LSST DESC Notes

Core Cosmology Library: Precision Cosmological Predictions for LSST

Husni Almoubayyed,[1] David Alonso,[2] Jonathan Blazek,[3,4] Philip Bull,[5,6] Jean-Éric Campagne,[7] N. Elisa Chisari,[2] Alex Drlica-Wagner,[8] Zilong Du,[9] Tim Eifler,[10,11] John Ellison,[9] Cristhian Garcia Quintero,[12] Renée Hlozek,[13] Mustapha Ishak,[12] Shahab Joudaki,[2] Matthew Kirby,[14] David Kirkby,[15] Elisabeth Krause,[10,16] Francois Lanusse,[1] C. Danielle Leonard,[1] Christiane S. Lorenz,[2] Phil Marshall,[17] Thomas McClintock,[14] Sean McLaughlin,[18] Alexander Mead,[19] Jérémy Neveu,[7] Stéphane Plaszczynski,[7] Javier Sanchez,[15] Sukhdeep Singh,[1] Anže Slosar,[20] Tilman Tröster,[21] Antonio Villarreal,[22] Michal Vrastil,[23] and Joe Zuntz[21] (LSST Dark Energy Science Collaboration)

[1] *McWilliams Center for Cosmology, Department of Physics, Carnegie Mellon University, Pittsburgh, PA 15213, USA*
[2] *Department of Physics, University of Oxford, Denys Wilkinson Building, Keble Road, Oxford OX1 3RH, United Kingdom*
[3] *Center for Cosmology and Astroparticle Physics, Ohio State, Columbus, OH 43210, USA*
[4] *Laboratory of Astrophysics, École Polytechnique Fédérale de Lausanne (EPFL), Observatoire de Sauverny, 1290 Versoix, Switzerland*
[5] *California Institute of Technology, Pasadena, CA 91125, USA*
[6] *Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California, USA*
[7] *Laboratoire de l'Accélérateur Linéaire, Université Paris-Sud, CNRS/IN2P3, Université Paris-Saclay, Orsay, France*
[8] *Fermi National Accelerator Laboratory, P. O. Box 500, Batavia, IL 60510, USA*
[9] *Department of Physics and Astronomy, University of California, Riverside, CA 92521, USA*
[10] *Steward Observatory/Department of Astronomy, University of Arizona, 933 North Cherry Avenue, Tucson, AZ 85721, USA*
[11] *Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA*
[12] *Department of Physics, The University of Texas at Dallas, Richardson, TX 75083, USA*
[13] *Dunlap Institute for Astronomy and Astrophysics & Department for Astronomy and Astrophysics, University of Toronto, ON M5S 3H4*
[14] *University of Arizona, Tucson, AZ 85721, USA*
[15] *Department of Physics and Astronomy, University of California, Irvine, CA 92697, USA*
[16] *Kavli Institute for Particle Astrophysics and Cosmology, Stanford, CA 94305-4085, USA*
[17] *SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA*
[18] *Stanford University, Stanford, CA, 94305, USA*
[19] *Department of Physics and Astronomy, University of British Columbia, 6224 Agricultural Road, Vancouver, BC V6T 1Z1, Canada*
[20] *Brookhaven National Laboratory, Physics Department, Upton, NY 11973, USA*
[21] *Institute for Astronomy, Royal Observatory Edinburgh, Edinburgh EH9 3HJ, UK*
[22] *Department of Physics and Astronomy, University of Pittsburgh, Pittsburgh PA 15260*
[23] *Institute of Physics CAS, Prague, 182 21, CZ*

The Core Cosmology Library (CCL) provides routines to compute basic cosmological observables with validated numerical accuracy. These routines have been validated to an accuracy level, documented here, against the results of independent implementations. In the current version, predictions are provided for distances and background quantities, angular auto- and cross-spectra of cosmic shear, galaxy-galaxy lensing, intrinsic alignments and clustering, halo bias and the halo mass function. CCL uses different schemes to obtain the matter power spectrum, including analytical, phenomenological and other schemes calibrated through sim-

ulations. CCL is written in C, with a Python interface. In this note, we explain the functionality of the CCL library.

Contents

1. Introduction

In preparation for constraining cosmology with the Large Synoptic Survey Telescope (LSST), it is necessary to be able to produce rigorous theoretical predictions for the cosmological quantities that will be measured. The Core Cosmology Library[1] (CCL) aims to provide, in one library, a way of making predictions that are validated to a well-documented numerical accuracy, for the purpose of constraining cosmology with LSST. By constructing a cosmology library specifically with LSST in mind, it is possible to ensure that it is flexible, adaptable, and validated for all cases of interest, as well as user-friendly and appropriate for the needs of all working groups. This note describes the underlying equations and conventions of the CCL library. See the GitHub repository for installation and usage instructions.

2. Functionality

2.1. Physical constants

We have performed a comparison of the physical constants used in CCL and included dependencies and external sources. See Table 1 for absolute fractional differences of the constants between these sources. Our final choice of constants for CCL mainly relies on CODATA 2014 (Mohr et al. 2016) in as much as possible, except for $\mathrm{M}_{\odot}$, where we adopt the IAU 2015 value (Mamajek et al. 2015), and for the conversion between parsec and meters, where we take the PDG 2013[2] value. Notice that NIST[3] adopts the CODATA 2014 values.

Notice there are some inconsistencies with the constants adopted by CLASS. This includes the value of the gravitational constant, the Boltzmann constant, the Planck constant, the speed of light, and the electron charge. Also, the value of ρ_c is derived from other constants, while PDG 2013 fixes it to a given value (this is the reason there is only one entry for that column).

After comparison between the physical constants used in CCL and those of the sources mentioned above, we have found better than 0.01% agreement for all constants of interest except for the gravitational constant and the value of the solar mass.

[1] https://github.com/LSSTDESC/CCL
[2] http://pdg.lbl.gov/2013/
[3] https://physics.nist.gov/cuu/Constants/index.html

Table 1. Absolute fractional differences between different constants as tabulated in the sources listed below. Entries marked with zero indicate that this is the value adopted by CCL.

	G_{Newt}	k_b	σ_{SB}	h	c	eV	ρ_c	$M_\odot$	pc
PDG 2013	3e-05	2.1e-07	1.1e-06	7e-08	0.0e+00	3.5e-08	8.8e-10	2.2e-05	0.0e+00
NIST	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	–	–	–
GSL 2.4	1.6e-04	1.1e-06	5.8e-06	6e-09	1.5e-06	2.1e-06	–	2.2e-04	7.8e-07
CLASS	3.0e-05	1.4e-06	–	1.6e-07	0.0e+00	8.4e-08	–	–	1.2e-09
CODATA 2014	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	–	–	–
IAU 2015	0.0e+00	–	–	–	–	–	–	0.0e+00	–

2.2. **Supported cosmological models**

Ultimately, CCL plans to incorporate theoretical predictions for all cosmological models of interest to LSST. Currently, the following families of models are supported:

- Flat ΛCDM
- wCDM and the CPL model ($w_0 + w_a$, Chevallier & Polarski 2001 and Linder 2003)
- Non-zero curvature (K)
- All of the above, plus an arbitrary, user-defined modified growth function (see description in Section 2.7)
- Massive neutrinos, in combination with any of the above.

CCL also provides support for modeling the impact of baryons on the matter power spectrum, as described in Sec. 2.8.7. Not all features of CCL are available for all models. For a guide to which predictions are available for each model, see Table 2. Note that if users install their own version of CLASS, CCL can then make predictions for a more extended set of cosmologies. Users should take care to understand the validity of the CCL assumptions for their own models.

In its default configuration, CCL adopts the nonlinear matter power spectrum from CLASS through the Halofit implementation and the Tinker mass function for cluster number counts.

2.3. **Creating a cosmology**

Table 2. Cosmologies implemented in CCL and observables supported in each of them. Note that the only reason why angular power spectra appear not to be supported in non-flat cosmologies is that the hyperspherical Bessel functions are currently not implemented, although their impact is fairly limited. Likewise, number counts power spectra are strictly not supported in the presence of massive neutrino cosmologies due to the scale-dependent growth rate that affects the redshift-space distortions term, even though the impact of this is also small for wide tomographic bins. The halo model can make matter power spectrum predictions for all cosmologies except those with μ/Σ modified gravity, but should not be used for massive neutrino models because the current version does not distinguish between the cold matter, relevant for clustering, and all matter. Finally, notice that in addition to the support for the μ/Σ parameterisation of modified gravity, CCL can make predictions for the growth of perturbations for some modified gravity models through a user defined $\Delta f(a)$, and that other extensions are supported via integration of external modified gravity codes.

Observable/Model	flat ΛCDM	ΛCDM+K	ΛCDM + m_ν	wCDM	ΛCDM + μ/Σ MG
Distances	✓	✓	✓	✓	✓
Growth	✓	✓	X	✓	✓
$P_m(k,z)$	✓	✓	✓	✓	✓
Halo Mass Function	✓	✓	X	✓	✓
C_l, number counts	✓	X	X	✓	✓
C_l, weak/CMB lensing	✓	X	✓	✓	✓
Correlation function	✓	X	✓	✓	✓
Halo model	✓	✓	X	✓	X

To use CCL, the first step is to create a `ccl_cosmology` structure, containing all of the information required to compute cosmological observables. A `ccl_cosmology` structure can be generated using the information from a `ccl_parameters` object and a `ccl_configuration` object.

`ccl_parameters` objects contain information about the cosmological parameters of a given model, and are initialized using one of the following routines (the full syntax for each function can be found in the header file `ccl_core.h`):

- `ccl_parameters_create(double Omega_c, double Omega_b, double Omega_k, double N_eff, double *mnu, ccl_mnu_type_label mnu_type, double w0, double wa, double h, double norm_pk, double n_s, int nz_mgrowth,double *zarr_mgrowth,double *dfarr_mgrowth, int *status)`: general `ccl_parameters` constructor supporting all the models described above.

- `ccl_parameters_create_flat_lcdm(...)`: particular constructor for flat ΛCDM cosmologies.

The argument `norm_pk` can be used to specify the power spectrum normalization in terms of either σ_8 or A_s. The `ccl_parameters_create` functions assume σ_8 normalization if `norm_pk` $>$ `1.0e` $-$ `5` and A_s normalization otherwise.

