm\_fit\_covar\_fparam\_sd, 236  
ncm\_fit\_covar\_fparam\_var, 236  
ncm\_fit\_covar\_sd, 235  
ncm\_fit\_covar\_var, 235  
ncm\_fit\_data\_m2lnL\_val, 230  
NCM\_FIT\_DEFAULT\_M2LNL\_ABSTOL, 240  
NCM\_FIT\_DEFAULT\_M2LNL\_RELTOL, 240  
ncm\_fit\_dprob, 237  
ncm\_fit\_fishermatrix\_print, 229  
ncm\_fit\_free, 226  
ncm\_fit\_function\_cov, 239  
ncm\_fit\_function\_error, 239  
ncm\_fit\_get\_desc, 228  
ncm\_fit\_get\_maxiter, 227  
ncm\_fit\_gsl\_ls\_new, 245  
ncm\_fit\_gsl\_mm\_new, 246  
ncm\_fit\_gsl\_mm\_new\_by\_name, 247  
ncm\_fit\_gsl\_mm\_new\_default, 247  
ncm\_fit\_gsl\_mm\_set\_algo, 247  
ncm\_fit\_gsl\_mms\_new, 249  
ncm\_fit\_gsl\_mms\_new\_by\_name, 249  
ncm\_fit\_gsl\_mms\_new\_default, 249  
ncm\_fit\_gsl\_mms\_set\_algo, 250  
ncm\_fit\_has\_equality\_constraints, 228  
ncm\_fit\_has\_inequality\_constraints, 228  
ncm\_fit\_is\_least\_squares, 228  
ncm\_fit\_levmar\_new, 251  
ncm\_fit\_levmar\_new\_by\_name, 252  
ncm\_fit\_levmar\_new\_default, 251  
ncm\_fit\_levmar\_set\_algo, 252  
ncm\_fit\_log\_covar, 228  
ncm\_fit\_log\_end, 229  
ncm\_fit\_log\_info, 228  
ncm\_fit\_log\_start, 229  
ncm\_fit\_log\_state, 229  
ncm\_fit\_log\_step, 229  
ncm\_fit\_log\_step\_error, 229  
ncm\_fit\_lr\_test, 238  
ncm\_fit\_lr\_test\_range, 237  
ncm\_fit\_ls\_covar, 235  
ncm\_fit\_ls\_f, 230  
ncm\_fit\_ls\_f\_J, 233  
ncm\_fit\_ls\_f\_J\_an, 234  
ncm\_fit\_ls\_f\_J\_nd\_ce, 234  
ncm\_fit\_ls\_f\_J\_nd\_fo, 234  
ncm\_fit\_ls\_J, 233  
ncm\_fit\_ls\_J\_an, 233  
ncm\_fit\_ls\_J\_nd\_ce, 233  
ncm\_fit\_ls\_J\_nd\_fo, 233  
ncm\_fit\_ls\_set\_state, 226  
ncm\_fit\_m2lnL\_grad, 230  
ncm\_fit\_m2lnL\_grad\_an, 231  
ncm\_fit\_m2lnL\_grad\_nd\_ac, 231  
ncm\_fit\_m2lnL\_grad\_nd\_ce, 231  
ncm\_fit\_m2lnL\_grad\_nd\_fo, 231  
ncm\_fit\_m2lnL\_val, 230  
ncm\_fit\_m2lnL\_val\_grad, 231  
ncm\_fit\_m2lnL\_val\_grad\_an, 232  
ncm\_fit\_m2lnL\_val\_grad\_nd\_ac, 232  
ncm\_fit\_m2lnL\_val\_grad\_nd\_ce, 232  
ncm\_fit\_m2lnL\_val\_grad\_nd\_fo, 232  
NCM\_FIT\_MAXITER, 240  
ncm\_fit\_mc\_clear, 254  
ncm\_fit\_mc\_free, 253  
ncm\_fit\_mc\_gof\_pdf, 254  
ncm\_fit\_mc\_gof\_pdf\_print, 255  
ncm\_fit\_mc\_gof\_pdf\_pvalue, 255  
ncm\_fit\_mc\_mean\_covar, 254  
ncm\_fit\_mc\_new, 253  
ncm\_fit\_mc\_print, 254  
ncm\_fit\_mc\_run, 254  
ncm\_fit\_new, 225  
ncm\_fit\_nlopt\_local\_new, 242  
ncm\_fit\_nlopt\_new, 242  
ncm\_fit\_nlopt\_new\_by\_name, 243  
ncm\_fit\_nlopt\_new\_default, 243  
ncm\_fit\_nlopt\_set\_algo, 243  
ncm\_fit\_nlopt\_set\_local\_algo, 243  
NCM\_FIT\_NPARAM, 240  
ncm\_fit\_numdiff\_m2lnL\_covar, 234  
ncm\_fit\_numdiff\_m2lnL\_hessian, 234  
NCM\_FIT\_NUMDIFF\_SCALE, 240



ncm\_fit\_priors\_m2lnL\_val, 230  
 ncm\_fit\_prob, 238  
 ncm\_fit\_ref, 225  
 ncm\_fit\_remove\_equality\_constraints, 227  
 ncm\_fit\_remove\_inequality\_constraints, 227  
 ncm\_fit\_reset, 238  
 ncm\_fit\_run, 239  
 ncm\_fit\_set\_grad\_type, 226  
 ncm\_fit\_set\_maxiter, 226  
 ncm\_fit\_state\_clear, 216  
 ncm\_fit\_state\_free, 216  
 ncm\_fit\_state\_new, 216  
 ncm\_fit\_state\_ref, 216  
 ncm\_fit\_state\_reset, 217  
 ncm\_fit\_state\_set\_all, 216  
 ncm\_fit\_type\_constrain\_error, 239  
 NcmFitClass, 224  
 NcmFitConstraint, 224  
 NcmFitGradType, 223  
 NcmFitGSLLS, 244  
 NcmFitGSLLSClass, 244  
 NcmFitGSLMM, 246  
 NcmFitGSLMM:algorithm, 247  
 NcmFitGSLMMAlgos, 246  
 NcmFitGSLMMClass, 246  
 NcmFitGSLMMS, 249  
 NcmFitGSLMMS:algorithm, 250  
 NcmFitGSLMMSAlgos, 248  
 NcmFitGSLMMSClass, 248  
 NcmFitLevmar, 251  
 NcmFitLevmar:algorithm, 252  
 NcmFitLevmarAlgos, 251  
 NcmFitLevmarClass, 251  
 NcmFitMC, 253  
 NcmFitMC:fit, 255  
 NcmFitMCClass, 253  
 NcmFitNLOpt, 242  
 NcmFitNLOpt:algorithm, 244  
 NcmFitNLOpt:local-algorithm, 244  
 NcmFitNLOptClass, 242  
 NcmFitRunMsgs, 224  
 NcmFitState, 215  
 NcmFitState:data-len, 217  
 NcmFitState:dof, 217  
 NcmFitState:fparam-len, 217  
 NcmFitState:func-eval, 217  
 NcmFitState:grad-eval, 218  
 NcmFitState:is-best-fit, 218  
 NcmFitState:is-least-squares, 218  
 NcmFitState:niters, 218  
 NcmFitStateClass, 215  
 NcmFitType, 222  
 NCM\_FLOOR\_TRUNC, 108  
 ncm\_func\_eval\_set\_max\_threads, 92  
 ncm\_func\_eval\_threaded\_loop, 93  
 ncm\_function\_cache\_clear, 558  
 ncm\_function\_cache\_free, 558

ncm\_function\_cache\_get, 559  
 ncm\_function\_cache\_get\_near, 559  
 ncm\_function\_cache\_insert, 559  
 ncm\_function\_cache\_insert\_vector, 559  
 ncm\_function\_cache\_new, 558  
 NcmFunctionCache, 558  
 NcmFunctionCacheSearchType, 558

## G

g\_clear\_object, 112  
 ncm\_get\_smoothed\_uniform\_sample, 541  
 ncm\_get\_uniform\_sample, 541  
 NcmGrid, 552  
 ncm\_grid\_free, 552  
 ncm\_grid\_get\_double\_array, 554  
 ncm\_grid\_get\_name, 553  
 ncm\_grid\_get\_node\_d, 554  
 NCM\_GRID\_GET\_SEC, 552  
 ncm\_grid\_new, 552  
 ncm\_grid\_new\_from\_sections, 552  
 ncm\_grid\_read, 554  
 ncm\_grid\_set\_nodes\_d, 553  
 ncm\_grid\_set\_nodes\_si, 553  
 ncm\_grid\_set\_sections, 553  
 ncm\_grid\_write, 554  
 NcmGridNodesEndPoints, 551  
 NcmGridSection, 552  
 ncm\_gsl\_odeiv2\_solver, 382

## H

HEALPIX\_INT\_TO\_XY, 267  
 HEALPIX\_NPIX, 267  
 NCM\_HEALPIX\_NULLVAL, 267  
 HEALPIX\_XY\_TO\_INT, 267

## I

NCM\_INTEGRAL\_ABS\_ERROR, 568  
 NCM\_INTEGRAL\_ALG, 567  
 ncm\_integral\_cached\_0\_x, 565  
 ncm\_integral\_cached\_x\_inf, 565  
 NCM\_INTEGRAL\_ERROR, 567  
 ncm\_integral\_fixed\_calc\_nodes, 566  
 ncm\_integral\_fixed\_free, 566  
 ncm\_integral\_fixed\_integ\_mult, 567  
 ncm\_integral\_fixed\_integ\_posdef\_mult, 567  
 ncm\_integral\_fixed\_new, 566  
 ncm\_integral\_fixed\_nodes\_eval, 567  
 ncm\_integral\_get\_workspace, 563  
 ncm\_integral\_locked\_a\_b, 564  
 ncm\_integral\_locked\_a\_inf, 564  
 NCM\_INTEGRAL\_PARTITION, 567  
 NcmIntegralFixed, 563  
 NcmIntegrand2dim, 563  
 ncm\_integrate\_2dim, 565  
 ncm\_interp\_dd\_eval, 527  
 ncm\_interp\_dd\_eval\_2\_4, 527  
 ncm\_interp\_dd\_init, 526

ncm\_interp\_dd\_init\_2\_4, 527

## L

NCM\_LAPACK\_CHECK\_INFO, 28  
 ncm\_lapack\_dptsv, 28  
 ncm\_lh\_ratio1d\_clear, 257  
 ncm\_lh\_ratio1d\_find\_bounds, 257  
 ncm\_lh\_ratio1d\_free, 257  
 ncm\_lh\_ratio1d\_new, 256  
 ncm\_lh\_ratio1d\_set\_pindex, 257  
 ncm\_lh\_ratio2d\_clear, 260  
 ncm\_lh\_ratio2d\_conf\_region, 260  
 ncm\_lh\_ratio2d\_fisher\_border, 261  
 ncm\_lh\_ratio2d\_free, 260  
 ncm\_lh\_ratio2d\_new, 260  
 ncm\_lh\_ratio2d\_region\_clear, 261  
 ncm\_lh\_ratio2d\_region\_dup, 261  
 ncm\_lh\_ratio2d\_region\_free, 261  
 ncm\_lh\_ratio2d\_region\_print, 262  
 ncm\_lh\_ratio2d\_set\_pindex, 260  
 NcmLHRatio1d, 256  
 NcmLHRatio1d:constraint, 257  
 NcmLHRatio1d:fit, 258  
 NcmLHRatio1d:pi, 258  
 NcmLHRatio1dClass, 256  
 NcmLHRatio1dRoot, 256  
 NcmLHRatio2d, 259  
 NcmLHRatio2d:fit, 262  
 NcmLHRatio2d:pi1, 262  
 NcmLHRatio2d:pi2, 262  
 NcmLHRatio2dClass, 259  
 NcmLHRatio2dPoint, 259  
 NcmLHRatio2dRegion, 259  
 NcmLHRatio2dRoot, 259  
 NcmLikelihood, 206  
 NcmLikelihood:dataset, 210  
 ncm\_likelihood\_clear, 207  
 ncm\_likelihood\_copy, 207  
 ncm\_likelihood\_dup, 207  
 ncm\_likelihood\_free, 207  
 ncm\_likelihood\_has\_least\_squares\_J, 208  
 ncm\_likelihood\_has\_m2lnL\_grad, 208  
 ncm\_likelihood\_least\_squares\_f, 209  
 ncm\_likelihood\_least\_squares\_f\_J, 209  
 ncm\_likelihood\_least\_squares\_J, 209  
 ncm\_likelihood\_m2lnL\_grad, 210  
 ncm\_likelihood\_m2lnL\_val, 210  
 ncm\_likelihood\_m2lnL\_val\_grad, 210  
 ncm\_likelihood\_new, 206  
 ncm\_likelihood\_priors\_add, 207  
 ncm\_likelihood\_priors\_least\_squares\_f, 208  
 ncm\_likelihood\_priors\_length, 208  
 ncm\_likelihood\_priors\_m2lnL\_val, 209  
 ncm\_likelihood\_priors\_peek, 208  
 ncm\_likelihood\_ref, 207  
 NcmLikelihoodClass, 206  
 LOAD\_SAVE\_VECTOR\_MATRIX\_DEF, 107

