ltbToolkit-1.0

Generated by Doxygen 1.8.5

Tue Sep 10 2013 17:59:34

Contents

1 Data Structure Index **1**

1.1 Data Structures 1

2 File Index **3**

2.1 File List 3

3 Data Structure Documentation **5**

3.1 Interp_data Struct Reference 5

3.1.1 Detailed Description 5

3.1.2 Field Documentation 5

3.1.2.1 acc 5

3.1.2.2 cur 5

3.1.2.3 max 6

3.1.2.4 min 6

3.1.2.5 spline 6

3.1.2.6 x 6

3.1.2.7 y 6

3.2 Ltb_model Struct Reference 6

3.2.1 Detailed Description 7

3.2.2 Field Documentation 7

3.2.2.1 E_fun 7

3.2.2.2 E_vec 7

3.2.2.3 M_fun 7

3.2.2.4 M_vec 7

3.2.2.5 R_mat 7

3.2.2.6 r_size 7

3.2.2.7 r_vec 7

3.2.2.8 t_size 7

3.2.2.9 t_vec 8

3.2.2.10 tB_fun 8

3.2.2.11 tB_max 8

3.2.2.12 tB_vec 8

4 File Documentation **9**

4.1 src/gsl_vector_help_functions.h File Reference . . . 9

4.1.1 Detailed Description . . . 10

4.2 src/interp_data.h File Reference . . . 10

4.2.1 Detailed Description . . . 11

4.2.2 Function Documentation . . . 11

4.2.2.1 check_interp_data . . . 11

4.2.2.2 interp_data_alloc . . . 11

4.2.2.3 interp_data_free . . . 12

4.2.2.4 set_eta_max . . . 12

4.2.2.5 set_eta_min . . . 12

4.2.3 Variable Documentation . . . 12

4.2.3.1 dxi . . . 12

4.3 src/ltb.h File Reference . . . 13

4.3.1 Detailed Description . . . 15

4.3.2 Function Documentation . . . 15

4.3.2.1 cur_fun . . . 15

4.3.2.2 eta_fun . . . 15

4.3.2.3 exp_fun . . . 15

4.3.2.4 fi . . . 16

4.3.2.5 fiFlat . . . 16

4.3.2.6 fiHyperbolic . . . 16

4.3.2.7 fiSpherical . . . 17

4.3.2.8 get_chi . . . 17

4.3.2.9 get_min_max_for_interp . . . 17

4.3.2.10 get_xi . . . 17

4.3.2.11 grr_sq_fun . . . 18

4.3.2.12 interp_data_alloc_for_ltb . . . 18

4.3.2.13 ltb_model_alloc . . . 18

4.3.2.14 ltb_model_alloc_from_vec . . . 19

4.3.2.15 ltb_model_cpy . . . 19

4.3.2.16 ltb_model_exp . . . 20

4.3.2.17 ltb_model_free . . . 20

4.3.2.18 ltb_model_grr_sq . . . 20

4.3.2.19 ltb_model_R_r . . . 21

4.3.2.20 ltb_model_R_rt . . . 21

4.3.2.21 ltb_model_R_t . . . 22

4.3.2.22 ltb_model_rho . . . 22

4.3.2.23 ltb_model_Ricci . . . 23

4.3.2.24 ltb_model_set_fun . . . 23

4.3.2.25 ltb_model_set_R . . . 23
4.3.2.26 ltb_model_write_fun . . . 24
4.3.2.27 R_fun . . . 24
4.3.2.28 R_r_analytic_fun . . . 25
4.3.2.29 R_rt_analytic_fun . . . 26
4.3.2.30 R_t_analytic_fun . . . 26
4.3.2.31 rho_fun . . . 27
4.3.2.32 Ricci_fun . . . 27
4.3.2.33 set_cur . . . 27
4.3.2.34 xi_fun . . . 28
4.3.2.35 xi_fun_from_eta . . . 28
4.3.2.36 xiFlat . . . 28
4.3.2.37 xiHyperbolic . . . 29
4.3.2.38 xiSpherical . . . 29
4.4 src/ltb_error.h File Reference . . . 29
4.4.1 Detailed Description . . . 29
4.4.2 Macro Definition Documentation . . . 30
4.4.2.1 CHECK_M_DIM . . . 30
4.4.2.2 CHECK_PTR . . . 30
4.4.2.3 CHECK_SIZE . . . 30
4.4.2.4 CHECK_STAT . . . 30
4.4.3 Enumeration Type Documentation . . . 30
4.4.3.1 anonymous enum . . . 30
4.5 src/ltb_integrate.h File Reference . . . 31
4.5.1 Detailed Description . . . 31
4.5.2 Function Documentation . . . 31
4.5.2.1 gsl_matrix_integrate_trap_range . . . 31
4.6 src/macros.h File Reference . . . 32
4.6.1 Detailed Description . . . 32
4.7 src/phys_constant.h File Reference . . . 32
4.7.1 Detailed Description . . . 33
4.7.2 Function Documentation . . . 33
4.7.2.1 export_phys_const . . . 33
4.7.2.2 set_phys_const . . . 33
4.7.3 Variable Documentation . . . 33
4.7.3.1 c . . . 33
4.7.3.2 G_grav . . . 33
4.7.3.3 H_0 . . . 33
4.7.3.4 pi . . . 34

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Interp_data 5
Ltb_model 6

Chapter 2

File Index

2.1 File List

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

src/**gsl_vector_help_functions.c** . . . . . **??**
src/gsl_vector_help_functions.h
Custom extensions of gsl_vector, gsl_matrix from GSL library 9
src/**interp_data.c** . . . . . **??**
src/interp_data.h
Structure and functions needed for interpolation *eta(xi)* 10
src/**ltb.c** . . . . . **??**
src/ltb.h
Ltb model function 13
src/ltb_error.h
Errors Handling macros and functions 29
src/**ltb_integrate.c** . . . . . **??**
src/ltb_integrate.h
Functions for integrating 31
src/macros.h
Additional helpful macros 32
src/**phys_constant.c** . . . . . **??**
src/phys_constant.h
Physics constant 32

Chapter 3

Data Structure Documentation

3.1 Interp_data Struct Reference

`#include <src/interp_data.h>`

Data Fields

- int **n**
- int cur
- double * x
- double * y
- double max
- double min
- gsl_interp_accel * acc
- gsl_interp * spline

3.1.1 Detailed Description

Interp_data typdef stucture

Used in inverting xi function to get eta

Definition at line 14 of file interp_data.h.

3.1.2 Field Documentation

3.1.2.1 gsl_interp_accel* Interp_data::acc

minimum value op points to initialization

Definition at line 22 of file interp_data.h.

3.1.2.2 int Interp_data::cur

Numbers of point to initialize spline procedure

Definition at line 17 of file interp_data.h.