The arguments `mu_0` (μ_0) and `Sigma_0` (Σ_0) are the parameters governing the amplitude of modifications to the cosmological Poisson equation for massive and massless particles respectively (see sections 2.7 and 2.9 below for more details). We currently assume the functional forms (Ferreira & Skordis 2010)

$$\mu(z) = \mu_0 \frac{\Omega_\Lambda(z)}{\Omega_\Lambda(z=0)}\,, \quad \Sigma(z) = \Sigma_0 \frac{\Omega_\Lambda(z)}{\Omega_\Lambda(z=0)} \tag{1}$$

but expect to introduce functionality for a broader range of functional forms in time.

`ccl_configuration` objects contain information about the prescriptions to be used to compute transfer functions, power spectra, mass functions, etc. A default `ccl_configuration` object is made available as `default_config`, which specifies that CLASS will be used to compute transfer functions, HaloFit will be used to calculate the matter power spectrum, there will be no impact of baryons included and the Tinker 2010 prescription will be used to compute the halo mass function.

After initializing instances of `ccl_parameters` and `ccl_configuration`, use the function `ccl_cosmology_create(ccl_parameters, ccl_configuration)` to return a pointer to a `ccl_cosmology` structure. You will need to pass this pointer around to every CCL function.

An example of using `CCL` is provided in Section **??**. The `README` file has additional extensive documentation for the example run, as well as installation.

Creating a cosmology with massive neutrinos—In the parameter construction routines above, `mnu` is a pointer to either a single value, to be interpreted as the sum of neutrino masses, or to an array containing the three neutrino masses, both in eV. `mnu_type` is a label which indicates whether `mnu` points to sum of the masses or to an array of three mass values. In the former case, `mnu_type` also defines which convention should be used to split the sum into three neutrino masses for calculations (options are normal hierarchy, inverted hierarchy, or an equal split).

To set up a cosmology with massive neutrinos, the user should pass `mnu`, `mnu_type`, and `N_eff` to the parameter construction routine. When working from the python wrapper, it suffices to pass `mnu` and `N_eff`; `CCL` will infer whether `mnu` is a set of masses or a sum. In the latter case, the default behaviour is to split the sum into

masses consistent with the normal hierarchy, but an inverted hierarchy or equal splitting can be requested by passing `mnu_type`.

In the case where `mnu` corresponds to a sum, we first allocate the three neutrino masses according to the requested convention. The default convention is the normal hierarchy, but users may also request either an inverted hierarchy or a split of the sum into three equal masses. In the latter case it is trivial to compute the three masses. If splitting with respect to the normal or inverted hierarchy is desired, the mass calculation of the three masses is only marginally more complicated. The relevant known quantity which has been determined via particle physics experiments is the square of the difference of neutrino masses (up to a sign for one of the differences, hence the two possible hierarchies, see Lesgourgues & Pastor 2012; Lattanzi & Gerbino 2017). Because we know the square of the differences rather than the differences themselves, we must solve a set of quadratic equations for the neutrino masses. This is accomplished via a simple implementation of Newton's method, which converges to within machine precision in a few iterations.

Having then a set of three masses regardless of `mnu_type`, we check for which of the three masses the corresponding neutrino species is non-relativistic today ($m_\nu > 0.00017$, Lesgourgues & Pastor 2012), and obtain a number of massive neutrinos `N_nu_mass`. We use this along with the `N_eff` value to set the number of relativistic neutrinos species `N_nu_rel`. We must be slightly careful in doing so, as for massive neutrinos only we follow `CLASS` in modifying the relationship between the temperature of the CMB and the neutrino temperature:

$$T_\nu^{\rm eff} = T_{\rm CMB} T_{\rm NCDM} \tag{2}$$

where the above defines $T_{\rm NCDM}$, an adhoc modification to the equality between $T_{\rm CMB}$ and $T_\nu^{\rm eff}$. We follow the nomenclature of `CLASS` here, but we emphasize that $T_{\rm NCDM}$ is a dimensionless scaling factor, not a temperature. Setting $T_{\rm NCDM} = 0.71611$ ensures that $m_\nu/\Omega_\nu^0 = 93.14{\rm eV}$, in agreement with second-order theoretical calculations which correctly take into account QED effects and electron / positron annihilation (Mangano et al. 2005). Therefore to get `N_nu_rel` consistent with the `N_eff` passed by the user, we do:

$$\texttt{N\_nu\_rel} = \texttt{N\_eff} - (T_{\rm NCDM})^4 \left(\frac{4}{11}\right)^{-\frac{4}{3}} \texttt{N\_nu\_mass}. \tag{3}$$

For easier initiation of cosmologies with massive neutrinos, the suffix '`_nu`' may be appended to the last four `ccl_parameters_create` functions in the previous section. Using these functions without the `_nu` suffix will set the neutrino masses to 0 and the effective number of neutrino species to 3.046.

It may sometimes be preferable or necessarily to specify a cosmology in terms of Ω_ν^0 for massive neutrinos instead of m_ν. To facilitate this, CCL includes a convenience function ccl_nu_masses, which takes as input Ω_ν^0 for massive neutrinos, the temperature of the CMB, and a label specifying how the neutrino mass should be split amongst species similarly to above. It then outputs a pointer to the resulting neutrino mass(es).

2.4. **Distances**

The routines described in this subsection are implemented in ccl_background.c.

The Hubble parameter is calculated as

$$\frac{H(a)}{H_0} = a^{-3/2}\Big(\Omega_{M,0} + \Omega_{\Lambda,0}a^{-3(w_0+w_a)}\exp[3w_a(a-1)] + \Omega_{K,0}a + (\Omega_{g,0} + \Omega_{\nu,\mathrm{rel}})a^{-1} + \Omega_{\nu,\mathrm{m}}(a)a^3\Big)^{\frac{1}{2}}. \tag{4}$$

The radial comoving distance is calculated via a numerical integral,

$$\chi(a) = c\int_a^1 \frac{da'}{a'^2 H(a')}. \tag{5}$$

The transverse comoving distance is computed in terms of the radial comoving distance as:

$$r(\chi) = \begin{cases} k^{-1/2}\sin(k^{1/2}\chi) & k > 0 \\ \chi & k = 0 \\ |k|^{-1/2}\sinh(|k|^{1/2}\chi) & k < 0 \end{cases} \tag{6}$$

The usual angular diameter distance is $d_A = a\,r(a)$, and the luminosity distance is $d_L = r(a)/a$.

CCL can also compute the distance modulus, defined as,

$$\mu = 5\log_{10}(d_L/\mathrm{pc}) - 5 \tag{7}$$

and $a(\chi)$, the inverse of $\chi(a)$.

2.5. **Density parameter functions**

The routines described in this subsection are implemented in `ccl_background.c`.

The density parameter functions $\Omega_X(a)$ can be calculated for six components:

- matter density parameter $\Omega_M(a) = \Omega_{M,0}H_0^2/(a^3H^2(a))$,
- dark energy density parameter $\Omega_\Lambda(a) = \Omega_{\Lambda,0}a^{-3(1+w_0+w_a)}\exp[3w_a(a-1)]H_0^2/H^2(a)$,
- radiation density parameter $\Omega_g(a) = \Omega_{g,0}H_0^2/(a^4H^2(a))$,
- curvature density parameter $\Omega_K(a) = \Omega_{K,0}H_0^2/(a^2H^2(a))$,
- massless neutrino density parameter $\Omega_{\nu,\mathrm{rel}}(a) = \Omega_{\nu,\mathrm{rel},0}H_0^2/(a^4H^2(a))$,
- massive neutrino density parameter $\Omega_{\nu,\mathrm{m}}(a)$,

all using the Hubble parameter defined in equation 4.

For massive neutrinos, $\Omega_{\nu,\mathrm{m}}(a)$ is calculated by calling a set of functions contained in `ccl_neutrinos.c`. For each species of massive neutrino with mass m_ν^i, we define

$$\tilde{m}^i = \frac{m_\nu^i a}{T_\nu^{\mathrm{eff}}} \tag{8}$$

in units such that $\tilde{m}$ is dimensionless. We conduct once the phase-space integral required to get the neutrino density (the integral over x in equation 9 below), and store the result as a `gsl` spline for subsequent access. Given then the value of this integral, we multiply by appropriate factors to obtain $\Omega_{\nu,\mathrm{m}}(a)$:

$$\Omega_{\nu,\mathrm{m}}(a) = \sum_{i=1}^{N_\nu} \frac{8\pi^2(\pi k_b)^3 k_b}{15(ch_\mathrm{P})^3}\frac{8\pi G}{3h^2c^2}\left(\frac{T_\nu^{\mathrm{eff}}}{a}\right)^4\left(\frac{7}{8}\int_0^{x_{\mathrm{max}}} dx\, x^2\frac{\sqrt{x^2+(\tilde{m}^i)^2}}{\exp(x)+1}\right) \tag{9}$$

where h_P is Planck's constant and h is $H_0/100$ with H_0 in units of km / s / Mpc. x_{max} is set to 1000. The final bracketed term which includes the phase-space integral can be simplified in the limit where $\tilde{m}$ is very large or very small: for small $\tilde{m}$, it is set to $\frac{7}{8}$, and for large $\tilde{m}$, it becomes $\frac{5\zeta(3)}{18\pi^4}\tilde{m} \sim 0.2776\mu$.

2.6. **Functions of the physical density**

The routines described in this subsection are implemented in `ccl_background.c`.

The physical density $\rho_X(a)$ can be calculated for seven components:

- critical density $\rho_{\rm crit}(a) = \frac{3H^2(a)}{8\pi G} = \rho_{\rm crit,0} H^2(a)/H_0^2$,
- matter density $\rho_M(a) = \rho_{\rm crit}(a)\Omega_M(a) = \rho_{\rm crit,0}\Omega_{M,0}/a^3$,
- dark energy density parameter $\rho_\Lambda(a) = \rho_{\rm crit,0}\Omega_{\Lambda,0} a^{-3(1+w_0+w_a)} \exp[3w_a(a-1)]$,
- radiation density parameter $\rho_g(a) = \rho_{\rm crit,0}\Omega_{g,0}/a^4$,
- curvature density parameter $\rho_K(a) = \rho_{\rm crit,0}\Omega_{K,0}/a^2$,
- massless neutrino density parameter $\rho_{\nu,\rm rel}(a) = \rho_{\rm crit,0}\Omega_{\nu,\rm rel,0}/a^4$,
- massive neutrino density parameter $\Omega_{\nu,\rm m}(a) = \rho_{\rm crit,0}\Omega_{\nu,\rm m}(a) H^2(a)/H_0^2$,

where $\Omega_{\nu,\rm m}(a)$ is given by equation 9 and the Hubble parameter by equation 4. CCL moreover allows for comoving physical densities $\rho_{X,\rm comoving}(a) = \rho_X(a)a^3$.

