hankl

Release 1.1.1

Mar 02, 2023

Contents:

hankl is a lightweight Python implementation of the FFTLog algorithm for Cosmology

- See the *Installation* page for instruction on how to easily install hankl.
- See the *User Guide and API* page for detailed User Manual and API documentation.
- See the *Examples* page for a couple simple examples.

CHAPTER 1

Attribution

- Please cite arXiv:2106.06331 if you find this code useful in your research:

```
@article{karamanis2021hankl,
title={hankl: A lightweight Python implementation of the FFTLog algorithm for␣
↪Cosmology},
author={Karamanis, Minas and Beutler, Florian},
journal={arXiv preprint arXiv:2106.06331},
year={2021}
}
```

CHAPTER 2

Licence

CHAPTER 3

Changelog

1.1.0 (17/07/20)

- Added preprocessing and extrapolation tools.

1.0.0 (01/07/20)

- Initial version release.

CHAPTER 4

Contributions

We welcome all contributions to hankl via Pull Requests. Let us know about any issues or questions about hankl.

4.1 Installation

To install **hankl** using pip run:

```
pip install hankl
```

Alternatively, install the latest version of **hankl** from source:

```
git clone https://github.com/minaskar/hankl.git
cd hankl
pip install -r requirements.txt
pip install .
```

You need numpy and scipy to install hankl.

4.2 User Guide and API

4.2.1 Hankel Transformations and FFTLog

The FFTLog algorithm can be thought of as the Fast Fourier Transform (FFT) of a logarithmically spaced periodic sequence (= Hankel Transform).

hankl consists of two modules, the General FFTLog module and the Cosmology one. The latter is suited for modern cosmological application and relies heavily on the former to perform the Hankel transforms.

The user can find more information about the FFTLog algorithm here .

Here we will provide some usefull suggestions:

- Accuracy of the method usually improves as the range of integration is enlarged. FFTlog prefers an interval that spans many orders of magnitude.
- Resulution is important. Low resolution will introduce sharp features which in turn will cause ringing.
- When possible, use the provided preprocessing tools to zero/constant pad or extrapolate the function along a larger interval.
- FFTLog works better when the input arrays have size that is a power of 2. You can easily manage this by enabling extrapolation (ext).
- Use the lowringing value of kr (or xy).

For more information about how to use **hankl** see the API below and visit the *Examples* page.

4.2.2 Cosmology module

hankl.**P2xi**(*k*, *P*, *l*, *n=0*, *lowring=False*, *ext=0*, *range=None*, *return_ext=False*, *stirling_cutoff=200.0*)

Hankel Transform Power Spectrum Multipole to Correlation Function Multipole.

$$\xi_l^{(n)}(r) = i^l \int_0^\infty k^2 dk/(2\pi^2)(kr)^{-n} P_l^{(n)}(k) j_l(ks)$$

Parameters

- **k** (*array*) – Array of uniformly logarithmically spaced wavenumbers.
- **P** (*array*) – Array of respective Power Spectrum values.
- **l** (*int*) – Degree of Power Spectrum multipole.
- **n** (*int*) – Order of expansion (Default is 0, plane-parallel).
- **lowring** (*bool*) – If True then use low-ringing value of kr (Default is False).
- **ext** (*int or tuple or list*) – Controls the extrapolation mode. When ext is an integer then the same extrapolation method will be used for both ends of the input array. Alternatively, when ext is an tuple (ext_left, ext_right) or a list [ext_left, ext_right] then different methods can be used for the two ends of the the input array.
  - if ext=0 then no extrapolation is performed (Default).
  - if ext=1 then zero padding is performed.
  - if ext=2 then constant padding is performed.
  - if ext=3 then Power-Law extrapolation is performed.
- **range** (*tuple or list*) – The minimum extrapolation range in the form of a tuple (k_min, k_max) or list [k_min, k_max]. When range=None (Default) then the extended range is chosen automatically such that its array-size is the next power of two.
- **return_ext** (*bool*) – When False (Default) the result is cropped to fit the original k range.
- **stirling_cutoff** (*float*) – Cutoff threshold, above which the Stirling approximation is used to compute the Gamma function ratio (Default is 200).