NcmLoopFunc, 92

## M

ncm\_magnus\_iserles\_ode\_eval\_vec, 550  
 ncm\_magnus\_iserles\_ode\_new, 549  
 ncm\_magnus\_iserles\_ode\_step, 550  
 ncm\_magnus\_iserles\_ode\_step\_frac, 550  
 NCM\_MAP\_ALM\_INDEX, 110  
 NCM\_MAP\_ALM\_M\_START, 110  
 NCM\_MAP\_ALM\_SIZE, 109  
 NCM\_MAP\_MAX\_RING\_SIZE, 109  
 NCM\_MAP\_N\_DIFFERENT\_SIZED\_RINGS, 109  
 NCM\_MAP\_N\_IND\_PLM, 109  
 NCM\_MAP\_N\_RINGS, 110  
 NCM\_MAP\_RING\_PLAN\_INDEX, 110  
 NCM\_MAP\_RING\_SIZE, 110  
 NcmMatrix, 19  
 NcmMatrix:values, 27  
 ncm\_matrix\_add\_mul, 26  
 ncm\_matrix\_cholesky\_decomp, 26  
 ncm\_matrix\_clear, 26  
 NCM\_MATRIX\_COL\_LEN, 19  
 NCM\_MATRIX\_DATA, 19  
 ncm\_matrix\_dup, 26  
 ncm\_matrix\_exp\_2x2, 546  
 ncm\_matrix\_fast\_get, 25  
 ncm\_matrix\_fast\_set, 25  
 ncm\_matrix\_free, 26  
 ncm\_matrix\_get, 23  
 ncm\_matrix\_get\_array, 23  
 ncm\_matrix\_get\_col, 22  
 ncm\_matrix\_get\_row, 22  
 ncm\_matrix\_get\_submatrix, 22  
 NCM\_MATRIX\_GSL, 19  
 ncm\_matrix\_memcpy, 25  
 NCM\_MATRIX\_NCOLS, 19  
 ncm\_matrix\_new, 19  
 ncm\_matrix\_new\_array, 20  
 ncm\_matrix\_new\_data\_malloc, 21  
 ncm\_matrix\_new\_data\_slice, 20  
 ncm\_matrix\_new\_data\_static, 21  
 ncm\_matrix\_new\_data\_static\_tda, 21  
 ncm\_matrix\_new\_gsl, 20  
 ncm\_matrix\_new\_gsl\_const, 22  
 ncm\_matrix\_new\_gsl\_static, 20  
 NCM\_MATRIX\_NROWS, 19  
 ncm\_matrix\_ptr, 23  
 ncm\_matrix\_ref, 23  
 NCM\_MATRIX\_ROW\_LEN, 19  
 ncm\_matrix\_scale, 24  
 ncm\_matrix\_set, 24  
 ncm\_matrix\_set\_col, 25  
 ncm\_matrix\_set\_identity, 24  
 ncm\_matrix\_set\_zero, 24  
 ncm\_matrix\_transpose, 24  
 NcmMatrixClass, 19  
 NcmMatrixInternal, 18

`ncm_memory_pool_free`, 561  
`ncm_memory_pool_get`, 561  
`ncm_memory_pool_new`, 561  
`ncm_memory_pool_return`, 561  
`ncm_memory_pool_set_min_size`, 561  
`NcmMemoryPool`, 560  
`NcmMemoryPoolAlloc`, 560  
`NcmMemoryPoolSlice`, 560  
`ncm_message`, 102  
`ncm_message_wv`, 102  
`NcmMIODE`, 549  
`NcmMIODEFunction`, 549  
`NcmModel`, 52  
`NcmModel:implementation`, 67  
`NcmModel:name`, 67  
`NcmModel:nick`, 67  
`NcmModel:params`, 67  
`NcmModel:params-types`, 67  
`NcmModel:reparam`, 67  
`NcmModel:scalar-params-len`, 67  
`NcmModel:vector-params-len`, 68  
`ncm_model_class_add_params`, 53  
`ncm_model_class_check_params_info`, 55  
`ncm_model_class_get_property`, 53  
`ncm_model_class_set_name_nick`, 54  
`ncm_model_class_set_property`, 53  
`ncm_model_class_set_sparam`, 54  
`ncm_model_class_set_vparam`, 54  
`ncm_model_clear`, 56  
`ncm_model_copy`, 55  
`ncm_model_copyto`, 55  
`ncm_model_ctrl_clear`, 70  
`ncm_model_ctrl_copy`, 69  
`ncm_model_ctrl_force_update`, 69  
`ncm_model_ctrl_free`, 70  
`ncm_model_ctrl_get_model`, 69  
`ncm_model_ctrl_model_update`, 70  
`ncm_model_ctrl_new`, 69  
`ncm_model_ctrl_set_model`, 69  
`ncm_model_ctrl_update`, 70  
`ncm_model_free`, 55  
`NCM_MODEL_FUNC0_IMPL`, 66  
`NCM_MODEL_FUNC1_IMPL`, 66  
`ncm_model_id`, 56  
`ncm_model_id_by_type`, 56  
`ncm_model_impl`, 57  
`ncm_model_is_equal`, 56  
`ncm_model_len`, 57  
`NCM_MODEL_MAX_ID`, 73  
`ncm_model_name`, 58  
`ncm_model_nick`, 58  
`ncm_model_orig_param_get`, 60  
`ncm_model_orig_param_set`, 60  
`ncm_model_orig_params_update`, 59  
`ncm_model_orig_vparam_get`, 61  
`ncm_model_orig_vparam_get_vector`, 61  
`ncm_model_orig_vparam_set`, 60  
`ncm_model_orig_vparam_set_vector`, 60  
`ncm_model_param_finite`, 59  
`ncm_model_param_get`, 60  
`ncm_model_param_get_abstol`, 64  
`ncm_model_param_get_ftype`, 64  
`ncm_model_param_get_lower_bound`, 64  
`ncm_model_param_get_scale`, 63  
`ncm_model_param_get_upper_bound`, 64  
`ncm_model_param_index_from_name`, 63  
`ncm_model_param_name`, 63  
`ncm_model_param_set`, 59  
`ncm_model_param_set_abstol`, 65  
`ncm_model_param_set_default`, 59  
`ncm_model_param_set_ftype`, 66  
`ncm_model_param_set_lower_bound`, 65  
`ncm_model_param_set_scale`, 65  
`ncm_model_param_set_upper_bound`, 65  
`ncm_model_param_symbol`, 63  
`ncm_model_params_copyto`, 61  
`ncm_model_params_finite`, 58  
`ncm_model_params_get_all`, 63  
`ncm_model_params_log_all`, 62  
`ncm_model_params_print_all`, 62  
`ncm_model_params_save_as_default`, 61  
`ncm_model_params_set_all`, 61  
`ncm_model_params_set_all_data`, 62  
`ncm_model_params_set_default`, 61  
`ncm_model_params_set_model`, 62  
`ncm_model_params_set_vector`, 62  
`ncm_model_params_update`, 59  
`ncm_model_params_valid`, 63  
`ncm_model_peek_reparam`, 58  
`ncm_model_ref`, 56  
`ncm_model_reparam_df`, 66  
`ncm_model_reparam_J`, 66  
`NCM_MODEL_SET_IMPL_FUNC`, 66  
`ncm_model_set_reparam`, 56  
`ncm_model_sparam_len`, 57  
`ncm_model_vparam_array_len`, 57  
`ncm_model_vparam_index`, 57  
`ncm_model_vparam_len`, 58  
`NcmModelClass`, 52  
`NcmModelCtrl`, 69  
`NcmModelCtrl:model`, 70  
`NcmModelCtrlClass`, 68  
`NcmModelFunc0`, 53  
`NcmModelFunc1`, 53  
`NcmModelID`, 52  
`mp_rnd_t`, 263  
`ncm_mpfr_inp_raw`, 543  
`ncm_mpfr_out_raw`, 543  
`mpfr_ptr`, 263  
`mpfr_t`, 263  
`ncm_mpq_hash_new`, 547  
`ncm_mpq_inp_raw`, 543  
`ncm_mpq_out_raw`, 543  
`mpq_ptr`, 263

mpq\_t, 263  
 ncm\_mpq\_tree\_new, 547  
 ncm\_mpsf\_OF1\_d, 163  
 ncm\_mpsf\_OF1\_q, 162  
 ncm\_mpsf\_sbessel, 164  
 ncm\_mpsf\_sbessel\_d, 164  
 NCM\_MPSF\_SBESSEL\_INT\_MAP, 169  
 ncm\_mpsf\_sbessel\_integrate, 169  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral, 169  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_a\_b, 172  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_q, 169  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_calc\_d2jl, 171  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_calc\_djl, 171  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_free, 170  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_goto, 171  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_new, 170  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_next, 170  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_previous, 171  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_read, 170  
 ncm\_mpsf\_sbessel\_jl\_xj\_integral\_recur\_write, 172  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_cached\_new, 173  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_goto, 174  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_load, 173  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_new, 172  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_new\_from\_sections, 172  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_next, 174  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_prepare, 173  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_prepare\_new, 173  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_previous, 174  
 ncm\_mpsf\_sbessel\_jl\_xj\_integrate\_spline\_save, 174  
 ncm\_mpsf\_sbessel\_recur\_free, 165  
 ncm\_mpsf\_sbessel\_recur\_goto, 166  
 ncm\_mpsf\_sbessel\_recur\_new, 165  
 ncm\_mpsf\_sbessel\_recur\_next, 166  
 ncm\_mpsf\_sbessel\_recur\_previous, 166  
 ncm\_mpsf\_sbessel\_recur\_read, 165  
 ncm\_mpsf\_sbessel\_recur\_set\_d, 165  
 ncm\_mpsf\_sbessel\_recur\_set\_q, 165  
 ncm\_mpsf\_sbessel\_recur\_write, 166  
 ncm\_mpsf\_sin\_int\_mpfr, 162  
 NcmMpsfSBesselIntegRecur, 168  
 NcmMpsfSBesselIntSpline, 168  
 NcmMpsfSBesselRecur, 164  
 mpz\_clears, 110  
 ncm\_mpz\_clears, 544  
 mpz\_inits, 110  
 ncm\_mpz\_inits, 544  
 mpz\_t, 263  
 NcmMSet, 74  
 ncm\_mset\_clear, 77  
 ncm\_mset\_cmp, 80  
 ncm\_mset\_copy, 76  
 ncm\_mset\_copyto, 76  
 ncm\_mset\_dup, 76  
 ncm\_mset\_empty\_new, 75  
 ncm\_mset\_exists, 78  
 ncm\_mset\_fparam\_get, 87  
 ncm\_mset\_fparam\_get\_abstol, 86  
 ncm\_mset\_fparam\_get\_fpi, 87  
 ncm\_mset\_fparam\_get\_lower\_bound, 86  
 ncm\_mset\_fparam\_get\_pi, 87  
 ncm\_mset\_fparam\_get\_scale, 86  
 ncm\_mset\_fparam\_get\_upper\_bound, 86  
 ncm\_mset\_fparam\_len, 78  
 ncm\_mset\_fparam\_name, 85  
 ncm\_mset\_fparam\_set, 87  
 ncm\_mset\_fparams\_get\_vector, 84  
 ncm\_mset\_fparams\_len, 85  
 ncm\_mset\_fparams\_set\_array, 85  
 ncm\_mset\_fparams\_set\_gsl\_vector, 85  
 ncm\_mset\_fparams\_set\_vector, 85  
 ncm\_mset\_free, 77  
 ncm\_mset\_func\_array\_new, 90  
 ncm\_mset\_func\_eval, 90  
 ncm\_mset\_func\_eval0, 90  
 ncm\_mset\_func\_eval1, 91  
 ncm\_mset\_func\_free, 90  
 ncm\_mset\_func\_has\_params, 91  
 ncm\_mset\_func\_is\_const, 91  
 ncm\_mset\_func\_is\_scalar, 91  
 ncm\_mset\_func\_is\_vector, 91  
 ncm\_mset\_func\_new, 89  
 ncm\_mset\_func\_numdiff\_fparams, 92  
 ncm\_mset\_func\_ref, 90  
 ncm\_mset\_get, 77  
 ncm\_mset\_load, 88  
 ncm\_mset\_max\_fparam\_name, 79  
 ncm\_mset\_max\_model\_nick, 79  
 ncm\_mset\_max\_param\_name, 79  
 NCM\_MSET\_MODEL\_DECLARE\_ID, 74  
 NCM\_MSET\_MODEL\_ID\_FUNC, 74  
 NCM\_MSET\_MODEL\_REGISTER\_ID, 74  
 ncm\_mset\_model\_register\_id, 74  
 ncm\_mset\_new, 75  
 ncm\_mset\_new\_array, 76  
 ncm\_mset\_newv, 76  
 ncm\_mset\_orig\_param\_get, 81  
 ncm\_mset\_param\_get, 81  
 ncm\_mset\_param\_get\_abstol, 83  
 ncm\_mset\_param\_get\_ftype, 84  
 ncm\_mset\_param\_get\_lower\_bound, 83  
 ncm\_mset\_param\_get\_pi, 84  
 ncm\_mset\_param\_get\_scale, 83  
 ncm\_mset\_param\_get\_upper\_bound, 83  
 ncm\_mset\_param\_get\_vector, 82  
 ncm\_mset\_param\_len, 81  
 ncm\_mset\_param\_name, 81  
 ncm\_mset\_param\_set, 80  
 ncm\_mset\_param\_set\_all\_ftype, 82  
 ncm\_mset\_param\_set\_ftype, 82  
 ncm\_mset\_param\_set\_pi, 84  
 ncm\_mset\_param\_set\_vector, 82  
 ncm\_mset\_params\_log\_vals, 80

ncm\_mset\_params\_pretty\_print, 79  
 ncm\_mset\_params\_print\_vals, 80  
 ncm\_mset\_params\_valid, 80  
 ncm\_mset\_peek, 77  
 ncm\_mset\_pindex\_dup, 75  
 ncm\_mset\_pindex\_free, 75  
 ncm\_mset\_pindex\_new, 75  
 ncm\_mset\_prepare\_fparam\_map, 78  
 ncm\_mset\_pretty\_log, 79  
 ncm\_mset\_ref, 76  
 ncm\_mset\_remove, 77  
 ncm\_mset\_save, 88  
 ncm\_mset\_set, 78  
 ncm\_mset\_total\_len, 78  
 NcmMSetClass, 73  
 NcmMSetFunc, 89  
 NcmMSetFuncClass, 89  
 NcmMSetFuncN, 89  
 NcmMSetModelDesc, 73  
 NcmMSetPIndex, 74