2.7. **Growth function**

The routines described in this subsection are implemented in `ccl_background.c`. To compute the growth function, $D(a)$, the growth factor of matter perturbations, CCL solves the following differential equation:

$$\frac{d}{da}\left(a^3 H(a)\frac{dD}{da}\right) = \frac{3}{2}\Omega_M(a) a H(a) D, \tag{10}$$

using a Runge-Kutta Cash-Karp algorithm.

In doing this, CCL simultaneously computes the growth rate $f(a)$, defined as:

$$f(a) = \frac{d\ln D}{d\ln a}. \tag{11}$$

CCL provides different functions that return the growth normalized to $D(a=1) = 1$ and to $D(a \ll 1) \to a$.

Note that the above is strictly valid for a Universe containing only dust-like matter components. A scale-independent growth rate is, for example, ill-defined in the presence of massive neutrinos; therefore CCL will raise an error if the user attempts to calculate the growth rate or growth factor in a cosmology with massive neutrinos.

Currently, CCL allows for two version of alternative 'modified gravity' cosmological models. The first is defined by a regular background $(w_0 + w_a)$CDM (with arbitrary K) as well as a user-defined $\Delta f(a)$, such that the true growth rate in this model is

given by $f(a) = f_0(a) + \Delta f(a)$, where $f_0(a)$ is the growth rate in the background model. Note that this implementation of 'modified gravity' is only consistently implemented with regards to the computation of the linear growth factor and growth rates (which will also scale the linear power spectrum). All other CCL functions (including the non-linear power spectrum) will ignore these modifications. This model, and the interpretation of the predictions given by CCL, should therefore be used with care.

The second model for deviations from General Relativity supported by CCL is the quasistatic parameterization, with parameterizing functions $\mu(a)$ (the change to the Poisson equation for massive particles) and $\Sigma(a)$ (the change to the same for massless particles), with functional form assumed to be given as in equation 1. The background is once again allowed to be defined by $(w_0 + w_a)$CDM (with arbitrary K).

The growth factor and growth rate are altered when mu_0 $\neq 0$, with the above equation becoming

$$\frac{d}{da}\left(a^3 H(a)\frac{dD}{da}\right) = \frac{3}{2}\Omega_M(a) a H(a)(1+\mu(a))D. \tag{12}$$

As usual, the resulting growth factor can be returned normalized to $D(a=1)=1$ or to $D(a \ll 1) \to a$. The second normalization is used in returning the matter power spectrum with appropriate modification in this model, as discussed in section 2.8.

2.8. **Matter power spectrum**

There are several options for obtaining the linear and non-linear matter power spectrum in CCL. We parameterize the linear matter power spectrum via the transfer function via the relationship $P(k) = 2\pi^2\Delta^2(k)/k^3 \propto T^2(k)k^{n_s}$, where $\Delta(k)$ is the dimensionless power spectrum and n_s is the spectral index. The normalization of the power spectrum is defined at $z = 0$ by setting σ_8 to its value today or by setting the initial value A_s depending on the option. The non-linear matter power spectrum options either use the linear matter power spectrum as inputs (e.g., the halo model) or supply their own values (e.g., an emulator). Both power spectra are interpolated in a two-dimensional table of scale factor and redshift for later use. The routines described in this subsection are implemented in ccl_power.c and related files for each specific model.

2.8.1. **BBKS**

CCL implements the analytical BBKS approximation to the transfer function (Bardeen et al. 1986), given by

$$T(q \equiv k/\Gamma h\mathrm{Mpc}^{-1}) = \frac{\ln[1+2.34q]}{2.34q}[1+3.89q+(16.2q)^2+(5.47q)^3+(6.71q)^4]^{-0.25} \tag{13}$$

where $\Gamma = \Omega_m h$.

The BBKS power spectrum option is primarily used as a precisely-defined input for testing the numerical accuracy of CCL routines (as described in Section 3), and it is not recommended for other uses.

2.8.2. **Eisenstein and Hu**

CCL also provides an approximation to the transfer function as implemented by Eisenstein & Hu (1998) (E&H; we refer the reader to this paper for a detailed discussion of the fitting formulae).[4]

The Eisenstein & Hu and BBKS approximations are not very accurate (generally only to within a few percent), and so should not be used to derive precise cosmological constraints. They are, however, computationally faster, and therefore useful for code testing and comparison.

2.8.3. **CLASS**

The fiducial configuration calls the CLASS software (Blas et al. 2011) within CCL to obtain the linear matter power spectrum at given redshift. On initializing the cosmology object, we construct a bi-dimensional spline in k and the scale-factor which is then evaluated by the relevant routines to obtain the matter power spectrum at the desired wavenumber and redshift. The relevant routines can be found within `ccl_class.c`.

As discussed in Section **??**, the user can compile CCL with an external version of CLASS as well.

2.8.4. **Halofit**

[4] Note that the implementation in CCL modifies Eq. 5 of Eisenstein & Hu (1998) using $a^{-1} = 1+z$ instead of the approximation $a^{-1} \sim z$. The difference in the resulting power spectra is negligible, but larger than 1 part in 10^4 for $k < 10\,h\,\mathrm{Mpc}^{-1}$.

We provide a Halofit implementation that applies a correction to any input linear matter power spectrum to compute an approximation to the non-linear matter power spectrum. We use the model from Takahashi et al. (2012) which is accruate to roughly 10%.

2.8.5. **Halo Model**

We also provide a halo model implementation for computing the non-linear matter power spectrum. Halo model computations are known to be innacurate at the 10% level or more. See Section 2.12 for details.

2.8.6. **Cosmic emulator**

The cosmic emulator of Lawrence et al. (2017) is integrated into `CCL` and it is available as one of the non-linear matter power spectrum options. The cosmic emulator provides predictions for the non-linear matter power spectrum based on an interpolation over simulation outputs spanning a defined set of cosmological parameter values.

The emulator provides accurate predictions for the nonlinear matter power spectrum, at the 1% level, for $z \leq 2$ and in the wavenumber range $k = [10^{-3}, 5]$ Mpc^{-1}. If a redshift above $z = 2$ is passed, the emulator will quit and return an error message. For k values below and above the previously specified range, we extrapolate in the manner specified in the following section.

The allowed range of cosmological parameters is as follows:

$$\begin{aligned} 0.12 \leq \Omega_m h^2 \leq 0.155, \\ 0.0215 \leq \Omega_b h^2 \leq 0.0235, \\ 0.7 \leq \sigma_8 \leq 0.9, \\ 0.55 \leq h \leq 0.85, \\ 0.85 \leq n_s \leq 1.05, \\ -1.3 \leq w_0 \leq -0.7, \\ -1.73 \leq w_a \leq -0.7, \\ 0.0 \leq \Omega_\nu h^2 \leq 0.01. \end{aligned} \tag{14}$$

Actually, w_a and w_0 are constrained jointly to be $0.3 \leq (-w_0 - w_a)^{1/4}$. Note that `CCL` only allows a subregion within this parameter space. For models in which

w(z) crosses -1 at some given redshift, CLASS will crash because this value corresponds to a true cosmological constant, which by definition should have no perturbations.[5]

Neutrino species—The emulator is set up to consider $N_{\rm eff} = 3.04$ and to allow the user to provide $\Omega_\nu h^2$ in order to set the neutrino mass, where it is assumed that the corresponding mass is split equally amongst three neutrinos. This is different from the standard CCL configuration, which takes as input the mass(es) of neutrinos in eV, m_ν. The assumption of an equal split of masses amongst three neutrino species is also different from the default CCL choice to split the mass amongst species according to the normal hierarchy. For best compatibility with the emulator, we therefore include a ccl_configuration element emulator_neutrinos_method, which defaults to ccl_strict but can be set to ccl_equalize if desired:

- ccl_strict requires the user to pass either a set of three equal neutrino masses, or to pass a sum of masses and to set ccl_mnu_type such that the sum is split equally amongst species. If other options are selected when the emulator is in use, CCL will raise an error and exit. This default option ensures full self-consistency within quantities calculated with a single CCL cosmology.
- ccl_equalize allows the user to pass an unequal set of neutrino masses, or to pass a sum with ccl_mnu_type set such that the sum is not split equally amongst speices. In this case, CCL will pass redistributed equal neutrino masses to the emulator. This choice may result in internal inconsistency amongst different quantities calculated with the same CCL cosmology, or indeed even internal inconsistency between the linear and nonlinear parts of the same power spectrum. This option should only be selected for convenience if you are sure you don't care about the mass splitting between neutrino speices.

Note also that because the emulator is constructed with $N_{\rm eff} = 3.04$, the user is required to pass this when the emulator is in use. If this is not the case, CCL will quit and issue an error.

Notice that ideally we would like to pass a non-integer number of massive neutrino species for best compatibility with the emulator set-up. However, since this is not possible within CCL, we have opted to verify that neither the growth function nor the comoving radial distance computation are affected by this approximation to more than 10^{-4} in the range $0.01 < a < 1$, where a is the scale fac-

[5] We thank Emilio Bellini and Miguel Zumalacárregui for clarifying this for us.

tor. We have verified this by comparing this prediction for a fiducial cosmology $\{\Omega_c = 0.27, \Omega_b = 0.049, h = 0.67, \sigma_8 = 0.8, n_s = 0.96\}$ with the following neutrino parameters: $\{N_{\nu,\mathrm{rel}}, N_{\nu,\mathrm{mass}}, m_\nu\} = \{0.00641, 3, 0.06\mathrm{eV}\}$, $\{N_{\nu,\mathrm{rel}}, N_{\nu,\mathrm{mass}}, m_\nu\} = \{0, 3.04, 0.06\mathrm{eV}\}$ and $\{N_{\nu,\mathrm{rel}}, N_{\nu,\mathrm{mass}}, m_\nu\} = \{0, 3.046, 0.06\mathrm{eV}\}$. The discrepancy between distances and growth results between these choices of neutrino parameters can raise above 10^{-4} at $a < 0.01$ and the user should be mindful of this for their particular application.

