A&A 602, A72 (2017)
DOI: 10.1051/0004-6361/201730399

Astronomy & Astrophysics

Angpow: a software for the fast computation of accurate tomographic power spectra⋆

J.-E. Campagne, J. Neveu, and S. Plaszczynski

LAL, Univ. Paris-Sud, CNRS/IN2P3, Université Paris-Saclay, 91400 Orsay, France
e-mail: campagne@lal.in2p3.fr

Received 5 January 2017 / Accepted 27 April 2017

ABSTRACT

Aims. The statistical distribution of galaxies is a powerful probe to constrain cosmological models and gravity. In particular, the matter power spectrum $P(k)$ provides information about the cosmological distance evolution and the galaxy clustering. However the building of $P(k)$ from galaxy catalogs requires a cosmological model to convert angles on the sky and redshifts into distances, which leads to difficulties when comparing data with predicted $P(k)$ from other cosmological models, and for photometric surveys like the Large Synoptic Survey Telescope (LSST). The angular power spectrum $C_\ell(z_1, z_2)$ between two bins located at redshift z_1 and z_2 contains the same information as the matter power spectrum, and is free from any cosmological assumption, but the prediction of $C_\ell(z_1, z_2)$ from $P(k)$ is a costly computation when performed precisely.
Methods. The Angpow software aims at quickly and accurately computing the auto ($z_1 = z_2$) and cross ($z_1 \neq z_2$) angular power spectra between redshift bins. We describe the developed algorithm based on developments on the Chebyshev polynomial basis and on the Clenshaw-Curtis quadrature method. We validate the results with other codes, and benchmark the performance.
Results. Angpow is flexible and can handle any user-defined power spectra, transfer functions, and redshift selection windows. The code is fast enough to be embedded inside programs exploring large cosmological parameter spaces through the $C_\ell(z_1, z_2)$ comparison with data. We emphasize that the Limber's approximation, often used to speed up the computation, gives incorrect C_ℓ values for cross-correlations.

Key words. large-scale structure of Universe – methods: numerical

1. Introduction

Cosmology is entering the era of wide and deep surveys of galaxies, such as, for example, with the Dark Energy Spectroscopic Instrument (DESI; Levi et al. 2013), the Large Synoptic Survey Telescope (LSST; Ivezic et al. 2008), and the *Euclid* satellite (Laureijs et al. 2011), in order to investigate the mechanisms of cosmic acceleration (for a review see Weinberg et al. 2013). Cosmological models can be tested, that is, compared against actual measurements, by studying the statistical properties of galaxy clustering. Several methods exist for this, the most classical ones computing correlations in real (Landy & Szalay 1993) or Fourier space (Feldman et al. 1994). However, for wider and deeper surveys, one may also try to condense the clustering information into redshift bins ("shells") and compute the auto- and cross-correlations between redshift shells (Asorey et al. 2012). This is known as tomography, and allows for a more precise understanding of potential systematic errors in different redshift regions. Several studies compare the merits of this kind of approach with the more classical ones (Asorey et al. 2012, 2014; Di Dio et al. 2014; Nicola et al. 2014; Lanusse et al. 2015) and try to optimize the binning to keep most of the cosmological information. All previous studies have been based on the Fisher formalism, which considers observables as Gaussian; unrepresentative, however, of real life data.

To go on further and prepare the future tomographic analyses, one needs to implement a full pipeline and test the method with, for instance, a Monte-Carlo Markov chain (MCMC) exploration of the cosmological parameter space. But there is a technical bottleneck; running a typical MCMC algorithm in cosmology is already very lengthy and requires computing typically a few 10^5 models. Each model is the result of a numerical code that solves the cosmological equations (known as "Boltzmann solvers"), such as CLASS[1] (Blas et al. 2011), which takes typically 5–10 s on eight cores. Today this amounts to several days of computation.

For a tomographic method, one further needs to transform the output of the Boltzmann solver, the matter power spectrum, into the observable space, represented as $C_\ell(z_i, z_j)$ angular power spectra between two redshift shells located at z_i and z_j. This transformation is numerically challenging because of overlapping integrals between very oscillating spherical Bessel functions.

One then often makes use of the Limber's approximation, which essentiallyreplaces the Bessel functions by a single Dirac value. However, as we show here, this leads to a poor approximation for auto-correlations and may even be incorrect for cross-correlations, since it cannot capture any anti-correlation.

We therefore address here the issue of accurately and quickly computing the integrals required to derive the correlations between tomographic bins. Our goal, in computational terms, is that this computation be faster than one typical Boltzmann code computation time, that is, essentially at or below the 1s level (on eight cores). Another aspect of this work is to provide a

⋆ The C++ code is available from https://gitlab.in2p3.fr/campagne/AngPow

[1] http://class-code.net

stand-alone library that offers a generic interface where the user can plug any matter power spectrum. This is a different approach from `CLASSgal` (Di Dio et al. 2013) which also provides some theoretical computations related to tomography that are deeply rooted within the `CLASS` software.

The integrals defining the $C_\ell(z_i, z_j)$ angular power spectra are introduced in Sect. 2. In Sect. 3 we detail the algorithm implemented in `Angpow`, while we address some numerical tests in Sect. 4. Section 5 provides insight into the code design and we conclude in Sect. 6.

2. The position of the problem

Our aim is to compute the angular over density power spectrum $C_\ell(z_1, z_2)$ as a cross-correlation between two z-shells with mean values (z_1, z_2) and also the auto-correlation $C_\ell(z_1)$ with $z_1 = z_2$, taking into account, in both cases, possible redshift selection functions. Following notations of reference (Di Dio et al. 2013), for a couple of redshift (z_1, z_2) one computes $C_\ell(z_1, z_2)$ according to

$$C_\ell(z_1, z_2) = \frac{2}{\pi} \int_0^\infty \mathrm{d}k\, k^2\, P(k) \Delta_\ell(z_1, k) \Delta_\ell(z_2, k), \tag{1}$$

with on one hand $P(k)$ the non-normalized primordial power spectrum, and on the other hand, $\Delta_\ell(z, k)$, a general function used to describe physical processes down to redshift z (Di Dio et al. 2013; Bonvin & Durrer 2011). At the lowest order, $\Delta_\ell(z, k)$ can be expressed as the product of the bias b and a growth factor $D(z, k)$ to account for matter density contribution:

$$\Delta_\ell^{\mathrm{mat.}}(z, k) = bD(z, k) j_\ell(k\, r(z)), \tag{2}$$

where $j_\ell(x)$ is a first kind spherical Bessel function of parameter ℓ, and $r(z)$ is the radial comoving distance of the shell located at redshift z.