3.1.2.3 double Interp_data::max

values of function in point x for spline initialization

Definition at line 20 of file interp_data.h.

3.1.2.4 double Interp_data::min

maximum value of points to initialization

Definition at line 21 of file interp_data.h.

3.1.2.5 gsl_interp* Interp_data::spline

LUT (look up table) for accelerating of interpolation

Definition at line 23 of file interp_data.h.

3.1.2.6 double* Interp_data::x

curvature of model for which we want interpolation

Definition at line 18 of file interp_data.h.

3.1.2.7 double* Interp_data::y

x points for spline initialization

Definition at line 19 of file interp_data.h.

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

- src/interp_data.h

3.2 Ltb_model Struct Reference

`#include <src/ltb.h>`

Data Fields

- int **cur**
- size_t r_size
- size_t t_size
- gsl_vector * r_vec
- gsl_vector * t_vec
- gsl_vector * tB_vec
- gsl_vector * E_vec
- gsl_vector * M_vec
- gsl_matrix * R_mat
- double(* tB_fun)(double)
- double(* E_fun)(double)
- double(* M_fun)(double)
- double tB_max

3.2.1 Detailed Description

Ltb_model typdef struct Keeping all important information about specific LTB Model.

Definition at line 26 of file ltb.h.

3.2.2 Field Documentation

3.2.2.1 double(* Ltb_model::E_fun)(double)

E function

Definition at line 38 of file ltb.h.

3.2.2.2 gsl_vector* Ltb_model::E_vec

values of function E — gsl_vector

Definition at line 34 of file ltb.h.

3.2.2.3 double(* Ltb_model::M_fun)(double)

M function

Definition at line 39 of file ltb.h.

3.2.2.4 gsl_vector* Ltb_model::M_vec

values of function M — gsl_vector

Definition at line 35 of file ltb.h.

3.2.2.5 gsl_matrix* Ltb_model::R_mat

Matrix of R (areal radius) — gsl_matrix of size r_vec x r_mat

Definition at line 36 of file ltb.h.

3.2.2.6 size_t Ltb_model::r_size

Curvature of model values=-1,0,1 or -999 (error). Number of r points in model, size of r_vec.

Definition at line 29 of file ltb.h.

3.2.2.7 gsl_vector* Ltb_model::r_vec

values of r — gsl_vector.

Definition at line 31 of file ltb.h.

3.2.2.8 size_t Ltb_model::t_size

Number of t points in model. size of t_vec

Definition at line 30 of file ltb.h.

3.2.2.9 gsl_vector* Ltb_model::t_vec

values of t — gsl_vector.

Definition at line 32 of file ltb.h.

3.2.2.10 double(* Ltb_model::tB_fun)(double)

tB function

Definition at line 37 of file ltb.h.

3.2.2.11 double Ltb_model::tB_max

maximum value of tB function, needed for normalization

Definition at line 40 of file ltb.h.

3.2.2.12 gsl_vector* Ltb_model::tB_vec

values of function t_{b} — gsl_vector

Definition at line 33 of file ltb.h.

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

- src/ltb.h

Chapter 4

File Documentation

4.1 src/gsl_vector_help_functions.h File Reference

Custom extensions of gsl_vector, gsl_matrix from GSL library.

```
#include <string.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_matrix.h>
#include "ltb_error.h"
```

Macros

- #define **M**(m, i, j) (gsl_matrix_get(m,i,j))
- #define **MS**(m, i, j, x) (gsl_matrix_set(m,i,j,x))
- #define **V**(v, i) (gsl_vector_get(v,i))
- #define **VS**(v, i, x) (gsl_vector_set(v,i,x))

Functions

- int gsl_vector_gen (gsl_vector ∗vec, unsigned int n, double min, double max)

  Generate n values from min to max and kept them in gsl_vector vec.
- int gsl_vector_gen_fun (gsl_vector ∗vec, unsigned int n, gsl_vector ∗r, double(∗f)(double))

  Generate n values of function f(r) and kept them in gsl_vector vec.
- int min (gsl_vector ∗vec, double x)

  If ith element of vector vec is greater than x -> set x.
- int max (gsl_vector ∗vec, double x)

  If ith element of vector vec is lesser than x -> set x.
- int gsl_vector_substract (gsl_vector ∗a, gsl_vector ∗b)

  Subtraction of gsl_vector's a=a-b.
- int gsl_vector_substract_out (gsl_vector ∗a, gsl_vector ∗b, gsl_vector ∗out)

  Subtraction of gsl_vector's out=a-b.
- int gsl_vector_add_constant_out (gsl_vector ∗dest, gsl_vector ∗src, double x)

  Create new gsl_vector dest = src + x.
- int gsl_vector_set_range (gsl_vector ∗vec, unsigned int min, unsigned int max, double val)

  Set value val in vec over indexes from min to max.
- int gsl_vector_load_from_file (gsl_vector ∗vec, char ∗fileName)

Load gsl_vector vec from ascii file.

- int gsl_write_mat (gsl_matrix *matrix, char *fileName, char *header)

  Write gsl_matrix matrix to ascii file fileName with header.
- int gsl_write_two_vec (const gsl_vector *vec, const gsl_vector *vec2, char *fileName)

  Write to vectors to file in two columns.
- int find_indices_leq (const double *data, const double x, unsigned int n, unsigned int start)

  Find first indices of array data less equal than x.
- int gsl_vector_find_indices_leq (const gsl_vector *vec, const double x, unsigned int start)

  Find first indices of gsl_vector vec less equal than x.
- int find_indices_geq (const double *data, const double x, unsigned int n, unsigned int start)

  Find first indices of array data greater equal than x.
- int gsl_vector_find_indices_geq (const gsl_vector *vec, const double x, unsigned int start)

  Find first indices of gsl_vector vec greater equal than x.
- int find_indices_range (const double *data, const double xa, const double xb, unsigned int n, unsigned int start, int *ia, int *ib)

  Find indices indices from range from xa to xb in array data.
- int gsl_vector_indices_range (const gsl_vector *vec, const double xa, const double xb, unsigned int start, int *ia, int *ib)

  Find indices indices from range from xa to xb in gsl_vector vec.
- int data_to_string (double *data, char *format, unsigned int n, char *out)

  Convert double array to string.
- int gsl_vector_to_string (gsl_vector *vec, char *format, char *out)

  Convert gsl_vector to string.
- int gsl_matrix_row_normalize (gsl_matrix *mat)

  Normalize rows of matrix to 1.

4.1.1 Detailed Description

Custom extensions of gsl_vector, gsl_matrix from GSL library. Most of function from this file was used during testes. I only use functions for writing vectors and matrices and function for generating values from function (gsl_vector_gen_fun() and gsl_vector_gen())

Definition in file gsl_vector_help_functions.h.