2.8.7. **Impact of baryons**

`CCL` incorporates the impact of baryons on the total matter power spectrum via the "baryonic correction model" (BCM) of Schneider & Teyssier (2015). This is implemented through a flag in the configuration object, which currently accepts the following options: `ccl_nobaryons` or `ccl_bcm`. When `ccl_bcm` is set, the nonlinear matter power spectrum (whichever the method that provides it) is multiplied by a correction factor $F(k, z)$ which models the impact of baryons.

The main consequences of baryonic processes are: to suppress the power spectrum at intermediate scales ($k \sim$ a few $h/$Mpc) due to the ejection of gas by Active Galactic Nuclei feedback, and to enhance it at smaller scales due to enhanced cooling. Three effective parameters govern the contribution of baryonic processes to modifying the total matter power spectrum:

- $\log_{10}[M_c/(\mathrm{M}_\odot/h)]$: the mass of the clusters responsible for feedback, which regulates the amount of suppression of the matter power spectrum at intermediate scales;
- η_b: a dimensionless parameter which determines the scale at which suppression peaks;
- and k_s [$h/$Mpc]: the wavenumber that determines the scale of the stellar profile.

If the BCM parameters are not specified and the user sets `CCL` to compute the power spectrum with baryonic feedback included, `CCL` will assume the default parameters of Schneider & Teyssier (2015). The user may pass these explicitly if desired.

Other baryonic impact models might be incorporated in the future as they become available.

2.8.8. **Modified gravity (μ, Σ)**

CCL supports the quasistatic parameterization of modified gravity with scale-independent deviations from GR which arise at late times (currently with functional form given in equation 1). Under these conditions, the matter power spectrum in a modified gravity scenario can be computed simply by re-scaling the GR matter power spectrum with the modified gravity growth factor.

For each power spectrum calculation method above, we first compute the power spectrum in GR at a grid in a and k as usual, in preparation for producing splines in these variables. If mu_0 $\neq 0$, we then take the ratio of the growth factor D at the given value of mu_0 to the GR (mu_0 $= 0$) growth factor, and multiply the GR power spectrum by this ratio squared:

$$P(k,z) = \left(\frac{D^{\mu}(z)}{D^{\rm GR}(z)}\right)^2 P^{\rm GR}(k,z). \tag{15}$$

We then set up splines for the power spectrum as usual.

Note that the μ, Σ quasistatic parameterization of deviations from GR is technically only valid in the linear regime and at sub-horizon scales. We allow the user to do this re-scaling for the non-linear power spectrum, but issue a warning. Computing the matter power spectrum under this paramterisation of modified gravity will not produce physically meaningful results on very large or small scales.

2.8.9. **Extrapolation for the nonlinear power spectrum**

The computation of the power spectrum from CLASS can be significantly sped up by extrapolating in the range $k >$ K_MAX_SPLINE and $k <$ K_MIN_SPLINE. In this section, we describe the implementation of the extrapolation and the accuracy attained. These tests are performed in a flat ΛCDM cosmology with $\Omega_c = 0.25$, $\Omega_b = 0.05$, $A_s = 2.1 \times 10^{-9}$, $h = 0.7$ and $n_s = 0.96$.

We first describe the extrapolation at high wavenumbers. The introduction of the parameter K_MAX_SPLINE allows us to spline the matter power spectrum within the cosmo structure up to that value of k (in units of $1/$Mpc). A separate K_MAX parameter sets the limit for evaluation of the matter power spectrum. The range between K_MAX_SPLINE$< k <$K_MAX is evaluated by performing a second order Taylor expansion in $\ln k$ within the static routine ccl_power_extrapol_highk.

First, we compute the first and second derivative of the $\ln P(k,z)$ at $k_0 =$ K_MAX_SPLINE $- 2\Delta \ln k$ by computing the numerical derivatives by finite differences using GSL. The fiducial choice for $\Delta \ln k$ is 10^{-2}. We then apply a second order Taylor expansion to extrapolate the matter power spectrum to $k >$K_MAX_SPLINE. The Taylor expansion gives

$$\ln P(k,z) \simeq \ln P(k_0,z) + \frac{d \ln P}{d \ln k}(\ln k_0, z)(\ln k - \ln k_0) + \frac{1}{2}\frac{d^2 \ln P}{d \ln k^2}(\ln k_0, z)(\ln k - \ln k_0)^2. \tag{16}$$

The accuracy of this approximation is shown in Figure 1 for redshifts $z = 0$, $z = 3$ and $z = 20$. We compare the nonlinear matter power spectrum at these redshifts computed with the previously described approximation, to the matter power spectrum obtained by setting the power spectrum splines to high values. We find that for typical values of $\Delta \ln k = 10^{-2}$ and K_MAX_SPLINE$= 50$/Mpc, $\ln P$ has converged to an accuracy that surpasses the expected impact of baryonic effects on the matter power spectrum at $k > 10$/Mpc. (For an estimate of the impact of baryons on the total matter power spectrum, see Schneider & Teyssier 2015.) The lower K_MAX_SPLINE is, the faster CCL will run. The optimum choice of K_MAX_SPLINE is left to the user for their particular application.

We also extrapolate the power spectrum at small wavenumbers within the static routine ccl_power_extrapol_lowk. In this case, the power spectrum below K_MIN_SPLINE is obtained by a power-law extrapolation with index n_s:

$$\log P(k < \text{K\_MIN\_SPLINE}, z) = \log P(\text{K\_MIN\_SPLINE}, z) + n_s(\log k - \log \text{K\_MIN\_SPLINE}) \tag{17}$$

The value adopted for K_MIN_SPLINE depends on the choice of power spectrum method and is not currently settable by the user. For CLASS and the nonlinear power spectrum, we adopt K_MIN_SPLINE that coincides with the smallest wavenumber output by CLASS, K_MIN_SPLINE$= 7 \times 10^{-6}$/Mpc. Note that this parameter is different from K_MIN, which sets the minimum k for integrations and which is set by default to K_MIN$= 5 \times 10^{-5}$/Mpc. Hence, in practice, no extrapolation is occurring in this case, unless the user specifically asks for an output power spectra below K_MIN for their own purposes.

For BBKS, the power spectrum is computed analytically at all k, there is no extrapolation. For the Eisenstein & Hu implementation, the splines of the power spectrum span K_MIN$< k <$K_MAX_SPLINE, so there is only extrapolation at high k. For the nonlinear matter power spectrum from the emulator, K_MIN_SPLINE and K_MAX_SPLINE are set to fixed values that are determined from the range of validity of the emulator: K_MIN_SPLINE$= 10^{-3}$ Mpc^{-1} and K_MAX_SPLINE$= 5$ Mpc^{-1}.

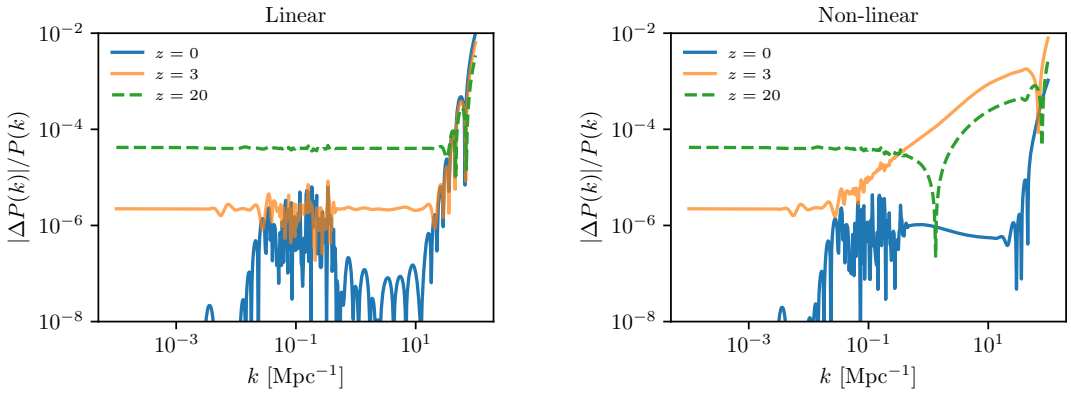

Figure 1. The relative error compared to power spectra produced with high values of the power spectrum splines, P_{fid}, produced by splining the matter power spectrum up to K_MAX_SPLINE$= 50\,\mathrm{Mpc}^{-1}$ and extrapolating beyond this value with a second order Taylor expansion the natural logarithm of the matter power spectrum. The left panel shows the relative errors for the linear matter power spectrum at $z = 0$, $z = 3$ and $z = 20$. The right panel shows the results for the non-linear matter power spectrum at the same redshifts. The standard CCL parameters adopted are those corresponding to the black dashed curve. For comparison, the impact of baryonic physics on the matter power spectrum is $\sim 10\%$ at $k = 1\,\mathrm{Mpc}^{-1}$ (Schneider & Teyssier 2015).

2.8.10. **Extrapolation for the linear power spectrum**

With the implementation described in the previous section, the power spectrum splines are initialized up to K_MAX_SPLINE. This is also true for the linear matter power spectrum, which is used within CCL in particular to obtain σ_8 (see Eq. 51). We have tested here how the procedure described in the previous section affects the convergence of the linear matter power spectrum. The result is shown in Figure 1. For some applications that use the linear power spectrum, the user might need to increase the value of K_MAX_SPLINE.

As in the previous section, the power spectrum at small wavenumber is extrapolated using a power-law. This extrapolation is performed below a fiducial value of K_MIN_SPLINE that coincides with the smallest wavenumber output by CLASS, as in the case of the nonlinear power spectrum described above.

We have found that changing N_A to 200, or changing the sampling of the wavenumber to 5000 points, does not change the results presented in Figures 1 in this section.

2.8.11. **Normalization of the power spectrum**

There are two alternative schemes for normalization of the matter power spectrum. The first one is to specify the value of A_s, the amplitude of the primordial power spectrum, which is passed directly to `CLASS`. This option is available in the case of the linear/nonlinear matter power spectrum implementation. For these, as well as for BBKS and E&H transfer functions, there is the additional option to set the normalization of the matter power spectrum by specifying σ_8, the RMS density contrast averaged over spheres of radius $8h^{-1}$Mpc. The computation of σ_8 is described in Section 2.11.