## N

NCM\_N2VECTOR, 7  
 nc\_bias\_mean\_prepare, 488  
 nc\_bias\_mean\_val, 489  
 nc\_ca\_mean\_bias, 489  
 nc\_ca\_mean\_bias\_denominator, 489  
 nc\_ca\_mean\_bias\_Mobs\_denominator, 489  
 nc\_ca\_mean\_bias\_Mobs\_numerator, 489  
 nc\_ca\_mean\_bias\_numerator, 489  
 nc\_cluster\_abundance\_clear, 484  
 nc\_cluster\_abundance\_copy, 483  
 nc\_cluster\_abundance\_d2n, 486  
 nc\_cluster\_abundance\_d2NdlnM\_val, 482  
 nc\_cluster\_abundance\_dNdlnM\_val, 482  
 nc\_cluster\_abundance\_dNdz\_val, 482  
 nc\_cluster\_abundance\_free, 484  
 nc\_cluster\_abundance\_get\_mass, 488  
 nc\_cluster\_abundance\_get\_redshift, 488  
 nc\_cluster\_abundance\_intp\_d2n, 487  
 nc\_cluster\_abundance\_lnm\_p\_d2n, 486  
 nc\_cluster\_abundance\_n, 487  
 nc\_cluster\_abundance\_N\_val, 482  
 nc\_cluster\_abundance\_new, 483  
 nc\_cluster\_abundance\_nodist\_new, 483  
 nc\_cluster\_abundance\_peek\_mass, 488  
 nc\_cluster\_abundance\_peek\_redshift, 488  
 nc\_cluster\_abundance\_prepare, 484  
 nc\_cluster\_abundance\_prepare\_inv\_dNdlnM\_z, 485  
 nc\_cluster\_abundance\_prepare\_inv\_dNdz, 484  
 nc\_cluster\_abundance\_ref, 484  
 nc\_cluster\_abundance\_set\_mass, 488  
 nc\_cluster\_abundance\_set\_redshift, 487  
 nc\_cluster\_abundance\_true\_n, 487  
 nc\_cluster\_abundance\_z\_p\_d2n, 485  
 nc\_cluster\_abundance\_z\_p\_lnm\_p\_d2n, 485  
 NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_A\_SZ, 473

NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_B\_SZ, 473  
 NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_C\_SZ, 473  
 NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_D\_SZ, 473  
 NC\_CLUSTER\_MASS\_BENSON\_DEFAULT\_PARAMS\_ABSTO, 473  
 NC\_CLUSTER\_MASS\_BENSON\_M\_LOWER\_BOUND, 473  
 NC\_CLUSTER\_MASS\_BENSON\_XI\_ZETA\_DIST\_CUT, 473  
 NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_A\_X, 477  
 NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_B\_X, 477  
 NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_C\_X, 477  
 NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_D\_X, 477  
 NC\_CLUSTER\_MASS\_BENSON\_XRAY\_DEFAULT\_PARAMS, 477  
 nc\_cluster\_mass\_clear, 463  
 nc\_cluster\_mass\_free, 463  
 nc\_cluster\_mass\_impl, 463  
 NC\_CLUSTER\_MASS\_IMPL\_ALL, 462  
 nc\_cluster\_mass\_intp, 464  
 NC\_CLUSTER\_MASS\_LNNORMAL\_BIAS, 467  
 NC\_CLUSTER\_MASS\_LNNORMAL\_SIGMA, 467  
 nc\_cluster\_mass\_log\_all\_models, 465  
 nc\_cluster\_mass\_n\_limits, 465  
 nc\_cluster\_mass\_new\_from\_name, 462  
 nc\_cluster\_mass\_obs\_len, 463  
 nc\_cluster\_mass\_obs\_params\_len, 463  
 nc\_cluster\_mass\_p, 464  
 nc\_cluster\_mass\_p\_limits, 465  
 nc\_cluster\_mass\_ref, 462  
 nc\_cluster\_mass\_resample, 464  
 NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_A\_SZ, 469  
 NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_B\_SZ, 469  
 NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_C\_SZ, 469  
 NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_D\_SZ, 469  
 NC\_CLUSTER\_MASS\_VANDERLINDE\_DEFAULT\_PARAMS, 469  
 NC\_CLUSTER\_PHOTOZ\_GAUSS\_BIAS, 457  
 nc\_cluster\_photoz\_gauss\_global\_get\_sigma0, 460  
 nc\_cluster\_photoz\_gauss\_global\_get\_z\_bias, 459  
 nc\_cluster\_photoz\_gauss\_global\_new, 459  
 nc\_cluster\_photoz\_gauss\_global\_set\_sigma0, 459  
 nc\_cluster\_photoz\_gauss\_global\_set\_z\_bias, 459  
 nc\_cluster\_photoz\_gauss\_new, 457  
 NC\_CLUSTER\_PHOTOZ\_GAUSS\_SIGMA, 457  
 nc\_cluster\_redshift\_clear, 453  
 nc\_cluster\_redshift\_free, 453  
 nc\_cluster\_redshift\_impl, 453  
 NC\_CLUSTER\_REDSHIFT\_IMPL\_ALL, 452  
 nc\_cluster\_redshift\_intp, 454



nc\_cluster\_redshift\_n\_limits, 455  
nc\_cluster\_redshift\_new\_from\_name, 452  
nc\_cluster\_redshift\_obs\_len, 453  
nc\_cluster\_redshift\_obs\_params\_len, 453  
nc\_cluster\_redshift\_p, 454  
nc\_cluster\_redshift\_p\_limits, 455  
nc\_cluster\_redshift\_ref, 453  
nc\_cluster\_redshift\_resample, 454  
nc\_data\_bao\_a\_get\_sample, 503  
nc\_data\_bao\_a\_get\_size, 502  
nc\_data\_bao\_a\_new, 502  
nc\_data\_bao\_a\_set\_sample, 503  
nc\_data\_bao\_a\_set\_size, 502  
nc\_data\_bao\_create, 501  
nc\_data\_bao\_dv\_get\_sample, 505  
nc\_data\_bao\_dv\_get\_size, 505  
nc\_data\_bao\_dv\_new, 504  
nc\_data\_bao\_dv\_set\_sample, 505  
nc\_data\_bao\_dv\_set\_size, 504  
nc\_data\_bao\_dvdv\_get\_sample, 509  
nc\_data\_bao\_dvdv\_new, 509  
nc\_data\_bao\_dvdv\_set\_sample, 509  
nc\_data\_bao\_rdv\_get\_sample, 507  
nc\_data\_bao\_rdv\_get\_size, 507  
nc\_data\_bao\_rdv\_new, 506  
nc\_data\_bao\_rdv\_set\_sample, 507  
nc\_data\_bao\_rdv\_set\_size, 507  
nc\_data\_cluster\_ncount\_bin\_data, 523  
nc\_data\_cluster\_ncount\_binned\_create\_func, 522  
nc\_data\_cluster\_ncount\_binned\_init\_from\_sampling, 521  
nc\_data\_cluster\_ncount\_binned\_init\_from\_text\_file\_gkey, 521  
nc\_data\_cluster\_ncount\_binned\_lnM\_z\_new, 522  
nc\_data\_cluster\_ncount\_binned\_new, 521  
nc\_data\_cluster\_ncount\_binned\_save, 522  
nc\_data\_cluster\_ncount\_catalog\_load, 524  
nc\_data\_cluster\_ncount\_catalog\_save, 523  
nc\_data\_cluster\_ncount\_free, 521  
nc\_data\_cluster\_ncount\_hist\_lnM\_z, 523  
nc\_data\_cluster\_ncount\_init\_from\_sampling, 522  
nc\_data\_cluster\_ncount\_new, 521  
nc\_data\_cluster\_ncount\_print, 523  
nc\_data\_cluster\_ncount\_ref, 521  
nc\_data\_cluster\_ncount\_true\_data, 522  
nc\_data\_cluster\_poisson\_new, 525  
nc\_data\_cluster\_poisson\_new\_cad, 525  
nc\_data\_cmb\_create, 492  
nc\_data\_cmb\_dist\_priors\_get\_sample, 495  
nc\_data\_cmb\_dist\_priors\_new, 494  
nc\_data\_cmb\_dist\_priors\_set\_sample, 495  
nc\_data\_cmb\_shift\_param\_get\_sample, 493  
nc\_data\_cmb\_shift\_param\_new, 493  
nc\_data\_cmb\_shift\_param\_set\_sample, 493  
nc\_data\_dist\_mu\_get\_sample, 514  
nc\_data\_dist\_mu\_get\_size, 513  
nc\_data\_dist\_mu\_new, 513  
nc\_data\_dist\_mu\_set\_sample, 513  
nc\_data\_dist\_mu\_set\_size, 513  
nc\_data\_hubble\_bao\_get\_sample, 500  
nc\_data\_hubble\_bao\_get\_size, 499  
nc\_data\_hubble\_bao\_new, 499  
nc\_data\_hubble\_bao\_set\_sample, 499  
nc\_data\_hubble\_bao\_set\_size, 499  
nc\_data\_hubble\_get\_sample, 497  
nc\_data\_hubble\_get\_size, 497  
nc\_data\_hubble\_new, 497  
nc\_data\_hubble\_set\_sample, 497  
nc\_data\_hubble\_set\_size, 497  
NC\_DATA\_SNIA\_COV\_COLOUR\_KEY, 518  
NC\_DATA\_SNIA\_COV\_DATA\_GROUP, 518  
NC\_DATA\_SNIA\_COV\_DATA\_KEY, 518  
NC\_DATA\_SNIA\_COV\_DATA\_LEN\_KEY, 518  
NC\_DATA\_SNIA\_COV\_END, 511  
NC\_DATA\_SNIA\_COV\_LEN, 511  
NC\_DATA\_SNIA\_COV\_LENGTH, 516  
nc\_data\_snica\_cov\_load, 517  
nc\_data\_snica\_cov\_load\_txt, 517  
NC\_DATA\_SNIA\_COV\_MAG\_COLOUR\_KEY, 518  
NC\_DATA\_SNIA\_COV\_MAG\_KEY, 518  
NC\_DATA\_SNIA\_COV\_MAG\_WIDTH\_KEY, 518  
nc\_data\_snica\_cov\_new, 516  
nc\_data\_snica\_cov\_new\_full, 517  
nc\_data\_snica\_cov\_save, 517  
nc\_data\_snica\_cov\_set\_size, 517  
NC\_DATA\_SNIA\_COV\_START, 511  
NC\_DATA\_SNIA\_COV\_WIDTH\_COLOUR\_KEY, 518  
NC\_DATA\_SNIA\_COV\_WIDTH\_KEY, 518  
nc\_data\_snica\_get\_catalog, 511  
nc\_data\_snica\_get\_catalog\_by\_id, 512  
nc\_data\_snica\_get\_fits, 511  
nc\_data\_snica\_load\_cat, 511  
NC\_DATA\_SNIA\_SIMPLE\_END, 511  
NC\_DATA\_SNIA\_SIMPLE\_LEN, 511  
NC\_DATA\_SNIA\_SIMPLE\_START, 511  
nc\_distance\_acoustic\_scale, 346  
nc\_distance\_angular\_diameter\_curvature\_scale, 346  
nc\_distance\_bao\_A\_scale, 349  
nc\_distance\_bao\_r\_Dv, 350  
nc\_distance\_clear, 344  
nc\_distance\_comoving, 346  
nc\_distance\_comoving\_lss, 345  
nc\_distance\_conformal\_lookback\_time, 350  
nc\_distance\_conformal\_time, 350  
nc\_distance\_conformal\_time\_mks\_scale, 350  
nc\_distance\_cosmic\_time, 350  
nc\_distance\_cosmic\_time\_mks\_scale, 350  
nc\_distance\_decoupling\_redshift, 344  
nc\_distance\_dilation\_scale, 348  
nc\_distance\_drag\_redshift, 345  
nc\_distance\_dsound\_horizon\_dz, 349  
nc\_distance\_free, 343  
nc\_distance\_func0\_new, 344  
nc\_distance\_func1\_new, 344  
nc\_distance\_hubble, 344  
nc\_distance\_lookback\_time, 350