4.2 src/interp_data.h File Reference

Structure and functions needed for interpolation *eta(xi)*

```
#include <gsl/gsl_spline.h>
#include "ltb_error.h"
```

Data Structures

- struct Interp_data

Functions

- Interp_data * interp_data_alloc (double max, double min, unsigned int n, int cur, double(*f)(double, int))

  Allocating Interp_data.
- void interp_data_free (Interp_data *idata)

Freeing Interp_data stuct.

- double set_eta_max (double xi_max)

  Calculating max value for Interp_data structure.

- double set_eta_min (double xi_min)

  Calculating min value for Interp_data structure.

- int check_interp_data (Interp_data *eta_interp, double xi_min, double xi_max, int cur)

  Check Interp_data.

Variables

- size_t **eta_n**
- double dxi

  Parameter used to calculate min and max value in Interp_data. It telling how much bigger (lower) max (min) should be than max (min) value of xi. Used in set_eta_max() and set_eta_min() functions.

4.2.1 Detailed Description

Structure and functions needed for interpolation *eta(xi)*

Author

Lukasz Matczynski

Copyright

GNU Public License

Definition in file interp_data.h.

4.2.2 Function Documentation

4.2.2.1 int check_interp_data (Interp_data * *eta_interp*, double *xi_min*, double *xi_max*, int *cur*)

Check Interp_data.

Return error if `cur != eta_interp->cur`, `xi_min < eta_interp->min` or `xi_max > eta_interp->max`

Parameters

in	*eta_interp*	— Interp_data to check
in	*xi_min*	
in	*xi_max*	
in	*cur*	curvature parameter

Definition at line 137 of file interp_data.c.

4.2.2.2 Interp_data* interp_data_alloc (double *max*, double *min*, unsigned int *n*, int *cur*, double(*)(double, int) *f*)

Allocating Interp_data.

Parameters

in	*max*	— maximum value of points to initialize spline function
in	*min*	— minimum value of points to initialize spline function
in	*n*	— number of points to initialize spline function
in	*cur*	curvature of model for which we want interpolate (xi)
in	*f*	pointer to function calculating eta

Returns

pointer to allocate Interp_data stuct or NULL pointer

Definition at line 7 of file interp_data.c.

4.2.2.3 void interp_data_free (Interp_data * *idata*)

Freeing Interp_data stuct.

Parameters

in	*idata*	pointer to Interp_data structure which we want to free

Definition at line 77 of file interp_data.c.

4.2.2.4 double set_eta_max (double *xi_max*)

Calculating max value for Interp_data structure.

Parameters

in	*xi_max*	maximum value of xi

Definition at line 103 of file interp_data.c.

4.2.2.5 double set_eta_min (double *xi_min*)

Calculating min value for Interp_data structure.

Parameters

in	*xi_mim*	minimum value of xi

Definition at line 120 of file interp_data.c.

4.2.3 Variable Documentation

4.2.3.1 double dxi

Parameter used to calculate min and max value in Interp_data. It telling how much bigger (lower) max (min) should be than max (min) value of xi. Used in set_eta_max() and set_eta_min() functions.

Default numbers of points to initialize spline interpolation

See Also

set_eta_max()
set_eta_min()

Definition at line 35 of file interp_data.h.