In practice, there is only one argument that encodes the normalization. This is the argument `norm_pk`, which can be passed the power spectrum normalization parameterized by σ_8 or A_s. As noted above, `ccl_parameters_create` switches to σ_8 normalization if `norm_pk` $> 10^{-5}$, and to A_s normalization otherwise.

In the `python` implementation, `CCL` allows for either σ_8 or `A_s` to be passed as parameters.

2.9. **Angular power spectra**

Angular power spectra between two quantities α and β will in general take the form:

$$C_\ell^{\alpha\beta} = \frac{2}{\pi}\int d\chi_1\, d\chi_2\, dk\, k^2 P_{\alpha\beta}(k,\chi_1,\chi_2)\, \Delta_\ell^\alpha(k,\chi_1)\, \Delta_\ell^\beta(k,\chi_2). \tag{18}$$

Here $P_{\alpha\beta}$ will be a generalized power spectrum, and the functions $\Delta_\ell^\alpha(k,\chi)$ will in general be a sum over different contributions, all of which take the form:

$$\Delta_\ell^\alpha(k,\chi) = f_\ell^\alpha\, W_\alpha(\chi)\, T_\alpha(k,\chi)\, j_\ell^{(n_\alpha)}(k\chi), \tag{19}$$

where we have defined the quantities:

1. f_ℓ^α: the *ℓ-dependent prefactor*, usually associated with angular derivatives.
2. $W_\alpha(\chi)$: the *radial kernel*, dependent only on redshift/distance.
3. $T_\alpha(k,\chi)$: the *transfer function*, dependent on both k and z/χ.
4. $j_\ell^{(n)}(x)$: generalized versions of the spherical Bessel functions, associated with radial derivatives or inverse laplacians.

Let us describe these in more details.

Radial kernels

Three important things must be noted about them:

- Line-of-sight integrals will be carried out over the variable χ, so it is important that the input arrays defining $W_\alpha(\chi)$ sample the kernel sufficiently well in that variable.
- `CCL` automatically determines the range of χ over which it will carry out any line-of-sight integral. These are determined as the lowest and highest value of χ in the input arrays defining $W_\alpha(\chi)$ at which the value of $W_\alpha(\chi)$ reaches 0.05% of its maximum value.
- If no input is passed, $W_\alpha(\chi)$ defaults to 1 everywhere.

Transfer functions

In the most general case, `CCL` accepts transfer functions as generic functions of k and a. For simplicity and speed it also supports the simpler case where these are factorisable: $T_\alpha(k,a) = K_\alpha(k)\,A_\alpha(a)$.

ℓ prefactors

We provide 3 options for these, encoded in a parameter that we will call d_θ here:

- $d_\theta = 0$: $f_\ell = 1$.
- $d_\theta = 1$: $= 1$: $f_\ell = \ell(\ell+1)$. This corresponds to taking the angular laplacian ∇^2.
- $d_\theta = 2$: $f_\ell = \sqrt{(\ell+2)!/(\ell-2)!}$, corresponding to the second angular derivatives of spin-2 quantities. To avoid computing the square root we make use of the following approximation:

$$\sqrt{\frac{(\ell+2)!}{(\ell-2)!}} \simeq \ell_{1/2}^2\left(1-\frac{5}{4}\ell_{1/2}^{-2}+\mathcal{O}(\ell_{1/2}^{-4})\right), \tag{20}$$

  where $\ell_{1/2} = \ell + 1/2$. This approximation is accurate at the level of 5×10^{-5} for $\ell = 10$, and for $\ell > 1000$ the $\mathcal{O}(\ell_{1/2}^{-2})$ term can be dropped altogether with an accuracy of $\sim 10^{-6}$.

Bessel functions

For $n \geq 0$, $j_\ell^{(n)}(x)$ represents the n-th derivative of the spherical bessel functions with respect to their argument. We allow values $n = 0$, 1 and 2, encoded in the variable `der_bessel`, and use the following identities:

$$j_\ell^{(1)}(x) = \frac{\ell\, j_\ell(x) - x\, j_{\ell+1}(x)}{x} \tag{21}$$

$$j_\ell^{(2)}(x) = \frac{[\ell(\ell-1) - x^2] j_\ell(x) + 2x j_{\ell+1}(x)}{x^2}. \tag{22}$$

We also allow a special value $n = -1$, for which:

$$j_\ell^{(-1)}(x) = \frac{j_\ell(x)}{x^2}. \tag{23}$$

This case is quite ubiquitous (see next section).

Tracer combinations

CCL also provides a way to generalize all the expressions in this sections to combinations of tracers. This is useful in many cases, such as galaxy number counts or cosmic shear, in which the actual physical observable (galaxy number overdensity or galaxy shape distortions) is made up of a combination of different effects. It is therefore possible to combine several tracers into a single one, such that the total $\Delta_\ell^X(k, \chi)$ of that tracer is the sum of the corresponding functions for all the tracers in the combination.

Standard tracers

CCL provides specific functionality (besides its generic features) to define the following types of tracers:

Here N.C. stands for "number counts", W.L. stands for "weak lensing" and CMB κ is the CMB lensing convergence. $s(z)$ is the magnification bias, given as the logarithmic derivative of the number of sources with magnitude limit, and $r(\chi)$ is the angular comoving distance (see Eq. 6). f is the growth rate, which CCL does not compute for massive neutrino cosmologies; therefore at this time an attempt to create a number count tracer in a cosmology with massive neutrinos will cause CCL to raise an error. C_ℓ is instead computed assuming a linear-theory relation between the matter overdensity and peculiar velocity fields. While this should not be

Tracer	$W_\alpha(\chi)$	$K_\alpha(k)$	$A_\alpha(a)$	d_θ^α	n_α
N.C., dens.	$H(z)\,N(z)$	1	$b(z)$	0	0
N.C., RSD	$H(z)\,N(z)$	1	$-f(z)$	0	2
N.C., mag.	$-\frac{3H_0^2\Omega_M}{a}\chi\int_z^\infty dz'\,N(z')\left[1-\frac{5s(z')}{2}\right]\frac{\chi'-\chi}{\chi'}$	1	1	1	-1
W.L., shear	$\frac{3H_0^2\Omega_M}{2a}\chi\int_z^\infty dz'\,N(z')\frac{\chi'-\chi}{\chi'}$	1	1	2	-1
W.L., I.A.	$H(z)\,N(z)$	1	$b_{\rm IA}(z)$	2	-1
CMB κ	$\frac{3H_0^2\Omega_M}{2a}\chi\frac{\chi_s-\chi}{\chi_s}$	1	1	1	-1

Table 3. Description of the standard tracers supported by CCL in terms of the framework introduced in Eq. 19. Note that, for modified gravity theories in the $\mu-\Sigma$ parametrization, the radial kernels for shear, magnification and CMB lensing take an extra multiplicative factor of $1+\Sigma(z)$.

problematic for wide photometric redshift bins, users should exercise care when interpreting results for narrow window functions. For intrinsic alignments, (I.A. in the table), CCL currently supports the so-called "non-linear alignment model", according to which the galaxy inertia tensor is proportional the local tidal tensor Hirata & Seljak (2004); Hirata et al. (2007). χ_s is the distance to the source plane for CMB lensing. $b_{\rm IA}$ is the corresponding alignment bias. Notice that in the CCL implementation, the normalization of this bias is chosen to coincide with the parameter $A_{\rm IA}$ used in current weak lensing analyses, i.e., the user is expected to pass $A_{\rm IA}$ to the routine, and this is connected to $b_{\rm IA}$ by the following expression(Joachimi et al. 2011, Eq. 6):

$$b_{\rm IA} = -A_{\rm IA}\frac{C_1\Omega_{\rm m}\rho_{\rm crit}}{D(z)} \tag{24}$$

where $\rho_{\rm crit}$ is in the default CCL units and $C_1 = 5\times10^{-14}(Mpc/h)^3/(M_\odot/h)$ (Brown et al. 2002; Bridle & King 2007). Note that although $\Omega_{\rm m}$ should in principle include non-relativistic neutrinos, this is not the case in CCL at this point.

Note that CCL currently does not compute relativistic corrections to number counts Challinor & Lewis (2011); Bonvin & Durrer (2011). Although these should be included in the future, their contribution to the total fluctuation is largely subdominant (see Alonso et al. (2015) and the two references above), and therefore it is safe to work without them in most cases.

It is also worth noting that the equations above should be modified for non-flat cosmologies by replacing the spherical Bessel functions j_ℓ with their hyperspherical counterparts Kamionkowski & Spergel (1994). This will be revisited in future versions of CCL.

Limber integrals

In the Limber approximation:

$$j_\ell(k\chi) \simeq \sqrt{\frac{\pi}{2\ell_{1/2}}}\delta(k\chi - \ell_{1/2}) \tag{25}$$

Defining $\chi_\ell \equiv \ell_{1/2}/k$ and $k_\ell \equiv \ell_{1/2}/\chi$, the power spectra can now be calculated through a single integral as:

$$C_\ell^{\alpha\beta} = \ell_{1/2}^{-1} \int dk\, P_{\alpha\beta}(k, \chi_\ell, \chi_\ell)\tilde{\Delta}_\ell^\alpha(k)\tilde{\Delta}_\ell^\beta(k), \tag{26}$$

where the $\tilde{\Delta}^\alpha$s depend on the value of n_α:

- $n_\alpha = 0$:

$$\tilde{\Delta}_\ell^\alpha(k) = f_\ell^\alpha\, W_\alpha(\chi_\ell) T_\alpha(k, \chi_\ell). \tag{27}$$

- $n_\alpha = 1$:

$$\tilde{\Delta}_\ell^\alpha(k) = f_\ell^\alpha \left[\frac{2\ell}{2\ell+1}\, W_\alpha(\chi_\ell) T_\alpha(k, \chi_\ell) - \sqrt{\frac{2\ell+1}{2\ell+3}}\, W_\alpha(\chi_{\ell+1}) T_\alpha(k, \chi_{\ell+1})\right] \tag{28}$$

- $n_\alpha = 2$:

$$\tilde{\Delta}_\ell^\alpha(k) = f_\ell^\alpha \left[\sqrt{\frac{2\ell+1}{2\ell+3}}\frac{4}{2\ell+3} W_\alpha(\chi_{\ell+1}) T_\alpha(k, \chi_{\ell+1}) - \frac{1+8\ell}{(2\ell+1)^2} W_\alpha(\chi_\ell) T_\alpha(k, \chi_\ell)\right] \tag{29}$$

- $n_\alpha = -1$:

$$\tilde{\Delta}_\ell^\alpha(k) = \frac{f_\ell^\alpha\, W_\alpha(\chi_\ell) T_\alpha(k, \chi_\ell)}{(\ell + 1/2)^2}. \tag{30}$$

Beyond limber

Native CCL computation.

