Study of the correlation function

by Álvaro Méndez R.T.

June 2025

Contents

Abstract

In this article we show that the correlation function may indicate some new tensions with the standard model ΛCDM, especialy with a flat geometry.

1 Introduction

2 Methodology

In this section we will discuss all the steps to reproduce these calculations.

2.1 Functions and statistics

The correlation function is defined like

$$C(\theta) = \sum_l \frac{2l+1}{2l(l+1)} D_l P_l(\cos(\theta)) \tag{1}$$

where $D_l =$

Code 1: Code to calculate the correlation function

```
def correlation_func(D_ell,xvals):
    # It uses (l+2) instead of l, since the first 2 multipoles
        are not available
    fac2= [(2*(l+2)+1)/(2*(l+2)*((l+2)+1))*c for l,c in
        enumerate(D_ell)]
    cor =0
    for l,f in enumerate(fac2):

        cor+=f*P(l+2,xvals) # P(l+2,xvals) is the Legendre
            polynomial of degree l+2 evaluated at xvals

    return cor
```

Once the correlation function is calculated

$$\langle\bar{\xi}\rangle_a^b = \frac{\int_a^b C(\theta) d\cos(\theta)}{b-a} \tag{2a}$$

$$= \int_a^b \frac{1}{b-a} \sum_l \frac{2l+1}{2l(l+1)} D_l P_l(\cos(\theta)) d\cos(\theta) \tag{2b}$$

$$= \sum_l \frac{2l+1}{2l(l+1)} D_l \int_a^b \frac{1}{b-a} P_l(\cos(\theta)) d\cos(\theta) \tag{2c}$$

$$= \sum_l \frac{2l+1}{2l(l+1)} D_l \frac{1}{b-a} \left[\frac{P_{l+1}(\cos(\theta)) - P_{l-1}(\cos(\theta))}{2l+1} \right]_a^b \tag{2d}$$

$$= \sum_l \frac{D_l}{2l(l+1)} \frac{1}{b-a} \left[P_{l+1}(\cos(\theta)) - P_{l-1}(\cos(\theta)) \right]_a^b \tag{2e}$$

Code 2: Code to calculate the new statistic

```
def xivar(D_ell, a,b):
    s=0
    for i,d in enumerate(D_ell):
        l=i+2
        fac= d / (2*l*(l+1))
        term =Decimal(legendre(l +1, b) -legendre(l -1,
            b)-legendre(l +1, a) +legendre(l -1, a))
        s+=fac*float(term)

    return s/(b-a)
```

The C_{180} is the value of the two points correlation function when they are separated 180°, this statistic has the intention of avoiding the discussion of the mask used to smooth the data and clean the noise.

$$C_{180} = C(180) \tag{3}$$

$$S_a^b = \int_a^b C(\theta)^2 \sin(\theta) d\theta = \int_a^b \left[\sum_l \frac{2l+1}{2l(l+1)} D_l P_l(\cos(\theta)) \right]^2 d\cos(\theta) \tag{4a}$$

$$= \sum_n \sum_m \frac{2n+1}{2n(n+1)} D_n \frac{2m+1}{2m(m+1)} D_m \underbrace{\int_a^b P_n(\cos(\theta)) P_m(\cos(\theta)) d\cos(\theta)}_{T_{nm}} \tag{4b}$$

Code 3: Code to calculate the old statistic

```
def S12(D_ell,M):
    # Set the precision high enough to handle large calculations
    # M is the T_mn matrix
    getcontext().prec =1000
    s=Decimal(0)
    for i,xn in enumerate(D_ell):
        n=i+2
        fac1=(((2*n+1)*xn)/(2*n*(n+1)))
        for j,xm in enumerate(D_ell):
            m=j+2
            fac2=(((2*m+1)*xm)/(2*m*(m+1)))

            integral=M[i,j]

            s += Decimal(fac1)*Decimal(fac2)*Decimal(integral)

    return float(s)
```