4.3 src/ltb.h File Reference

Ltb model function.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <gsl/gsl_spline.h>
#include "ltb_error.h"
#include "gsl_vector_help_functions.h"
#include "phys_constant.h"
#include "interp_data.h"
#include <omp.h>
```

Data Structures

- struct Ltb_model

Functions

- int set_cur (const gsl_vector *E_vec, int *cur)

  Setting curvature of model.
- int cur_fun (double E)

  Calculate curvature from E.
- double Ricci_fun (double E, double E_r, double R, double R_r)

  Calculate single values of Ricci Tensor.
- int ltb_model_Ricci (Ltb_model *ltb, gsl_vector *E_r, gsl_matrix *R_r, gsl_matrix *Ric)

  Calculate matrix of Ricci Tensor values.
- double rho_fun (double M_r, double R, double R_r)
- int ltb_model_rho (Ltb_model *ltb, gsl_vector *M_r, gsl_matrix *R_r, gsl_matrix *rho_mat)

  Calculate matrix of density in LTB model.
- double grr_sq_fun (double R_r, double E)

  Calculate root square of radial part of LTB metrics.
- int ltb_model_grr_sq (gsl_vector *E, gsl_matrix *R_r, gsl_matrix *grr_sq)

  Calculate matrix of root square of radial part of LTB model metrics .
- double exp_fun (double R, double R_r, double R_t, double R_rt)

  Calculate expansion parameter in LTB model.
- int ltb_model_exp (Ltb_model *ltb, gsl_matrix *R_r, gsl_matrix *R_t, gsl_matrix *R_rt, gsl_matrix *exp)

  Calculate matrix of expansion parameters in LTB model.
- int get_chi (int cur, gsl_vector *E, gsl_vector *chi_vec)

  Calculate vector of chi values.
- double fiFlat (double eta)

  Calculate values of fi in model with curvature = 0.
- double fiSpherical (double eta)

  Calculate values of fi in model with curvature > 0.
- double fiHyperbolic (double eta)

  Calculate values of fi in model with curvature < 0.
- double fi (double eta, int cur)

  Calculate values of fi.
- double xi_fun (double tB, double M, double E, double t, int cur)

  Calculet xi value using value of tB,M,E functions.

- int get_xi (gsl_vector *tB, gsl_vector *M, gsl_vector *t, gsl_vector *E, int cur, gsl_matrix *xi_mat)

  Calculate xi matrix using values of tB,M, functions.
- double xiHyperbolic (double eta)

  Calculate xi value in model with curvature < 0.
- double xiFlat (double eta)

  Calculate values of xi in model with curvature = 0.
- double xiSpherical (double eta)

  Calculate values of xi in model with curvature > 0.
- double xi_fun_from_eta (double eta, int cur)

  Calculate values of xi.
- Interp_data * interp_data_alloc_for_ltb (const Ltb_model *model, size_t n)

  Allocating interpolation data for specific Ltb_model.
- int get_min_max_for_interp (const Ltb_model *model, double *max, double *min)

  Finding value of max and min for Interp_data.
- double eta_fun (double xi, int cur, Interp_data *eta_interp)

  Calculate values of eta from xi.
- double R_fun (double M, double E, double eta, int cur)

  Calculate R (areal radius) value.
- Ltb_model * ltb_model_alloc (size_t _r_size, size_t _t_size, double(*M)(double), double(*E)(double), double(*tB)(double))

  Allocating Ltb_model structure.
- Ltb_model * ltb_model_alloc_from_vec (gsl_vector *_r_vec, gsl_vector *_t_vec, double(*M)(double), double(*E)(double), double(*tB)(double))

  Allocating Ltb_model structure from r vector and t vector.
- Ltb_model * ltb_model_cpy (Ltb_model *src)

  Allocating Ltb_model making copy of src.
- void ltb_model_free (Ltb_model *model)

  Freeing Ltb_model structure.
- int ltb_model_set_fun (Ltb_model *model)

  Setting values of tB, M, E function for Ltb_model model.
- int ltb_model_set_R (Ltb_model *model, Interp_data *eta_interp)

  Calculating matrix of function R(t,r)
- int ltb_model_write_fun (const Ltb_model *model, char *file_name)
- double R_t_analytic_fun (double R, double M, double E, int sgn)

  Calculating partial derivative of R over t.
- int ltb_model_R_t (const Ltb_model *model, Interp_data *eta_interp, gsl_matrix *R_t)

  Calculating partial derivatives of R over t.
- double R_r_analytic_fun (double R, double R_t, double E, double E_r, double M, double M_r, double tB, double tB_r, double t)

  Calculating partial derivative of R over r.
- int ltb_model_R_r (const Ltb_model *model, const gsl_matrix *R_t, const gsl_vector *E_r_vec, const gsl_vector *M_r_vec, const gsl_vector *tB_r_vec, gsl_matrix *R_r)

  Calculating value of partial derivatives of R over r.
- double R_rt_analytic_fun (double R, double R_r, double R_t, double E_r, double M, double M_r)

  Calculating value of partial mixed derivative of R over r and t.
- int ltb_model_R_rt (const Ltb_model *model, const gsl_matrix *R_r, const gsl_matrix *R_t, const gsl_vector *E_r, const gsl_vector *M_r, gsl_matrix *R_rt)

  Calculating value of partial mixed derivative of R over r and t.

Variables

- int **tB_norm**
- unsigned int **chunk_size**

4.3.1 Detailed Description

Ltb model function. Details(TODO)

Author

Lukasz Matczynski

Copyright

GNU Public License

Definition in file ltb.h.

4.3.2 Function Documentation

4.3.2.1 int cur_fun (double *E*)

Calculate curvature from E.

Parameters

in	*E*	value of $E(r)$ function from LTB model

Returns

curvature parameter from set {-1,0,1}

Definition at line 170 of file ltb.c.

4.3.2.2 double eta_fun (double *xi,* int *cur,* Interp_data * *eta_interp*)

Calculate values of eta from xi.

If curvature (cur = 0) is flat — use analytic formula for *eta(xi)*. Otherwise use eta_interp parameter for spline interpolation of *eta(xi)*. If cur != 0 && eta_interp == NULL throw up error and return GSL_NAN.

Parameters

in	*xi*	value of xi
in	*cur*	curvature parameter
in	*eta_interp*	Interp_data needed for spline interpolation Using gsl spline interpolation

Returns

Single eta value.

Definition at line 423 of file ltb.c.

4.3.2.3 double exp_fun (double *R,* double *R_r,* double *R_t,* double *R_rt*)

Calculate expansion parameter in LTB model.

Parameters

in	*R_r*	partial derivative of areal radius *R* ($\frac{\partial R}{\partial r}$) in point (*t*,*r*)
in	*R_t*	partial derivative of areal radius *R* ($\frac{\partial R}{\partial t}$) in point (*t*,*r*)
in	*R_rt*	partial derivative of areal radius *R* ($\frac{\partial^2 R}{\partial r \partial t}$) in point (*t*,*r*)

Returns

single value of expansion parameter

Definition at line 113 of file ltb.c.

4.3.2.4 double fi (double *eta*, int *cur*)

Calculate values of fi.

Depending on cur parameter choosing function to calculate *fi(eta)*

Parameters

in	*eta*	value of eta
in	*cur*	curvature parameter

Returns

fi value

See Also

fiFlat()
fiSpherical()
fiHyperbolic()

4.3.2.5 double fiFlat (double *eta*)

Calculate values of fi in model with curvature = 0.

Returns

fi value

See Also

fi()

Definition at line 276 of file ltb.c.

4.3.2.6 double fiHyperbolic (double *eta*)

Calculate values of fi in model with curvature < 0.

Returns

fi value

See Also

fi()

Definition at line 281 of file ltb.c.

4.3.2.7 double fiSpherical (double *eta*)

Calculate values of fi in model with curvature > 0.

Returns

fi value

See Also

fi()

Definition at line 271 of file ltb.c.

4.3.2.8 int get_chi (int *cur*, gsl_vector * *E*, gsl_vector * *chi_vec*)

Calculate vector of chi values.