Returns r, xi – Array of uniformly logarithmically spaced r values and respective array of xi_{l}^{(n)}(r) values.

Return type array, array

hankl.**xi2P**(*r*, *xi*, *l*, *n=0*, *lowring=False*, *ext=0*, *range=None*, *return_ext=False*, *stirling_cutoff=200.0*)
Hankel Transform Correlation Function Multipole to Power Spectrum Multipole.

$$P_l^{(n)}(k) = 4\pi(-i)^l \int_0^\infty r^2 dr (kr)^n \xi_l^{(n)}(r) j_l(kr)$$

Parameters

- **r** (*array*) – Array of uniformly logarithmically spaced separations.
- **xi** (*array*) – Array of respective two point correlation function values.
- **l** (*int*) – Degree of Power Spectrum multipole.
- **n** (*int*) – Order of expansion (Default is 0, plane-parallel).
- **lowring** (*bool*) – If True then use low-ringing value of kr (Default is False).
- **ext** (*int or tuple or list*) – Controls the extrapolation mode. When ext is an integer then the same extrapolation method will be used for both ends of the input array. Alternatively, when ext is an tuple (ext_left, ext_right) or a list [ext_left, ext_right] then different methods can be used for the two ends of the the input array.
  - if ext=0 then no extrapolation is performed (Default).
  - if ext=1 then zero padding is performed.
  - if ext=2 then constant padding is performed.
  - if ext=3 then Power-Law extrapolation is performed.
- **range** (*tuple or list*) – The minimum extrapolation range in the form of a tuple (k_min, k_max) or list [k_min, k_max]. When range=None (Default) then the extended range is chosen automatically such that its array-size is the next power of two.
- **return_ext** (*bool*) – When False (Default) the result is cropped to fit the original k range.
- **stirling_cutoff** (*float*) – Cutoff threshold, above which the Stirling approximation is used to compute the Gamma function ratio (Default is 200).

Returns k, P – Array of uniformly logarithmically spaced k values and array of respective P_{l}^{(n)}(k) values.

Return type array, array

4.2.3 General FFTLog module

hankl.**FFTLog**(*x*, *f_x*, *q*, *mu*, *xy=1.0*, *lowring=False*, *ext=0*, *range=None*, *return_ext=False*, *stirling_cutoff=200.0*)
Hankel Transform based on the FFTLog algorithm of [1] and [2].

Defined as:

$$f(y) = \int_0^\infty F(x)(xy)^q J_\mu(xy) y dx$$

Parameters

- **x** (*array*) – Array of uniformly logarithmically spaced x values.
- **f_x** (*array*) – Array of respective F(x) values.

- **q** (*float*) – Exponent of power law bias; q may be any real number, positive (for forward transform) or negative (for inverse transform).
- **mu** (*float*) – Index of J_mu in Hankel transform; mu may be any real number, positive or negative.
- **xy** (*float*) – Input value of xy (Default is 1).
- **lowring** (*bool*) – If True, then use low-ringing value of xy closest to input value of xy (Default is False).
- **ext** (*int or tuple or list*) – Controls the extrapolation mode. When ext is an integer then the same extrapolation method will be used for both ends of the input array. Alternatively, when ext is an tuple (ext_left, ext_right) or a list [ext_left, ext_right] then different methods can be used for the two ends of the the input array.
  - if ext=0 then no extrapolation is performed (Default).
  - if ext=1 then zero padding is performed.
  - if ext=2 then constant padding is performed.
  - if ext=3 then Power-Law extrapolation is performed.
- **range** (*tuple or list*) – The minimum extrapolation range in the form of a tuple (x_min, x_max) or list [x_min, x_max]. When range=None (Default) then the extended range is chosen automatically such that its array-size is the next power of two.
- **return_ext** (*bool*) – When False (Default) the result is cropped to fit the original x range.
- **stirling_cutoff** (*float*) – Cutoff threshold, above which the Stirling approximation is used to compute the Gamma function ratio (Default is 200).

Returns **y, f(y)** – Array of uniformly logarithmically spaced y values and array of respecive f(y) values.

Return type array, array

References

[1] J. D. Talman. Numerical Fourier and Bessel Transforms in Logarithmic Variables. Journal of Computational Physics, 29:35-48, October 1978.

