Power Spectrum Normalizations for HERA

April 4, 2017

by Aaron Parsons

1 Background

The relation between the delay-transformed visibility, $\tilde{V}$, and the three-dimensional power spectrum of reionization, $P_{21}(\mathrm{k})$, is given by

$$\tilde{V}_{21}^2(u,v,\eta) \approx \left(\frac{2k_\mathrm{B}}{\lambda^2}\right)^2 \frac{\Omega\, B}{X^2 Y} \widehat{P}_{21}(\mathrm{k}), \tag{1}$$

which follows from equation 12 of Parsons et al. (2012a), where λ is the observing wavelength, k_B is Boltzmann's constant, X^2Y is a cosmological scalar with units of $\frac{h^{-3}\,\mathrm{Mpc}^3}{\mathrm{sr}\cdot\mathrm{Hz}}$, and Ω is the angular area. As shown in Parsons et al. (2014), relevant beam area for the above equation is the power-square beam, Ω_{PP}, given by

$$\Omega_{\mathrm{PP}} \equiv \int dl\; dm\, |A(l,m)|^2. \tag{2}$$

This contrasts with the standard metric for beam area — the integrated power beam — which we will call Ω_{P}, and is given by

$$\Omega_{\mathrm{P}} \equiv \int dl\; dm A(l,m), \tag{3}$$

This beam area metric is used to convert visibility measurements from Jy units to mK, but is incorrect for normalizing power spectra that relate to the two-point correlation function of a temperature field. Similarly, the relevant bandwidth B is given by

$$B_{\mathrm{PP}} \equiv \int d\nu\; |W(\nu)|^2, \tag{4}$$

where W is the windowing (or spectral tapering) function multiplied to the data prior to taking the delay transform.

Let us walk through this explicitly, following the delay-spectrum power spectrum analysis pipeline, beginning with V_{Jy}, a visibility measured in Jy. First, we typically construct a visibility in temperature units, V_{mK}, using the relation

$$V_{\mathrm{mK}} = V_{\mathrm{Jy}} \frac{\lambda^2}{2k_B\Omega_{\mathrm{P}}} 10^{-23} \frac{\mathrm{erg}/(\mathrm{s}\cdot\mathrm{cm}^2\cdot\mathrm{Hz})}{\mathrm{Jy}} \cdot 10^3 \frac{\mathrm{mK}}{\mathrm{K}}. \tag{5}$$

Next, we construct a delay-transformed visibility $\tilde{V}'_{\rm mK}$ using an inverse discrete Fourier Transform (typically, numpy.fft.ifft). Following the inverse DFT definition, numpy.fft.ifft divides by the number of channels over which the transform is performed. This, combined with the Jy calibration (which divides by the channel bandwidth), and the conversion to brightness temperature, results in a delay-transformed visibility that is related to the original (un-primed) definition used in Parsons et al. (2012a,b) by

$$\tilde{V}'_{\rm mK} = \frac{\lambda^2}{2k_B \Omega_{\rm P} B_{\rm P}} \tilde{V}, \tag{6}$$

where $B_{\rm P} \equiv \int d\nu$ is the total bandwidth over which the transform is performed. Thus, we relate $V'_{21,{\rm mK}}$ to the estimated 21cm power spectrum, $\widehat{P}({\rm k})$, as

$$\widehat{P}({\rm k}_{t\tau}) = X^2 Y \frac{\Omega_{\rm P}^2}{\Omega_{\rm PP}} \frac{B_{\rm P}^2}{B_{\rm PP}} \left\langle \left| \tilde{V}'_{\rm mK}(\tau, t) \right|^2 \right\rangle. \tag{7}$$

In Parsons et al. (2014), we define $\Omega_{\rm eff} \equiv \Omega_{\rm P}^2 / \Omega_{\rm PP}$, which is precisely the term that should be used in the above equation. Similarly, we could define $B_{\rm eff} \equiv B_{\rm P}^2 / B_{\rm PP}$. It turns out this is equivalent to $B_{\rm eff} = B_{\rm P} \cdot {\rm NEB}$, where NEB is the noise-equivalent bandwidth of the chosen windowing function W. Thus, we ultimately have

$$\widehat{P}({\rm k}_{t\tau}) = X^2 Y \Omega_{\rm eff} B_{\rm P} \cdot {\rm NEB} \cdot \left\langle \left| \tilde{V}'_{\rm mK}(\tau, t) \right|^2 \right\rangle. \tag{8}$$