Note: This capability has been deprecated in `v1`, superseded by `Angpow`, described below. But it is still available in previous versions of the library.

CCL incorporates routines to compute the C_ℓ^{ab} angular power spectra as described above but without the Limber approximation. The algorithm performs first the integrals over z for both tracers, and ends with the k integral. This computation is much slower than using the Limber approximation, but it ends up with precise angular power spectra at low ℓ, and correct cross-correlations between tracers (the Limber approximation fails at reproducing the interference terms $j_\ell(x) \times j_\ell(x')$).

Some parameters are needed to define this integral. First, to fasten the computation the C_ℓ^{ab} function is computed for a selection of ℓ values (linearly spaced at low-ℓ and logarithmicly spaced at high-ℓ) and then the function is splined. Then the integration step of the redshift integrals must be specified in terms of a comoving distance. It is recommanded to start with a low value (< 3 Mpc) for a high precision integration, and then to release this parameter to achieve the user needs in terms of precision and rapidity. A logarithmic step in k must also be provided: it does not correspond to the integration step but to a computation step before splining the transfer functions $\Delta_\ell^a(k)$. Last parameter, a minimum redshift can be given for the redshift integral bound so that the transfer functions are not computed near $z \approx 0$ where they can be numerically undefined.

The integration bounds in redshift are estimated automatically given the a redshift window so that the integration is preformed where the redshift window is relevant within a given precision (set by the CCL_FRAC_RELEVANT parameter). The integration bounds for the k integrals are defined automatically given the ℓ multipole and the comoving distances at play.

It is worth noting that the user can define a threshold multipole ℓ from which the C_ℓ^{ab} computation switches to the Limber approximation which is faster and generally relevant at high ℓ values.

Angpow.

The aim of the Angpow software (Campagne et al. 2017) is to compute the angular power spectra C_ℓ^{ab} without any Limber numerical approximation. CCL has been linked to the Angpow code, which is briefly described here.

The angular power spectrum for two shells C_ℓ^{ab} is computed in Angpow according to the following expression

$$C_\ell^{ab} = \iint_0^\infty \mathrm{d}z\mathrm{d}z' p_{z_1}(z_1) p_{z_2}(z') \times \int_0^\infty \mathrm{d}k\, f_\ell(z,k) f_\ell(z',k). \tag{31}$$

The auxiliary function $f_\ell(z,k)$ can be defined without loss of generality as

$$f_\ell(z,k) \equiv \sqrt{\frac{2}{\pi}}\, k\sqrt{P(k,z)}\, \widetilde{\Delta}_\ell(z,k) \tag{32}$$

with

- $P(k,z)$: the matter power spectrum at redshift z
- $\widetilde{\Delta}_\ell(z,k)$: a function describing the physical processes such as matter density fluctuations, redshift-space distortions as described for instance in references Durrer (2008); Yoo et al. (2009); Yoo (2010); Challinor & Lewis (2011); Bonvin & Durrer (2011). Currently, the `Angpow` version delivered with `CCL` only can deal with galaxy clustering tracers (no lensing) and this without the magnification lensing term (equation **??**). The incorporation of those transfer functions is left for future work, but in principle `Angpow` has already the capability to treat them. For now, for galaxy clustering tracers we defined $\widetilde{\Delta}_\ell(z,k)$ as
$$\widetilde{\Delta}_\ell(z,k) \approx b j_\ell(k\chi(z)) - f(z) j''_\ell(k\chi(z)) \tag{33}$$
with $j_\ell(x)$ and $j''_\ell(x)$ the spherical Bessel function of order ℓ and its second derivative, and $f(a)$ the growth rate as defined subsection 2.7 (derivative of the growth function with respect to the scale factor a).

To proceed to a numerical evaluation of equation 31, `Angpow` first conducts inside the rectangle $[z_{1\min}, z_{1\max}] \times [z_{2\min}, z_{2\max}]$ given by the $p_z(z)$ selection functions a Cartesian product of one-dimensional (1D) quadrature defined by the set of sample nodes z_i and weights w_i. In practice, the Clenshaw-Curtis quadrature is used. The corresponding sampling points (z_{1i}, z_{2j}) are weighted by the product $w_i w_j$ using the 1D quadrature sample points and weights on both redshift regions with $i = 0, \ldots, N_{z_1} - 1$ and $j = 0, \ldots, N_{z_2} - 1$. Then, one gets the following approximation:

$$C_\ell^{ab} \approx \sum_{i=0}^{N_{z_1}-1} \sum_{j=0}^{N_{z_2}-1} w_i w_j p_{z_1}(z_i) p_{z_2}(z_j) \widehat{P}_\ell(\chi_i, \chi_j) \tag{34}$$

with the notations $z_i = z_{1i}$, $z_j = z_{2j}$ and $\chi_i = \chi(z_{1i})$, $\chi_j = \chi(z_{2j})$ and

$$\widehat{P}_\ell(z_i, z_j) = \int_0^\infty dk\, f_\ell(z_i,k) f_\ell(z_j,k), \tag{35}$$

To conduct the computation of such integral of highly oscillating functions we use the 3C-algorithm described in details in reference (Campagne et al. 2017). In brief this algorithm proceeds the following way:

1. the total integration k interval (eg. $[k_{\rm min}, k_{\rm max}]$) in equation (35) is cut on several k-sub-intervals;

2. on each sub-interval the functions $f_{i\,\ell}(k) = f_\ell(z_i, k)$ and $f_{j\,\ell}(k) = f_\ell(z_j, k)$ are projected onto Chebyshev series of order 2^N;

3. the product of the two Chebyshev series is performed with a 2^{2N} Chebyshev series;

4. then, the integral on the sub-interval is computed thanks to the Clenshaw-Curtis quadrature.

All the Chebyshev expansions and the Clenshaw-Curtis quadrature are performed via the DCT-I fast transform of FFTW.

Thanks to the 3C-algorithm, the `Angpow` is able to compute the C_ℓ^{ab} in a fast and accurate way. It was tested against `CLASS` and the native `CCL` computation and can performed the computation an order of magnitude faster, which is more suitable for an extensive exploration of the cosmological parameter space ($\mathcal{O}(1s)$). Note that `Angpow` is written in C++ with OpenMP, to distribute the computation of a single C_ℓ on a single thread. The computation can also be switch by the user to the faster Limber approximation setting a threshold multipole ℓ.

Precision tests.

The code has been compared with `CLASS` and the native `CCL` computation and all three softwares agrees perfectly if precision parameters are pushed to high levels.

A few parameters must be provided to set the precision of this computation. First the order of the Chebyshev polynomials is set to 2^{10} by default, and the number of k sub-intervals to 200, and we checked this is enough for the current uses. Then the redshift quadrature stepping is set automatically given the redshift windows to recover the native `CCL` computation boosted with high precision parameters: its precision is optimised so that the relative numerical error compared with the native method is two orders of magnitude below the relative cosmic variance $\sqrt{2/(2\ell+1)}$, from $\ell = 2$ to $\ell = 1000$. The $k_{\rm min}$ and $k_{\rm max}$ bounds are also automatically set given the current multipole ℓ and the comoving distance χ involved in the inner integral.

2.10. **Correlation functions**

The following expressions relating the angular power spectra and correlation functions are valid in the flat-sky approximation[6]. In all cases, $f_K(\chi)$ is comoving angular diameter distance, which differs from the radial comoving distance χ only in the case of cosmologies with non-zero curvature. The routines described in this subsection are implemented in `ccl_correlation.c`.

Galaxy-galaxy. The angular correlation function between two number-count tracers (labeled a and b here) is given by

$$\xi^{ab}(\theta) = \int d\ell \frac{\ell}{2\pi} C_\ell^{ab} J_0(\ell\theta), \tag{36}$$

where C_{ab} is the angular power spectrum between both tracers.

Lensing-lensing. The lensing correlation functions are [7]

$$\xi_+^{ab}(\theta) = \int_0^\infty d\ell \frac{\ell}{2\pi} J_0(\ell\theta) C_\ell^{ab}, \tag{37}$$

$$\xi_-^{ab}(\theta) = \int_0^\infty d\ell \frac{\ell}{2\pi} J_4(\ell\theta) C_\ell^{ab}, \tag{38}$$

where the angular lensing convergence power spectrum C_ℓ^{ab} is given above (see Equations **??** and **??**).

Galaxy-lensing. The correlation between a number count tracer a and a shear tracer b is given by

$$\xi^{ab}(\theta) = \int d\ell \frac{\ell}{2\pi} C_\ell^{ab} J_2(\ell\theta), \tag{39}$$

Note that, in the above, "Galaxy" and "Lensing" can be replaced by any spin-0 and spin-2 fields on the sphere respectively (e.g. the CMB lensing convergence would play the same role as the galaxy overdensity field in all the formulas above).

3d spatial correlation function. In addition to the angular correlation functions, the 3-dimensional spatial correlation function $\xi(r)$ is also calculated from the power spectrum $P(k)$ using

$$\xi(r) = \frac{1}{2\pi^2} \int dk\, k^2 P(k) \frac{\sin(kr)}{kr} \tag{40}$$

To evaluate the numerical integral in the correlation functions, we make use of the public code `FFTlog`[8](Hamilton 2000; Talman 2009). In brief, `FFTlog` works on

[6] See the weak lensing review by Bartelmann & Schneider (2001), page 44 and Joachimi & Bridle (2010).

[7] from Schneider 2002 and Bartelmann & Schneider section 6.4.1

[8] http://casa.colorado.edu/~ajsh/FFTLog/

functions periodic in log space, by writing the Hankel Transform as a convolution between Bessel functions and the function of interest (in this case either C_ℓ or $P(k)$). A version of this code is included in CCL with minor modifications.