[2] A. J. S. Hamilton. Uncorrelated modes of the non-linear power spectrum. MNRAS, 312:257-284, February 2000.

4.3 Examples

This section includes two examples. The first one uses the General FFTLog module and the second uses the Cosmology module.

4.3.1 General FFTLog Example

This is a simple example, used by Hamilton to illustrate how the FFTLog algorithm works. We will use **hankl**'s General FFTLog module to compute the following transform:

$$\int_0^\infty r^{\mu+1} exp\Big(-\frac{r^2}{2}\Big) J_\mu(kr) k dr = k^{\mu+1} exp\Big(-\frac{k^2}{2}\Big)$$

Now in this example we know the analytical form of the result so we will use this to demonstrate the accuracy of the transformation.

The general form of the Hankel transform that **hankl** computes is the following:

$$g(k) = \int_0^\infty f(r)(kr)^q J_\mu(kr) k dr$$

This means that the function that we want to transform is:

$$f(r) = r^{\mu+1} exp\Big(-\frac{r^2}{2}\Big)$$

and the result should be

$$g(k) = k^{\mu+1} exp\Big(-\frac{k^2}{2}\Big)$$

which is of course the right hand side (RHS) of the first equation.

Now let's start by importing everything we need:

```
import hankl
import numpy as np
import matplotlib.pyplot as pyplot
```

The next thing we want is to define the functions f(r) and g(r) as well as the integration range for *r*:

```
def f(r, mu=0.0):
    return r**(mu+1.0) * np.exp(-r**2.0 / 2.0)

def g(k, mu=0.0):
    return k**(mu+1.0) * np.exp(- k**2.0 / 2.0)

r = np.logspace(-5, 5, 2**10)
```

As you can see, we used an integration range for *r* which is quite wide and we also chose its size to be a power of two, this will make the algorithm faster and more accurate.

Now let's perform the Hankel transform:

```
k, G = hankl.FFTLog(r, f(r, mu=0.0), q=0.0, mu=0.0)
```

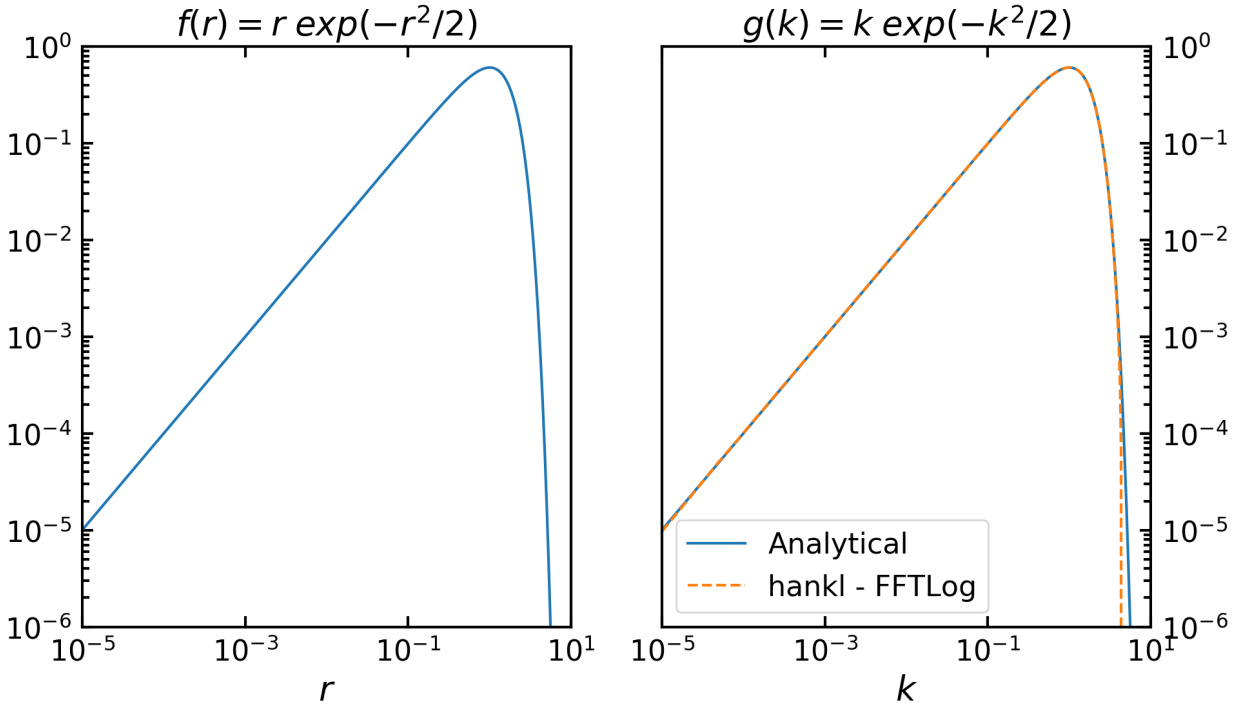
Finally, we can plot the results:

```
plt.figure(figsize=(10,6))

ax1 = plt.subplot(121)
plt.loglog(r, f(r))
```