For thick redshift shells, one introduces two normalized redshift selection functions $W_1(z; z_1, \sigma_1)$ and $W_2(z'; z_2, \sigma_2)$ around z_1 and z_2 with typical width σ_1 and σ_2, respectively. Then, one extends Eq. (1) by

$$C_\ell^{\mathrm{thick}}(z_1, z_2; \sigma_1, \sigma_2) = \frac{2}{\pi} \iint_0^\infty \mathrm{d}z\, \mathrm{d}z'\, W_1(z; z_1, \sigma_1) W_2(z'; z_2, \sigma_2) \times \int_0^\infty \mathrm{d}k\, k^2\, P(k) \Delta_\ell(z, k) \Delta_\ell(z', k). \tag{3}$$

It is convenient to introduce the auxiliary function justified in the following section

$$\begin{aligned} f_\ell(z, k) &\equiv \sqrt{\frac{2}{\pi}}\, k \sqrt{P(k)}\, \Delta_\ell(z, k) \\ &= \sqrt{\frac{2}{\pi}}\, k \sqrt{P(k)} \left\{ bD(z, k) j_\ell(k\, r(z)) + \dots \right\} \\ &= \sqrt{\frac{2}{\pi}}\, k \sqrt{P(k) D(z, k)^2} \left\{ b\, j_\ell(k\, r(z)) + \dots \right\} \\ &\equiv \sqrt{\frac{2}{\pi}}\, k \sqrt{P(k, z)}\, \widetilde{\Delta}_\ell(z, k), \end{aligned} \tag{4}$$

where we have used the factorization of the growth factor $D(k, z)$ from the matter density contribution to introduce the notation $P(k, z)$ and $\widetilde{\Delta}_\ell(z, k)$. The dots signify that other contributions may be introduced as the redshift distortions and lensing effects that we ignore here for simplicity. Then,

$$C_\ell^{\mathrm{thick}}(z_1, z_2; \sigma_1, \sigma_2) = \iint_0^\infty \mathrm{d}z\mathrm{d}z'\, W_1(z; z_1, \sigma_1) W_2(z'; z_2, \sigma_2) \times \int_0^\infty \mathrm{d}k\, f_\ell(z, k) f_\ell(z', k). \tag{5}$$

The auto-correlation is a particular case where the redshift selection function $W_2(z'; z_2, \sigma_2)$ is reduced to $W_1(z'; z_1, \sigma_1)$ and we can use a single W function, which leads to

$$C_\ell^{\mathrm{thick}}(z_1; \sigma_1) = \iint_0^\infty \mathrm{d}z\ \mathrm{d}z'\ W(z; z_1, \sigma_1) W(z'; z_1, \sigma_1) \times \int_0^\infty \mathrm{d}k\, f_\ell(z, k) f_\ell(z', k). \tag{6}$$

To account for infinite redshift precision at $z = z_1$, the use of a Dirac selection function for W yields

$$C_\ell^\delta(z_1) = \frac{2}{\pi} \int_0^\infty \mathrm{d}k\, k^2 P(k, z_1) \widetilde{\Delta}_\ell^2(z_1, k). \tag{7}$$

`Angpow` is designed to efficiently compute these C_ℓ expressions once one provides access to the power spectra $P(k, z)$, the $\tilde{\Delta}_\ell(z, k)$ extra function, and the cosmological distance $r(z)$. To simplify the notation, we do not write the width σ of the selection functions if not explicitly needed.

3. A brief description of the computational algorithm

The redshift integral computations of Eq. (5) can be conducted in practice inside the rectangle $[z_{1\min}, z_{1\max}] \times [z_{2\min}, z_{2\max}]$ given by the W selection functions using a Cartesian product of a one-dimensional (1D) quadrature defined by the set of sample nodes z_i and weights w_i. In practice, we use the Clenshaw-Curtis quadrature. 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 $i = 0, \dots, N_{z_1} - 1$ and $j = 0, \dots, N_{z_2} - 1$. Then, one gets the following approximation:

$$C_\ell^{\mathrm{thick}}(z_1, z_2) \approx \sum_{i=0}^{N_{z_1}-1} \sum_{j=0}^{N_{z_2}-1} w_i w_j W_1(z_i, z_1) W_2(z_j, z_2) \widehat{P}_\ell(r_i, r_j) \tag{8}$$

with the notations $z_i = z_{1i}$, $z_j = z_{2j}$ and $r_i = r(z_{1i})$, $r_j = r(z_{2j})$ and

$$\widehat{P}_\ell(z_i, z_j) = \int_0^\infty \mathrm{d}k\, f_\ell(z_i, k) f_\ell(z_j, k), \tag{9}$$

defined with the $f_\ell(z, k)$ function of Eq. (4).

We use a piecewise integration over a user-defined range $[k_{\min}, k_{\max}]$ such that

$$\widehat{P}_\ell(z_i, z_j) \approx \sum_{p=0}^{N_k - 1} I_\ell(k_p^\ell, k_{p+1}^\ell; z_i, z_j), \tag{10}$$

where the k_p^ℓ bounds are related to the roots of $j_\ell(x)$ noted $q_{\ell p}$ and the user-defined number of roots $q_{\ell p}$ per sub-interval $[k_p^\ell, k_{p+1}^\ell]$

(see Appendix A). Then, Eq. (8) may be rewritten as

$$C_\ell^{\rm thick}(z_1,z_2) \approx \sum_{i=0}^{N_{z_1}}\sum_{j=0}^{N_{z_2}} w_i w_j W_1(z_i,z_1) W_2(z_j,z_2) \times \sum_{p=0}^{N_{\rm k}-1} I_\ell(k_p^\ell, k_{p+1}^\ell; z_i, z_j). \quad (11)$$

