HERA Memo 127
Received: 1 Dec 2023

Definition of the Expected Power Spectrum In The H1C Validation Simulations

Zachary Martinot and Zhilei Xu

September, 2023

This note provides an explicit definition for true input cosmological power spectrum of the signal component of the end-to-end simulation used in Aguirre et al. (2022). The definition recorded here should allow the use of the simulated visibility files stored on the NRAO cluster to be reused for further tests where you would like to test recovery of the input power spectrum.

The effective true power spectrum of the simulated EoR signal in the file `/lustre/aoc/projects/hera/Validation/test-4.0.0/data/visibilities/eor.uvh5`, using the default units and conventions of `hera_pspec`, is given by

$$P_{true}(k,z) = \left(10^6 \frac{\mathrm{mK}^2}{\mathrm{K}^2}\right)\left(\frac{\nu_e}{\nu_z}\right)^{-2}\left(100\left(\frac{10^8\mathrm{Hz}}{\nu_z}\right)^3\right)^2 A_0 k^{-2} \qquad [\mathrm{mK}^2 h^{-3}\mathrm{Mpc}^3]. \tag{1}$$

The variables are:

- k: spatial wavenumber with units of $h\mathrm{Mpc}^{-1}$.
- z: redshift, dimensionless.
- h: dimensionless Hubble constant "little-h"; little-h units are used by default in `hera_pspec`.
- $\nu_e = 1420405751.7667\mathrm{Hz}$: H I line emission frequency.
- $\nu_z = \dfrac{\nu_e}{1+z}$: observed frequency with units of Hz
- $A_0 = h \cdot 10\left(\dfrac{1}{0.2}\right)^{-2}$ $[\mathrm{K}^2 h^{-1}\mathrm{Mpc}]$: amplitude factor from the original power spectrum definition in `test-0.2`, but redefined here with little-h factors.

This possibly bewildering combination of factors arise from unit conversions relative to original definitions, a purposeful redefinition of the power spectrum amplitude to make it detectable at the simulated noise level of Aguirre et al. (2022), and the physical redshift dimming factor. A further complication is that Equation 1 is still not the approprate definition of the *expected* power spectrum. The simulation contains a strong aliasing effect because there was no simulation of frequency channel integration - the spectrum of the simulated visibilities is as if the instrument had a delta-function response at each of the frequency samples. The resulting aliasing effect can be mostly accounted for analytically, and as a result the *expected* power spectrum, appropriate for evaluating the correctness of a power spectrum estimate obtained from the simulated data, is defined as

$$P_{expected}(k,z) = \left(10^6 \frac{\mathrm{mK}^2}{\mathrm{K}^2}\right)\left(\frac{\nu_e}{\nu_z}\right)^{-2}\left(100\left(\frac{10^8\mathrm{Hz}}{\nu_z}\right)^3\right)^2 \frac{A_0\pi^2}{4k_s^2 \sin^2\left(\dfrac{\pi k}{2k_s}\right)} \tag{2}$$

where k_s is an approximate Nyquist frequency. This is defined as the average Nyquist frequency over the frequency sub-band that goes into the delay spectrum

$$k_s = \frac{\pi}{N-1} \sum_{n=0}^{N-2} \frac{1}{|r(z_{n+1}) - r(z_n)|} \tag{3}$$

where $r(z)$ is the comoving distance and $z_n = \frac{\nu_e}{\nu_n} - 1$ is the redshift for each frequency sample in the sub-band $\{\nu_n\}_{n=0}^{N-1}$. The quantity k_s is thus dependent on the frequency sub-band used.

References

Aguirre, J. E., Murray, S. G., Pascua, R., et al. 2022, The Astrophysical Journal, 924, 85, doi: 10.3847/1538-4357/ac32cd

signal_power_spectrum_for_h1c_idr2_validation_sim

September 25, 2023

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt

import h5py, os, glob, copy

import hera_pspec as hp

from pyuvdata import UVData

from pyuvdata import utils as uvd_utils
```

[2]:
```python
vis_data_path = '/lustre/aoc/projects/hera/Validation/test-4.0.0/data/
 ↪visibilities/eor.uvh5'

omegas_data_path = '/lustre/aoc/projects/hera/Validation/HERA_dipole_Omegas/
 ↪HERA_dipole_Omegas.h5'

uvd = UVData.from_file(vis_data_path, read_data=False)
```

```
Telescope RIMEz calculation is not in known_telescopes.
LST values stored in
/lustre/aoc/projects/hera/Validation/test-4.0.0/data/visibilities/eor.uvh5 are
not self-consistent with time_array and telescope location. Consider recomputing
with utils.get_lst_for_time.
```

[3]:
```python
# This is the cosmological parameter set in the input simulation
def astropyPlanck15_for_hera_pspec():
    H0 = 67.74
    h = H0/100.