nc\_distance\_luminosity, 347  
nc\_distance\_luminosity\_hef, 347  
nc\_distance\_modulus, 347  
nc\_distance\_modulus\_hef, 348  
nc\_distance\_new, 343  
nc\_distance\_Omega\_k, 346  
nc\_distance\_prepare, 343  
nc\_distance\_prepare\_if\_needed, 343  
nc\_distance\_ref, 343  
nc\_distance\_shift\_parameter, 348  
nc\_distance\_shift\_parameter\_lss, 345  
nc\_distance\_sound\_horizon, 349  
nc\_distance\_transverse, 347  
NC\_FUNCTION\_CACHE, 559  
nc\_galaxy\_acf\_new, 450  
nc\_galaxy\_acf\_prepare\_psi, 451  
nc\_galaxy\_acf\_psi, 451  
nc\_growth\_func\_clear, 395  
nc\_growth\_func\_copy, 395  
nc\_growth\_func\_eval, 396  
nc\_growth\_func\_eval\_both, 396  
nc\_growth\_func\_eval\_deriv, 396  
nc\_growth\_func\_free, 395  
nc\_growth\_func\_new, 395  
nc\_growth\_func\_prepare, 396  
nc\_halo\_bias\_func\_clear, 449  
nc\_halo\_bias\_func\_copy, 449  
nc\_halo\_bias\_func\_free, 449  
nc\_halo\_bias\_func\_integrand, 449  
nc\_halo\_bias\_func\_new, 448  
nc\_halo\_bias\_type\_clear, 435  
nc\_halo\_bias\_type\_eval, 435  
nc\_halo\_bias\_type\_free, 435  
nc\_halo\_bias\_type\_new\_from\_name, 435  
nc\_halo\_bias\_type\_ps\_get\_delta\_c, 436  
nc\_halo\_bias\_type\_ps\_new, 436  
nc\_halo\_bias\_type\_ps\_set\_delta\_c, 436  
nc\_halo\_bias\_type\_st\_ellip\_get\_a, 442  
nc\_halo\_bias\_type\_st\_ellip\_get\_b, 442  
nc\_halo\_bias\_type\_st\_ellip\_get\_c, 442  
nc\_halo\_bias\_type\_st\_ellip\_get\_delta\_c, 441  
nc\_halo\_bias\_type\_st\_ellip\_new, 441  
nc\_halo\_bias\_type\_st\_ellip\_set\_a, 441  
nc\_halo\_bias\_type\_st\_ellip\_set\_b, 442  
nc\_halo\_bias\_type\_st\_ellip\_set\_c, 442  
nc\_halo\_bias\_type\_st\_ellip\_set\_delta\_c, 441  
nc\_halo\_bias\_type\_st\_spher\_get\_a, 439  
nc\_halo\_bias\_type\_st\_spher\_get\_delta\_c, 438  
nc\_halo\_bias\_type\_st\_spher\_get\_p, 439  
nc\_halo\_bias\_type\_st\_spher\_new, 438  
nc\_halo\_bias\_type\_st\_spher\_set\_a, 438  
nc\_halo\_bias\_type\_st\_spher\_set\_delta\_c, 438  
nc\_halo\_bias\_type\_st\_spher\_set\_p, 439  
nc\_halo\_bias\_type\_tinker\_get\_B, 445  
nc\_halo\_bias\_type\_tinker\_get\_b, 446  
nc\_halo\_bias\_type\_tinker\_get\_c, 446  
nc\_halo\_bias\_type\_tinker\_get\_Delta, 447  
nc\_halo\_bias\_type\_tinker\_get\_delta\_c, 445  
nc\_halo\_bias\_type\_tinker\_new, 444  
nc\_halo\_bias\_type\_tinker\_set\_B, 445  
nc\_halo\_bias\_type\_tinker\_set\_b, 446  
nc\_halo\_bias\_type\_tinker\_set\_c, 446  
nc\_halo\_bias\_type\_tinker\_set\_Delta, 446  
nc\_halo\_bias\_type\_tinker\_set\_delta\_c, 445  
nc\_hicosmo\_as\_drag, 278  
nc\_hicosmo\_c\_H0, 280  
nc\_hicosmo\_cd, 279  
nc\_hicosmo\_create\_mset\_func0, 284  
nc\_hicosmo\_create\_mset\_func1, 284  
nc\_hicosmo\_d2E2\_dz2, 279  
nc\_hicosmo\_dE2\_dz, 279  
NC\_HICOSMO\_DE\_DEFAULT\_H0, 294  
NC\_HICOSMO\_DE\_DEFAULT\_OMEGA\_B, 295  
NC\_HICOSMO\_DE\_DEFAULT\_OMEGA\_C, 294  
NC\_HICOSMO\_DE\_DEFAULT\_OMEGA\_X, 294  
NC\_HICOSMO\_DE\_DEFAULT\_SIGMA8, 295  
NC\_HICOSMO\_DE\_DEFAULT\_SPECINDEX, 295  
NC\_HICOSMO\_DE\_DEFAULT\_T\_GAMMA0, 295  
nc\_hicosmo\_de\_dweff\_dz, 296  
NC\_HICOSMO\_DE\_LINDER\_DEFAULT\_W0, 301  
NC\_HICOSMO\_DE\_LINDER\_DEFAULT\_W1, 301  
NC\_HICOSMO\_DE\_LINDER\_N, 301  
nc\_hicosmo\_de\_linder\_new, 302  
nc\_hicosmo\_de\_new\_add\_bbn, 295  
nc\_hicosmo\_de\_omega\_x2omega\_k, 295  
NC\_HICOSMO\_DE\_PAD\_DEFAULT\_W0, 303  
NC\_HICOSMO\_DE\_PAD\_DEFAULT\_W1, 303  
nc\_hicosmo\_de\_pad\_new, 304  
NC\_HICOSMO\_DE\_PM\_N, 303  
NC\_HICOSMO\_DE\_QE\_DEFAULT\_W0, 305  
NC\_HICOSMO\_DE\_QE\_DEFAULT\_W1, 305  
NC\_HICOSMO\_DE\_QE\_N, 306  
nc\_hicosmo\_de\_qe\_new, 306  
nc\_hicosmo\_de\_set\_dweff\_dz\_impl, 296  
nc\_hicosmo\_de\_set\_weff\_impl, 296  
nc\_hicosmo\_de\_set\_wmap5\_params, 295  
nc\_hicosmo\_de\_weff, 296  
NC\_HICOSMO\_DE\_XCDM\_DEFAULT\_W0, 299  
NC\_HICOSMO\_DE\_XCDM\_N, 299  
nc\_hicosmo\_de\_xcdm\_new, 300  
nc\_hicosmo\_dec, 283  
NC\_HICOSMO\_DEFAULT\_PARAMS\_ABSTOL, 287  
NC\_HICOSMO\_DEFAULT\_PARAMS\_RELTOL, 287  
nc\_hicosmo\_dH\_dz, 282  
nc\_hicosmo\_E, 281  
nc\_hicosmo\_E2, 279  
nc\_hicosmo\_Em2, 281  
nc\_hicosmo\_free, 284  
nc\_hicosmo\_H, 282  
nc\_hicosmo\_h, 280  
nc\_hicosmo\_H0, 277  
nc\_hicosmo\_h2, 280  
nc\_hicosmo\_j, 282  
nc\_hicosmo\_lcdm\_new, 290

---

nc\_hicosmo\_log\_all\_models, 283  
 nc\_hicosmo\_new\_from\_name, 283  
 nc\_hicosmo\_Omega\_b, 277  
 nc\_hicosmo\_Omega\_bh2, 281  
 nc\_hicosmo\_Omega\_c, 278  
 nc\_hicosmo\_Omega\_ch2, 281  
 nc\_hicosmo\_Omega\_k, 280  
 nc\_hicosmo\_Omega\_m, 280  
 nc\_hicosmo\_Omega\_mh2, 281  
 nc\_hicosmo\_Omega\_r, 277  
 nc\_hicosmo\_Omega\_rh2, 281  
 nc\_hicosmo\_Omega\_t, 278  
 nc\_hicosmo\_powspec, 279  
 nc\_hicosmo\_prior\_top\_add, 289  
 nc\_hicosmo\_prior\_top\_free, 288  
 nc\_hicosmo\_prior\_top\_new, 288  
 nc\_hicosmo\_prior\_top\_set, 288  
 nc\_hicosmo\_q, 283  
 NC\_HICOSMO\_QCONST\_DEFAULT\_CD, 318  
 NC\_HICOSMO\_QCONST\_DEFAULT\_E, 318  
 NC\_HICOSMO\_QCONST\_DEFAULT\_H0, 318  
 NC\_HICOSMO\_QCONST\_DEFAULT\_OMEGA\_T, 318  
 NC\_HICOSMO\_QCONST\_DEFAULT\_Q, 318  
 NC\_HICOSMO\_QCONST\_DEFAULT\_Z1, 318  
 nc\_hicosmo\_qconst\_new, 319  
 nc\_hicosmo\_qg\_alphaprime2, 312  
 nc\_hicosmo\_qg\_beta, 311  
 nc\_hicosmo\_qg\_cs2, 311  
 nc\_hicosmo\_qg\_cs2\_lambda, 314  
 nc\_hicosmo\_qg\_cs2\_xxbar2, 311  
 nc\_hicosmo\_qg\_cs2zeta2\_int\_1\_zeta2\_lambda, 314  
 nc\_hicosmo\_qg\_d2sqrtxxbarzeta\_sqrtxxbarzeta, 310  
 nc\_hicosmo\_qg\_dalphaprime2\_dalpha, 312  
 nc\_hicosmo\_qg\_dcs2, 311  
 nc\_hicosmo\_qg\_ddzeta\_zeta, 311  
 nc\_hicosmo\_qg\_dxxbarzeta2\_xxbarzeta2, 310  
 nc\_hicosmo\_qg\_dzeta\_zeta, 311  
 nc\_hicosmo\_qg\_eta\_lambda, 313  
 nc\_hicosmo\_qg\_evolvefunc, 316  
 nc\_hicosmo\_qg\_gbar2, 310  
 nc\_hicosmo\_qg\_gbar\_lambda, 313  
 nc\_hicosmo\_qg\_gbarbar, 310  
 nc\_hicosmo\_qg\_gbarbar\_lambda, 313  
 nc\_hicosmo\_qg\_get\_eta\_b, 310  
 nc\_hicosmo\_qg\_get\_lambda\_d, 313  
 nc\_hicosmo\_qg\_get\_lambda\_f, 313  
 nc\_hicosmo\_qg\_get\_lambda\_i, 313  
 nc\_hicosmo\_qg\_h\_to\_R\_matrix, 312  
 nc\_hicosmo\_qg\_int\_1\_zeta2\_lambda, 314  
 nc\_hicosmo\_qg\_lambda\_k\_cross, 314  
 nc\_hicosmo\_qg\_lambda\_x, 312  
 nc\_hicosmo\_qg\_max\_z, 310  
 nc\_hicosmo\_qg\_modefunc, 315  
 nc\_hicosmo\_qg\_modefunc\_evolve, 316  
 nc\_hicosmo\_qg\_modefunc\_init, 316  
 nc\_hicosmo\_qg\_modefunc\_set\_opts, 316  
 nc\_hicosmo\_qg\_modefunc\_sol, 316  
 nc\_hicosmo\_qg\_modefuncm\_cvode\_init, 316  
 nc\_hicosmo\_qg\_new, 309  
 nc\_hicosmo\_qg\_past\_sol, 312  
 nc\_hicosmo\_qg\_pert\_evolve, 315  
 nc\_hicosmo\_qg\_pert\_h\_to\_R, 317  
 nc\_hicosmo\_qg\_pert\_init, 315  
 nc\_hicosmo\_qg\_pert\_new, 315  
 nc\_hicosmo\_qg\_pert\_powerspectrum, 315  
 nc\_hicosmo\_qg\_pert\_prepare\_pw\_spline, 315  
 nc\_hicosmo\_qg\_pert\_R\_to\_h, 316  
 nc\_hicosmo\_qg\_pert\_set\_opts, 315  
 nc\_hicosmo\_qg\_R\_to\_h\_matrix, 313  
 nc\_hicosmo\_qg\_V, 312  
 nc\_hicosmo\_qg\_V\_lambda, 314  
 nc\_hicosmo\_qg\_x\_lambda, 313  
 nc\_hicosmo\_qg\_xbar, 310  
 nc\_hicosmo\_qg\_xddzeta\_zeta\_mxdzeta\_zeta2\_dzeta\_zeta, 311  
 nc\_hicosmo\_qg\_xxbarzeta2, 310  
 nc\_hicosmo\_qg\_zeta, 311  
 nc\_hicosmo\_qlinear\_dE, 323  
 NC\_HICOSMO\_QLINEAR\_DEFAULT\_CD, 322  
 NC\_HICOSMO\_QLINEAR\_DEFAULT\_E, 323  
 NC\_HICOSMO\_QLINEAR\_DEFAULT\_H0, 322  
 NC\_HICOSMO\_QLINEAR\_DEFAULT\_OMEGA\_T, 322  
 NC\_HICOSMO\_QLINEAR\_DEFAULT\_Q, 323  
 NC\_HICOSMO\_QLINEAR\_DEFAULT\_QP, 323  
 NC\_HICOSMO\_QLINEAR\_DEFAULT\_Z1, 323  
 nc\_hicosmo\_qlinear\_new, 323  
 nc\_hicosmo\_qp, 282  
 nc\_hicosmo\_qpw\_add\_asymptotic\_cdm\_prior, 329  
 nc\_hicosmo\_qpw\_add\_continuity\_prior, 329  
 nc\_hicosmo\_qpw\_add\_continuity\_priors, 329  
 nc\_hicosmo\_qpw\_change\_params, 329  
 nc\_hicosmo\_qpw\_change\_params\_qpp, 330  
 NC\_HICOSMO\_QPW\_DEFAULT\_H0, 327  
 NC\_HICOSMO\_QPW\_DEFAULT\_OMEGA\_T, 328  
 NC\_HICOSMO\_QPW\_DEFAULT\_Q0, 328  
 NC\_HICOSMO\_QPW\_DEFAULT\_QP, 328  
 NC\_HICOSMO\_QPW\_DEFAULT\_QP\_LEN, 328  
 nc\_hicosmo\_qpw\_index, 330  
 nc\_hicosmo\_qpw\_new, 328  
 nc\_hicosmo\_qspline\_add\_continuity\_constraint, 335  
 nc\_hicosmo\_qspline\_add\_continuity\_constraints, 336  
 nc\_hicosmo\_qspline\_add\_continuity\_prior, 335  
 nc\_hicosmo\_qspline\_add\_continuity\_priors, 335  
 nc\_hicosmo\_qspline\_cont\_prior\_free, 336  
 nc\_hicosmo\_qspline\_cont\_prior\_get\_lnsigma, 338  
 nc\_hicosmo\_qspline\_cont\_prior\_get\_nknots, 337  
 nc\_hicosmo\_qspline\_cont\_prior\_new, 336  
 nc\_hicosmo\_qspline\_cont\_prior\_ref, 336  
 nc\_hicosmo\_qspline\_cont\_prior\_set\_all\_lnsigma, 337  
 nc\_hicosmo\_qspline\_cont\_prior\_set\_lnsigma, 337  
 nc\_hicosmo\_qspline\_cont\_prior\_set\_nknots, 337  
 NC\_HICOSMO\_QSPLINE\_DEFAULT\_AS\_DRAG, 334  
 NC\_HICOSMO\_QSPLINE\_DEFAULT\_H0, 334  
 NC\_HICOSMO\_QSPLINE\_DEFAULT\_OMEGA\_T, 334  
 NC\_HICOSMO\_QSPLINE\_DEFAULT\_Q, 334