Parameters

in	*cur*	curvature parameter
in	*E*	value of $E(r)$ LTB function
out	*chi_vec*	vector of chi values

Returns

error code or GSL_SUCCESS;

Definition at line 186 of file ltb.c.

4.3.2.9 int get_min_max_for_interp (const Ltb_model * *model*, double * *max*, double * *min*)

Finding value of max and min for Interp_data.

Parameters

in	*model*	pointer to Ltb_model
out	*max*	— max value of points to initialize spline procedure [out] min — mina value of points to initialize spline procedure

Returns

GSL_SUCCESS or error code

Definition at line 344 of file ltb.c.

4.3.2.10 int get_xi (gsl_vector * *tB*, gsl_vector * *M*, gsl_vector * *t*, gsl_vector * *E*, int *cur*, gsl_matrix * *xi_mat*)

Calculate xi matrix using values of *tB*,*M*, functions.

If cur parameter is not in {-1,0,1}, return GSL_NAN and throw up error

Parameters

in	*tB*	values of *tB* function
in	*M*	values of *M* function
in	*E*	values of *E* function
in	*t*	values of time coordinates
out	*xi_mat*	values of xi

Returns

GSL_SUCESS or error code

See Also

xi_fun_from_eta()

Definition at line 231 of file ltb.c.

4.3.2.11 double grr_sq_fun (double *R_r*, double *E*)

Calculate root square of radial part of LTB metrics.

Parameters

in	*R_r*	partial derivative of areal radius *R* ($\frac{\partial R}{\partial r}$) in point (*t*,*r*)
in	*E*	value of $E(r)$ LTB function

Returns

value or sqrt of radial part of LTB metrics

Definition at line 76 of file ltb.c.

4.3.2.12 Interp_data* interp_data_alloc_for_ltb (const Ltb_model * *model*, size_t *n*)

Allocating interpolation data for specific Ltb_model.

Parameters

in	*model*	pointer to Ltb_model
in	*n*	numbers of point to initialize spline

Returns

pointer to Interp_data* (or null pointer)

Definition at line 396 of file ltb.c.

4.3.2.13 Ltb_model* ltb_model_alloc (size_t *_r_size*, size_t *_t_size*, double(*)(double) *M*, double(*)(double) *E*, double(*)(double) *tB*)

Allocating Ltb_model structure.

`r_size` and `t_size` have to be > 0 otherwise throw up error and return NULL ptr. If can't allocate any of Ltb_model pointers return NULL.

Parameters

in	*_r_size*	number of *r* points (radial coordinates)
in	*_t_size*	number of *t* points (time coordinates)
in	*M*	pointer to *M(r)* function
in	*E*	pointer to *E(r)* function
in	*tB*	pointer to *tB(r)* function

Returns

pointer to Ltb_model

See Also

ltb_model_alloc_from_vec()
ltb_model_cpy()

Definition at line 519 of file ltb.c.

4.3.2.14 Ltb_model* ltb_model_alloc_from_vec (gsl_vector * *_r_vec*, gsl_vector * *_t_vec*, double(*)(double) *M*, double(*)(double) *E*, double(*)(double) *tB*)

Allocating Ltb_model structure from r vector and t vector.

After allocating Ltb_model with ltb_model_alloc() function (with r parameter equal to _r_vec->size and t parameter to _t_vec->size) copy _r_vec and _t_vec to Ltb_model `r_vec` and `t_vec`. If copying data fail throw up error and return null.

Parameters

in	*_r_vec*	vector of *r* values (radial coordinate)
in	*_t_vec*	vector of *t* values (time coordinate)
in	*M*	pointer to *M(r)* function
in	*E*	pointer to *E(r)* function
in	*tB*	pointer to *tB(r)* function

Returns

pointer to Ltb_model

See Also

ltb_model_alloc()
ltb_model_cpy()

Definition at line 597 of file ltb.c.

4.3.2.15 Ltb_model* ltb_model_cpy (Ltb_model * *src*)

Allocating Ltb_model making copy of src.

Allocate Ltb_model with ltb_model_alloc() function, after this copying of data from Ltb_model *src to newly created. If copying of any data fail throw up error and return NULL.

Parameters

in	*src*	Ltb_model to copy

Returns

pointer to Ltb_model

See Also

ltb_model_alloc_from_vec()
ltb_model_alloc()

Definition at line 634 of file ltb.c.

4.3.2.16 int ltb_model_exp (Ltb_model * *ltb*, gsl_matrix * *R_r*, gsl_matrix * *R_t*, gsl_matrix * *R_rt*, gsl_matrix * *exp*)

Calculate matrix of expansion parameters in LTB model.

Parameters

in	*ltb*	pointer to `Ltb_model` (using `ltb->R_mat`)
in	*R_r*	partial derivative of areal radius R ($\frac{\partial R}{\partial r}$) in point ($t$,$r$)
in	*R_t*	partial derivative of areal radius R ($\frac{\partial R}{\partial t}$) in point ($t$,$r$)
in	*R_rt*	partial derivative of areal radius R ($\frac{\partial^2 R}{\partial r \partial t}$) in point ($t$,$r$)
out	*exp*	matrix of expansion parameters

Returns

error code or GSL_SUCCESS;

See Also

expansion_fun()

Definition at line 119 of file ltb.c.

4.3.2.17 void ltb_model_free (Ltb_model * *model*)

Freeing Ltb_model structure.

Parameters

in	*model*	pointer to Ltb_model

Definition at line 689 of file ltb.c.

4.3.2.18 int ltb_model_grr_sq (gsl_vector * *E*, gsl_matrix * *R_r*, gsl_matrix * *grr_sq*)

Calculate matrix of root square of radial part of LTB model metrics .

Parameters

in	*E*	value of $E(r)$ LTB function
in	*R_r*	partial derivative of areal radius *R* ($\frac{\partial R}{\partial r}$) in point (*t*,*r*)
out	*matrix*	of grr_sq values

Returns

error code or GSL_SUCCESS;

See Also

grr_sq_fun()

Definition at line 87 of file ltb.c.

4.3.2.19 int ltb_model_R_r (const Ltb_model * *model*, const gsl_matrix * *R_t*, const gsl_vector * *E_r_vec*, const gsl_vector * *M_r_vec*, const gsl_vector * *tB_r_vec*, gsl_matrix * *R_r*)

Calculating value of partial derivatives of *R* over *r*.

Parameters

in	*R*	areal radius value
in	*R_t*	partial derivative of R over t
in	*E*	*LTB* E(r) function
in	*E_r*	derivative of *E(r)* function
in	*M*	*LTB* M(r) function
in	*M_r*	derivative of *M(r)* function
in	*tB*	LTB *tB(r)* function
in	*tB_r*	derivative of *tB(r)* function
out	*R_r*	matrix of partial derivative of *R* over *r*

Returns

GSL_SUCCESS or error code

See Also

ltb_model_R_r_analytic_fun()

Definition at line 910 of file ltb.c.

4.3.2.20 int ltb_model_R_rt (const Ltb_model * *model*, const gsl_matrix * *R_r*, const gsl_matrix * *R_t*, const gsl_vector * *E_r*, const gsl_vector * *M_r*, gsl_matrix * *R_rt*)

Calculating value of partial mixed derivative of *R* over *r* and *t*.

Parameters

in	*model*	pointer to Ltb_model
in	*R_r*	partial derivative of R over t
in	*R_t*	partial derivative of R over t
in	*E_r*	derivative of *E(r)* function

in	*M_r*	derivative of *M(r)* function
out	*R_rt*	matrix of mixed partial derivative

Returns

GSL_SUCCESS or error code

See Also

R_rt_analytic_fun()

Definition at line 955 of file ltb.c.

4.3.2.21 int ltb_model_R_t (const Ltb_model * *model*, Interp_data * *eta_interp*, gsl_matrix * *R_t*)

Calculating partial derivatives of *R* over *t*.

Parameters

in	*model*	pointer to Ltb_model
in	*eta_interp*	pointer to Interp_data needed for calculating sign of derivative.
out	*R_t*	matrix of derivatives

Returns

GSL_SUCCESS or error code

See Also

R_t_analytic_fun()

Definition at line 860 of file ltb.c.

4.3.2.22 int ltb_model_rho (Ltb_model * *ltb*, gsl_vector * *M_r*, gsl_matrix * *R_r*, gsl_matrix * *rho_mat*)

Calculate matrix of density in LTB model.

All pointers should be allocated otherwise return GSL_EFAULT. All matrix should have equals dimensions all vector should have size equal matrices second dimension otherwise return GSL_EBADLEN.

Parameters