The integral $I_\ell(k_p^\ell, k_{p+1}^\ell; r_i, r_j)$ defined as

$$I_\ell(k_p^\ell, k_{p+1}^\ell; r_i, r_j) = \int_{k_p^\ell}^{k_{p+1}^\ell} {\rm d}k\; f_\ell(k, z_i) f_\ell(k, z_j), \quad (12)$$

is computed using the 3C-algorithm of appendix A. Investigating the different steps of the algorithm, one notices that the sampling of the function $f_\ell(k, z_i)$ along the k axis for a given $[k_p^\ell, k_{p+1}^\ell]$ interval depends on z_i but is independent from z_j and vice versa for the $f_\ell(k, z_j)$ function (those samplings are independent from z_i). So, one may proceed to k-sampling before performing the double sum over (i, j), which is particularly efficient as the CPU bottleneck is the computation of the spherical Bessel function j_ℓ. Angpow uses a spherical Bessel function implementation based on Numerical Recipes (Press et al. 1992). The brute force Cartesian quadrature evolves as $O(N_{z_1} \times N_{z_2})$, while the optimized version reduces the CPU times scaling to $O(N_{z_1} + N_{z_2})$. Therefore, as a matter of efficiency, it is more appropriate to exchange the order of the p and (i, j) summations leading to

$$C_\ell^{\rm obs}(z_1,z_2) \approx \sum_{p=0}^{N_{\rm k}-1}\sum_{i=0}^{N_{z_1}}\sum_{j=0}^{N_{z_2}} w_i w_j W(z_i,z_1) \times W(z_j,z_2) I_\ell(k_p, k_{p+1}; z_i, z_j). \quad (13)$$

As a first hint for the 3C-algorithm, we use typically 2^8-2^9 polynomial approximations for $\ell_{\rm max} \approx 500$–$1000$ of the $f_\ell(k, r_i)$ and $f_\ell(k, r_j)$ functions, 100 spherical Bessel roots per sub-interval, 99-point Clenshaw-Curtis quadrature nodes, and weights for z integration in case of $\sigma = 0.02$.

4. Numerical tests

4.1. Comparison with other codes

We proceed now to a numerical comparison of the estimation of $C_\ell(z_1)$ and $C_\ell(z_1, z_2)$ computed by CLASSgal (Di Dio et al. 2013) and Mathematica (Wolfram Research Inc. 2016) with Dirac redshift selection functions. Concerning CLASSgal (v1.1.3), we have started with the provided explanatory.ini file where we have modified the cosmological parameters to: $h = 0.679$, $\Omega_{\rm b} = 0.0483$, $\Omega_{\rm cdm} = 0.2582$ and $\Omega_{\rm k} = \Omega_{\rm fld} = 0$. We have also set k_scalar_max_tau0_over_l_max to fix the upper bound of the k-integration taking into account the maximal ℓ value and the redshift mean value. Concerning Angpow we have taken advantage of the possibility to read an external file produced by CLASSgal as an input $P(k)$ computed at $z = 0$ in association to the growth function computed internally given by Lahav et al. (1991), Carroll et al. (1992). Finally, to avoid the Limber's approximation for CLASSgal, we have set the "Limber" threshold much higher than the ℓ upper limit.

Results of the auto-correlation $C_\ell^\delta(z)$ computations at $z = 1$ using Dirac selection functions are shown in Fig. 1. As the three softwares use the same primordial power spectrum, all the results agree with one other within a maximal relative error of 0.06% on the whole ℓ range.

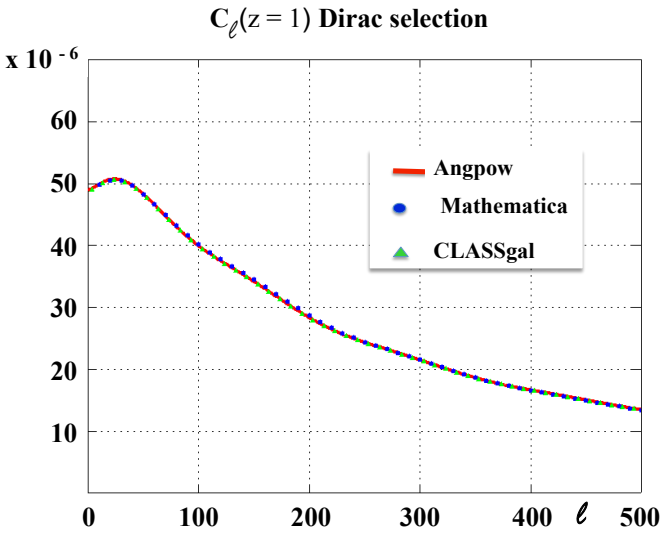

Fig. 1. Comparison of the computations of the $C_\ell^\delta(z = 1)$ given by Angpow, Mathematica, and CLASSgal for a Dirac selection function with $k_{\rm max} = 10\ {\rm Mpc}^{-1}$.

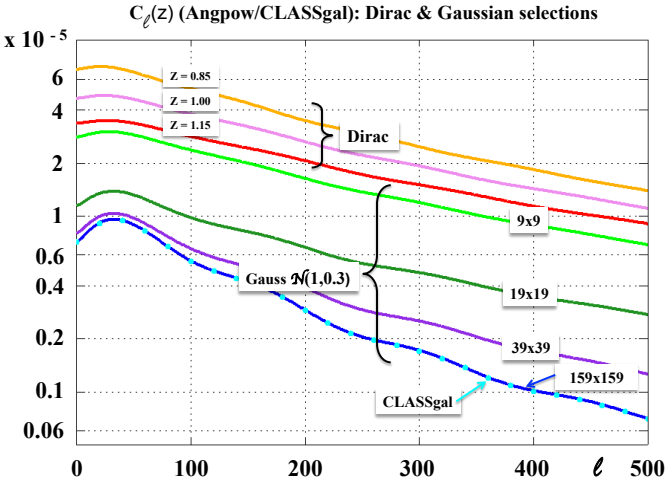

Fig. 2. Computations of $C_l^\delta(z)$ with Dirac selections centered at $z \in \{0.85, 1.00, 1.15\}$ with Angpow alone and $C_l^{\rm thick}(z)$ with Angpow and CLASSgal using a Gaussian selection function with mean $z = 1$, a width of $\sigma = 0.3$, and a redshift cut at $\pm 5\sigma$ (in all cases $k_{\rm max} = 0.44\ {\rm Mpc}^{-1}$). For Angpow we use the power spectrum produced by CLASSgal and we have varied the redshift grid sampling resolution from 9×9 to 159×159 points to reach the converged result (blue curve) in good agreement with the CLASSgal result (cyan points). Comparing the $C_l^\delta(z = 1)$ to the $C_l^{\rm thick}(z = 1, \sigma = 0.3)$ results we measure the effect of self-cross-correlation inside a thick redshift shell which washes out the matter fluctuation contrast.