---



NC\_HICOSMO\_QSPLINE\_DEFAULT\_Q\_LEN, 334  
 nc\_hicosmo\_qspline\_new, 334  
 nc\_hicosmo\_set\_as\_drag\_impl, 286  
 nc\_hicosmo\_set\_cd\_impl, 286  
 nc\_hicosmo\_set\_d2E2\_dz2\_impl, 286  
 nc\_hicosmo\_set\_dE2\_dz\_impl, 286  
 nc\_hicosmo\_set\_E2\_impl, 286  
 nc\_hicosmo\_set\_H0\_impl, 284  
 nc\_hicosmo\_set\_Omega\_b\_impl, 284  
 nc\_hicosmo\_set\_Omega\_c\_impl, 285  
 nc\_hicosmo\_set\_Omega\_r\_impl, 284  
 nc\_hicosmo\_set\_Omega\_t\_impl, 285  
 nc\_hicosmo\_set\_powspec\_impl, 287  
 nc\_hicosmo\_set\_sigma\_8\_impl, 285  
 nc\_hicosmo\_set\_T\_gamma0\_impl, 285  
 nc\_hicosmo\_set\_z\_lss\_impl, 285  
 nc\_hicosmo\_sigma\_8, 278  
 nc\_hicosmo\_T\_gamma0, 278  
 nc\_hicosmo\_wec, 283  
 nc\_hicosmo\_z\_lss, 278  
 nc\_hydrodyn\_adiabatic\_stub, 383  
 NC\_LINEAR\_PERTURBATIONS, 379  
 NC\_LINEAR\_PERTURBATIONS\_GET\_SPLINE, 379  
 NC\_LINEAR\_PERTURBATIONS\_SPLINE\_ALL, 379  
 nc\_mass\_function\_alpha\_eff, 433  
 nc\_mass\_function\_clear, 429  
 nc\_mass\_function\_copy, 429  
 nc\_mass\_function\_d2n\_dzdlm, 431  
 nc\_mass\_function\_dn\_dlnm, 430  
 nc\_mass\_function\_dn\_dz, 431  
 nc\_mass\_function\_dn\_M1\_to\_M2\_dv, 432  
 nc\_mass\_function\_dn\_M\_to\_inf\_dv, 432  
 nc\_mass\_function\_dv\_dzdomaga, 431  
 nc\_mass\_function\_free, 429  
 nc\_mass\_function\_n, 432  
 nc\_mass\_function\_new, 428  
 nc\_mass\_function\_prepare, 430  
 nc\_mass\_function\_prepare\_if\_needed, 430  
 nc\_mass\_function\_set\_area, 429  
 nc\_mass\_function\_set\_area\_sd, 429  
 nc\_mass\_function\_set\_eval\_limits, 430  
 nc\_mass\_function\_sigma, 432  
 nc\_matter\_var\_clear, 399  
 nc\_matter\_var\_copy, 399  
 nc\_matter\_var\_dlnvar0\_dlnR, 400  
 nc\_matter\_var\_dlnvar0\_dR, 400  
 nc\_matter\_var\_dsigma0\_dR, 402  
 nc\_matter\_var\_free, 399  
 nc\_matter\_var\_integrand\_over\_window2, 401  
 nc\_matter\_var\_lnM\_to\_lnR, 401  
 nc\_matter\_var\_lnR\_to\_lnM, 401  
 nc\_matter\_var\_mass\_to\_R, 400  
 nc\_matter\_var\_new, 399  
 nc\_matter\_var\_prepare, 399  
 nc\_matter\_var\_R\_to\_mass, 401  
 nc\_matter\_var\_sigma8\_sqrtvar0, 403  
 nc\_matter\_var\_spectral\_moment\_over\_growth2, 402

nc\_matter\_var\_spectral\_moment\_over\_growth2\_gaussian, 402  
 nc\_matter\_var\_spectral\_moment\_over\_growth2\_tophat, 402  
 nc\_matter\_var\_var0, 400  
 nc\_multiplicity\_func\_clear, 405  
 nc\_multiplicity\_func\_eval, 404  
 nc\_multiplicity\_func\_free, 405  
 nc\_multiplicity\_func\_jenkins\_get\_A, 412  
 nc\_multiplicity\_func\_jenkins\_get\_A\_tCDM, 412  
 nc\_multiplicity\_func\_jenkins\_get\_B, 413  
 nc\_multiplicity\_func\_jenkins\_get\_B\_tCDM, 413  
 nc\_multiplicity\_func\_jenkins\_get\_epsilon, 413  
 nc\_multiplicity\_func\_jenkins\_get\_epsilon\_tCDM, 414  
 nc\_multiplicity\_func\_jenkins\_new, 411  
 nc\_multiplicity\_func\_jenkins\_set\_A, 412  
 nc\_multiplicity\_func\_jenkins\_set\_A\_tCDM, 412  
 nc\_multiplicity\_func\_jenkins\_set\_B, 412  
 nc\_multiplicity\_func\_jenkins\_set\_B\_tCDM, 413  
 nc\_multiplicity\_func\_jenkins\_set\_epsilon, 413  
 nc\_multiplicity\_func\_jenkins\_set\_epsilon\_tCDM, 414  
 nc\_multiplicity\_func\_new\_from\_name, 404  
 nc\_multiplicity\_func\_ps\_get\_delta\_c, 406  
 nc\_multiplicity\_func\_ps\_new, 406  
 nc\_multiplicity\_func\_ps\_set\_delta\_c, 406  
 nc\_multiplicity\_func\_st\_get\_A, 408  
 nc\_multiplicity\_func\_st\_get\_b, 408  
 nc\_multiplicity\_func\_st\_get\_delta\_c, 409  
 nc\_multiplicity\_func\_st\_get\_p, 409  
 nc\_multiplicity\_func\_st\_new, 408  
 nc\_multiplicity\_func\_st\_set\_A, 408  
 nc\_multiplicity\_func\_st\_set\_b, 408  
 nc\_multiplicity\_func\_st\_set\_delta\_c, 409  
 nc\_multiplicity\_func\_st\_set\_p, 409  
 nc\_multiplicity\_func\_tinker\_crit\_get\_Delta, 426  
 nc\_multiplicity\_func\_tinker\_crit\_new, 425  
 nc\_multiplicity\_func\_tinker\_crit\_set\_Delta, 426  
 nc\_multiplicity\_func\_tinker\_get\_A0, 420  
 nc\_multiplicity\_func\_tinker\_get\_a0, 421  
 nc\_multiplicity\_func\_tinker\_get\_b0, 421  
 nc\_multiplicity\_func\_tinker\_get\_c, 421  
 nc\_multiplicity\_func\_tinker\_get\_Delta, 422  
 nc\_multiplicity\_func\_tinker\_mean\_get\_Delta, 424  
 nc\_multiplicity\_func\_tinker\_mean\_new, 424  
 nc\_multiplicity\_func\_tinker\_mean\_set\_Delta, 424  
 nc\_multiplicity\_func\_tinker\_new, 420  
 nc\_multiplicity\_func\_tinker\_set\_A0, 420  
 nc\_multiplicity\_func\_tinker\_set\_a0, 420  
 nc\_multiplicity\_func\_tinker\_set\_b0, 421  
 nc\_multiplicity\_func\_tinker\_set\_c, 421  
 nc\_multiplicity\_func\_tinker\_set\_Delta, 422  
 nc\_multiplicity\_func\_warren\_get\_A, 416  
 nc\_multiplicity\_func\_warren\_get\_a, 417  
 nc\_multiplicity\_func\_warren\_get\_b, 417  
 nc\_multiplicity\_func\_warren\_get\_c, 418  
 nc\_multiplicity\_func\_warren\_new, 416  
 nc\_multiplicity\_func\_warren\_set\_A, 416  
 nc\_multiplicity\_func\_warren\_set\_a, 417  
 nc\_multiplicity\_func\_warren\_set\_b, 417

---

nc\_multiplicity\_func\_warren\_set\_c, 417  
 nc\_pert\_cov\_direct, 383  
 NC\_PERT\_dB0, 376  
 NC\_PERT\_dTHETA0, 376  
 nc\_pert\_get\_default\_los\_table, 382  
 nc\_pert\_linear\_calc\_Nc\_spline, 381  
 nc\_pert\_linear\_clear, 380  
 nc\_pert\_linear\_create\_los\_table, 382  
 nc\_pert\_linear\_free, 380  
 nc\_pert\_linear\_los\_integrate, 381  
 nc\_pert\_linear\_new, 379  
 nc\_pert\_linear\_prepare\_splines, 380  
 nc\_pert\_linear\_spline\_set\_source\_at, 381  
 nc\_pert\_linear\_splines\_clear, 381  
 nc\_pert\_linear\_splines\_free, 380  
 nc\_pert\_linear\_splines\_new, 380  
 NC\_PERT\_T, 376  
 NC\_PERT\_THETA, 377  
 NC\_PERT\_THETA\_P, 377  
 nc\_pert\_transfer\_function\_get, 383  
 nc\_pert\_transfer\_function\_new, 382  
 nc\_pert\_transfer\_function\_prepare, 382  
 NC\_PERT\_U, 377  
 NC\_PERT\_V, 376  
 NC\_PERTURBATION\_BASE\_SIZE, 376  
 NC\_PERTURBATION\_START\_X, 379  
 NC\_PERTURBATIONS\_LAMBDA2X, 377  
 NC\_PERTURBATIONS\_X2LAMBDA, 377  
 NC\_QUATERNION\_MEMCPY, 535  
 NC\_QUATERNION\_NEW, 534  
 NC\_QUATERNION\_NEW\_I, 534  
 NC\_QUATERNION\_NORM, 534  
 NC\_QUATERNION\_SET\_0, 534  
 NC\_QUATERNION\_SET\_I, 534  
 nc\_recomb\_d2tau\_dlambda2, 357  
 nc\_recomb\_d2v\_tau\_dlambda2, 359  
 nc\_recomb\_d3tau\_dlambda3, 358  
 nc\_recomb\_dtau\_dlambda, 357  
 nc\_recomb\_dtau\_dlambda\_Xe, 361  
 nc\_recomb\_dtau\_dx, 357  
 nc\_recomb\_dv\_tau\_dlambda, 359  
 nc\_recomb\_equilibrium\_Xe, 356  
 nc\_recomb\_free, 354  
 nc\_recomb\_He\_fully\_ionized\_dtau\_dlambda, 361  
 nc\_recomb\_He\_fully\_ionized\_Xe, 356  
 nc\_recomb\_HeI\_ion\_saha, 355  
 nc\_recomb\_HeII\_ion\_saha, 355  
 nc\_recomb\_HeII\_ion\_saha\_x, 356  
 nc\_recomb\_HeII\_ion\_saha\_x\_by\_HeIII\_He, 356  
 nc\_recomb\_HI\_ion\_saha, 355  
 nc\_recomb\_hummer\_HeI\_case\_B, 362  
 nc\_recomb\_hummer\_HeI\_case\_B\_dTm, 362  
 nc\_recomb\_log\_v\_tau, 359  
 nc\_recomb\_new\_from\_name, 354  
 nc\_recomb\_pequignot\_HI\_case\_B, 362  
 nc\_recomb\_pequignot\_HI\_case\_B\_dTm, 362  
 nc\_recomb\_prepare, 355  
 nc\_recomb\_prepare\_if\_needed, 355  
 nc\_recomb\_ref, 354  
 nc\_recomb\_seager\_new, 364  
 nc\_recomb\_seager\_new\_full, 364  
 nc\_recomb\_tau, 358  
 nc\_recomb\_tau\_cutoff, 361  
 nc\_recomb\_tau\_lambda0\_lambda1, 358  
 nc\_recomb\_tau\_zdrag, 361  
 nc\_recomb\_tau\_zstar, 360  
 nc\_recomb\_v\_tau, 359  
 nc\_recomb\_v\_tau\_lambda\_features, 360  
 nc\_recomb\_v\_tau\_lambda\_mode, 360  
 nc\_recomb\_weinberg\_HII\_ion\_rate, 363  
 nc\_recomb\_Xe, 357  
 nc\_scale\_factor\_a\_t, 373  
 nc\_scale\_factor\_copy, 372  
 nc\_scale\_factor\_free, 372  
 nc\_scale\_factor\_new, 372  
 nc\_scale\_factor\_prepare, 372  
 nc\_scale\_factor\_prepare\_if\_needed, 372  
 nc\_scale\_factor\_t\_x, 373  
 nc\_scale\_factor\_t\_z, 373  
 nc\_scale\_factor\_z\_t, 373  
 nc\_snia\_dist\_cov\_calc, 367  
 nc\_snia\_dist\_cov\_clear, 367  
 NC\_SNIA\_DIST\_COV\_DEFAULT\_ALPHA, 367  
 NC\_SNIA\_DIST\_COV\_DEFAULT\_BETA, 368  
 NC\_SNIA\_DIST\_COV\_DEFAULT\_M1, 368  
 NC\_SNIA\_DIST\_COV\_DEFAULT\_M2, 368  
 NC\_SNIA\_DIST\_COV\_DEFAULT\_PARAMS\_ABSTOL, 368  
 NC\_SNIA\_DIST\_COV\_DEFAULT\_SIGMA\_INT, 368  
 nc\_snia\_dist\_cov\_free, 366  
 nc\_snia\_dist\_cov\_mean, 367  
 nc\_snia\_dist\_cov\_new, 366  
 nc\_snia\_dist\_cov\_prepare, 367  
 nc\_snia\_dist\_cov\_prepare\_if\_needed, 367  
 nc\_snia\_dist\_cov\_ref, 366  
 NC\_SNIA\_DIST\_COV\_SIGMA\_INT, 368  
 NC\_SNIA\_DIST\_COV\_SIGMA\_INT\_DEFAULT\_LEN, 368  
 NC\_SNIA\_DIST\_COV\_VPARAM\_LEN, 368  
 nc\_transfer\_func\_bbks\_new, 392  
 nc\_transfer\_func\_camb\_new, 393  
 nc\_transfer\_func\_clear, 391  
 nc\_transfer\_func\_eh\_new, 392  
 nc\_transfer\_func\_eval, 390  
 nc\_transfer\_func\_free, 391  
 nc\_transfer\_func\_matter\_powerspectrum, 390  
 nc\_transfer\_func\_new\_from\_name, 390  
 nc\_transfer\_func\_prepare, 390  
 NC\_TRIVEC\_DOT, 534  
 NC\_TRIVEC\_MEMCPY, 533  
 NC\_TRIVEC\_NEW, 533  
 NC\_TRIVEC\_NORM, 533  
 NC\_TRIVEC\_NORMALIZE, 534  
 NC\_TRIVEC\_SCALE, 533  
 NC\_TRIVEC\_SET\_0, 533  
 nc\_window\_clear, 386