in	*ltb*	pointer to `Ltb_model` (using `ltb->R_mat`)
in	*M_r*	first derivative of *M(r)* function in point *r*
in	*R_r*	partial derivative of areal radius *R* ($\frac{\partial R}{\partial r}$) in point (*t*,*r*)
out	*rho_mat*	matrix of density

Returns

error code or GSL_SUCCESS;

See Also

rho_fun()

Definition at line 47 of file ltb.c.

4.3.2.23 int ltb_model_Ricci (Ltb_model * *ltb*, gsl_vector * *E_r*, gsl_matrix * *R_r*, gsl_matrix * *Ric*)

Calculate matrix of Ricci Tensor values.

All pointers should be allocated otherwise return GSL_EFAULT. All matrix should have equals dimensions all vector should have size equal matrices second dimension otherwise return GSL_EBADLEN.

Parameters

in	*ltb*	pointer to `Ltb_model` (using `ltb->R_mat`)
in	*E_r*	first derivative of E
in	*R_r*	partial derivative over r of R
out	*Ric*	pointer to Ricci matrix

Returns

error code or GSL_SUCCESS;

See Also

Ricci_fun()

Definition at line 13 of file ltb.c.

4.3.2.24 int ltb_model_set_fun (Ltb_model * *model*)

Setting values of tB, M, E function for Ltb_model model.

Calculating values of LTB functions (*M*, *E*, *tB*) using pointers tu function `model->M_fun`, `model->E_fun`, `model->tB_fun` . If one of this pointers is NULL return GSL_EFAULT. Calculated values are stored in Ltb_model gsl_vectors (`M_vec`, `E_vec` and `tB_vec`). For tB is calculate `model->tB_max` if global parameter tB_norm is set values in `tB_vec` will be normalized (`gsl_vector_max(tB_vec)` == 0). Depending on `E_vec` curvature parameter is calculated (E $<$ 0 - cur = 1, E $>$ 0 – cur = -1 , E == 0 cur=0, otherwise error will be throw up and cur == -999).

Parameters

in	*model*	pointer to Ltb_model

Returns

GSL_SUCCESS or error code

See Also

ltb_model_alloc()
ltb_model_set_R

Definition at line 730 of file ltb.c.

4.3.2.25 int ltb_model_set_R (Ltb_model * *model*, Interp_data * *eta_interp*)

Calculating matrix of function *R(t,r)*

eta_interp pointer is needed to calculate eta in curved model. If you only interested in flat model you can pass NULL in place of eta_interp.

Parameters

in	*model*	pointer to Ltb_model
in	*eta_interp*	pointer to Inter_data structure

Returns

GSL_SUCCESS or error code

See Also

ltb_model_set_fun()
ltb_model_alloc()

Definition at line 472 of file ltb.c.

4.3.2.26 int ltb_model_write_fun (const Ltb_model * *model*, char * *file_name*)

Write basis function E,tB,M of Ltb_model model to ascii file file_name

Writing to file content of `r_vec`, `M_vec`, `E_vec`, `tB_vec` in four columns.

Parameters

in	*model*	pointer to Ltb_model with functions to write
in	*file_name*	name of file

Returns

GSL_SUCCESS or error code

See Also

ltb_model_write_fun()

Definition at line 793 of file ltb.c.

4.3.2.27 double R_fun (double *M*, double *E*, double *eta*, int *cur*)

Calculate *R* (areal radius) value.

Parameters

in	*M*	value of *M(r)* function
in	*E*	value of *E(r)* function
in	*tB*	value of *tB(r)* function
in	*eta*	value of eta
in	*cur*	value of curvature parameter

Returns

value of R

See Also

ltb_model_set_R()

Definition at line 454 of file ltb.c.

4.3.2.28 double R_r_analytic_fun (double *R*, double *R_t*, double *E*, double *E_r*, double *M*, double *M_r*, double *tB*, double *tB_r*, double *t*)

Calculating partial derivative of *R* over *r*.

Parameters

in	*R*	areal radius value
in	*M*	value of *M(r)* function
in	*E*	value of *E(r)* function
in	*sgn*	sign of derivative (have to be calculated separately)

Returns

Partial derivative of *R* over *r*

See Also

ltb_model_R_r()

Definition at line 894 of file ltb.c.

4.3.2.29 double R_rt_analytic_fun (double *R*, double *R_r*, double *R_t*, double *E_r*, double *M*, double *M_r*)

Calculating value of partial mixed derivative of *R* over *r* and *t*.

Parameters

in	*R*	areal radius value
in	*R_r*	partial derivative of R over t
in	*R_t*	partial derivative of R over t
in	*E_r*	derivative of *E(r)* function
in	*M*	*LTB* M(r) function
in	*M_r*	derivative of *M(r)* function

Returns

value of partial mixed derivative of *R* over *r* and *t*

See Also

ltb_model_R_rt()

Definition at line 950 of file ltb.c.

4.3.2.30 double R_t_analytic_fun (double *R*, double *M*, double *E*, int *sgn*)

Calculating partial derivative of *R* over *t*.

Parameters

in	*R*	areal radius value
in	*M*	value of *M(r)* function
in	*E*	value of *E(r)* function
in	*sgn*	sign of derivative (have to be calculated separately)

Returns

Partial derivative of *R* over *t*

See Also

ltb_model_R_t()

Definition at line 822 of file ltb.c.

4.3.2.31 double rho_fun (double *M_r*, double *R*, double *R_r*)

Calculate density in point (*t*,*r*).

Calculate density using equation [**?**] :

$$\rho = \frac{M_{,r}}{4\pi R^2 R_{,r}}$$

Parameters

M_r	first derivative of *M(r)* function in point *r*
R	value of areal radius in point (*t*,*r*)
R_r	partial derivative of areal radius *R* ($\frac{\partial R}{\partial r}$) in point (*t*,*r*)

Returns

value of density

See Also

ltb_model_rho()

Definition at line 42 of file ltb.c.

4.3.2.32 double Ricci_fun (double *E*, double *E_r*, double *R*, double *R_r*)

Calculate single values of Ricci Tensor.

Parameters

in	*E*	value of LTB $E(r)$ function
in	*E_r*	first derivative of E
in	*R*	areal radius
in	*R_r*	partial derivative over r of R

Returns

value of Ricci Tensor (double)

Definition at line 8 of file ltb.c.

4.3.2.33 int set_cur (const gsl_vector * *E_vec*, int * *cur*)

Setting curvature of model.

Checking all values of E gsl_vector If all values have the same sign set cur parameter and return GSL_SUCCESS. If E values have different sign set cur parameter to error value -999 and return LTB_ECURV.

Parameters

in	*E*	vector of values of $E(r)$ LTB function
out	*cur*	curvature parameter

Returns

GSL_SUCCESS or LTB_ECURV

See Also

set_cur()

Definition at line 152 of file ltb.c.

4.3.2.34 double xi_fun (double *tB*, double *M*, double *E*, double *t*, int *cur*)

Calculet xi value using value of tB,M,E functions.

Parameters

in	*tB*	value of *tB* function
in	*M*	value of *M* function
in	*E*	value of *E* function
in	*t*	value of time coordinate
in	*cur*	curvature parameter

Returns

value of xi

Definition at line 210 of file ltb.c.

4.3.2.35 double xi_fun_from_eta (double *eta*, int *cur*)

Calculate values of xi.

Depending on cur parameter choosing function to calculate *xi(eta)*

Parameters

in	*eta*	value of eta
in	*cur*	curvature parameter

Returns

fi value

See Also

xiFlat()
xiSpherical()
xiHyperbolic()

Definition at line 307 of file ltb.c.

4.3.2.36 double xiFlat (double *eta*)

Calculate values of xi in model with curvature = 0.

Returns

xi value

See Also

xi_fun_from_eta()

Definition at line 328 of file ltb.c.

4.3.2.37 double xiHyperbolic (double *eta*)

Calculate xi value in model with curvature < 0.

Returns

xi value

See Also

xi_fun_from_eta()

Definition at line 333 of file ltb.c.

4.3.2.38 double xiSpherical (double *eta*)

Calculate values of xi in model with curvature > 0.

Returns

xi value

See Also

xi_fun_from_eta()

Definition at line 338 of file ltb.c.

4.4 src/ltb_error.h File Reference

Errors Handling macros and functions.