The $C_\ell^{\rm thick}(z)$ auto-correlation computation within a thick redshift band, selected by a Gaussian of mean $z = 1$ and $\sigma = 0.03$ cut at $\pm 5\sigma$, has been used as a test bench. But, for this test we have neglected the Mathematica software, which is too slow, and have restricted testing to comparison of Angpow to CLASSgal. Figure 2 shows computation results. The orange, violet, and red curves are produced by Angpow using Dirac selection function in the range $\pm 5\sigma$ around the mean redshift $z = 1$, while the green, forest green, purple, and blue curves are results of the above mentioned Gaussian selection function but sampled using

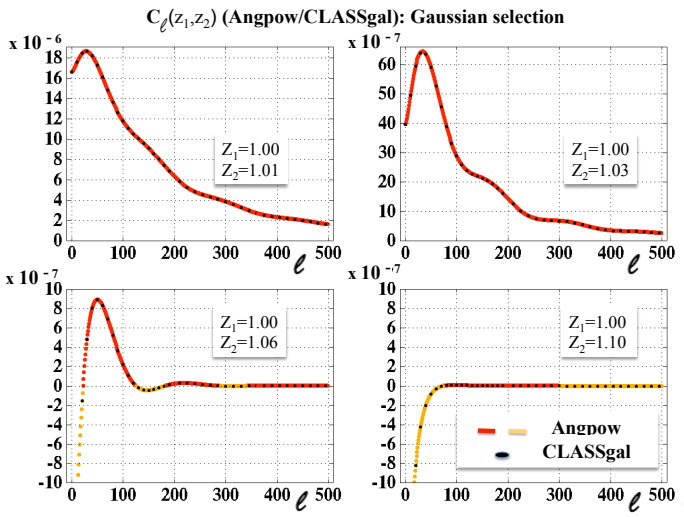

Fig. 3. Comparison between `Angpow` (red/orange points) and `CLASSgal` (black points) for several cross-correlation spectra with Gaussian (σ = 0.01) selection and $k_{\max} = 1$ Mpc^{-1}. The orange points are used to emphasize negative C_ℓ.

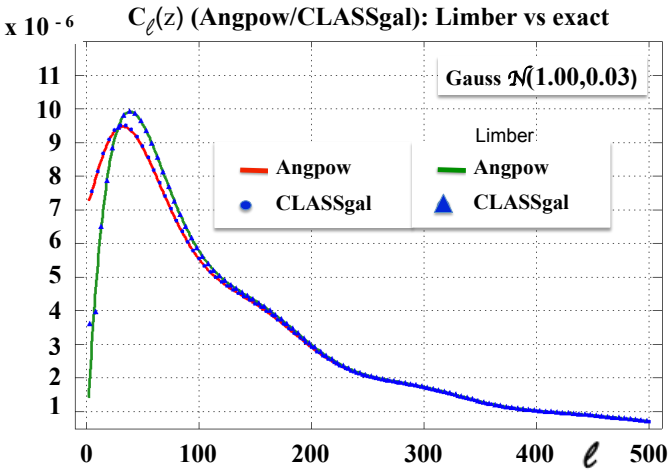

Fig. 4. Comparison of the computations of the $C_\ell^{\rm thick}(z)$ given by `Angpow` and `CLASSgal` either with the Limber's approximation or the exact computation.

different grid sizes: 9×9, 19×19, 39×39 and 159×159 Clenshaw-Curtis sample points. As the number of points, or equivalently the quadrature order, increases, the $C_\ell^{\rm thick}(z)$ computation converges towards the `CLASSgal` result (cyan points). We also address the cross-correlation computations performed by `Angpow` and `CLASSgal` ($C_\ell^{\rm thick}(z_1, z_2)$) using Gaussian selection functions ($\sigma = 0.01$ and a $\pm 5\sigma$ cut). The results are shown in Fig. 3. One should not be surprised by negative values since we are cross-correlating two different quantities. In both tests we have used the power spectrum computed at $z = 0$ by `CLASSgal` as input to `Angpow`. The agreement between the two software codes is very good, keeping the relative residuals at values less than 0.02%.

4.2. Note on Limber's approximation

The `Angpow` library can also be used to compute, if desired, the first order Limber's approximation (Loverde & Afshordi 2008). In such an approximation, the spherical Bessel function is formally reduced to

$$j_\ell(x) \approx \sqrt{\frac{\pi}{2\ell+1}}\delta^D\left(x - \left(\ell + \frac{1}{2}\right)\right). \tag{14}$$

In such conditions, the product $kr(z)$ is constrained, and if one uses the following notation for the comoving distance computation with $d_{\rm H} = c/H_0$, the Hubble distance ($H_0 = 100h$ km s^{-1} Mpc^{-1} and c the speed of light) and $E(z)$, the dimensionless Hubble parameter,

$$r(z_\ell(k)) = \frac{l+1/2}{k} = d_{\rm H}\int_0^{z_\ell(k)}\frac{{\rm d}z}{E(z)}. \tag{15}$$

Then, Eq. (5) is transformed to the following expression

$$C_\ell^{\rm thick}(z_1, z_2; \sigma_1, \sigma_2) \approx \frac{2}{d_{\rm H}^2(2\ell+1)}\int_0^\infty {\rm d}k\; W_1(z_\ell(k); z_1, \sigma_1) \times W_2(z_\ell(k); z_2, \sigma_2)E^2(z_\ell(k))P(k, z_\ell(k)). \tag{16}$$

This integral can be computed using a divide-and-conquer recursive method with the Gauss-Kronrod quadrature (Laurie 1997). The Gauss sample points are a subset of the Gauss-Kronrod sample points and can easily be used to set up an error estimate to drive the recursive algorithm.

Looking at Eq. (16) one realizes that all the terms are positive, indicating that such approximation is not suitable for cross-correlation where $C_\ell(z_1, z_2)$ is not guaranteed to be positive as can be explicitly seen in Fig. 3. We have proceeded to the computation of $C_\ell(z)$ in the case of a Gaussian selection function of mean $z = 1$ and $\sigma = 0.03$ for both `Angpow` and `CLASSgal`, with/without the Limber's approximation. The results are shown in Fig. 4. The two software codes agree very well and show that the Limber's approximation can give sizeable errors compared to the exact computation; of the order of the cosmic variance in the given example. So, this Limber's approximation, even if it runs 100 times faster than the exact computation, should then be used with extreme care not only for cross-correlation but also for auto-correlation.