Where the matrix elements of T_{nm} can be obtained with the following expression:

$$T_{mn} = \int_a^b P_n(\cos(\theta)) P_m(\cos(\theta)) d\cos(\theta) \tag{5}$$

$$= \frac{A_r \cdot A_{m-r} \cdot A_{n-r}}{A_{m+n-r}(2m+2n-2r+1)} \left[(P_{m+n-2r+1}(b) - P_{m+n-2r-1}(b)) - (P_{m+n-2r+1}(a) - P_{m+n-2r-1}(a)) \right]$$

with $A_r = \frac{1 \cdot 3 \cdot 5 \cdots (r-1)}{r!}$

Code 4: Code to calculate the matrix elements

```
def A_r(r):
    """Compute A_r using Decimal for high precision."""
    numerator =Decimal(1)
    for i in range(1, r+1):
        numerator *=Decimal(2*i-1)
    denominator =factorial(r)
    return numerator / denominator


def Tmn(l,l1,l2,a=-1,b=1/2):
    # Set the precision high enough to handle large calculations
    getcontext().prec =1000

    matrix =np.zeros((l, l), dtype=float)
    for i in range(l):
        n=i+2

        for j in range(l):
            m=j+2

            for r in range(min(m,n)+1):
                integral=A_r(r)*A_r(m-r)*A_r(n-r)/A_r(m+n-r)
                    /(2*m+2*n-2*r+1) *Decimal((legendre(m+n-2*r +
                    1, b) -legendre(m+n-2*r -1,
                    b))-(legendre(m+n-2*r +1, a) -legendre(m+n-2*r
                    -1, a)))
                matrix[i,j]+=np.float64(integral)

    np.save(f"Tmn_{l1}_{l2}.npy", matrix)
```

2.2 Experimental values

The experimental values were gathered from the Planck's data official site[1]. The data used are the observed multipoles of the CMB that were measured according to (Aghanim et al. 2020) assuming that the errors are given with a significance of 1σ. The treatment of the Planck's mission data is not for us to discuss.

2.3 Simulations

The simulations were made using the *"Code for Anisotropies in the Microwave Background"* (Lewis n.d.) with the following parameters for each simulation:

- `ombh2`: physical baryon density ($\Omega_b h^2$)

[1] That can be found here (Planck Collaboration 2015)

- `omch2`: physical cold dark matter density ($\Omega_c h^2$)
- `H0`: Hubble constant today (H_0), in $\mathrm{km{\cdot}s^{-1}{\cdot}Mpc^{-1}}$
- `omk`: curvature density parameter (Ω_k)
- `YHe`: helium mass fraction (Y_{He})
- `nnu`: effective number of neutrino species (N_{eff})
- `nrun`: running of the scalar spectral index ($dn_s/d\ln k$)
- `Alens`: phenomenological lensing amplitude ($\mathrm{A_{lens}}$)
- `ns`: scalar spectral index (n_s)
- `As`: amplitude of the primordial curvature perturbations (A_s), here expressed as $\exp(\mathrm{logA}) \times 10^{-10}$
- `w`: dark energy equation of state parameter $w = -1$[2]
- `wa`: evolution of dark energy equation of state (w_a)
- `mnu`: sum of neutrino masses ($\sum m_\nu$), in eV
- `tau`: optical depth to reionization (τ)

In order to simulate the correlation function the following steps were made:

1. We get the last thousand values if the MCMC chain in order to get a representative sample that yet is close enough to the convergence value of the different parameters stated previously.
2. We create a simulated universe with each one of the thousand set of parameters gathered from (Planck Collaboration 2015)
3. We then compute the temperature power spectrum and the correlation function using the specialized functions offered by CAMB in each one of those universes.
4. Then the mean and the standard deviation of the simulated power spectrum and correlation function are computed in order to calculate the statistic (using the mean) and compare the errors (using the standard deviation)
5. The final step is to compute the statistics using the expressions defined previously and the errors in the appendix.

[2]This is the only fixed parameter, the other values where obtained as the results of the MCMC simulations done by Planck's team

A Error propagation

The error propagation assumes that the multipoles are independent of each other and have a Gaussian distribution. Also, it is assumed that the angle has no error at all.