```
#include <gsl/gsl_errno.h>
#include <gsl/gsl_nan.h>
```

Macros

- #define CHECK_STAT(status)
- #define CHECK_PTR(stptr, errno)
- #define CHECK_M_DIM(m1, m2, reason)
- #define CHECK_SIZE(s1, s2, reason)

Enumerations

- enum { LTB_ECURV = 100, LTB_EEFUN = 101 }

4.4.1 Detailed Description

Errors Handling macros and functions.

Definition in file ltb_error.h.

4.4.2 Macro Definition Documentation

4.4.2.1 #define CHECK_M_DIM(*m1, m2, reason*)

Value:

```
if (m1->size1 != m2->size1 || m1->size2 != m2->size2){ \
        gsl_error(reason,__FILE__, __LINE__, GSL_EBADLEN);\
        return GSL_EBADLEN; \
    }
```

Checking if two matrices have the same shape

Definition at line 37 of file ltb_error.h.

4.4.2.2 #define CHECK_PTR(*stptr, errno*)

Value:

```
if(!stptr){ \
        gsl_error ("Bad pointer", __FILE__, __LINE__, errno) ; \
        return errno; \
    }
```

Checking pointer macro - if pointer is NULL return specific error errno

Definition at line 30 of file ltb_error.h.

4.4.2.3 #define CHECK_SIZE(*s1, s2, reason*)

Value:

```
if (s1 != s2){ \
        gsl_error(reason,__FILE__, __LINE__, GSL_EBADLEN);\
        return GSL_EBADLEN; \
    }
```

Checking if two values differ

Definition at line 44 of file ltb_error.h.

4.4.2.4 #define CHECK_STAT(*status*)

Value:

```
if(status){ \
        return status; \
    }
```

Checking status macro - if status is $>$ 0 then return status

Definition at line 23 of file ltb_error.h.

4.4.3 Enumeration Type Documentation

4.4.3.1 anonymous enum

Ltb erro codes

Enumerator

LTB_ECURV Wrong value of curvature

LTB_EEFUN E(r)<-0.5 metric degeneration

Definition at line 15 of file ltb_error.h.

4.5 src/ltb_integrate.h File Reference

Functions for integrating.

```
#include <gsl/gsl_matrix.h>
#include "ltb_error.h"
```

Functions

- int gsl_matrix_integrate_trap_range (const gsl_matrix *mat, double dr, unsigned int rstart, unsigned int rstop, unsigned int tstart, unsigned int tstop, gsl_matrix *integral)

  Integrate using trapezoid method points kept in gsl_matrix.

4.5.1 Detailed Description

Functions for integrating. Details(TODO)

Author

Lukasz Matczynski

Copyright

GNU Public License

Definition in file ltb_integrate.h.

4.5.2 Function Documentation

4.5.2.1 int gsl_matrix_integrate_trap_range (const gsl_matrix * *mat*, double *dr*, unsigned int *rstart*, unsigned int *rstop*, unsigned int *tstart*, unsigned int *tstop*, gsl_matrix * *integral*)

Integrate using trapezoid method points kept in gsl_matrix.

This function is needed for calculating radial proper length d(t,r). To get it we mast integrate points in rows of grr_sq matrix (ltb_model_grr_sq()). To integrate I'm using trapezoid method:

$$\int_a^b f(x)\mathrm{dx} \approx \frac{h}{2}\sum_{i=1}^{n}\left(f(x_i)+f(x_{i+1})\right)$$

Where h size of integration interval. When r dimension of matrix grr_sq is n we calculate n integrals

$$\int_{m[0]}^{m[0]} f(x)\mathrm{dx}, \int_{m[0]}^{m[1]} f(x)\mathrm{dx}, ..., \int_{m[0]}^{m[n-1]} f(x)\mathrm{dx}$$

Where $m[i]$ are matrix elements. First of this integrals is always 0.

Parameters

in	*mat*	matrix to integrate
in	*dr*	size of integration interval
in	*rstart*	element of row from which start to integrate
in	*rstop*	element of row on which stop integrating
in	*tstart*	row from which start to integrate
in	*tstop*	row on which stop integrating
out	*integral*	matrix where result of integration is kept

See Also

ltb_model_grr_sq()

Definition at line 3 of file ltb_integrate.c.

4.6 src/macros.h File Reference

Additional helpful macros.

4.6.1 Detailed Description

Additional helpful macros.

Definition in file macros.h.

4.7 src/phys_constant.h File Reference

Physics constant.

