CosmoMAD

0.9

Generated by Doxygen 1.8.6

Fri Jun 5 2015 15:34:36

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 Csm_params Struct Reference

Basic parameter structure in CosmoMAD.

3.1.1 Detailed Description

Basic parameter structure in CosmoMAD.

A Csm_params structure contains all the information loaded by the user regarding a given cosmological model.

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

- /home/damonge/Science/Codes/CosmoMad_dev/include/cosmo_mad.h

Chapter 4

File Documentation

4.1 /home/damonge/Science/Codes/CosmoMad_dev/include/cosmo_mad.h File Reference

Header file for the CosmoMAD library.

Data Structures

- struct Csm_params

  Basic parameter structure in CosmoMAD.

Functions

- void csm_unset_gsl_eh (void)

  Unset GSL error handler.
- void csm_set_verbosity (int verb)

  Sets verbosity level.
- void csm_params_free (Csm_params ∗par)

  Csm_params destructor.
- Csm_params ∗ csm_params_new (void)

  Csm_params creator.
- double csm_omega_matter (Csm_params ∗par, double aa)

  Matter parameter.
- double csm_hubble (Csm_params ∗par, double aa)

  Hubble parameter.
- double csm_cosmic_time (Csm_params ∗par, double aa)

  Cosmic time.
- double csm_particle_horizon (Csm_params ∗par, double aa)

  Particle horizon.
- double csm_radial_comoving_distance (Csm_params ∗par, double aa)

  Radial comoving distance.
- double csm_curvature_comoving_distance (Csm_params ∗par, double aa)

  Curvature comoving distance.
- double csm_angular_diameter_distance (Csm_params ∗par, double aa)

  Angular diameter distance.
- double csm_luminosity_distance (Csm_params ∗par, double aa)

Luminosity distance.
- double csm_growth_factor (Csm_params ∗par, double aa)

  Growth factor.
- double csm_f_growth (Csm_params ∗par, double aa)

  Growth rate.
- double csm_growth_factor_and_growth_rate (Csm_params ∗par, double aa, double ∗gf, double ∗fg)

  Growth factor AND growth rate.
- double csm_scale_factor (Csm_params ∗par, double t)

  Scale factor.
- double csm_theta_BAO (Csm_params ∗par, double aa)

  Angular BAO scale.
- double csm_Dz_BAO (Csm_params ∗par, double aa)

  Radial BAO scale.
- void csm_background_set (Csm_params ∗par, double OmegaM, double OmegaL, double OmegaB, double ww, double wwa, double hh, double T_CMB)

  Set background cosmology.
- void csm_set_linear_pk (Csm_params ∗par, char ∗fname, double lkmn, double lkmx, double dlk, double nns, double s8)

  Set linear power spectrum.
- void csm_set_nonlinear_pk (Csm_params ∗par, char ∗fnamePkHFIT)

  Set non-linear power spectrum.
- double csm_Pk_linear_0 (Csm_params ∗par, double kk)

  Linear power spectrum.
- double csm_Pk_nonlinear (Csm_params ∗par, double kk)

  Non-linear power spectrum.
- double csm_xi2p_L (Csm_params ∗par, double r, double R1, double R2, char ∗wf1, char ∗wf2, double errfac)

  Linear correlation function.
- double csm_sig0_L (Csm_params ∗par, double R1, double R2, char ∗wf1, char ∗wf2)

  Linear covariance.
- void csm_set_Pk_params (Csm_params ∗par, double beta, double gf, double bias, int l_max)

  Set power spectrum params.
- double csm_Pk_full (Csm_params ∗par, double kk, double muk)

  Full power spectrum.
- double csm_Pk_multipole (Csm_params ∗par, double kk, int l)

  Power spectrum multipole.
- double csm_p_leg (int l, double x)

  Legendre polynomial.
- double csm_j_bessel (int l, double x)

  Spherical Bessel function.
- double csm_xi_multipole (Csm_params ∗par, double rr, int l)

  Correlation function multipole.
- void csm_set_xi_multipole_splines (Csm_params ∗par)

  Set multipole splines.
- void csm_unset_xi_multipole_splines (Csm_params ∗par)

  Unset multipole splines.
- double csm_xi_3D (Csm_params ∗par, double rr, double mu)

  3D correlation function (polar coords)
- double csm_xi_pi_sigma (Csm_params ∗par, double pi, double sigma, int use_multipoles)

  3D correlation function (orthogonal coords)
- double csm_M2R (Csm_params ∗par, double mass)

  Radius of a comoving sphere of mass M.