---

nc\_window\_deriv\_fourier, 386  
nc\_window\_eval\_fourier, 385  
nc\_window\_eval\_realspace, 386  
nc\_window\_free, 386  
nc\_window\_gaussian\_new, 388  
nc\_window\_new\_from\_name, 385  
nc\_window\_tophat\_new, 387  
nc\_window\_volume, 385  
NC\_WINDOW\_VOLUME\_GAUSSIAN, 388  
NC\_WINDOW\_VOLUME\_TOPHAT, 387  
NcClusterAbundance, 483  
NcClusterAbundance:mass, 490  
NcClusterAbundance:mass-function, 490  
NcClusterAbundance:mean-bias, 490  
NcClusterAbundance:redshift, 490  
NcClusterAbundanceClass, 483  
NcClusterAbundanceDataBin, 490  
NcClusterAbundanceDataBinM, 489  
NcClusterAbundanceDataBinZ, 482  
NcClusterAbundanceDataP, 490  
NcClusterAbundanceIntPd2N, 482  
NcClusterAbundanceN, 482  
NcClusterMass, 462  
NcClusterMassBenson, 473  
NcClusterMassBenson:Asz, 474  
NcClusterMassBenson:Asz-fit, 474  
NcClusterMassBenson:Bsz, 474  
NcClusterMassBenson:Bsz-fit, 474  
NcClusterMassBenson:Csz, 474  
NcClusterMassBenson:Csz-fit, 474  
NcClusterMassBenson:Dsz, 475  
NcClusterMassBenson:Dsz-fit, 475  
NcClusterMassBenson:M0, 475  
NcClusterMassBenson:signif-obs-max, 475  
NcClusterMassBenson:signif-obs-min, 475  
NcClusterMassBenson:z0, 475  
NcClusterMassBensonClass, 473  
NcClusterMassBensonParams, 472  
NcClusterMassBensonXRay, 477  
NcClusterMassBensonXRay:Ax, 477  
NcClusterMassBensonXRay:Ax-fit, 478  
NcClusterMassBensonXRay:Bx, 478  
NcClusterMassBensonXRay:Bx-fit, 478  
NcClusterMassBensonXRay:Cx, 478  
NcClusterMassBensonXRay:Cx-fit, 478  
NcClusterMassBensonXRay:Dx, 478  
NcClusterMassBensonXRay:Dx-fit, 478  
NcClusterMassBensonXRay:M0x, 479  
NcClusterMassBensonXRay:Y0, 479  
NcClusterMassBensonXRay:Yx-obs-max, 479  
NcClusterMassBensonXRay:Yx-obs-min, 479  
NcClusterMassBensonXRayClass, 477  
NcClusterMassBensonXRayParams, 476  
NcClusterMassClass, 462  
NcClusterMassImpl, 462  
NcClusterMassLnnormal, 467  
NcClusterMassLnnormal:lnMobs-max, 468  
NcClusterMassLnnormal:lnMobs-min, 468  
NcClusterMassLnnormalClass, 467  
NcClusterMassNodist, 466  
NcClusterMassNodist:lnM-max, 466  
NcClusterMassNodist:lnM-min, 466  
NcClusterMassNodistClass, 466  
NcClusterMassVanderlinde, 470  
NcClusterMassVanderlinde:Asz, 470  
NcClusterMassVanderlinde:Asz-fit, 470  
NcClusterMassVanderlinde:Bsz, 470  
NcClusterMassVanderlinde:Bsz-fit, 470  
NcClusterMassVanderlinde:Csz, 470  
NcClusterMassVanderlinde:Csz-fit, 470  
NcClusterMassVanderlinde:Dsz, 471  
NcClusterMassVanderlinde:Dsz-fit, 471  
NcClusterMassVanderlinde:M0, 471  
NcClusterMassVanderlinde:signif-obs-max, 471  
NcClusterMassVanderlinde:signif-obs-min, 471  
NcClusterMassVanderlinde:z0, 471  
NcClusterMassVanderlindeClass, 469  
NcClusterMassVanderlindeParams, 469  
NcClusterPhotozGauss, 457  
NcClusterPhotozGauss:pz-max, 457  
NcClusterPhotozGauss:pz-min, 458  
NcClusterPhotozGaussClass, 457  
NcClusterPhotozGaussGlobal, 459  
NcClusterPhotozGaussGlobal:pz-max, 460  
NcClusterPhotozGaussGlobal:pz-min, 460  
NcClusterPhotozGaussGlobal:sigma0, 460  
NcClusterPhotozGaussGlobal:z-bias, 460  
NcClusterPhotozGaussGlobalClass, 458  
NcClusterRedshift, 452  
NcClusterRedshiftClass, 452  
NcClusterRedshiftImpl, 452  
NcClusterRedshiftNodist, 456  
NcClusterRedshiftNodist:z-max, 456  
NcClusterRedshiftNodist:z-min, 456  
NcClusterRedshiftNodistClass, 456  
NcDataBaoA, 502  
NcDataBaoA:dist, 503  
NcDataBaoA:sample-id, 503  
NcDataBaoAClass, 502  
NcDataBaoDV, 504  
NcDataBaoDV:dist, 505  
NcDataBaoDV:sample-id, 505  
NcDataBaoDVClass, 504  
NcDataBaoDVDV, 508  
NcDataBaoDVDV:dist, 509  
NcDataBaoDVDV:sample-id, 509  
NcDataBaoDVDVClass, 508  
NcDataBaoId, 500  
NcDataBaoRDV, 506  
NcDataBaoRDV:dist, 507  
NcDataBaoRDV:sample-id, 508  
NcDataBaoRDVClass, 506  
NcDataClusterAbundanceId, 520  
NcDataClusterNCount, 520

- NcDataClusterNCount:cluster-abundance, 524
- NcDataClusterNCountClass, 520
- NcDataClusterPoisson, 525
- NcDataClusterPoissonClass, 524
- NcDataCMBDistPriors, 494
- NcDataCMBDistPriors:dist, 495
- NcDataCMBDistPriors:sample-id, 495
- NcDataCMBDistPriorsClass, 494
- NcDataCMBId, 491
- NcDataCMBShiftParam, 493
- NcDataCMBShiftParam:dist, 493
- NcDataCMBShiftParam:sample-id, 493
- NcDataCMBShiftParamClass, 492
- NcDataDistMu, 513
- NcDataDistMu:dist, 514
- NcDataDistMu:sample-id, 514
- NcDataDistMuClass, 512
- NcDataHubble, 496
- NcDataHubble:sample-id, 498
- NcDataHubbleBao, 499
- NcDataHubbleBao:dist, 500
- NcDataHubbleBao:sample-id, 500
- NcDataHubbleBaoClass, 499
- NcDataHubbleBaoId, 498
- NcDataHubbleClass, 496
- NcDataHubbleId, 496
- NcDataSNIACov, 516
- NcDataSNIACov:data-filename, 518
- NcDataSNIACov:sigma-pecz, 519
- NcDataSNIACovClass, 516
- NcDataSNIACovData, 515
- NcDataSNIId, 510
- NcDistance, 343
- NcDistance:zf, 351
- NcDistanceClass, 342
- NcDistanceFunc0, 342
- NcDistanceFunc1, 342
- NcGalaxyAcf, 450
- NcGalaxyAcfClass, 450
- NcGrowthFunc, 395
- NcGrowthFuncClass, 395
- NcHaloBiasFunc, 448
- NcHaloBiasFunc:bias-type, 449
- NcHaloBiasFunc:mass-function, 450
- NcHaloBiasFuncClass, 448
- NcHaloBiasType, 434
- NcHaloBiasTypeClass, 434
- NcHaloBiasTypePS, 436
- NcHaloBiasTypePS:critical-delta, 437
- NcHaloBiasTypePSClass, 436
- NcHaloBiasTypeSTEllip, 441
- NcHaloBiasTypeSTEllip:a, 443
- NcHaloBiasTypeSTEllip:b, 443
- NcHaloBiasTypeSTEllip:c, 443
- NcHaloBiasTypeSTEllip:critical-delta, 443
- NcHaloBiasTypeSTEllipClass, 440
- NcHaloBiasTypeSTSpher, 438
- NcHaloBiasTypeSTSpher:a, 439
- NcHaloBiasTypeSTSpher:critical-delta, 439
- NcHaloBiasTypeSTSpher:p, 439
- NcHaloBiasTypeSTSpherClass, 437
- NcHaloBiasTypeTinker, 444
- NcHaloBiasTypeTinker:B, 447
- NcHaloBiasTypeTinker:b, 447
- NcHaloBiasTypeTinker:c, 447
- NcHaloBiasTypeTinker:critical-delta, 447
- NcHaloBiasTypeTinker:Delta, 447
- NcHaloBiasTypeTinkerClass, 444
- NcHICosmo, 277
- NcHICosmoClass, 277
- NcHICosmoDE, 295
- NcHICosmoDE:H0, 296
- NcHICosmoDE:H0-fit, 296
- NcHICosmoDE:ns, 298
- NcHICosmoDE:ns-fit, 298
- NcHICosmoDE:Omegab, 297
- NcHICosmoDE:Omegab-fit, 297
- NcHICosmoDE:Omegac, 297
- NcHICosmoDE:Omegac-fit, 297
- NcHICosmoDE:Omegax, 297
- NcHICosmoDE:Omegax-fit, 297
- NcHICosmoDE:sigma8, 298
- NcHICosmoDE:sigma8-fit, 298
- NcHICosmoDE:Tgamma0, 298
- NcHICosmoDE:Tgamma0-fit, 298
- NcHICosmoDEClass, 295
- NcHICosmoDEImpl, 294
- NcHICosmoDELinder, 301
- NcHICosmoDELinder:w0, 302
- NcHICosmoDELinder:w0-fit, 302
- NcHICosmoDELinder:w1, 302
- NcHICosmoDELinder:w1-fit, 302
- NcHICosmoDELinderClass, 301
- NcHICosmoDELinderParams, 301
- NcHICosmoDEPad, 304
- NcHICosmoDEPad:w0, 304
- NcHICosmoDEPad:w0-fit, 304
- NcHICosmoDEPad:w1, 304
- NcHICosmoDEPad:w1-fit, 304
- NcHICosmoDEPadClass, 303
- NcHICosmoDEPadParams, 303
- NcHICosmoDEParams, 294
- NcHICosmoDEQe, 306
- NcHICosmoDEQe:w0, 306
- NcHICosmoDEQe:w0-fit, 306
- NcHICosmoDEQe:w1, 306
- NcHICosmoDEQe:w1-fit, 307
- NcHICosmoDEQeClass, 306
- NcHICosmoDEQEParams, 305
- NcHICosmoDEXcdm, 300
- NcHICosmoDEXcdm:w, 300
- NcHICosmoDEXcdm:w-fit, 300
- NcHICosmoDEXcdmClass, 299
- NcHICosmoDEXCDMParams, 299