Redshift-space distortions. In redshift space and under the linear approximation (Kaiser 1987) the correlation function $\xi(s,\mu)$ can be expanded in multipoles:

$$\xi(s,\mu) = \sum_{l\geq 0} \xi_l(s) L_l(\mu), \tag{41}$$

where s is the magnitude of the galaxy separation vector in redshift space $\mathbf{s}$, μ is the cosine of the separation angle between $\mathbf{s}$ and the line of sight, and $L_l(x)$ are the Legendre polynomials. The multipoles of the correlation function are given by

$$\xi_l(s) = \frac{i^l}{2\pi^2} \int_0^\infty P_l(k) j_l(ks) k^2 dk, \tag{42}$$

where $P_l(k)$ are multipoles of the power spectrum in redshift space $P(k,\mu_{\mathbf{k}})$ defined by

$$P(k,\mu_{\mathbf{k}}) = \sum_{l\geq 0} P_l(k) L_l(\mu_{\mathbf{k}}). \tag{43}$$

In the linear approximation only the $l = 0, 2, 4$ multipoles are non-zero. They are related to the real-space power spectrum $P(k)$ by

$$P_0(k) = \left(1 + \tfrac{2}{3}\beta + \tfrac{1}{5}\beta^2\right) P(k), \tag{44}$$

$$P_2(k) = \left(\tfrac{4}{3}\beta + \tfrac{4}{7}\beta^2\right) P(k), \tag{45}$$

$$P_4(k) = \tfrac{8}{35}\beta^2 P(k), \tag{46}$$

where β is the ratio of the growth rate f and bias factor b, $\beta = f/b$.

CCL implements the redshift space correlation function $\xi(s,\mu)$, it's average at constant s, $\xi(s)$, the multiploes $\xi_l(s)$, and $\xi(\pi,\sigma)$, where π is the galaxy pair separation parallel to the line of sight and σ is the separation perpendicular to the line of sight. We use FFTlog to calculate $\xi_l(s)$ from $P_l(k)$ in the implementation of $\xi(s,\mu)$.

In order to speed up the calculations, we provide the option to create a spline of the multipole functions $\xi_l(s)$ and store them in global splines for subsequent access. After calculating the redshift-space correlation functions for a given cosmology the splines should be freed using the provided function before calculating for a different cosmology.

2.11. **Halo mass & halo bias functions**

The routines described in this subsection are implemented in `ccl_massfunc.c`.

The halo mass function is implemented using several different definitions from the literature: Tinker et al. (2008), Tinker et al. (2010), Angulo et al. (2012), and Watson et al. (2013). All four models are tuned to simulation data and tested against observational results. In addition, each of these fits has been implemented using the common halo definition of $\Delta = 200$, where a halo is defined with:

$$\bar{\rho}(r_\Delta) = \Delta \times \rho_{\rm m}, \tag{47}$$

where a halo with size r_Δ has an average density $\bar{\rho}$ equal to the overdensity parameter Δ times the mean background density of the universe, $\rho_{\rm m}$. Note that another common definition utilizes the critical density of the universe, $\rho_{\rm c}$; currently `CCL` requires that an external conversion by the end user between values of Δ with respect to the critical density to values of Δ as defined with respect to the mean density. In the future we plan to allow for self-consistent handling of critical density based definitions, though it is not implemented as of this build.

In addition to the usage of the most common definition, we have implemented an extension for two of the models. The Tinker 2010 model allows for a value of Δ to be given between the values of 200 and 3200 and interpolates the fitting parameters within this range in a space of $\log\Delta$ using splines. We also have implemented interpolation in the same range of Tinker 2008 Δ values. For both Tinker 2008 and Tinker 2010 models we have utilized spline interpolation through `GSL` routines in order to guarantee a match to specified fitting parameters at exact values of Δ. This fitting has slight deviation from the fit as expressed in Tinker 2010.

The halo mass function models implemented in CCL are tuned to simulations without massive neutrinos, therefore are not valid in cosmologies with massive neutrinos. Attempts to calculate the halo mass function, halo bias, or other related quantities within cosmologies with massive neutrinos will cause `CCL` to raise an error and quit.

With the exception of the Tinker 2010 model, we attempt to keep a common form to the multiplicity function whenever possible for ease of extension:

$$f(\sigma) = A\left[\left(\frac{\sigma}{b}\right)^{-a} + 1\right]e^{-c/\sigma^2}, \tag{48}$$

where A, a, b, and c are fitting parameters that have additional redshift scaling and σ is the RMS variance of the density field smoothed on some scale M at some scale factor a. This basic form is modified for the Angulo et al. (2012) formulation. The resulting form is

$$f(\sigma) = A\left[\left(\frac{b}{\sigma} + 1\right)^{-a}\right]e^{-c/\sigma^2}, \tag{49}$$

where the only change is in the formulation of the second term. Note that the fitting parameters in the Angulo et al. (2012) formulation do not contain any redshift dependence and the use of it is primarily for testing and benchmark purposes.

Each call to the halo mass function requires an assumed model (defined within the `ccl_configuration` structure contained in `ccl_cosmology`), in addition to a value of the halo mass and scale factor for which to evaluate the halo mass function. The currently implemented models can be called with the tags `config.mass_function_method = ccl_tinker`, `ccl_tinker10`, `ccl_angulo`, or `ccl_watson`. It returns the number density of halos in logarithmic mass bins, in the form $dn/d\log_{10} M$, where n is the number density of halos of a given mass and M is the input halo mass.

The halo mass M is related to σ by first computing the radius R that would enclose a mass M in a homogeneous Universe at $z = 0$:

$$M = \frac{H_0^2}{2G} R^3 \rightarrow \frac{M}{M_\odot} = 1.162 \times 10^{12} \Omega_M h^2 \left(\frac{R}{1\,\mathrm{Mpc}}\right)^3. \tag{50}$$

The rms density contrast in spheres of radius R can then be computed as

$$\sigma_R^2 = \frac{1}{2\pi^2} \int dk\, k^2\, P_k\, \tilde{W}_R^2(k) \tag{51}$$

where P_k is the matter power spectrum and $\tilde{W}(kR)$ is the Fourier transform of a spherical top hat window function,

$$\tilde{W}_R(k) = \frac{3}{(kR)^3}[\sin(kR) - kR\cos(kR)] \tag{52}$$

This function is directly implemented in `CCL` as well as a specific σ_8 function.

The Tinker et al. (2010) model parameterizes both the halo mass function and the halo bias in terms of the peak height, $\nu = \delta_c/\sigma(M)$, where δ_c is the critical density for collapse and is chosen to be 1.686 for this particular parameterization. We can then parameterize the halo function and halo bias as

$$b(\nu) = 1 - A\frac{\nu^a}{\nu^a + {\delta_c}^a} + B\nu^b + C\nu^c, f(\nu) = \alpha[1 + (\beta\nu)^{-2\phi}]\nu^{2\eta} e(-\gamma\nu^2/2). \tag{53}$$

The currently implemented model in `CCL` allows for an arbitrary overdensity Δ to be chosen, using the fitting functions provided in Tinker et al. (2010). Other halo model definitions are not included in the halo bias calculation, though this remains an area of active work to improve upon.

2.12. **Halo model**

The routines described in this subsection are implemented in `ccl_halomod.c`.

In this section we review a basic halo-model computation (Seljak 2000; Peacock & Smith 2000; Cooray & Sheth 2002) of the cross-correlation between any two cosmological fields and only requires knowledge of the halo profiles of the field in question. For example, in the case of the matter-density auto spectrum we need only know the halo density profiles. For the galaxy spectrum we require knowledge of the number of, and distribution of, galaxies as a function of halo mass. In this simple form the halo model is approximate and makes the assumption that haloes are *linearly* biased with respect to the *linear* matter field and also assumes that haloes are spherical with properties that are determined solely by the halo mass. It is possible to go beyond these simplified assumptions, and we direct the interested reader to Cooray & Sheth (2002); Smith et al. (2007); Giocoli et al. (2010); Smith & Markovic (2011).

The eventual aim for CCL is to have a halo model that can calculate the auto- and cross-spectra for any cosmological field combinations with parameters that can be taken either from numerical simulations or observational data. Currently we have only implemented the case of the matter-density auto spectrum, but we keep the notation as general as possible in the following:

Consider two 3D cosmological fields ρ_i and ρ_j, the cross power spectrum at a given redshift can be written as a sum of a two- and a one-halo term given by

$$P_{2\mathrm{H},ij}(k) = P_{\mathrm{lin}}(k) \prod_{n=i,j} \left[\int_0^\infty b(M) \frac{\mathrm{d}n}{\mathrm{d}M} W_n(M,k)\ \mathrm{d}M \right] , \tag{54}$$

$$P_{1\mathrm{H},ij}(k) = \int_0^\infty \frac{\mathrm{d}n}{\mathrm{d}M} W_i(M,k) W_j(M,k)\ \mathrm{d}M , \tag{55}$$

where M is the halo mass, $\mathrm{d}n/\mathrm{d}M$ is the halo mass function defined as the first of equations (53) and $b(M)$ is the linear halo bias with respect to the linear matter density field, defined as the large-scale limit of the second of equations (53).

Equations (54) and (55) contain the (spherical) Fourier transform of the halo profile, or halo 'window function':

$$W_i(M,k) = \int_0^\infty 4\pi r^2 \frac{\sin(kr)}{kr} \rho_{\mathrm{H},i}(M,r)\ \mathrm{d}r , \tag{56}$$

where $\rho_{\mathrm{H},i}(M,r)$ is the radial profile for the field i in a host halo of mass M. For example, if one is interested in matter fields then this would be the halo density

profile, if one were interested in galaxies then this would be the number density and distribution of galaxies around a halo of mass M.

Note that the halo mass function and bias *must* satisfy the following properties for the total power spectrum to have the correct large-scale limit[9]:

$$\frac{1}{\bar{\rho}_{\rm m}}\int_0^\infty M\frac{{\rm d}n}{{\rm d}M}\,{\rm d}M = 1\,, \tag{57}$$

$$\frac{1}{\bar{\rho}_{\rm m}}\int_0^\infty Mb(M)\frac{{\rm d}n}{{\rm d}M}\,{\rm d}M = 1\,. \tag{58}$$

If one uses a mass function and bias pair that are related via the peak-background split formalism (Mo & White 1996; Sheth et al. 2001), these conditions are automatically satisfied. In words these equations enforce that all matter is associated to a halo and that matter is on average unbiased with respect to itself. In the convention used in CCL the units of $P(k)$ will be exactly the units of $\rho_i\rho_j/{\rm Mpc}^3$. The units of the W_i are those of ρ_i multiplied by volume.