- double csm_R2M (Csm_params *par, double radius)

  Mass of a sphere of comoving radius R.
- double csm_collapsed_fraction (Csm_params *par, double mass, char *mf_model)

  Collapsed fraction.

4.1.1 Detailed Description

Header file for the CosmoMAD library.

Author

David Alonso

Date

13 Oct 2013 This header file contains the definitions of all the available CosmoMAD functions and macros.

4.1.2 Function Documentation

4.1.2.1 double csm_angular_diameter_distance (Csm_params * *par,* double *aa*)

Angular diameter distance.

Returns the angular diameter distance $d_A(a)$ at $a = \mathtt{aa}$ as $d_A(a) = a\,r(a)$, where $r(a)$ is the curvature comoving distance calculated with csm_curvature_comoving_distance.

4.1.2.2 void csm_background_set (Csm_params * *par,* double *OmegaM,* double *OmegaL,* double *OmegaB,* double *ww,* double *wwa,* double *hh,* double *T_CMB*)

Set background cosmology.

Sets background cosmology for `par`: $\Omega_M = \mathtt{OM}$, $\Omega_\Lambda = \mathtt{OL}$, $\Omega_b = \mathtt{OB}$, $w_0 = \mathtt{w0}$, $w_a = \mathtt{wa}$, $h = \mathrm{hh}$ and $T_{\rm CMB} = \mathtt{T\_CMB}$. This function must be called before calculating any a-dependent quantity.

4.1.2.3 double csm_collapsed_fraction (Csm_params * *par,* double *mass,* char * *mf_model*)

Collapsed fraction.

Returns the fraction of the Universe that has collapsed into halos of mass larger than `mass` according to the mass function parametrization given by `mf_model`. Three models are supported:

- "PS" (Press & Schechter, 1974):
$$F_{\rm PS}(<M) = {\rm erfc}(\nu/\sqrt{2})$$
- "JAP" (Peacock, 2007):
$$F_{\rm JAP}(<M) = \frac{\exp(-c\,\nu^2)}{1+a\,\nu^b},$$
  with $(a,b,c) = (1.529, 0.704, 0.412)$.
- "ST" (Sheth & Tormen, 2002):
$$F_{\rm ST}(<M) = A\left[{\rm erfc}\left(\sqrt{\frac{a}{2}}\nu\right) + \frac{\Gamma(1/2-p, a\,\nu^2/2)}{\sqrt{\pi}\,2^p}\right]$$
  with $(A,a,p) = (0.322, 0.707, 0.3)$.

4.1.2.4 double csm_cosmic_time (**Csm_params** $*$ *par,* double *aa*)

Cosmic time.

Returns the cosmic time $t(a)$ at $a = \mathtt{aa}$ by calculating the integral

$$t(a) = \int_0^a \frac{da'}{a' H(a')}.$$

4.1.2.5 double csm_curvature_comoving_distance (**Csm_params** $*$ *par,* double *aa*)

Curvature comoving distance.

Returns the curvature comoving distance $r(a)$ at $a = \mathtt{aa}$ as

$$r(a) = \frac{1}{H_0\sqrt{|\Omega_k|}} \mathrm{sinn}\left(H_0\sqrt{|\Omega_k|}\chi(a)\right),$$

where $\chi(a)$ is the radial comoving distance calculated with csm_radial_comoving_distance.

4.1.2.6 double csm_Dz_BAO (**Csm_params** $*$ *par,* double *aa*)

Radial BAO scale.