```
#include "macros.h"
```

Macros

- #define G_DEF 6.67300e-11 ∗ 1.98892e40 / 3.08568025e25 / (299792458.0∗299792458.0)

  Default G parameter c=1, Gpc / 10^10M_{sun}.
- #define H_0_DEF 100.0 / 299792.458 ∗ 1000

  Default Hubble parameter c=1 units Gpc∗h^{-1}.
- #define C_DEF 299792458.0;

  Default c — speed of light.

Functions

- void export_phys_const (char ∗file_name)

  Write physical constant to file.
- void set_phys_const (double G_in, double H_0_in, double c_in)

  Set physical constant.

Variables

- static const double pi = 3.14159265
- double G_grav
- double H_0
- double c
- static const double Gyr_per_Gpc = 3.08568025e16 / 299792458.0 / (365.25∗24.0∗3600.0)

  Gyr/Gpc value Parameter used to conver time values to distance values.

4.7.1 Detailed Description

Physics constant.

Definition in file phys_constant.h.

4.7.2 Function Documentation

4.7.2.1 void export_phys_const (char ∗ *file_name*)

Write physical constant to file.

Parameters

in	*file*	

Definition at line 8 of file phys_constant.c.

4.7.2.2 void set_phys_const (double *G_in*, double *H_0_in*, double *c_in*)

Set physical constant.

Parameters

in	*G_in*	Gravity Const.
in	*H_0_in*	actual value of Hubbel Parameter
in	*c_in*	speed of light

Definition at line 20 of file phys_constant.c.

4.7.3 Variable Documentation

4.7.3.1 double c

c speed of light

Definition at line 6 of file phys_constant.c.

4.7.3.2 double G_grav

Gravity constant G

Definition at line 4 of file phys_constant.c.

4.7.3.3 double H_0

Hubbel parameter

Definition at line 5 of file phys_constant.c.

4.7.3.4 const double pi = 3.14159265 [static]

pi constant

Definition at line 10 of file phys_constant.h.

Index

acc
    Interp_data, 5

c
    phys_constant.h, 33
CHECK_M_DIM
    ltb_error.h, 30
CHECK_PTR
    ltb_error.h, 30
CHECK_SIZE
    ltb_error.h, 30
CHECK_STAT
    ltb_error.h, 30
check_interp_data
    interp_data.h, 11
cur
    Interp_data, 5
cur_fun
    ltb.h, 15

dxi
    interp_data.h, 12

E_fun
    Ltb_model, 7
E_vec
    Ltb_model, 7
eta_fun
    ltb.h, 15
exp_fun
    ltb.h, 15
export_phys_const
    phys_constant.h, 33

fi
    ltb.h, 16
fiFlat
    ltb.h, 16
fiHyperbolic
    ltb.h, 16
fiSpherical
    ltb.h, 16

G_grav
    phys_constant.h, 33
get_chi
    ltb.h, 17
get_min_max_for_interp
    ltb.h, 17
get_xi
    ltb.h, 17
grr_sq_fun
    ltb.h, 18
gsl_matrix_integrate_trap_range
    ltb_integrate.h, 31

H_0
    phys_constant.h, 33

Interp_data, 5
    acc, 5
    cur, 5
    max, 5
    min, 6
    spline, 6
    x, 6
    y, 6
interp_data.h
    check_interp_data, 11
    dxi, 12
    interp_data_alloc, 11
    interp_data_free, 12
    set_eta_max, 12
    set_eta_min, 12
interp_data_alloc
    interp_data.h, 11
interp_data_alloc_for_ltb
    ltb.h, 18
interp_data_free
    interp_data.h, 12

LTB_ECURV
    ltb_error.h, 30
LTB_EEFUN
    ltb_error.h, 30
ltb.h
    cur_fun, 15
    eta_fun, 15
    exp_fun, 15
    fi, 16
    fiFlat, 16
    fiHyperbolic, 16
    fiSpherical, 16
    get_chi, 17
    get_min_max_for_interp, 17
    get_xi, 17
    grr_sq_fun, 18
    interp_data_alloc_for_ltb, 18
    ltb_model_R_r, 21
    ltb_model_R_rt, 21
    ltb_model_R_t, 22

ltb_model_Ricci, 22
ltb_model_alloc, 18
ltb_model_alloc_from_vec, 19
ltb_model_cpy, 19
ltb_model_exp, 20
ltb_model_free, 20
ltb_model_grr_sq, 20
ltb_model_rho, 22
ltb_model_set_R, 23
ltb_model_set_fun, 23
ltb_model_write_fun, 24
R_fun, 24
R_r_analytic_fun, 24
R_rt_analytic_fun, 26
R_t_analytic_fun, 26
rho_fun, 26
Ricci_fun, 27
set_cur, 27
xi_fun, 27
xi_fun_from_eta, 28
xiFlat, 28
xiHyperbolic, 28
xiSpherical, 29
ltb_error.h
LTB_ECURV, 30
LTB_EEFUN, 30
ltb_error.h
CHECK_M_DIM, 30
CHECK_PTR, 30
CHECK_SIZE, 30
CHECK_STAT, 30
ltb_integrate.h
gsl_matrix_integrate_trap_range, 31
Ltb_model, 6
E_fun, 7
E_vec, 7
M_fun, 7
M_vec, 7
R_mat, 7
r_size, 7
r_vec, 7
t_size, 7
t_vec, 7
tB_fun, 8
tB_max, 8
tB_vec, 8
ltb_model_R_r
ltb.h, 21
ltb_model_R_rt
ltb.h, 21
ltb_model_R_t
ltb.h, 22
ltb_model_Ricci
ltb.h, 22
ltb_model_alloc
ltb.h, 18
ltb_model_alloc_from_vec
ltb.h, 19
ltb_model_cpy
ltb.h, 19
ltb_model_exp
ltb.h, 20
ltb_model_free
ltb.h, 20
ltb_model_grr_sq
ltb.h, 20
ltb_model_rho
ltb.h, 22
ltb_model_set_R
ltb.h, 23
ltb_model_set_fun
ltb.h, 23
ltb_model_write_fun
ltb.h, 24

M_fun
Ltb_model, 7
M_vec
Ltb_model, 7
max
Interp_data, 5
min
Interp_data, 6

phys_constant.h
c, 33
export_phys_const, 33
G_grav, 33
H_0, 33
pi, 33
set_phys_const, 33
pi
phys_constant.h, 33

R_fun
ltb.h, 24
R_mat
Ltb_model, 7
R_r_analytic_fun
ltb.h, 24
R_rt_analytic_fun
ltb.h, 26
r_size
Ltb_model, 7
R_t_analytic_fun
ltb.h, 26
r_vec
Ltb_model, 7
rho_fun
ltb.h, 26
Ricci_fun
ltb.h, 27

set_cur
ltb.h, 27
set_eta_max
interp_data.h, 12

set_eta_min
    interp_data.h, 12
set_phys_const
    phys_constant.h, 33
spline
    Interp_data, 6
src/gsl_vector_help_functions.h, 9
src/interp_data.h, 10
src/ltb.h, 13
src/ltb_error.h, 29
src/ltb_integrate.h, 31
src/macros.h, 32
src/phys_constant.h, 32

t_size
    Ltb_model, 7
t_vec
    Ltb_model, 7
tB_fun
    Ltb_model, 8
tB_max
    Ltb_model, 8
tB_vec
    Ltb_model, 8

x
    Interp_data, 6
xi_fun
    ltb.h, 27
xi_fun_from_eta
    ltb.h, 28
xiFlat
    ltb.h, 28
xiHyperbolic
    ltb.h, 28
xiSpherical
    ltb.h, 29

y
    Interp_data, 6