(continues on next page)

(continued from previous page)

```
plt.title('$f(r) = r \; exp(-r^{2}/2)$')
plt.xlabel('$r$')
plt.ylim(10**(-6), 1)
plt.xlim(10**(-5), 10)

ax1.yaxis.tick_left()
ax2.yaxis.set_label_position("left")

ax2 = plt.subplot(122, sharey=ax1)
plt.loglog(k, g(k), label='Analytical')
plt.loglog(k, G, ls='--', label='hankl - FFTLog')
plt.title('$g(k) = k \; exp(-k^{2}/2)$')
plt.xlabel('$k$')
plt.ylim(10**(-6), 1)
plt.xlim(10**(-5), 10)
plt.legend()

ax2.yaxis.tick_right()
ax2.yaxis.set_label_position("right")
plt.tight_layout()

plt.show()
```



We can further impove the performace of **hankl** by enabling the 'lowring' option, extrapolating or zero/constant padding the function (See the API for more information).

4.3.2 Cosmology Example

In this example we will start from a model of the Galaxy Power Spectrum and our goal is to transform it to Configuration space to get the respective model for the Galaxy 2-Point Correlation Function. Furthermore, we want to transform

both the Monopole and the Quadrupole of the Power Spectrum.

Let's start by importing all the required packages; we will use *classylss* to get the *Linear Matter Power Spectrum*:

```
import hankl
import numpy as np
import matplotlib.pyplot as plt
import classylss
import classylss.binding as CLASS
```

Now that we have imported everything we need let's initialize the fiducial Cosmology of our model:

```
engine = CLASS.ClassEngine({'H0':70, 'Omega_m':0.31})
bg = CLASS.Background(engine)
cosmo = CLASS.ClassEngine({'output': 'dTk vTk mPk', 'non linear': 'halofit', 'P_k_max_
↪h/Mpc' : 20., "z_max_pk" : 100.0})
sp = CLASS.Spectra(cosmo)
```

The next step is to define a suitable *k range* and get the model for the *Linear Matter Power Spectrum*:

```
k = np.logspace(-4, 1, 2**10)
pk_lin = sp.get_pklin(k=k, z=0.5)
```

where we also needed to specify the effective redshift *z=0.5*. We can plot easily the Linear Power Spectrum:

```
plt.loglog(k, pk_lin)
plt.xlabel(r'$k\: [h\; Mpc^{-1}]$')
plt.ylabel(r'$P_{linear}(k)\: [h^{-1}\; Mpc]^{3}$')
plt.show()
```

Now, this was the linear Matter Power Spectrum, to get the Galaxy Power Spectrum multipoles we will use *Kaiser*'s formula:

$$P(k,\mu) = (b + f\mu^2)^2 P_{lin}(k)$$

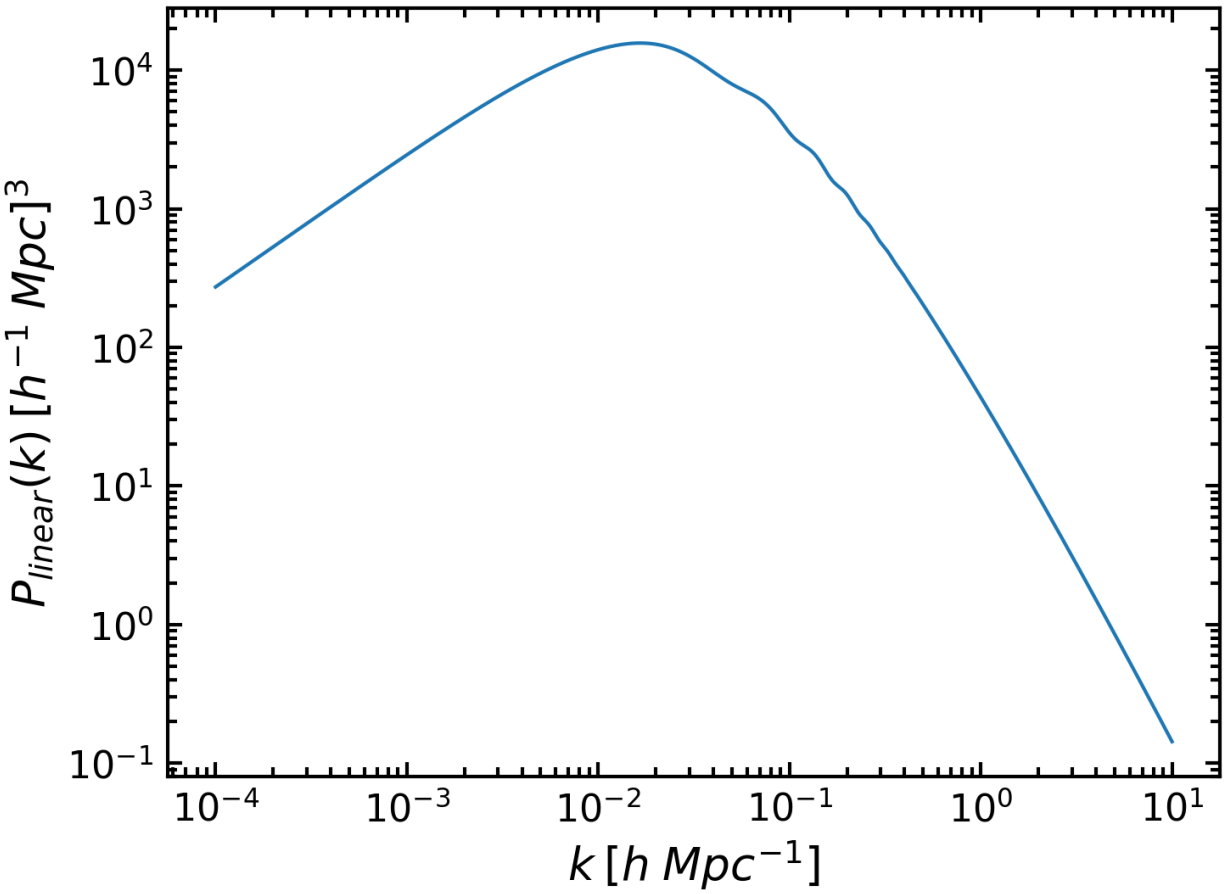
where b is the *linear bias parameter*, f is the *logarithmic growth rate*, and μ is the cosine of the angle between teh Fourier modes and the line-of-sight. By decomposing the aforementioned function in terms of the *Legendre polynomials* we get the Monopole ($l = 0$), Quadrupole ($l = 2$) and Hexadecapole ($l = 4$) of the galaxy Power Spectrum:

```
def get_multipoles(b, f):

    p0 = (b*b + 2.0 * b *f / 3.0 + f*f/5.0 ) * pk_lin

    p2 = (4.0*b*f/3.0 + 4.0*f*f/7.0) * pk_lin

    p4 = (8.0* f*f / 35.0) * pk_lin

return p0, p2, p4
```

Alright, let's compute and plot those multipoles for two realistic values of b and f:

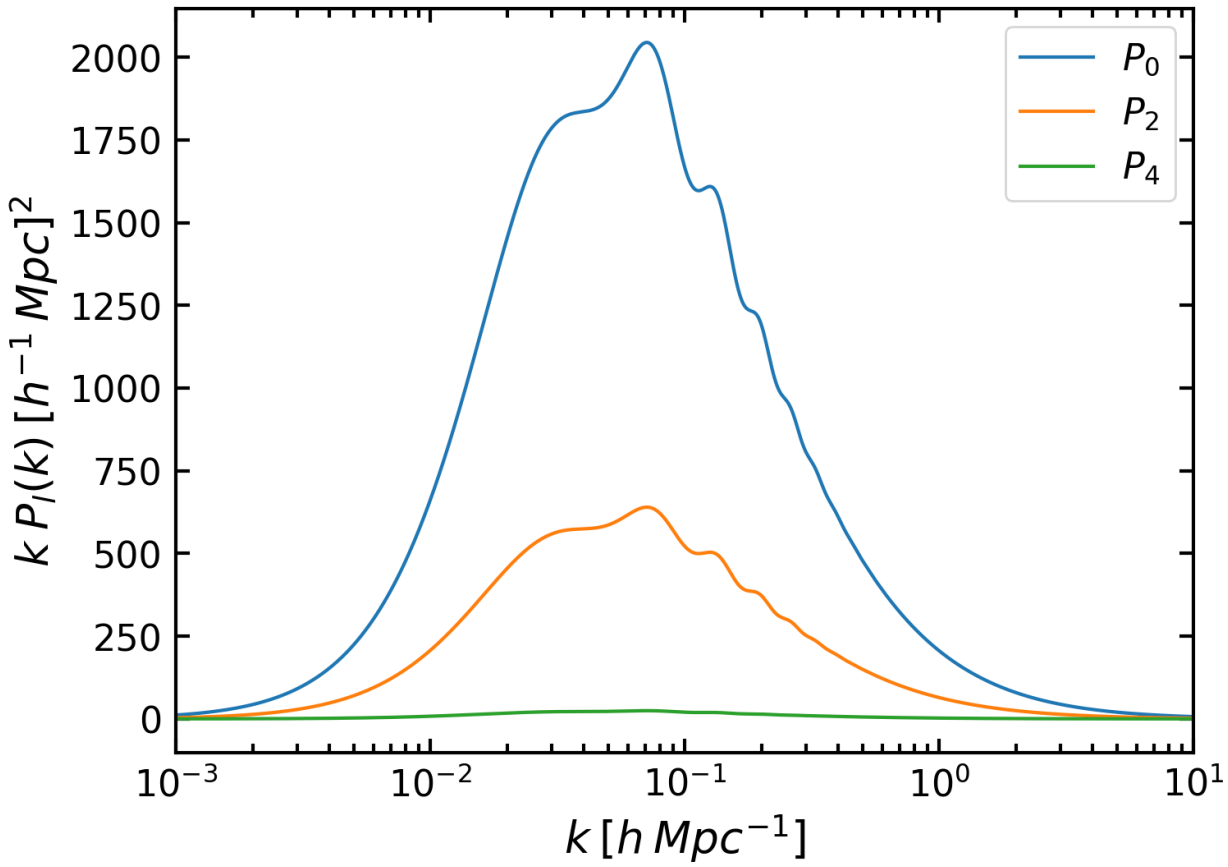
```
P0, P2, P4 = get_multipoles(b=2.0, f=0.5)

plt.semilogx(k, k * P0, label=r'$P_{0}$')
```

(continues on next page)


$P_{linear}(k)\ [h^{-1}\ Mpc]^3$
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^0
10^1
$k\ [h\ Mpc^{-1}]$

(continued from previous page)