Returns the radial scale of the BAO as a redshift separation $a = \mathtt{aa}$ as $\Delta z_{\mathrm{BAO}}(a) = H(a)\, r_s$, where r_s is the sound horizon scale estimated through the Eiseinstein & Hu fitting formula and $H(a)$ is the Hubble paramteter calculated with csm_hubble.

4.1.2.7 double csm_f_growth (**Csm_params** $*$ *par,* double *aa*)

Growth rate.

Returns the linear growth rate $f(a) \equiv d\,\log D(a)/d\,\log a$ at $a = \mathtt{aa}$.

4.1.2.8 double csm_growth_factor (**Csm_params** $*$ *par,* double *aa*)

Growth factor.

Returns the linear growth factor $D(a)$ at $a = \mathtt{aa}$ (normalized to $D(a \ll 1) \simeq a$) by solving the equation for matter perturbations:

$$\frac{d}{da}\left(a^3 H(a)\frac{dD}{da}\right)(a) = \frac{3}{2}H(a)\, a\, \Omega_M(a) D.$$

4.1.2.9 double csm_growth_factor_and_growth_rate (**Csm_params** $*$ *par,* double *aa,* double $*$ *gf,* double $*$ *fg*)

Growth factor AND growth rate.

Returns the linear growth factor $D(a)$ and growth rate $f(a)$ simultaneously at $a = \mathtt{aa}$ in the variables `gf` and `fg`. If both quantities are needed at the same time, calling this function once is more efficient than calling csm_growth_-factor and csm_f_growth separately, since both are simultaneously estimated when solving the evolution equation for matter perturbations.

4.1.2.10 double csm_hubble (**Csm_params** $*$ *par,* double *aa*)

Hubble parameter.

Returns the Hubble parameter $H(a)$ at $a = \mathtt{aa}$ in $h/$Mpc.

4.1.2.11 double csm_j_bessel (int *l*, double *x*)

Spherical Bessel function.

Returns the spherical Bessel function of order `l` at `x`.

4.1.2.12 double csm_luminosity_distance (**Csm_params** ∗ *par*, double *aa*)

Luminosity distance.

Returns the luminosity distance $d_L(a)$ at $a = \mathtt{aa}$ as $d_L(a) = r(a)/a$, where $r(a)$ is the curvature comoving distance calculated with csm_curvature_comoving_distance.

4.1.2.13 double csm_M2R (**Csm_params** ∗ *par*, double *mass*)

Radius of a comoving sphere of mass M.

Returns the comoving radius of a sphere of mass M (in $M_\odot/h$).

4.1.2.14 double csm_omega_matter (**Csm_params** ∗ *par*, double *aa*)

Matter parameter.

Returns the matter parameter $\Omega_M(a)$ at $a = \mathtt{aa}$.

4.1.2.15 double csm_p_leg (int *l*, double *x*)

Legendre polynomial.

Returns the Legendre polynomial of order `l` at `x`.

4.1.2.16 void csm_params_free (**Csm_params** ∗ *par*)

Csm_params destructor.

Frees up all memory associated with `par`.

4.1.2.17 **Csm_params**∗ csm_params_new (void)

Csm_params creator.

Returns an initialized Csm_params structure, without any associated cosmological information (see csm_background_set).

4.1.2.18 double csm_particle_horizon (**Csm_params** ∗ *par*, double *aa*)

Particle horizon.

Returns the particle horizon $\chi_p(a)$ at $a = \mathtt{aa}$ by calculating the integral

$$\chi_p(a) \int_0^a \frac{da'}{a'^2 H(a')}.$$

4.1.2.19 double csm_Pk_full (Csm_params * *par,* double *kk,* double *muk*)

Full power spectrum.

Returns the full redshift-space power spectrum for $k = \mathtt{kk}$ and $\mu_k = \mathtt{muk}$. The current implementation uses the Kaiser approximation for RSDs:

$$P_s(a,k,\mu_k) = b^2(1+\beta(a)\mu_k^2)^2 P_{\rm NL}(a,k),$$

where $P_{\rm NL}(a,k)$ is the non-linear power spectrum calculated using csm_Pk_nonlinear, which is normalized using the supplied growth factor.