4.3. Correlations in real space

`Angpow` can also quickly compute the correlation function in real space from the power spectrum

$$C(\theta; z_1, z_2) = \frac{1}{4\pi}\sum_{\ell=0}^{\ell_{\max}}(2\ell+1)\tilde{C}_\ell(z_1, z_2)P_\ell(\cos\theta), \tag{17}$$

where P_ℓ denotes the ℓth order Legendre polynomial. Because the $C_\ell(z_1, z_2)$ values are generally cut at a given $\ell_{\max}$, one needs to introduce an apodization to avoid ringing due to a sharp cut-off. We introduce a Gaussian one (which is the smoothest in both real and harmonic spaces) as in Di Dio et al. (2013) so that the $\tilde{C}_\ell(z_1, z_2)$ term in Eq. (17) is

$$\tilde{C}_\ell(z_1, z_2) = C_\ell(z_1, z_2)e^{-\ell(\ell+1)/\ell_{\rm a}^2}. \tag{18}$$

The apodization length $\ell_{\rm a}$ may depend on the signal but for the standard cosmology shown here (around $z = 1$) we noticed that using $l_{\rm a} \simeq 0.4\ell_{\max}$ gives good results. Correlations in real space are generally easier to comprehend as is shownin Fig. 5, which represents the analogue of Fig. 3 but in real space. Here one may identify a peak, named the "Baryonic Acoustic Oscillation" (e.g., Weinberg et al. 2013) that decreases in the cross-correlation when the distance between shells increases and is finally washed out.

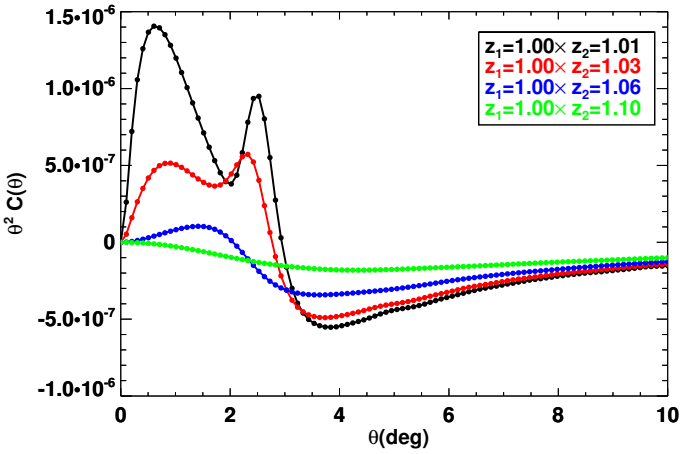

Fig. 5. Cross-correlations in real space corresponding to the spectra shown on Fig. 3. The points show where the function was evaluated.

4.4. Speed tests

Angpow is designed and written in C++ and parallelization is achieved through OpenMP. To qualify the code we provide four input parameter files and their corresponding outputs obtained in one of our runs. In all tests, we used a ΛCDM reference cosmology, a $P(k)$ at $z = 0$ provided by CLASSgal, $\ell_{\rm max} = 1000$, a 3C-algorithm with 2^9 Chebyshev polynomial order, and 100 roots per sub-k-interval:

Test 1: auto-correlation with a Dirac selection function at $z = 1$ and $k_{\rm max} = 10$ Mpc^{-1};

Test 2: cross-correlation with two Dirac selection functions at $z = 1$ and $z = 1.05$ and $k_{\rm max} = 10$ Mpc^{-1};

Test 3: auto-correlation with a Gaussian selection function with ($z_{\rm mean} = 1$, $\sigma_z = 0.02$, $5\sigma_z$-cut) and $k_{\rm max} = 1$ Mpc^{-1} and a radial quadrature based on $N_{\rm pts} = 139$ sample points;

Test 4: cross-correlation with two Gaussian selection functions with ($z_{\rm mean,1} = 0.90$, $z_{\rm mean,2} = 1.00$) both with ($\sigma_z = 0.02$, $5\sigma_z$-cut) and $k_{\rm max} = 1$ Mpc^{-1} and a radial quadrature based on $N_{\rm pts} = 139$ sample points.

We have tested the code both on laptop (Linux, MacOSX) as well as on Computer Center (CCIN2P3 in France and NERSC in the USA). We use OpenMP to distribute the computation of a given C_ℓ on a single thread. Table 1 gives Central Processing Unit (CPU) execution times averaged over ten processes. The code wall time decreases reasonably well with the number of threads and a wall time at the O(1s) level can be reached to reconstruct these accurate spectra. Such performances are much higher than those obtained with CLASSgal when not using the Limber approximation. For example, on our Test-3 setup, running the latter takes about 15s (on 16 threads), which is to be compared to about 0.5s in Table 1.

Figure 6 shows the dependence of the wall time with respect to the width of the selection function (σ) in the conditions of Test 3 using 16 threads. For a given σ, we have chosen the minimal radial_order value such that the relative accuracy on the C_ℓ is of the order of 0.01% compared to a computation with a much larger radial_order value (see Sect. 5). If one uses a looser criteria on the accuracy of the C_ℓ or if the number of sigma is lower than 5, then one may use a lower radial_order and gain on the wall time.

Table 1. Wall time (in seconds) measured at CCIN2P3 (on Intel Xeon CPU E5-2640 v3 processors) for the test benches described in the text, according to the number of OpenMP threads used.

# Threads	1	2	4	8	16
Linux/icpc					
Test 1	0.38	0.21	0.13	0.09	0.08
Test 2	0.76	0.41	0.23	0.15	0.11
Test 3	3.72	1.96	1.05	0.64	0.44
Test 4	9.97	5.25	2.79	1.60	1.01
Linux/gcc					
Test 1	0.56	0.30	0.17	0.12	0.09
Test 2	1.14	0.60	0.33	0.20	0.14
Test 3	5.01	2.59	1.38	0.81	0.50
Test 4	13.80	7.07	3.71	2.12	1.27

Notes. Results are given for the intel icpc 15.0 and gcc 5.2 compilers.