1.1 System Temperature and Estimated Noise Levels

Thermal fluctuations produce a white-noise signal with root-mean-square (RMS) brightness temperature $T_{\rm N}$. The thermal noise contributes a component to the RMS amplitude of the visibility $V_{\rm N}$ equal to:

$$V_{\rm N} = \frac{2k_B}{\lambda^2} T_{\rm N} \, \Omega_{\rm P}. \tag{9}$$

We can then use $\tilde{V}_{\rm N} = V_{\rm N} \sqrt{B_{\rm P}}$ for $\tilde{V}$ our original equation up top to get the noise contribution to the power spectrum, $P_{\rm N}^2(k)$, yielding:

$$P_{\rm N}(k) \approx X^2 Y \Omega_{\rm eff} \, B_{\rm eff} \, T_{\rm N}^2. \tag{10}$$

For a single-sideband, real-sampled correlator like the one used in PAPER and HERA, there are $B_{\rm P} t / {\rm NEB}$ independent measurements of the noise for integration time, t. (There has been a long-standing confusion on relating to a factor of 2 on the number of independent measurements. In a real-sampled system, there are twice as many time samples as frequency channels produced in the DFT — this is a restatement of the Nyquist-Shannon criterion — but each frequency channel coefficient is a complex number. In the end, these two factors of 2 cancel each other out to yield the number of independent measurements written above. This allows us to relate $T_{\rm N}$ to a system temperature, $T_{\rm sys}$, as

$$T_{\rm N}^2 = \frac{{\rm NEB} \, T_{\rm sys}^2}{B_{\rm P} t}. \tag{11}$$

With this substitution,

$$P_{\rm N}(k) \approx X^2 Y \Omega_{\rm eff} \cdot {\rm NEB} \cdot \frac{T_{\rm sys}^2}{N_{\rm pol} t}, \tag{12}$$

where t is the integration time for sampling a particular (u, v, η)-mode, and the factor of $N_{\rm pol}$ in the denominator explicitly counts the two orthogonal polarizations to measure the total unpolarized signal.

```
In [1]: import numpy as np, capo
        def noise(size, sig=1.):
            sig = sig/np.sqrt(2)
            return np.random.normal(scale=sig,size=size) + \
                    1j*np.random.normal(scale=sig,size=size)

        Tsys_mK = 500e3
        jy2T = capo.pspec.jy2T(.150)
        Tsys_jy = Tsys_mK / jy2T
        Jy_cal = 1 # Jy
        B_tot = 100e6 # Hz
        t_samp = 5e-9 # s
        t_int = 10.7 # s

        Nchan = 1024
        B_ch = B_tot / Nchan
        t_window = 2 * Nchan * t_samp
        Nwindow = int(t_int / t_window)

        # F Engine
        ant_volt = np.random.normal(scale=np.sqrt(Jy_cal*B_tot), size=2*Nchan)
        _ant_volt = np.fft.rfft(ant_volt)[:-1]
        _ant_jy = np.abs(_ant_volt)**2 / B_tot / (2*Nchan)
        print 'Check Jy normalization:', np.average(_ant_jy) / Jy_cal, '?= 1'

        # X Engine
        ant_volt_i = np.random.normal(scale=np.sqrt(Tsys_jy*B_tot), size=2*Nchan)
        _ant_volt_i = np.fft.rfft(ant_volt_i)[:-1]
        ant_volt_j = np.random.normal(scale=np.sqrt(Tsys_jy*B_tot), size=2*Nchan)
        _ant_volt_j = np.fft.rfft(ant_volt_j)[:-1]
        _ant_jy_ij = _ant_volt_i * np.conj(_ant_volt_j) / B_tot / (2*Nchan)
        print 'Check Tsys recovered w/o sqrt(2):',
        print np.std(_ant_jy_ij) * jy2T * np.sqrt(B_ch * t_window), '?=', Tsys_mK

Check Jy normalization: 1.00276977894 ?= 1
Check Tsys recovered w/o sqrt(2): 504441.313361 ?= 500000.0
```

2 Power Spectrum Normalizations for HERA

Here we compute the appropriate power spectrum normalizations for HERA and compare them to those previously computed for PAPER. Parsons et al. (2014) focused on the PAPER beam, computing angular areas of $\Omega_{\rm P} \approx 0.72$ sr, and $\Omega_{\rm PP} \approx 0.31$ sr. Following the definition above, $\Omega_{\rm eff} \approx 1.69$ for PAPER.