    Om_b = 0.02230/h**2.
    Om_c = 0.1188/h**2.
    Om_L = 0.6911
    Om_k = 1. - (Om_b + Om_c + Om_L)

    hp_cosmo = hp.conversions.Cosmo_Conversions(Om_L=Om_L,
                                                Om_b=Om_b,
                                                     Om_c=Om_c,
                                                     H0=H0,)

    return hp_cosmo

cosmo = astropyPlanck15_for_hera_pspec()

# Load beam-dependent normalization data
with h5py.File(omegas_data_path, 'r') as h5f:
    Omega = h5f['Omega'][()]
    Omegapp = h5f['Omegapp'][()]

# hera_pspec beam initiated from pre-computed integral data
hpsb = hp.pspecbeam.PSpecBeamFromArray(Omega, Omegapp, uvd.freq_array[0],␣
  ↪cosmo=cosmo)
```

[4]:
```
%%time
uvd = UVData.from_file(vis_data_path, run_check=False)

# uvd = get_VI_data(uvd)
uvd = hp.pstokes.construct_pstokes(uvd, uvd, pstokes='pI', run_check=False)

nu_e = 1420405751.7667 # Hz
nu_hz = uvd.freq_array[0]
```

```
Telescope RIMEz calculation is not in known_telescopes.
LST values stored in
/lustre/aoc/projects/hera/Validation/test-4.0.0/data/visibilities/eor.uvh5 are
not self-consistent with time_array and telescope location. Consider recomputing
with utils.get_lst_for_time.
```

```
CPU times: user 2min 32s, sys: 1min 5s, total: 3min 38s
Wall time: 4min 7s
```

[5]:
```
ds = hp.PSpecData(dsets=[uvd, uvd], beam=hpsb)
ds.Jy_to_mK()
```

```
Cannot convert dset 1 Jy -> mK because vis_units = mK
```

[6]:
```
# get antenna pairs for 63 unique baselines, excluding auto-correlation
ant_pairs = [ant_pair for ant_pair in uvd.get_antpairs() if ant_pair[0] !=␣
  ↪ant_pair[1]]

# approximate subbands used in H1C IDR2 results
spw_ranges = [(175, 325), (575, 725)]
```

[7]:
```
%%time
uvp = ds.pspec(ant_pairs, ant_pairs,
              dsets=(0,1),
              pols=('pI', 'pI'),
              input_data_weight='identity',
              norm='I',
              taper='blackman-harris',
              spw_ranges=spw_ranges,
              store_window=False,
              verbose=True,
              allow_fft=True,
              exact_norm=False)
```

```
CPU times: user 5min 34s, sys: 12.3 s, total: 5min 46s
Wall time: 56.3 s
```

[8]:
```
%%time
k_bin_widths = 0.05

k_bins = np.arange(0,2.5,k_bin_widths)

uvp_sphr = copy.deepcopy(uvp)

# time average for each (delay, baseline)
uvp_sphr.average_spectra(time_avg=True, inplace=True)

# average over (delay, baseline) for each k
uvp_sphr = hp.grouping.spherical_average(uvp_sphr, k_bins, k_bin_widths,␣
 ↪run_check=False)
```

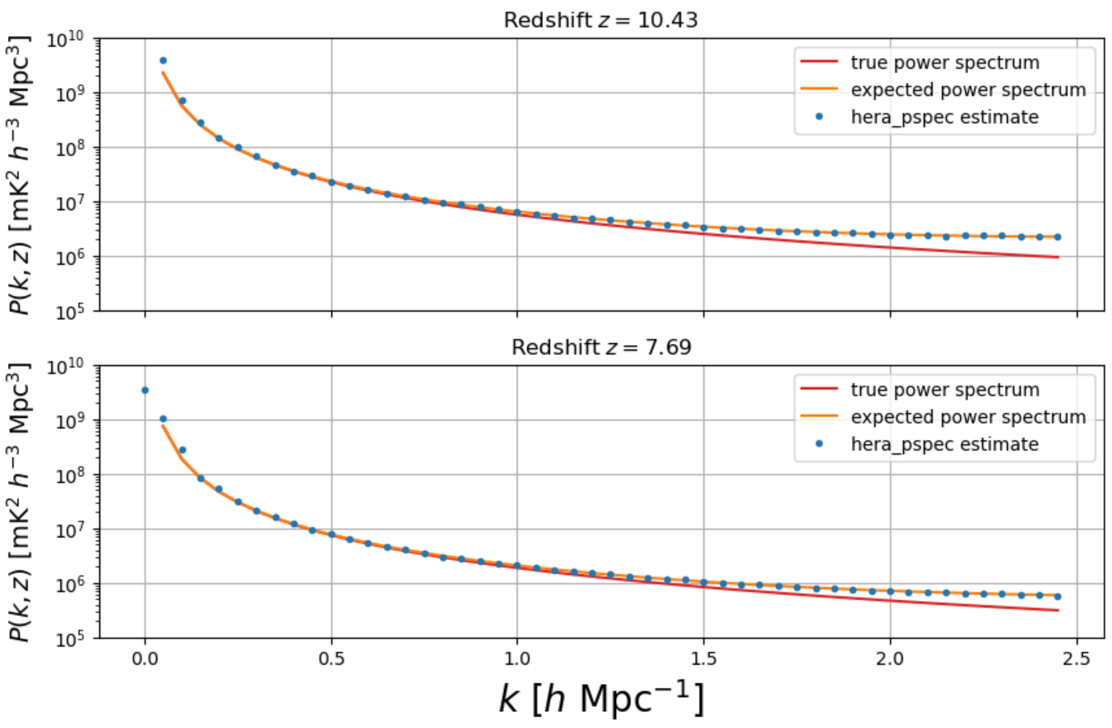
```
CPU times: user 7.9 s, sys: 861 ms, total: 8.76 s
Wall time: 6.53 s
```

[9]:
```
# evaluates the true input power spectrum
def true_power_spectrum(k, z):
    A0 = 1e1 * (1./0.2)**-2. # Kelvin^2