For the matter power spectrum we use the halo profiles of Navarro, Frenk, & White (NFW; 1997):

$$\rho_{\rm H}(M,r) \propto \frac{1}{r/r_{\rm s}(1+r/r_{\rm s})^2}\,, \tag{59}$$

which is written in terms of a scale radius $r_{\rm s}$. The constant of proportionality fixed by the condition that the halo has total mass M when the boundary is set at the virial radius $r_{\rm v}$, which is set such that the halo has a fixed density $\Delta_{\rm v}$ with respect to the mean

$$M = 4\pi r_{\rm v}^3\Delta_{\rm v}\bar{\rho}\,. \tag{60}$$

Finally, the scale radius is usually expressed in terms of the mass-dependent halo concentration parameter $c(M) = r_{\rm v}/r_{\rm s}$. We use the simple mass-concentration relation from Bullock et al. (2001)

$$c(M) = 9\left(\frac{M}{M_*}\right)^{-0.13}\,, \tag{61}$$

where $\delta_{\rm c}/\sigma(M_*) = 1$. Note that, in order to be consistent, one should use a value of $\Delta_{\rm v}$ and $c(M)$ that is consistent with the halo definition used for the halo mass function and bias.

[9] Note that achieving these correct limits for some fields is difficult numerically because of the large amount of mass contained in low mass haloes according to most popular mass functions. Special care must be taken with the two-halo integral in the case of matter power spectra.

3. Tests and validation

Our goal is for outputs of CCL to be validated against independent benchmark codes. This process is documented in the CCL paper that can be found in the doc/ccl_paper folder.

For each CCL prediction, at least one independent code was used to produce the same result. Predictions were compared and the resulting numerical accuracy, documented in the paper. (See Table 2 for a summary of these results.) Potential differences in the implementation of the relevant algorithms were discussed there as well. For specific cases, such as the matter power spectrum provided by the Cosmic Emulator, angular power spectra and projected correlation functions, we established target accuracies for CCL to achieve, though a more detailed forecasting exercise should be pursued in the future to establish whether results are compatible with the expected requirements of LSST DESC cosmological analyses in the next decade.

All benchmark codes are either made public within the CCL repository or made available online and described in the CCL wiki[10].

We would like to thank the organisers of the the DESC collaboration meetings at: Oxford (July 2016), SLAC (March 2016), and ANL (2015), and the LSST-DESC Hack Week organisers (CMU, November 2016), where this work was partly developed. We would also like to acknowledge the contribution of the participants of the TJP Code Comparison Project, some of whom are among the CCL contributors, for providing the benchmarks for testing CCL. Finally, we are grateful for the feedback received from other working groups of DESC, including Strong Lensing, Supernovae and Photometric Redshifts.

Author contributions are listed below.
Husni Almoubayyed: wrote an mcmc jupyter notebook example, reviewed code/contributed to issues.
David Alonso: Co-led project; developed structure for angular power spectra; implemented autotools; integrated into LSS pipeline; contributed to: background, power spectrum, mass function, documentation and benchmarks; reviewed code
Jonathan Blazek: Planning capabilities and structure; documentation and testing.
Philip Bull: Implemented the Python wrapper and wrote documentation for it; general bug fixes, maintenance, and code review; enhanced the installer and error handling system.

[10] https://github.com/LSSTDESC/CCL/wiki

Jean-Éric Campagne: Angpow builder and contributed to the interface with CCL.
N. Elisa Chisari: Co-led project, coordinated hack projects & communication, contributed to: correlation function & power spectrum implementation, documentation, and comparisons with benchmarks.
Alex Drlica-Wagner: Helped with document preparation.
Zilong Du: Implemented the 3d correlation function, redshift-space correlation functions, and corresponding benchmarks.
Tim Eifler: Reviewed/tested code.
John Ellison: Implemented the 3d correlation function, redshift-space correlation functions, and corresponding benchmarks; wrote documentation.
Cristhian Garcia Quintero: Produced benchmarks for correlation functions in modified gravity.
Renée Hlozek: Contributed initial code for error handling structures, reviewed other code edits.
Mustapha Ishak: Contributed to planning of code capabilities and structure; reviewed code; identified and fixed bugs.
Shahab Joudaki: Created physical density function and documentation.
Matthew Kirby: Performed comparison of physical constants.
David Kirkby: Writing, testing and reviewing code. Asking questions.
Elisabeth Krause: Initiated and co-led project; developed CLASS interface and error handling; contributed to other code; reviewed pull requests.
Francois Lanusse: Worked on install procedure
C. Danielle Leonard: Wrote and tested code for LSST specifications, user-defined photo-z interface, and support of massive neutrinos; reviewed other code; wrote text for this note.
Christiane S. Lorenz: Contributed to accurate high-redshift cosmological background quantities and benchmarked background splines.
Phil Marshall: Helped with document preparation.
Thomas McClintock: Wrote Python documentation.
Sean McLaughlin: Wrote doxygen documentation and fixed bugs/added functionality to distances.
Alexander Mead: Wrote halo model code
Jérémy Neveu: Contributed to Angpow and built the interface with CCL.
Stéphane Plaszczynski: Contributed to Angpow and contributed to the interface with CCL.
Javier Sanchez: Modified setup.py to allow pip installation and uninstall.
Sukhdeep Singh: Contributed to the correlation functions code.
Anže Slosar: Wrote and reviewed code.
Tilman Tröster: Wrote code for user-changable precision parameters, added distance and growth factor tests, found and fixed bugs.
Antonio Villarreal: Contributed to initial benchmarking, halo mass function code,

and general code and issues review.
Michal Vrastil: Wrote documentation and example code, reviewed code.
Joe Zuntz: Wrote initial infrastructure, C testing setup, and reviewed code.

References

Alonso, D., Bull, P., Ferreira, P. G., Maartens, R., & Santos, M. G. 2015, ApJ, 814, 145

Angulo, R. E., Springel, V., White, S. D. M., et al. 2012, MNRAS, 426, 2046

Bardeen, J. M., Bond, J. R., Kaiser, N., & Szalay, A. S. 1986, ApJ, 304, 15

Bartelmann, M., & Schneider, P. 2001, PhR, 340, 291

Blas, D., Lesgourgues, J., & Tram, T. 2011, CLASS: Cosmic Linear Anisotropy Solving System, Astrophysics Source Code Library, ascl:1106.020

Bonvin, C., & Durrer, R. 2011, PhRvD, 84, 063505

Bridle, S., & King, L. 2007, New Journal of Physics, 9, 444

Brown, M. L., Taylor, A. N., Hambly, N. C., & Dye, S. 2002, MNRAS, 333, 501

Bullock, J. S., Kolatt, T. S., Sigad, Y., et al. 2001, MNRAS, 321, 559

Campagne, J.-E., Neveu, J., & Plaszczynski, S. 2017, ArXiv e-prints, arXiv:1701.03592

Challinor, A., & Lewis, A. 2011, PhRvD, 84, 043516

Chevallier, M., & Polarski, D. 2001, International Journal of Modern Physics D, 10, 213

Cooray, A., & Sheth, R. 2002, Physics Reports, 372, 1

Durrer, R. 2008, The Cosmic Microwave Background (Cambridge University Press)

Eisenstein, D. J., & Hu, W. 1998, ApJ, 496, 605

Ferreira, P. G., & Skordis, C. 2010, Physical Review D, 81, 104020

Giocoli, C., Bartelmann, M., Sheth, R. K., & Cacciato, M. 2010, MNRAS, 408, 300

Hamilton, A. J. S. 2000, MNRAS, 312, 257

Hirata, C. M., Mandelbaum, R., Ishak, M., et al. 2007, MNRAS, 381, 1197

Hirata, C. M., & Seljak, U. 2004, PhRvD, 70, 063526

Joachimi, B., & Bridle, S. L. 2010, A&A, 523, A1

Joachimi, B., Mandelbaum, R., Abdalla, F. B., & Bridle, S. L. 2011, A&A, 527, A26

Kaiser, N. 1987, Monthly Notices of the Royal Astronomical Society, 227, 1

Kamionkowski, M., & Spergel, D. N. 1994, ApJ, 432, 7

Lattanzi, M., & Gerbino, M. 2017, ArXiv e-prints, arXiv:1712.07109

Lawrence, E., Heitmann, K., Kwan, J., et al. 2017, ApJ, 847, 50

Lesgourgues, J., & Pastor, S. 2012, ArXiv e-prints, arXiv:1212.6154

Linder, E. V. 2003, Physical Review Letters, 90, 091301

Mamajek, E. E., Prsa, A., Torres, G., et al. 2015, ArXiv e-prints, arXiv:1510.07674

Mangano, G., Miele, G., Pastor, S., et al. 2005, Nuclear Physics B, 729, 221

Mo, H. J., & White, S. D. M. 1996, MNRAS, 282, 347

Mohr, P. J., Newell, D. B., & Taylor, B. N. 2016, Reviews of Modern Physics, 88, 035009

Navarro, J. F., Frenk, C. S., & White, S. D. M. 1997, ApJ, 490, 493

Peacock, J. A., & Smith, R. E. 2000, MNRAS, 318, 1144

Schneider, A., & Teyssier, R. 2015, JCAP, 12, 049

Seljak, U. 2000, MNRAS, 318, 203

Sheth, R. K., Mo, H. J., & Tormen, G. 2001, MNRAS, 323, 1
Smith, R. E., & Markovic, K. 2011, PhRvD, 84, 063507
Smith, R. E., Scoccimarro, R., & Sheth, R. K. 2007, PhRvD, 75, 063512
Takahashi, R., Sato, M., Nishimichi, T., Taruya, A., & Oguri, M. 2012, ApJ, 761, 152
Talman, J. 2009, Computer Physics Communications, 180, 332
Tinker, J., Kravtsov, A. V., Klypin, A., et al. 2008, ApJ, 688, 709
Tinker, J. L., Robertson, B. E., Kravtsov, A. V., et al. 2010, ApJ, 724, 878
Watson, W. A., Iliev, I. T., D'Aloisio, A., et al. 2013, MNRAS, 433, 1230
Yoo, J. 2010, PhRvD, 82, 083508
Yoo, J., Fitzpatrick, A. L., & Zaldarriaga, M. 2009, PhRvD, 80, 083514