In [2]:
```
%matplotlib inline
import ipywidgets
import numpy as np, pylab as plt, aipy, capo, omnical
import os, ephem, glob
from mpl_toolkits.basemap import Basemap
```

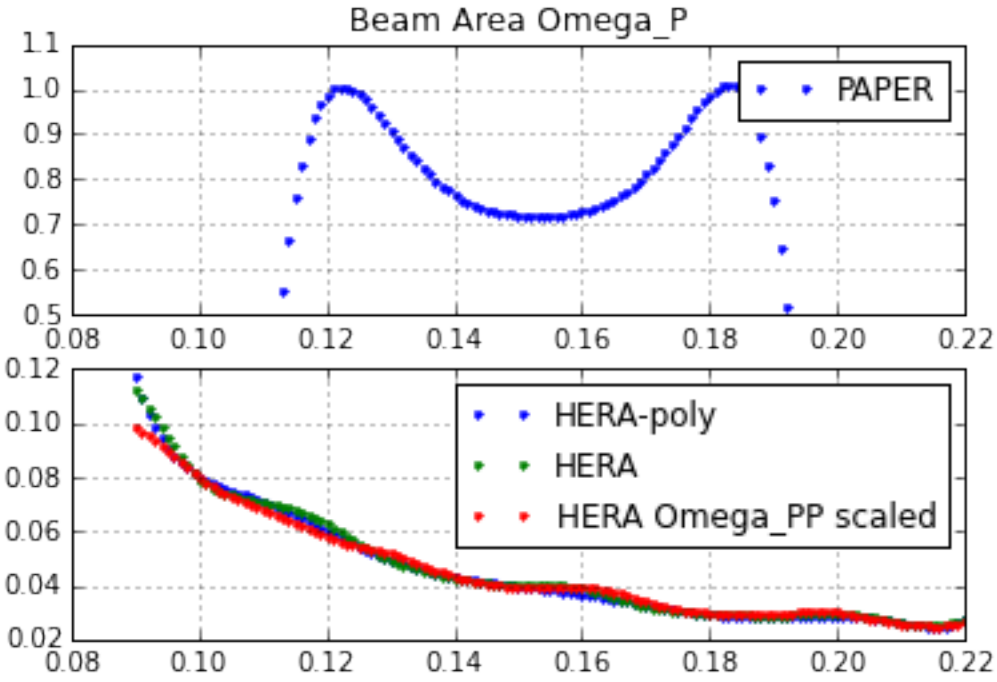
In [3]:
```
def beam_area(dat):
    return np.sum(dat) * 4 * np.pi / dat.size

def beamsq_area(dat):
    return np.sum(dat**2) * 4 * np.pi / dat.size
```

In [4]:
```
DIRECTORY = '/Users/aparsons/projects/eor/beam/hera-cst/mdl04'
files = glob.glob(DIRECTORY+'/*.hmap')

def parse_filename(f):
    return float(f.split('_')[-1][:-len('.hmap')])/1e3

fqs = np.empty(len(files), dtype=np.float)
Omega_P = np.empty_like(fqs)
Omega_PP = np.empty_like(fqs)

for i,f in enumerate(files):
    #print i,'/',len(files), f
    hmap = aipy.map.Map(fromfits=f)
    tx,ty,tz = hmap.px2crd(np.arange(hmap.npix()))
    d = np.where(tz > 0, hmap[tx,ty,tz], 0)
    d /= d.max()
    Omega_P[i], Omega_PP[i] = beam_area(d), beamsq_area(d)
    fqs[i] = parse_filename(f)
```