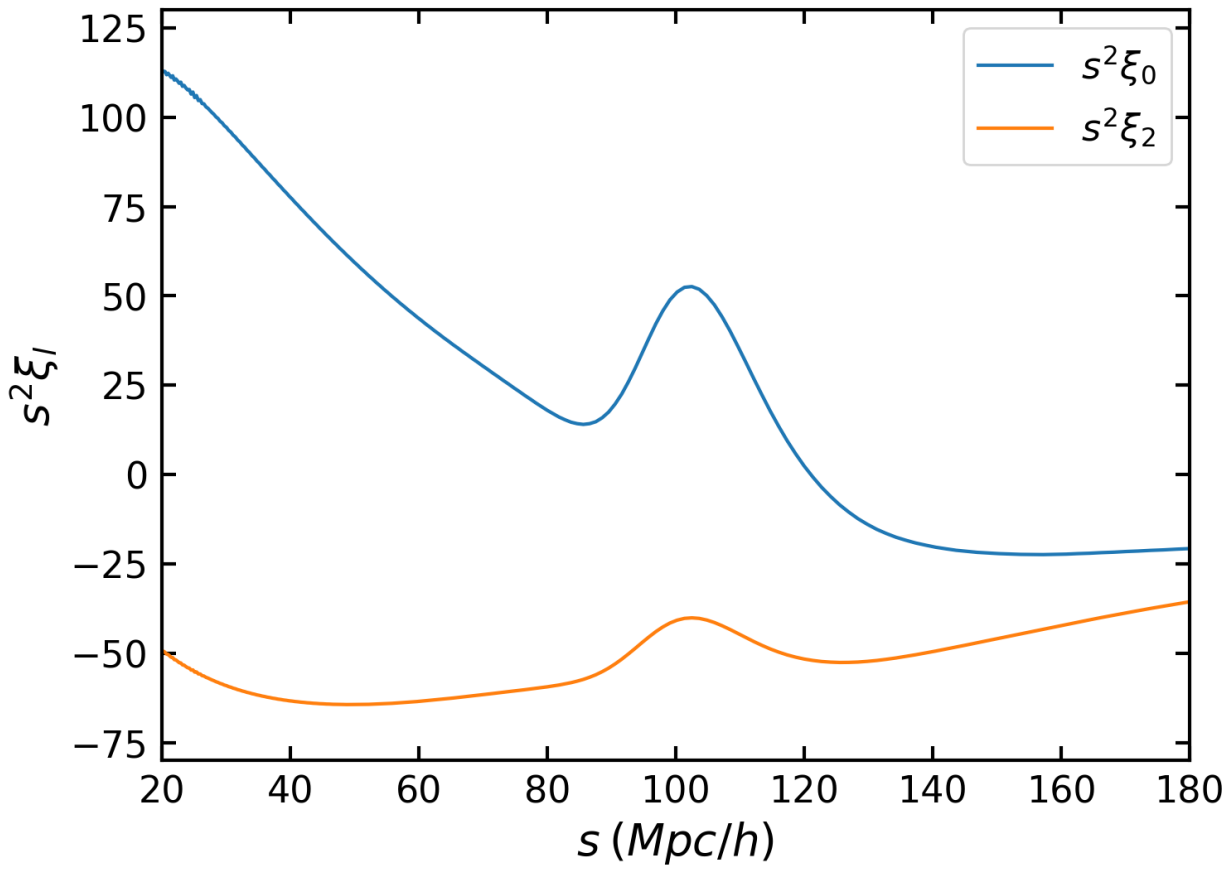
```
plt.semilogx(k, k * P2, label=r'$P_{2}$')
plt.semilogx(k, k * P4, label=r'$P_{4}$')
plt.xlabel(r'$k \: [h\: Mpc^{-1}]$')
plt.ylabel(r'$k\; P_{l}(k) \: [h^{-1}\: Mpc]^{2}$')
plt.xlim(1e-3, 10)
plt.legend()
plt.show()
```



The last step involves using the Cosmology module P2xi to transform the above power spectrum multipoles to correlation function multipoles and plot them. Since the Hexadecapole (l=4) is almost zero, we will do the transform only for the Monopole and Quadrupole:

```
s, xi0 = hankl.P2xi(k, P0, l=0)
s, xi2 = hankl.P2xi(k, P2, l=2)

plt.plot(s, s*s*xi0, label=r'$s^{2}\xi_{0}$')
plt.plot(s, s*s*xi2, label=r'$s^{2}\xi_{2}$')
plt.xlim(20,180)
plt.ylim(-80,130)
plt.xlabel(r'$s\: (Mpc/h)$')
plt.ylabel(r'$s^{2} \xi_{1}$')
plt.legend()
plt.show()
```

As expected we can see the Baryon Acoustic Oscillations (BAO) peak in both multipoles.


125
100
75
50
25
0
−25
−50
−75
20
40
60
80
100
120
140
160
180
$s^2\xi_l$
s (*Mpc/h*)
$s^2\xi_0$
$s^2\xi_2$

Index

F

P

X