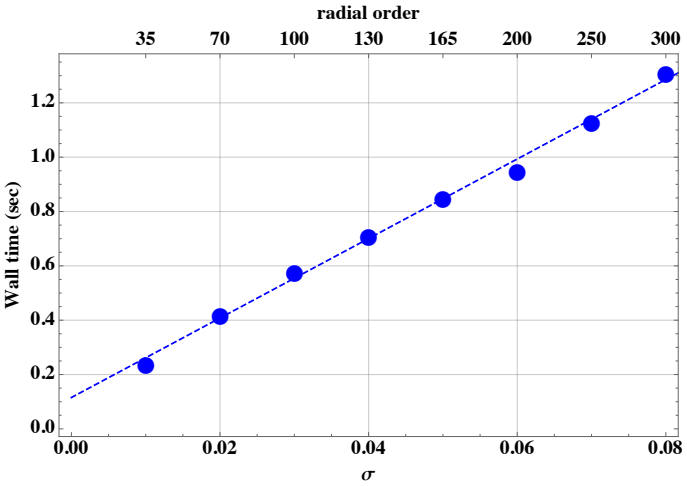

Fig. 6. Evolution of the wall time with respect to the width of the selection function (σ) illustrated in the condition of Test 3 and using 16 threads. In the upper scale of the frame is shown an indication of the radial_order used to sample the along the z direction for a given σ (see Sect. 5).

5. Code design and input parameters

Angpow is written in C++ which allows both good CPU performances and keeps the code flexible. A front end interface to Python is also foreseen and the code is distributed publicly[2]. The angpow.cc file is an example of the library usage to perform C_ℓ and $C(\theta)$ computations. We also provide the limber.cc file if one wants to test the Limber's approximation (Sect. 4.2). The different files are located in self-explained directories: src, inc/Angpow, lib, data. Finally, a README file provides details for the installation and build procedures.

The two main classes Pk2Cl and KIntegrator (located in angpow_pk2cl.h and angpow_kinteg.h files) are generic codes using abstract base classes. They define interface to the power spectrum function $P(\ell, k, z)$ (class PowerSpecBase), the generalisation of $P(k, z)$ used in Eq. (5); to the comoving distance computation $r(z)$ (class CosmoCoorBase); and to the

[2] https://gitlab.in2p3.fr/campagne/AngPow

radial/redshift selection functions $W(z)$ (class `RadSelectBase`). The user can derive their own concrete classes to access a third party library or use the ones implemented by default. For instance, we have implemented a file access to a $(k, P(k))$-tuple saved by the `CLASSgal` output. In this implementation, we have coded the growth factor defined in Lahav et al. (1991), Carroll et al. (1992) as minimal $\tilde{\Delta}_\ell(z, k)$ function (Eq. (2)).

To run the executable, one provides an ascii file defining the input parameters that drive the computational conditions of the algorithm and define the I/O locations. Some ready-made input parameter files are also provided (`angpow_bench<n>.ini`) as well as the C_ℓ output files (`angpow_bench<n>_cl.txt.REF`) corresponding to the `icpc` outputs of Table 1.

Among the different input parameters some are more sensitive than others, as those that deal with the radial/redshift 1D quadrature and the 3C algorithm. Here is a closer look at these parameters:

- `cl_kmax`: this is the maximal value of k in the k-integral. We have not set up an internal algorithm to determine this upper bound. As a hint, one may consider a relation with the factor $\ell_{\max}/r(z_{\min})$. The lower bound on k is internally fixed using the cut-off $x_{\min}(\ell)$ defined as $x < x_{\min}(\ell) \Leftrightarrow j_\ell(x) < \text{cut}$ (see the input parameters `jl_xmin_cut` and `Lmax_for_xmin` set as default to 5×10^{-10} and 2000, respectively).
- `radial_order`: if noted n, this fixes the number of sample points along one z direction, that is, $N_{\text{pts}} = 2n - 1$. The accuracy on the selection function as well as the CPU increase with n but we keep a $O(n)$ complexity of the k-sampling (see Fig. 6);
- `chebyshev_order`: if noted N, this fixes the degree d of the Chebyshev polynomial approximation of the $f_\ell(k, z_i)$ and $f_\ell(k, z_j)$ functions (Eq. (4)); that is, $d = 2^N$. Keeping the same degree of approximation for both functions guarantees a power of 2 for the product approximation. Even if not mandatory, this helps in getting CPU performance for the DCT-I transform (using the `FFTW` library). When $\ell_{\max}$ increases it may be worth updating this parameter by 1 unit step. For $\ell_{\max} = 500$ `chebyshev_order` is set to 8 by default. Increasing the angular spectrum computation up to $\ell_{\max} = 1000$ keeping this default value leads to absolute error of the order of 10^{-6} for Tests 1 and 2 and 5×10^{-10} for Test 3 and 4, then to get better accuracy in this case we use `chebyshev_order` = 9.
- `n_bessel_roots_per_interval`: this is the number of Bessel roots $q_{\ell p}$ used to define the bounds of the integral $I_\ell(k_p^\ell, k_{p+1}^\ell; r_i, r_j)$ (Eq. (12)). By default it is set to 100. There is an interplay with the `chebyshev_order` parameter as a lower `n_bessel_roots_per_interval` value is coherent with a lower `chebyshev_order`.
- `total_weight_cut` and `deltaR_cut`: these two threshold parameters are used to avoid unnecessary 3C algorithm processing (especially for the k-sampling of the $f_\ell(k, z_i)$ or $f_\ell(k, z_j)$ functions). So, we do not consider a couple (z_i, z_j) for which either the product $w_i w_j W(z_i, z_1) W(z_j, z_2)$ is too low (`total_weight_cut` cut) or the radial distance $|r(z_i) - r(z_j)|$ is too large to produce a sizeable contribution to the final C_ℓ. The `deltaR_cut` cut is in Mpc units and is used in conjunction with `has_deltaR_cut` set to 1. These two threshold parameters depend on the use case under consideration and for preliminary tests we recommend to set both `total_weight_cut` and `has_deltaR_cut` to 0.