    Pk = 1e6*A0 * k**-2

    nu_e = 1420405751.7667 # Hz
    nu_z = nu_e/(1. + z)

    # redshift dimming factor
    Pk *= (nu_e/nu_z)**-2

    # rescaling in /lustre/aoc/projects/hera/Validation/test-4.0.0/setup/
 ↪rescale_eor_data.py
    Pk *= (100.*(100e6/nu_z)**3)**2

    # using little-h units
    Pk *= cosmo.h
    return Pk

# evaluates the expected power spectrum
def aliased_power_spectrum(k, ks, z):
    A0 = 1e1 * (1./0.2)**-2. # Kelvin^2

    Pk = 1e6*A0 * (np.pi)**2. * 1./(2. * ks * np.sin(np.pi*k / ks /2.))**2.

    nu_e = 1420405751.7667 # Hz
    nu_z = nu_e/(1. + z)

    # redshift dimming factor
    Pk *= (nu_e/nu_z)**-2

    # rescaling in /lustre/aoc/projects/hera/Validation/test-4.0.0/setup/
 ↪rescale_eor_data.py
    Pk *= (100.*(100e6/nu_z)**3)**2

    # using little-h units
    Pk *= cosmo.h
    return Pk

def get_ks(spw):
    spw_freqs = uvp.freq_array[uvp.spw_freq_array == spw]
    z_s = cosmo.f2z(spw_freqs)
    dr = np.diff(np.array([cosmo.DC(z, little_h=True) for z in z_s]))

    return np.pi * np.mean(1/np.abs(dr)), np.mean(z_s)
```

[12]:
```
keys = uvp_sphr.get_all_keys()

fig, axes = plt.subplots(2, 1, figsize=(10,6), sharex=True)

for ii, spw in enumerate(uvp_sphr.data_array.keys()):
    pspec_est = uvp_sphr.get_data(keys[spw])[0].real

    ks, z_mean = get_ks(spw)

    pspec_true = true_power_spectrum(k_bins, z_mean)
    pspec_aliased_true = aliased_power_spectrum(k_bins, ks, z_mean)

    ax = axes[ii]

    ax.plot(k_bins, pspec_true, label='true power spectrum', color='tab:red')
    ax.plot(k_bins, pspec_aliased_true, label='expected power spectrum',␣
 ↪color='tab:orange')
```

```
    ax.plot(k_bins, pspec_est, '.', label='hera_pspec estimate', color='tab:
 ↪blue')

    ax.set_yscale('log')
    ax.set_ylim(1e5, 1e10)
    ax.set_ylabel(r'$P(k,z)\ [{\rm mK}^2\ h^{-3}\ {\rm Mpc}^3]$', fontsize=14)
    if spw == 1:
        ax.set_xlabel(r'$k\ [h\ {\rm Mpc}^{-1}]$', fontsize=20)
    ax.set_title(r'Redshift $z =${0}'.format(str(np.around(z_mean,2))))

    ax.grid()
    ax.legend()
plt.show()
```

```
divide by zero encountered in power
divide by zero encountered in true_divide
```


Redshift $z = 10.43$
Redshift $z = 7.69$
true power spectrum
expected power spectrum
hera_pspec estimate
$P(k,z)$ [mK^2 h^{-3} Mpc^3]
10^{10}
10^{9}
10^{8}
10^{7}
10^{6}
10^{5}
k [h Mpc^{-1}]