4.1.2.20 double csm_Pk_linear_0 (Csm_params * *par,* double *kk*)

Linear power spectrum.

Returns the linear power spectrum at $a = 1$ and wave number $k = \mathtt{kk}$. If kk lies outside the interpolation limits, $P(k)$ is approximated by $P(k) \propto k^{n_s}$ for small k and by $P(k) \propto k^{-3}$ for large k.

4.1.2.21 double csm_Pk_multipole (Csm_params * *par,* double *kk,* int *l*)

Power spectrum multipole.

Returns the l-th power spectrum multipole:

$$P_l(k) = \frac{2l+1}{2}\int_{-1}^{1} L_l(\mu_k)\,P(k,\mu_k),$$

where $L_l(x)$ is the Legendre polynomial of order l.

4.1.2.22 double csm_Pk_nonlinear (Csm_params * *par,* double *kk*)

Non-linear power spectrum.

Returns the non-linear power spectrum for $k = \mathtt{kk}$. If kk lies outside the interpolation limits, $P(k)$ is approximated by $P(k) \propto k^{n_s}$ for small k and by $P(k) \propto k^{-3}$ for large k. This function returns the power spectrum normalized with the growth factor supplied by csm_set_Pk_params, but without bias or RSDs.

4.1.2.23 double csm_R2M (Csm_params * *par,* double *radius*)

Mass of a sphere of comoving radius R.

Returns the comoving mass (in $M_\odot/h$) inside a sphere of comoving radius radius.

4.1.2.24 double csm_radial_comoving_distance (Csm_params * *par,* double *aa*)

Radial comoving distance.

Returns the radial comoving distance $\chi(a)$ at $a = \mathtt{aa}$ as $\chi(a) = \chi_p(1) - \chi_p(a)$, where χ_p is the particle horizon calculated with csm_particle_horizon.

4.1.2.25 double csm_scale_factor (Csm_params * *par,* double *t*)

Scale factor.

Returns the value of the scale factor $a(t)$ at cosmic time $t = \mathtt{t}$. The first time this function is called, the relation $t(a)$ is calculated for several values of $a \in [0,1]$ using csm_cosmic_time, and a spline is used to invert this relation in all subsequent calls.

4.1.2.26 void csm_set_linear_pk (**Csm_params** ∗ *par,* char ∗ *fname,* double *lkmn,* double *lkmx,* double *dlk,* double *nns,* double *s8*)

Set linear power spectrum.

Initializes the linear power spectrum at $a = 1$. This function does different things depending on the value of `fname`:

- If `fname` is "BBKS", the power spectrum will be calculated using the BBKS transfer function in the interval $\mathtt{lkmn} < \log_1 0(k) < \mathtt{lkmx}$ in intervals of $\Delta \log_1 0(k) = \mathtt{dlk}$.
- If `fname` is "EH", the power spectrum will be calculated using the Eisenstein & Hu transfer function.
- If `fname` is "EH_smooth" the power spectrum will be calculated from the Eisenstein & Hu transfer function **without acoustic oscillations** .
- Finally, `fname` can be set to the path to an ASCII file containing the power spectrum. This file must be in CAMB format, i.e.: two columns $(k, P(k))$ with k in units of $h/$Mpc and evenly spaced in $\log_1 0(k)$.

Once the $P(k)$ is read (or calculated) it is normalized to $\sigma_8 = \mathtt{s8}$. If `s8` is a negative number, the normalization of the power spectrum is preserved (note that this only makes sense for power spectra read from a CAMB file, since we only use the BBKS of EH transfer functions with the other options). After this is done, a spline is used for fast interpolation. The normalization for $P(k)$ used here is such that

$$\langle \delta(\mathbf{x})\delta(\mathbf{x}+\mathbf{r})\rangle = \frac{1}{2\pi^2}\int_0^\infty P(k)\,\frac{\sin(kr)}{kr}k^2\,dk.$$

4.1.2.27 void csm_set_nonlinear_pk (**Csm_params** ∗ *par,* char ∗ *fnamePkHFIT*)

Set non-linear power spectrum.