6. Summary and outlooks

We have set up a fast and generic software to compute the tomographic $C_\ell(z_1, z_2)$ with redshift selection functions. `Angpow` is versatile enough to accept user-defined matter power spectrum $P(k)$, transfer functions $\tilde{\Delta}_\ell(k, z)$, and cosmology. The code provides an accurate computation of the auto and cross-correlation power spectra, checked by comparison with other codes, which is fast enough to be included inside MCMC cosmology softwares. The rapidity of the software relies on the use of the 3C-algorithm, adapted to the computation of integrals over spherical Bessel functions, while other codes rely on the Limber's approximation. We emphasize that the Limber's approximation can lead to incorrect $C_\ell(z_1, z_2)$, especially in the case of cross-correlations, as Limber's Dirac functions cannot model the interferences between two spherical Bessel functions at different redshifts.

This code is thus fast and accurate enough to be used to test cosmological parameters, and perform tomographic analysis of the galaxy distribution. The definition of the $\tilde{\Delta}_\ell(k, z)$ function is general and can accept zero order function as $\tilde{\Delta}_\ell^{\text{mat}}(k, z)$, but also relativistic corrections such as the redshift space distortions or the gravitational lensing; despite these corrections, it can also contain spherical Bessel functions (see e.g., Di Dio et al. 2013). Because `Angpow` provides the correct angular power spectra for cross-correlations, it can be a key tool to perform an integrated approach to cosmology, as advertised in Nicola et al. (2016). In particular, we propose this tool for deriving the auto- and cross-correlation angular power spectra for galaxy clustering, but also with angular power spectra from cosmic shear and the cosmological microwave background.

`Angpow` is publicly available[3] and can be interfaced to other codes; a Python interface is foreseen. At the moment the code only accepts two redshift bins but soon it will be generalized to any number of bins. Feedback from `Angpow` users would be greatly accepted.

Acknowledgements. The authors want to thank M. Reinecke who kindly provided pieces of the code, and J. D. McEwen for fruitful discussion on the Chebyshev transform.

References

Asorey, J., Crocce, M., Gaztañaga, E., & Lewis, A. 2012, MNRAS, 427, 1891
Asorey, J., Crocce, M., & Gaztañaga, E. 2014, MNRAS, 445, 2825
Baszenski, G., & Tasche, M. 1997, Linear Algebra and its Application, 252, 1
Blas, D., Lesgourgues, J., & Tram, T. 2011, JCAP, 2011, 034
Bonvin, C., & Durrer, R. 2011, Phys. Rev. D, 84, 063505
Carroll, S. M., Press, W. H., & Turner, E. L. 1992, ARA&A, 30, 499
Di Dio, E., Montanari, F., Lesgourgues, J., & Durrer, R. 2013, JCAP, 11, 044
Di Dio, E., Montanari, F., Durrer, R., & Lesgourgues, J. 2014, JCAP, 1, 042
Feldman, H. A., Kaiser, N., & Peacock, J. A. 1994, ApJ, 426, 23
Frigo, M., & Johnson, S. G. 2005, special issue on "Program Generation, Optimization, and Platform Adaptation, Proc. IEEE, 93, 216
Giorgi, P. 2012, IEEE Trans. Comput., 61, 780
Glasser, M. L., & Montaldi, E. 1993, ArXiv e-prints [arXiv:math/9307213]
Ivezic, Z., Tyson, J. A., Abel, B., et al. 2008, ArXiv e-prints [arXiv:0805.2366]
Lahav, O., Lilje, P. B., Primack, J. R., & Rees, M. J. 1991, MNRAS, 251, 128
Landy, S. D. & Szalay, A. S. 1993, ApJ, 412, 64
Lanusse, F., Rassat, A., & Starck, J.-L. 2015, A&A, 578, A10
Laureijs, R., Amiaux, J., Arduini, S., et al. 2011, ArXiv e-prints [arXiv:1110.3193]

[3] From `https://gitlab.in2p3.fr/campagne/AngPow`

Laurie, D. P. 1997, Math. Comput., 466, 1133
Levi, M., Bebek, C., Beers, T., et al. 2013, ArXiv e-prints [arXiv:1308.0847]
Loverde, M., & Afshordi, N. 2008, Phys. Rev. D, 78, 123506
Lucas, S., & Stone, H. 1995, J. Comput. Appl. Math., 64, 217
Nicola, A., Refregier, A., Amara, A., & Paranjape, A. 2014, Phys. Rev. D, 90, 063515
Nicola, A., Refregier, A., & Amara, A. 2016, Phys. Rev. D, 94, 083517
Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992, Numerical Recipes in C, 2nd edn.: The Art of Scientific Computing (New York: Cambridge University Press)
Waldvogel, J. 2006, BIT Numerical Mathematics, 46, 195
Weinberg, D. H., Mortonson, M. J., Eisenstein, D. J., et al. 2013, Phys. Rep., 530, 87
Wolfram Research Inc. 2016, Mathematica 11.0, Champaign, Illinois, USA

Appendix A: Clenshaw-Curtis-Chebyshev algorithm (3C-algorithm)

Each integral type of Eq. (12) involves the product of "highly" oscillatory functions. The purpose of this section is not to provide a review of all the integration methods used in the different fields of physics to tackle such a difficult task. To focus on our case, where we have to deal with (at least) the product of spherical Bessel functions, the authors point out that the reader may find either specific integral solving rules as in Glasser & Montaldi (1993) or general methods as in Lucas & Stone (1995). However we need a precise and also very fast method. We cannot rely on methods that solve the problem of a product of spherical Bessel functions multiplied by a regular function. In fact, both the primordial power spectrum and the extension beyond the matter density $\tilde{\Delta}^{\rm mat.}(z,k)$ may show oscillation features in the form of derivative of spherical Bessel functions. So, we have searched and found a general method that meets our requirements of precision and speed.

Equation (12) is a special case of the following generic integral after a proper change of variable

$$I = \int_{-1}^{1} \mathrm{d}x f(x) g(x). \tag{A.1}$$

To get an approximate value of this integral, we use in this section the Clenshaw-Curtis quadrature at order $N_{\rm cc}$ (noting $h = f \times g$):

$$I \approx \sum_{k=0}^{N_{\rm cc}} w_k f(x_k) g_(x_k) = \sum_{k=0}^{N_{\rm cc}} w_k h(x_k), \tag{A.2}$$

where the sampling points are defined as $x_k = \cos k\pi/N_{\rm cc}$ ($k = 0, \ldots, N_{\rm cc}$) and the corresponding weights w_k are addressed later in the section. But, if the functions f and g have a highly oscillatory behavior, one needs, in principle, to use large values of $N_{\rm cc}$ to reach a sufficient accuracy level. In that case, dealing with the above sum may not be computationally efficient. The idea is then to use Chebyshev series to approximate both functions f and g. Then, one performs the product of both Chebyshev series, which is also a Chebyshev series but with a higher order, and finally one uses a fast Clenshaw-Curtis weights computation to perform the last weighted sum. We briefly describe those steps leaving the details of the demonstration that the interested reader can find in Baszenski & Tasche (1997).