A.1 Error of the correlation function

The error of the correlation function can be calculated according to this derivation

$$\Delta^2 C(\theta) = \Big(\frac{\partial C(\theta)}{\partial D_l}\,\Delta D_l\Big)^2 \tag{6a}$$

$$= \Big(\sum_l \Delta D_l \frac{2l+1}{2l(l+1)} P_l(\cos(\theta))\Big)^2 \tag{6b}$$

$$\Delta C(\theta) = \sqrt{\Big(\sum_l \Delta D_l \frac{2l+1}{2l(l+1)} P_l(\cos(\theta))\Big)^2} \tag{6c}$$

with this function

Code 5: Code to calculate the error of the correlation function

```
def correlation_func_err(error,xvals):
    cor_err=0

    fac_err= [(2*(l+2)+1)/(2*(l+2)*((l+2)+1))*c for l,c in
        enumerate(error)]
    for l,f in enumerate(fac_err):
        cor_err+=(f*P(l+2,xvals))**2
    return cor_err**0.5
```

A.2 Error of $S_{1/2}$

$$\Delta^2 S_a^b = \Big(\frac{\partial S_a^b}{\partial D_n}\,\Delta D_n\Big)^2 + \Big(\frac{\partial S_a^b}{\partial D_m}\,\Delta D_m\Big)^2 \tag{7a}$$

$$= \Big(\sum_n \sum_m \frac{2n+1}{2n(n+1)} \frac{2m+1}{2m(m+1)} D_m T_{nm}\,\Delta D_n\Big)^2 \tag{7b}$$

$$+ \Big(\sum_n \sum_m \frac{2n+1}{2n(n+1)} D_n \frac{2m+1}{2m(m+1)} T_{nm}\,\Delta D_m\Big)^2$$

$$= \sum_n \sum_m \Big(\frac{2n+1}{2n(n+1)} \frac{2m+1}{2m(m+1)} T_{nm}\Big)^2 \Big(D_m^2 \Delta D_n^2 + D_n^2 \Delta D_m^2\Big) \tag{7c}$$

$$\Delta S_a^b = \sqrt{\sum_n \sum_m \Big(\frac{2n+1}{2n(n+1)} \frac{2m+1}{2m(m+1)} T_{nm}\Big)^2 \Big(D_m^2 \Delta D_n^2 + D_n^2 \Delta D_m^2\Big)} \tag{7d}$$

with this function

Code 6: Code to calculate the error of the old statistic

```
def S12_err(D_ell,D_ell_err,M):
    # Set the precision high enough to handle large calculations
    getcontext().prec =1000
    s=Decimal(0)
    for i,xn in enumerate(D_ell_err):
        n=i+2
        fac1=(((2*n+1))/(2*n*(n+1)))
        for j,xm in enumerate(D_ell):
            m=j+2
            fac2=(((2*m+1))/(2*m*(m+1)))
            Amn=fac1*fac2
            integral=M[i,j]

            s +=
                Decimal(Amn**2)*Decimal(integral**2)*Decimal(D_ell[i]**2*xm**2
                +D_ell[j]**2*xn**2)

    return float(s)**0.5
```

A.3 Error of $\langle \xi \rangle_a^b$

The error of new statistic defined in this paper can be computed using the following function:

Code 7: Code to calculate the error of the new statistic

```
def xivar_err(D_ell_err, a,b):
    s=0
    for i,d in enumerate(D_ell_err):
        l=i+2
        fac=d/(2*l*(l+1))


        integral =Decimal(legendre(l +1, b) -legendre(l -1,
            b)-legendre(l +1, a) +legendre(l -1, a))

        s+=(fac*float(integral)/(b-a))**2

    return (s)**0.5
```

References

Aghanim, N. et al. (Sept. 2020). "Planck2018 results: V. CMB power spectra and likelihoods". In: *Astronomy amp; Astrophysics* 641, A5.

Lewis, A. (n.d.). *CAMB: Code for Anisotropies in the Microwave Background*. `https://github.com/cmbant/CAMB`. Accessed: 2025-02-26.

Planck Collaboration (2015). *Planck Legacy Archive*. `https://pla.esac.esa.int/`. Accedido el 25 de febrero de 2025.