Initializes the non-linear power spectrum at $a = 1$. Three options are available depending on the value of `fnamePkHFIT` function does different things depending on the value of `fname`:

- If `fname` is "RPT", the mildly non-linear power spectrum is approximated by including a Gaussian damping term arising in renormalized perturbation theory:

  $$P(k,z) = P_L(k,z)\,e^{-k^2\sigma_v^2(z)},$$

  where

  $$\sigma_v^2(z) = \frac{1}{6\pi^2}\int_0^\infty P_L(k,z)\,dk.$$

  Here $P_L(k,z)$ is the linear power spectrum (see csm_Pk_linear_0). Thus, in this case $\sigma_v^2(z=0)$ is calculated and used as above whenever csm_Pk_nonlinear.
- If `fnamePkHFIT` is "RPT_ss", the Gaussian damping factor described above is used only on the oscillatory part of the power spectrum:

  $$P(k,z) = \left[P_L(k,z) - P_L^{\rm no\,BAO}(k,z)\right]e^{-k^2\sigma_v^2(z)} + P_L^{\rm no\,BAO}(k,z),$$

  where the no-BAO power spectrum is obtained using the Eisenstein & Hu fitting formula.
- Finally, `fnamePkHFIT` may be set to the path to a file containing the $z = 0$ non-linear power spectrum (e.g.: using HALOFIT). The format for this file must be the same as the one used in csm_set_linear_pk. Note that in this case there is no way to normalize the power spectrum to the value of σ_8 used for the linear case. Therefore the user must make sure that the files for both the linear and non-linear spectra were generated using the same normalization.

4.1.2.28 void csm_set_Pk_params (Csm_params ∗ *par,* double *beta,* double *gf,* double *bias,* int *l_max*)

Set power spectrum params.

Sets the parameters necessary to calculate the full redshift-space power spectrum: $\beta(a) = \mathtt{beta}$, $D(a) = \mathtt{gf}$ and $b = \mathtt{bias}$ (see csm_Pk_full). `l_max` is the maximum multipole that will be used in the calculation of the power spectrum and 3D correlation function (e.g. 4 for the Kaiser approximation).

4.1.2.29 void csm_set_verbosity (int *verb*)

Sets verbosity level.

Determines the amount of information to be output. Only two values for `verb` are supported: 0 (nothing) and 1 (everything). The default verbosity level is 1 (all messages are shown).

4.1.2.30 void csm_set_xi_multipole_splines (Csm_params ∗ *par*)

Set multipole splines.

If the correlation function must be calculated repeatedly, it may be faster to calculate first the multipole once for a set of values of r and interpolate between these afterwards. This function initializes a set of spline objects that are used thereafter when calling csm_xi_multipole (directly or indirectly). Specifically, a logarithmic-spaced spline is used for $0.1 < rh/\mathrm{Mpc} < 15$, and a linear-spaced spline is used for $15 < rh/\mathrm{Mpc} < 500$. Hence subsequent calls to this function will not calculate the integral in csm_xi_multipole, but will perform a much faster interpolation. If this function is call for $r > 500\,\mathrm{Mpc}/h$, csm_xi_multipole will return 0 and for $r < 0.1\,\mathrm{Mpc}/h$ it will return the value at $0.1\,\mathrm{Mpc}/h$.

4.1.2.31 double csm_sig0_L (Csm_params ∗ *par,* double *R1,* double *R2,* char ∗ *wf1,* char ∗ *wf2*)

Linear covariance.

Returns the covariance of the linear density field smoothed over scales `R1` and `R2` (i.e.: it is equivalent to calling csm_xi2p_L with r = 0).

4.1.2.32 double csm_theta_BAO (Csm_params ∗ *par,* double *aa*)

Angular BAO scale.