NcHICosmoFunc0, 277  
NcHICosmoFunc1, 277  
NcHICosmoImpl, 276  
NcHICosmoLCDM, 290  
NcHICosmoLCDM:H0, 290  
NcHICosmoLCDM:H0-fit, 290  
NcHICosmoLCDM:ns, 292  
NcHICosmoLCDM:ns-fit, 292  
NcHICosmoLCDM:Omegab, 290  
NcHICosmoLCDM:Omegab-fit, 291  
NcHICosmoLCDM:Omegac, 291  
NcHICosmoLCDM:Omegac-fit, 291  
NcHICosmoLCDM:Omegax, 291  
NcHICosmoLCDM:Omegax-fit, 291  
NcHICosmoLCDM:sigma8, 292  
NcHICosmoLCDM:sigma8-fit, 292  
NcHICosmoLCDM:Tgamma0, 291  
NcHICosmoLCDM:Tgamma0-fit, 292  
NcHICosmoLCDMClass, 290  
NcHICosmoPriorTop, 288  
NcHICosmoQConst, 319  
NcHICosmoQConst:cd, 320  
NcHICosmoQConst:cd-fit, 320  
NcHICosmoQConst:E, 319  
NcHICosmoQConst:E-fit, 319  
NcHICosmoQConst:H0, 319  
NcHICosmoQConst:H0-fit, 319  
NcHICosmoQConst:Omegat, 320  
NcHICosmoQConst:Omegat-fit, 320  
NcHICosmoQConst:q, 320  
NcHICosmoQConst:q-fit, 320  
NcHICosmoQConst:zs, 321  
NcHICosmoQConst:zs-fit, 321  
NcHICosmoQConstClass, 319  
NcHICosmoQConstParams, 318  
NcHICosmoQGMode, 314  
NcHICosmoQGPerType, 314  
NcHICosmoQLinear, 323  
NcHICosmoQLinear:cd, 325  
NcHICosmoQLinear:cd-fit, 325  
NcHICosmoQLinear:E, 324  
NcHICosmoQLinear:E-fit, 324  
NcHICosmoQLinear:H0, 324  
NcHICosmoQLinear:H0-fit, 324  
NcHICosmoQLinear:Omegat, 324  
NcHICosmoQLinear:Omegat-fit, 324  
NcHICosmoQLinear:q, 325  
NcHICosmoQLinear:q-fit, 325  
NcHICosmoQLinear:qp, 325  
NcHICosmoQLinear:qp-fit, 325  
NcHICosmoQLinear:zs, 326  
NcHICosmoQLinear:zs-fit, 326  
NcHICosmoQLinearClass, 323  
NcHICosmoQLinearParams, 322  
NcHICosmoQPW, 328  
NcHICosmoQPW:flat, 331  
NcHICosmoQPW:H0, 330  
NcHICosmoQPW:H0-fit, 330  
NcHICosmoQPW:Omegat, 330  
NcHICosmoQPW:Omegat-fit, 330  
NcHICosmoQPW:q0, 331  
NcHICosmoQPW:q0-fit, 331  
NcHICosmoQPW:qp, 331  
NcHICosmoQPW:qp-fit, 331  
NcHICosmoQPW:qp-length, 331  
NcHICosmoQPW:zf, 332  
NcHICosmoQPWAsymCDMPrior, 328  
NcHICosmoQPWClass, 328  
NcHICosmoQPWContPrior, 328  
NcHICosmoQPWSPParams, 327  
NcHICosmoQPWVParams, 327  
NcHICosmoQSpline, 334  
NcHICosmoQSpline:asdrag, 338  
NcHICosmoQSpline:asdrag-fit, 339  
NcHICosmoQSpline:H0, 338  
NcHICosmoQSpline:H0-fit, 338  
NcHICosmoQSpline:Omegat, 338  
NcHICosmoQSpline:Omegat-fit, 338  
NcHICosmoQSpline:qparam, 339  
NcHICosmoQSpline:qparam-fit, 339  
NcHICosmoQSpline:qparam-length, 339  
NcHICosmoQSpline:spline, 339  
NcHICosmoQSpline:zf, 339  
NcHICosmoQSplineClass, 334  
NcHICosmoQSplineContPrior, 336  
NcHICosmoQSplineContPrior:nknots, 339  
NcHICosmoQSplineContPriorClass, 336  
NcHICosmoQSplineSPParams, 333  
NcHICosmoQSplineVParams, 334  
NcLinearPert, 377  
NcLinearPertConf, 377  
NcLinearPertCreate, 377  
NcLinearPertEvol, 377  
NcLinearPertGet, 378  
NcLinearPertGetN, 378  
NcLinearPertOdeSolver, 378  
NcLinearPertSources, 378  
NcLinearPertSplines, 379  
NcLinearPertSplineTypes, 378  
NcLinearPertTest, 377  
NcLinearPertTF, 379  
NcLinearPertVars, 375  
NcRecomb, 354  
NcRecomb:init-frac, 363  
NcRecomb:prec, 363  
NcRecomb:zi, 363  
NcRecombClass, 354  
NcRecombSeager, 364  
NcRecombSeagerClass, 364  
NcScaleFactor, 371  
NcScaleFactorTimeType, 371  
NcSNIADistCov, 366  
NcSNIADistCov:alpha, 369  
NcSNIADistCov:alpha-fit, 369



NcSNIADistCov:beta, 369  
 NcSNIADistCov:beta-fit, 370  
 NcSNIADistCov:dist, 370  
 NcSNIADistCov:M1, 368  
 NcSNIADistCov:M1-fit, 369  
 NcSNIADistCov:M2, 369  
 NcSNIADistCov:M2-fit, 369  
 NcSNIADistCov:sigma-int, 370  
 NcSNIADistCov:sigma-int-fit, 370  
 NcSNIADistCov:sigma-int-length, 370  
 NcSNIADistCovClass, 366  
 NcSNIADistCovParams, 366  
 NcTransferFunc, 389  
 NcTransferFuncBBKS, 391  
 NcTransferFuncBBKSClass, 391  
 NcTransferFuncCAMB, 393  
 NcTransferFuncCAMBClass, 393  
 NcTransferFuncClass, 389  
 NcTransferFuncEH, 392  
 NcTransferFuncEHClass, 392  
 NcTransferFuncPert, 394  
 NcTransferFuncPertClass, 394  
 NcWindow, 385  
 NcWindowClass, 385  
 NcWindowGaussian, 388  
 NcWindowGaussianClass, 388  
 NcWindowTophat, 387  
 NcWindowTophatClass, 387  
 ncm\_numdiff\_1, 544  
 ncm\_numdiff\_2, 545  
 ncm\_numdiff\_2\_err, 545

## O

ncm\_ode\_spline\_clear, 147  
 ncm\_ode\_spline\_free, 147  
 ncm\_ode\_spline\_new, 147  
 ncm\_ode\_spline\_prepare, 147  
 NcmOdeSpline, 147  
 NcmOdeSplineDydx, 146

## P

NcmParamType, 30  
 ncm\_poly\_eval, 531  
 ncm\_poly\_eval\_diff, 531  
 ncm\_poly\_eval\_int, 531  
 ncm\_poly\_eval\_int\_P\_over\_x, 532  
 ncm\_poly\_new, 531  
 ncm\_prior\_add\_gaussian, 213  
 ncm\_prior\_add\_gaussian\_const\_func, 214  
 ncm\_prior\_add\_gaussian\_data, 214  
 ncm\_prior\_add\_gaussian\_func, 214  
 ncm\_prior\_add\_oneside\_a\_inf, 213  
 ncm\_prior\_add\_oneside\_a\_inf\_const\_func, 212  
 ncm\_prior\_add\_oneside\_a\_inf\_func, 212  
 ncm\_prior\_add\_oneside\_a\_inf\_param, 212  
 ncm\_prior\_add\_positive, 213  
 ncm\_prior\_add\_twoside\_a\_b, 213

NcmPriorGauss, 211

## Q

NcmQ, 534  
 NcmQuadFilon, 555  
 NcmQuadFilonError, 555  
 ncm\_quadrature\_filon\_calc\_inter\_point, 556  
 ncm\_quadrature\_filon\_calc\_mu\_dxn, 556  
 ncm\_quadrature\_filon\_calc\_vandermonde, 556  
 ncm\_quadrature\_filon\_eval, 557  
 ncm\_quadrature\_filon\_new, 555  
 ncm\_quadrature\_filon\_solve\_vandermonde, 556  
 ncm\_quaternion\_conjugate, 536  
 ncm\_quaternion\_conjugate\_q\_mul, 538  
 ncm\_quaternion\_conjugate\_u\_mul, 537  
 ncm\_quaternion\_free, 536  
 ncm\_quaternion\_inv\_rotate, 538  
 ncm\_quaternion\_lmul, 537  
 ncm\_quaternion\_mul, 537  
 ncm\_quaternion\_new, 535  
 ncm\_quaternion\_new\_from\_data, 535  
 ncm\_quaternion\_new\_from\_vector, 535  
 ncm\_quaternion\_normalize, 536  
 ncm\_quaternion\_rmul, 537  
 ncm\_quaternion\_rotate, 538  
 ncm\_quaternion\_set\_from\_data, 536  
 ncm\_quaternion\_set\_random, 536

## R

r1, 140  
 r12, 140  
 r2, 140  
 ncm\_random\_seed, 540  
 ncm\_rational\_coarce\_double, 542  
 NCM\_READ\_DOUBLE, 112  
 NCM\_READ\_INT32, 112  
 NCM\_READ\_INT64, 112  
 NCM\_READ\_UINT32, 112  
 NCM\_READ\_UINT64, 112  
 NcmReparam, 45  
 NcmReparam:length, 47  
 ncm\_reparam\_clear, 47  
 ncm\_reparam\_copy, 45  
 ncm\_reparam\_copyto, 45  
 ncm\_reparam\_free, 47  
 ncm\_reparam\_grad\_old2new, 46  
 ncm\_reparam\_jac, 46  
 ncm\_reparam\_linear\_new, 48  
 ncm\_reparam\_M\_old2new, 47  
 ncm\_reparam\_new2old, 46  
 ncm\_reparam\_old2new, 46  
 ncm\_reparam\_ref, 45  
 NcmReparamClass, 45  
 NcmReparamJ, 45  
 NcmReparamLinear, 48  
 NcmReparamLinear:matrix, 48  
 NcmReparamLinear:vector, 49

NcmReparamLinearClass, 48  
NcmReparamV, 44  
NCM\_RETURN\_IF\_INF, 108  
NCM\_ROUND\_TRUNC, 109

## S

ncm\_sf\_0F1, 163  
ncm\_sf\_sbessel, 177  
ncm\_sf\_sbessel\_deriv, 178  
ncm\_sf\_sbessel\_jl\_xj\_integral, 180  
ncm\_sf\_sbessel\_jl\_xj\_integral\_a\_b, 185  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_cached\_new, 181  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_free, 182  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_goto, 182  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_load, 181  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_new, 181  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_new\_from\_section, 181  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_next, 182  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_previous, 182  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_read, 181  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_save, 183  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_set, 182  
ncm\_sf\_sbessel\_jl\_xj\_integral\_recur\_write, 183  
ncm\_sf\_sbessel\_jl\_xj\_integral\_spline, 185  
ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_cached\_new, 183  
ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_eval, 185  
ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_goto, 184  
ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_new, 183  
ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_next, 184  
ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_previous, 184  
ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_reset, 184  
ncm\_sf\_sbessel\_jl\_xj\_integrate\_spline\_set, 184  
ncm\_sf\_sbessel\_recur\_free, 176  
ncm\_sf\_sbessel\_recur\_goto, 177  
ncm\_sf\_sbessel\_recur\_new, 175  
ncm\_sf\_sbessel\_recur\_next, 176  
ncm\_sf\_sbessel\_recur\_previous, 176  
ncm\_sf\_sbessel\_recur\_read, 176  
ncm\_sf\_sbessel\_recur\_set, 176  
ncm\_sf\_sbessel\_recur\_write, 177  
ncm\_sf\_sbessel\_spline, 178  
ncm\_sf\_sbessel\_taylor, 178  
ncm\_sf\_sbessel\_taylor\_coeff\_jl\_jlp1, 177  
NcmSFSBesselRecur, 175  
NcmSFSphericalBesselIntegRecur, 180  
NcmSFSphericalBesselIntSpline, 180  
ncm\_smoothd, 540  
NcmSParam, 30  
NcmSParam:absolute-tolerance, 34  
NcmSParam:default-value, 34  
NcmSParam:fit-type, 34  
NcmSParam:lower-bound, 35  
NcmSParam:name, 35  
NcmSParam:scale, 35  
NcmSParam:symbol, 35  
NcmSParam:upper-bound, 35  
ncm\_sparam\_clear, 31  
ncm\_sparam\_copy, 30  
ncm\_sparam\_free, 31  
ncm\_sparam\_get\_absolute\_tolerance, 34  
ncm\_sparam\_get\_default\_value, 34  
ncm\_sparam\_get\_fit\_type, 34  
ncm\_sparam\_get\_lower\_bound, 33  
ncm\_sparam\_get\_scale, 33  
ncm\_sparam\_get\_upper\_bound, 33  
ncm\_sparam\_name, 33  
ncm\_sparam\_new, 30  
ncm\_sparam\_ref, 31  
ncm\_sparam\_set\_absolute\_tolerance, 32  
ncm\_sparam\_set\_default\_value, 32  
ncm\_sparam\_set\_fit\_type, 32  
ncm\_sparam\_set\_lower\_bound, 31  
ncm\_sparam\_set\_scale, 32  
ncm\_sparam\_set\_upper\_bound, 31  
ncm\_sparam\_symbol, 33  
ncm\_sparam\_take\_name, 32  
ncm\_sparam\_take\_symbol, 33  
NcmSParamClass, 29  
ncm\_sphere\_healpix\_nest2ring, 265  
ncm\_sphere\_healpix\_pix2ang\_nest, 266  
ncm\_sphere\_healpix\_pix2ang\_ring, 266  
ncm\_sphere\_healpix\_pix2vec\_nest, 266  
ncm\_sphere\_healpix\_pix2vec\_ring, 267  
ncm\_sphere\_healpix\_read\_map, 265  
ncm\_sphere\_healpix\_ring2nest, 266  
ncm\_sphere\_healpix\_vec2pix\_ring, 267  
ncm\_sphere\_healpix\_write\_map, 265  
ncm\_sphere\_map\_clone, 270  
ncm\_sphere\_map\_copy, 270  
ncm\_sphere\_map\_homogenize\_noise, 272  
ncm\_sphere\_map\_init\_coord, 270  
ncm\_sphere\_map\_new, 269  
ncm\_sphere\_map\_rotate\_avg, 273  
ncm\_sphere\_map\_set\_order, 270  
ncm\_sphere\_mapalm\_init, 271  
ncm\_sphere\_mapalm\_new, 270  
ncm\_sphere\_mapsht\_alm2map, 272  
ncm\_sphere\_mapsht\_alm2map\_circle, 272  
ncm\_sphere\_mapsht\_map2alm, 272  
ncm\_sphere\_mapsht\_map2alm\_circle, 271  
ncm\_sphere\_mapsht\_new, 271  
NcmSphereMap, 269  
NcmSphereMapAlm, 269  
NcmSphereMapOrder, 268  
NcmSphereMapSHT, 269  
NcmSphereMapType, 269  
ncm\_sphPlm\_test\_theta, 542  
ncm\_sphPlm\_x, 542  
NcmSpline, 134  
NcmSpline2d, 149  
NcmSpline2d:spline, 155  
NCM\_SPLINE2D\_BICUBIC\_00, 158  
NCM\_SPLINE2D\_BICUBIC\_01, 158  
NCM\_SPLINE2D\_BICUBIC\_10, 158