Let f_N be a polynomial approximation of f of degree $N-1$. We expend f_N onto the following basis of the first kind of Chebyshev polynomials $\{T_n; n = 0, \ldots, N-1\}$ which have the property $T_n(\cos\theta) = \cos n\theta$:

$$f_N(x) = \frac{a_0}{2} + \sum_{k=1}^{N-1} a_k T_k(x). \tag{A.3}$$

To determine the a_k coefficients one uses the following sampling vector

$$\boldsymbol{f}^{(N)} = (f(t_\mu^{(N)}))^T \qquad \text{with} \qquad t_\mu^{(N)} \equiv \cos\frac{\mu\pi}{N};\ \mu = 0, \ldots, N, \tag{A.4}$$

of length $N+1$ and related to the vector $\boldsymbol{a}^{(N)} = (a_0, \ldots, a_{N-1}, 0)^T$ by the linear algebra relation

$$\boldsymbol{a}^{(N)} = \frac{2}{N} \boldsymbol{C}_N^I \boldsymbol{f}^{(N)}, \tag{A.5}$$

with $\boldsymbol{C}_N^I$ being a discrete cosine transform of type-I (DCT-I) matrix of dimension $(N+1)^2$. Similarly, we note g_M a polynomial approximation of g of degree $M-1$ from which we determine the sampling vector $\boldsymbol{g}^{(M)}$ using the sample points $t_\mu^{(M)}$. The coefficient vector $\boldsymbol{b}^{(M)} = (b_0, \ldots, b_{M-1}, 0)^T$ is related to $\boldsymbol{g}^{(M)}$ using a relation similar to Eq. (A.5), namely

$$\boldsymbol{b}^{(M)} = \frac{2}{M} \boldsymbol{C}_M^I \boldsymbol{g}^{(M)}. \tag{A.6}$$

By combining the polynomial approximations f_N and g_M, the function h is then approximated by a Chebyshev series of degree $N+M-2$ with coefficient vector $\boldsymbol{c}^{(P)}$ of length $P+1$ with $P \geq N+M-1$. Using a relation of type Eq. (A.5), the vector $\boldsymbol{c}^{(P)}$ is related to the sampling vector

$$\boldsymbol{h}^{(P)} = \left(h\left(t_\mu^{(P)}\right)\right)^T;\ \mu = 0, \ldots, P. \tag{A.7}$$

To get $\boldsymbol{h}^{(P)}$ it is not necessary to compute $\boldsymbol{c}^{(P)}$ and proceed to an inversion of a DCT-I matrix. The key point is that if we note $\odot$, the component-wise multiplication, one has

$$\boldsymbol{h}^{(P)} = \boldsymbol{f}^{(P)} \odot \boldsymbol{g}^{(P)}. \tag{A.8}$$

Moreover, $\boldsymbol{f}^{(P)}$ ($\boldsymbol{g}^{(P)}$) is obtained from $\boldsymbol{a}^{(N)}$ ($\boldsymbol{b}^{(M)}$) of length $N+1$ ($M+1$) by an extension to a larger vector at least of length $N+M$ noted $\tilde{\boldsymbol{a}}^{(P)}$ ($\tilde{\boldsymbol{b}}^{(P)}$) by appending with zeros:

$$\begin{aligned} \tilde{\boldsymbol{a}}^{(P)} &= (\boldsymbol{a}^{(N)}, 0, \ldots, 0), \\ \tilde{\boldsymbol{b}}^{(P)} &= (\boldsymbol{b}^{(M)}, 0, \ldots, 0). \end{aligned} \tag{A.9}$$

Then, the sampling vector used in Eq. (A.2) where one identifies $N_{\rm cc} = P$ is determined by

$$\boldsymbol{h}^{(N_{\rm cc})} = \left(\boldsymbol{C}_{N_{\rm cc}}^I\, \tilde{\boldsymbol{a}}^{(N_{\rm cc})}\right) \odot \left(\boldsymbol{C}_{N_{\rm cc}}^I\, \tilde{\boldsymbol{b}}^{(N_{\rm cc})}\right), \tag{A.10}$$

using $\boldsymbol{C}_{N_{\rm cc}}^I$ the DCT-I matrix of dimension $(N_{\rm cc}+1)^2$ and the inversion property $(\boldsymbol{C}_P^I)^{-1} = (2/P)\boldsymbol{C}_P^I$. In some sense, for both f and g approximation sampling vectors, we have performed a Chebyshev basis change to a larger parameter space compatible with the polynomial degree involved in the product $f^{(N)} \times g^{(M)}$.

The second key point is that the Clenshaw-Curtis weights associated to $\boldsymbol{h}^{(N_{\rm cc})}$ in Eq. (A.2) can also be computed with a DCT-I transform from the vector $(2/N_{\rm cc})(1, 0, -1/3, 0, -1/15, \ldots, ((-1)^k + 1)/2(1 - k^2), \ldots)$ of length $N_{cc}+1$ (Waldvogel 2006) (the normalization depends on the exact definition of the DCT-I used).

So, to perform the integral given by Eq. (A.2), one needs $4+1$ DCT-I transforms, 1 for the Clenshaw-Curtis weights and 4 to transform the Chebyshev coefficients vectors. The DCT-I transform may be implemented using $O(n \log n)$ efficient algorithm, for example, the FFTW library (Frigo & Johnson 2005) used by `Angpow`. `Angpow` uses a power of 2 for N, M, and also P (keeping $P \geq N+M-1$) for a fast implementation. The 3C-algorithm is a special case of a more general class of algorithms dealing with the product of polynomials. We note that according to reference (Giorgi 2012) an even faster algorithm (although moderate) might be implemented in a future version of `Angpow` if necessary. We note finally that this general method can be applied to use cases beyond the power spectrum computation in other fields of interest.