Returns the angular scale of the BAO (in deg) at $a = \mathtt{aa}$ as

$$\theta_{\mathrm{BAO}}(a) = \frac{r_s}{r(a)},$$

where r_s is the sound horizon scale estimated through the Eiseinstein & Hu fitting formula and $r(a)$ is the curvature comoving distance calculated with csm_curvature_comoving_distance.

4.1.2.33 void csm_unset_gsl_eh (void)

Unset GSL error handler.

Disables the default GSL error handler. The user will then be notified if any errors or warnings are found regarding the GSL functions (e.g.: some integral was not able to reach the required precision). These warnings will often be unimportant, and the code will not exit.

4.1.2.34 void csm_unset_xi_multipole_splines (Csm_params ∗ *par*)

Unset multipole splines.

Undoes all the operations in csm_set_xi_multipole_splines, freeing up the allocated memory. It is not necessary to call this function unless the splines need to be reinitialized, since it is implicitly called by csm_params_free.

4.1.2.35 double csm_xi2p_L (Csm_params * *par,* double *r,* double *R1,* double *R2,* char * *wf1,* char * *wf2,* double *errfac*)

Linear correlation function.

Let $\delta(\mathbf{x},R,T)$ be the density contrast smoothed over a scale R with window function T. This function returns the correlation function

$$\langle\delta(\mathbf{x},\mathtt{R1},\mathtt{wf1})\delta(\mathbf{x}+\mathbf{r},\mathtt{R2},\mathtt{wf2})\rangle.$$

The possible values for `wf1` and `wf2` are "TopHat" and "Gauss":

$$W_{\rm TH}(x)=3\,\frac{\sin x-x\,\cos x}{x^3},\qquad W_{\rm G}(x)=\exp(-x^2/2).$$

For some values of the parameters it may be impossible for the GSL integrator to obtain the required accuracy, in which case the error tolerance can be scaled by the argument `errfac`. For most cases the default tolerance (`errfac` = 1) is OK.

4.1.2.36 double csm_xi_3D (Csm_params * *par,* double *rr,* double *mu*)

3D correlation function (polar coords)

Returns the anisotropic 3D correlation function as a sum over multipoles

$$\xi(r,\mu)=\sum_{l=0}^{\infty}\xi_l(r)\,L_l(\mu).$$

Note that under the Kaiser approximation used by CosmoMAD in its present version only the first three multipoles ($l=0,2,4$) are used. When may calls to this function are necessary it may be wise to call csm_set_xi_multipole_splines first for a better performance.

4.1.2.37 double csm_xi_multipole (Csm_params * *par,* double *rr,* int *l*)

Correlation function multipole.

Returns the l-th multipole of the redshift-space correlation function through the integral

$$\xi_l(r)=\frac{i^l}{2\pi^2}\int_0^\infty P_l(k)\,j_l(kr),$$

where $P_l(k)$ is the l-th power spectrum multipole (as returned by csm_Pk_multipole). The first time this function is called a spline is created for each power spectrum multipole in order to accelerate the calculation of the integral above.

4.1.2.38 double csm_xi_pi_sigma (Csm_params * *par,* double *pi,* double *sigma,* int *use_multipoles*)

3D correlation function (orthogonal coords)

Returns the anisotropic 3D correlation function using longitudinal ($\pi=r\mu$) and transverse ($\sigma=r\sqrt{1-\mu^2}$) coordinates. If `use_multipoles` is set to 1, the sum over multipoles described ind csm_xi_3D is used. If set to 0 the following double integral is performed:

$$\xi(\pi,\sigma)=\frac{1}{2\pi^2}\int_0^\infty dk_\parallel\,\cos(k_\parallel\pi)\int_0^\infty dk_\perp\,k_\perp\,J_0(k_\perp\sigma)P(k_\parallel,k_\perp),$$

where $J_0(x)$ is the 0-th order cylindrical Bessel function. Note that the latter approach, although exact, will be much slower than the former, unless a large number of multipoles is needed.