- NCM\_SPLINE2D\_BICUBIC\_11, 158  
 ncm\_spline2d\_bicubic\_bi, 160  
 ncm\_spline2d\_bicubic\_bi\_bip1, 160  
 NCM\_SPLINE2D\_BICUBIC\_COEFF, 159  
 NCM\_SPLINE2D\_BICUBIC\_COEFF\_INDEX, 159  
 ncm\_spline2d\_bicubic\_eval\_poly, 159  
 NCM\_SPLINE2D\_BICUBIC\_F, 159  
 ncm\_spline2d\_bicubic\_fij\_to\_aij, 159  
 NCM\_SPLINE2D\_BICUBIC\_FX, 159  
 NCM\_SPLINE2D\_BICUBIC\_FXY, 159  
 NCM\_SPLINE2D\_BICUBIC\_FY, 159  
 ncm\_spline2d\_bicubic\_integ\_dx\_coeffs, 160  
 ncm\_spline2d\_bicubic\_integ\_dy\_coeffs, 160  
 ncm\_spline2d\_bicubic\_integ\_eval2d, 160  
 ncm\_spline2d\_bicubic\_new, 158  
 ncm\_spline2d\_bicubic\_notaknot\_new, 158  
 NCM\_SPLINE2D\_BICUBIC\_STRUCT, 159  
 ncm\_spline2d\_clear, 152  
 ncm\_spline2d\_copy, 151  
 ncm\_spline2d\_copy\_empty, 151  
 ncm\_spline2d\_eval, 152  
 ncm\_spline2d\_free, 151  
 ncm\_spline2d\_gsl\_natural\_new, 161  
 ncm\_spline2d\_gsl\_new, 161  
 ncm\_spline2d\_integ\_dx, 152  
 ncm\_spline2d\_integ\_dx\_spline, 153  
 ncm\_spline2d\_integ\_dx\_spline\_val, 154  
 ncm\_spline2d\_integ\_dxdy, 153  
 ncm\_spline2d\_integ\_dxdy\_spline\_x, 154  
 ncm\_spline2d\_integ\_dxdy\_spline\_y, 155  
 ncm\_spline2d\_integ\_dy, 152  
 ncm\_spline2d\_integ\_dy\_spline, 153  
 ncm\_spline2d\_integ\_dy\_spline\_val, 154  
 ncm\_spline2d\_min\_size, 151  
 ncm\_spline2d\_new, 151  
 ncm\_spline2d\_prepare, 150  
 ncm\_spline2d\_set, 150  
 ncm\_spline2d\_set\_function, 150  
 ncm\_spline2d\_spline\_new, 156  
 NcmSpline2dBicubic, 158  
 NcmSpline2dBicubicClass, 158  
 NcmSpline2dBicubicCoeffs, 157  
 NcmSpline2dClass, 149  
 NcmSpline2dGsl, 161  
 NcmSpline2dGslClass, 161  
 ncm\_spline2dim\_integ\_total, 155  
 NcmSpline2dSpline, 156  
 NcmSpline2dSplineClass, 156  
 ncm\_spline\_clear, 138  
 ncm\_spline\_copy, 135  
 ncm\_spline\_copy\_empty, 134  
 ncm\_spline\_cubic\_notaknot\_new, 144  
 ncm\_spline\_cubic\_notaknot\_new\_full, 144  
 ncm\_spline\_eval, 138  
 ncm\_spline\_eval\_deriv, 139  
 ncm\_spline\_eval\_deriv2, 139  
 ncm\_spline\_eval\_deriv\_nmax, 139  
 ncm\_spline\_eval\_integ, 139  
 ncm\_spline\_free, 138  
 NCM\_SPLINE\_FUNC\_DEFAULT\_MAX\_NODES, 146  
 ncm\_spline\_get\_index, 140  
 ncm\_spline\_get\_xv, 137  
 ncm\_spline\_get\_yv, 138  
 ncm\_spline\_gsl\_new, 142  
 ncm\_spline\_gsl\_new\_full, 142  
 ncm\_spline\_gsl\_set\_type, 142  
 ncm\_spline\_gsl\_set\_type\_by\_id, 142  
 ncm\_spline\_is\_empty, 139  
 NCM\_SPLINE\_KNOT\_DIFF\_TOL, 146  
 ncm\_spline\_min\_size, 140  
 ncm\_spline\_new, 135  
 ncm\_spline\_new\_array, 135  
 ncm\_spline\_new\_data, 135  
 ncm\_spline\_prepare, 138  
 ncm\_spline\_prepare\_base, 138  
 ncm\_spline\_ref, 136  
 ncm\_spline\_set, 136  
 ncm\_spline\_set\_array, 137  
 ncm\_spline\_set\_data\_static, 137  
 ncm\_spline\_set\_func, 145  
 ncm\_spline\_set\_xv, 136  
 ncm\_spline\_set\_yv, 137  
 NcmSplineClass, 134  
 NcmSplineCubic, 143  
 NcmSplineCubicClass, 143  
 NcmSplineCubicNotaknot, 144  
 NcmSplineCubicNotaknotClass, 144  
 NcmSplineFuncType, 145  
 NcmSplineGsl, 142  
 NcmSplineGsl:type, 143  
 NcmSplineGsl:type-name, 143  
 NcmSplineGslClass, 142  
 NcmSplineGslType, 141  
 ncm\_sqrt1px\_m1, 545  
 ncm\_string\_ww, 102  
 ncm\_sum, 544
- ## T
- NCM\_TEST\_GSL\_RESULT, 109  
 NCM\_THREAD\_POOL\_MAX, 109  
 ncm\_topology\_comoving\_a0\_lss, 541  
 ncm\_topology\_sigma\_comoving\_a0\_lss, 542  
 NcmTriVector, 533
- ## U
- NeMultiplicityFunc, 404  
 NeMultiplicityFuncClass, 404  
 NeMultiplicityFuncJenkins, 411  
 NeMultiplicityFuncJenkins:A, 414  
 NeMultiplicityFuncJenkins:A-tCDM, 414  
 NeMultiplicityFuncJenkins:B, 414  
 NeMultiplicityFuncJenkins:B-tCDM, 414  
 NeMultiplicityFuncJenkins:epsilon, 415  
 NeMultiplicityFuncJenkins:epsilon-tCDM, 415



NcMultiplicityFuncJenkinsClass, 411  
 NcMultiplicityFuncPS, 406  
 NcMultiplicityFuncPS:critical-delta, 406  
 NcMultiplicityFuncPSClass, 405  
 NcMultiplicityFuncST, 407  
 NcMultiplicityFuncST:A, 409  
 NcMultiplicityFuncST:b, 409  
 NcMultiplicityFuncST:critical-delta, 410  
 NcMultiplicityFuncST:p, 410  
 NcMultiplicityFuncSTClass, 407  
 NcMultiplicityFuncTinker, 420  
 NcMultiplicityFuncTinker:A0, 422  
 NcMultiplicityFuncTinker:a0, 422  
 NcMultiplicityFuncTinker:b0, 422  
 NcMultiplicityFuncTinker:c, 423  
 NcMultiplicityFuncTinker:Delta, 422  
 NcMultiplicityFuncTinkerClass, 419  
 NcMultiplicityFuncTinkerCrit, 425  
 NcMultiplicityFuncTinkerCrit:Delta, 426  
 NcMultiplicityFuncTinkerCritClass, 425  
 NcMultiplicityFuncTinkerMean, 423  
 NcMultiplicityFuncTinkerMean:Delta, 424  
 NcMultiplicityFuncTinkerMeanClass, 423  
 NcMultiplicityFuncWarren, 416  
 NcMultiplicityFuncWarren:A, 418  
 NcMultiplicityFuncWarren:a, 418  
 NcMultiplicityFuncWarren:b, 418  
 NcMultiplicityFuncWarren:c, 418  
 NcMultiplicityFuncWarrenClass, 416

## V

NcmVector, 7  
 NcmVector:values, 16  
 ncm\_vector\_add, 13  
 ncm\_vector\_addto, 12  
 ncm\_vector\_clear, 16  
 ncm\_vector\_const\_free, 16  
 ncm\_vector\_const\_gsl, 15  
 NCM\_VECTOR\_DATA, 7  
 ncm\_vector\_div, 13  
 ncm\_vector\_dup, 15  
 ncm\_vector\_dup\_array, 14  
 ncm\_vector\_fast\_get, 11  
 ncm\_vector\_fast\_set, 11  
 ncm\_vector\_fast\_subfrom, 12  
 ncm\_vector\_free, 16  
 ncm\_vector\_get, 11  
 ncm\_vector\_get\_array, 14  
 ncm\_vector\_get\_subvector, 10  
 ncm\_vector\_get\_variant, 10  
 ncm\_vector\_gsl, 15  
 ncm\_vector\_len, 15  
 ncm\_vector\_memcpy, 14  
 ncm\_vector\_memcpy2, 14  
 ncm\_vector\_new, 7  
 ncm\_vector\_new\_array, 8  
 ncm\_vector\_new\_data\_const, 10

ncm\_vector\_new\_data\_malloc, 9  
 ncm\_vector\_new\_data\_slice, 8  
 ncm\_vector\_new\_data\_static, 9  
 ncm\_vector\_new\_gsl, 8  
 ncm\_vector\_new\_gsl\_const, 10  
 ncm\_vector\_new\_gsl\_static, 8  
 ncm\_vector\_new\_variant, 9  
 ncm\_vector\_nvector, 16  
 ncm\_vector\_ptr, 11  
 ncm\_vector\_ref, 9  
 ncm\_vector\_scale, 13  
 ncm\_vector\_set, 11  
 ncm\_vector\_set\_all, 12  
 ncm\_vector\_set\_zero, 13  
 ncm\_vector\_stride, 15  
 ncm\_vector\_sub, 13  
 ncm\_vector\_subfrom, 12  
 NcmVectorClass, 7  
 NcmVectorInternal, 7  
 NcmVParam, 37  
 NcmVParam:default-sparam, 43  
 ncm\_vparam\_clear, 39  
 ncm\_vparam\_copy, 38  
 ncm\_vparam\_free, 38  
 ncm\_vparam\_full\_new, 38  
 ncm\_vparam\_get\_absolute\_tolerance, 42  
 ncm\_vparam\_get\_default\_value, 43  
 ncm\_vparam\_get\_fit\_type, 43  
 ncm\_vparam\_get\_len, 39  
 ncm\_vparam\_get\_lower\_bound, 42  
 ncm\_vparam\_get\_scale, 42  
 ncm\_vparam\_get\_sparam, 40  
 ncm\_vparam\_get\_upper\_bound, 42  
 ncm\_vparam\_new, 37  
 ncm\_vparam\_peek\_sparam, 40  
 ncm\_vparam\_set\_absolute\_tolerance, 41  
 ncm\_vparam\_set\_default\_value, 41  
 ncm\_vparam\_set\_fit\_type, 42  
 ncm\_vparam\_set\_len, 39  
 ncm\_vparam\_set\_lower\_bound, 40  
 ncm\_vparam\_set\_scale, 41  
 ncm\_vparam\_set\_sparam, 39  
 ncm\_vparam\_set\_sparam\_full, 39  
 ncm\_vparam\_set\_upper\_bound, 41  
 NcmVParamClass, 37

## W

NCM\_WRITE\_DOUBLE, 112  
 NCM\_WRITE\_INT32, 111  
 NCM\_WRITE\_INT64, 111  
 NCM\_WRITE\_UINT32, 111  
 NCM\_WRITE\_UINT64, 112

## Z

NCM\_ZERO\_LIMIT, 109