In [10]:
```
HERA_BEAM_POLY = np.around(np.polyfit(fqs, Omega_P, deg=8), 2)
print 'PAPER Omega_P Poly:\n', np.array(capo.pspec.DEFAULT_BEAM_POLY)
print 'HERA Omega_P Poly:\n', HERA_BEAM_POLY
print 'PAPER Omega_P/Omega_PP', 2.35
print 'HERA Omega_P/OMEGA_PP', np.median(Omega_P/Omega_PP)
plt.subplot(211)
plt.title('Beam Area Omega_P')
plt.plot(fqs, np.polyval(capo.pspec.DEFAULT_BEAM_POLY, fqs),
         '.', label='PAPER')
plt.grid(); plt.legend()
plt.ylim(.5,1.1)
```

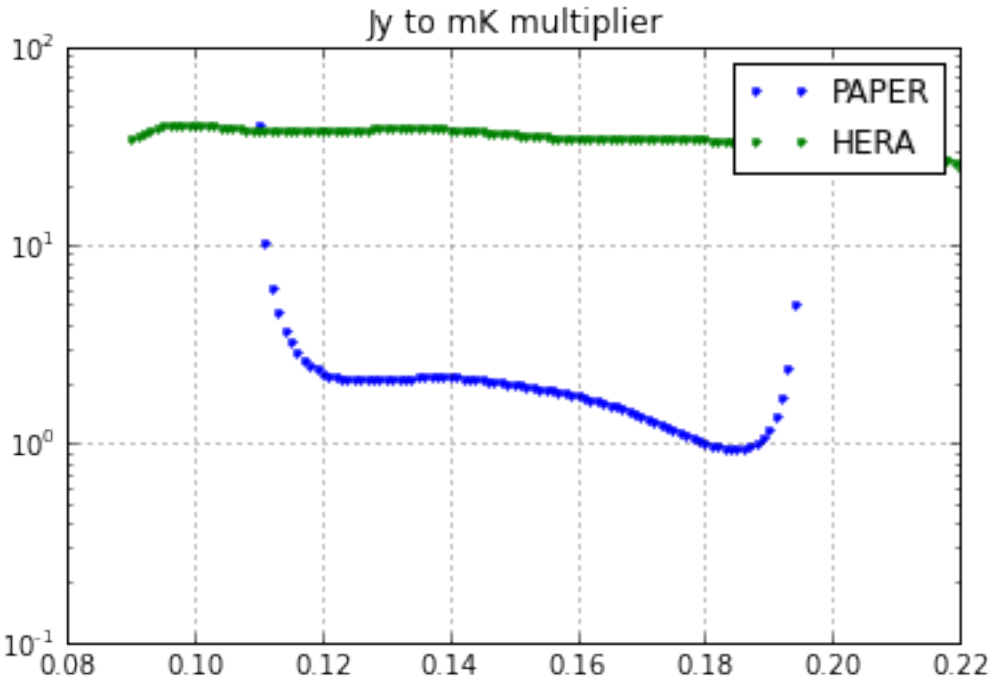
```
        plt.subplot(212)
        plt.plot(fqs, np.polyval(HERA_BEAM_POLY, fqs),
                 '.', label='HERA-poly')
        plt.plot(fqs, Omega_P, '.', label='HERA')
        #plt.semilogy(fqs, Omega_PP, '.')
        plt.plot(fqs, Omega_PP * np.median(Omega_P/Omega_PP),
                '.', label='HERA Omega_PP scaled' )
        plt.grid(); plt.legend()
        plt.ylim(.02,.12)
        plt.show()
```

```
PAPER Omega_P Poly:
[ -1.55740671e+09   1.14162351e+09  -2.80887022e+08   9.86929340e+06
   7.80672834e+06  -1.55085596e+06   1.20087809e+05  -3.47520109e+03]
HERA Omega_P Poly:
[  8.07774113e+08  -1.02194430e+09   5.59397878e+08  -1.72970713e+08
   3.30317669e+07  -3.98798031e+06   2.97189690e+05  -1.24980700e+04
   2.27220000e+02]
PAPER Omega_P/Omega_PP 2.35
HERA Omega_P/OMEGA_PP 2.1752891255
```




```
In [6]: plt.semilogy(fqs, capo.pspec.jy2T(fqs),
                     '.', label='PAPER')
        plt.semilogy(fqs, capo.pspec.jy2T(fqs, HERA_BEAM_POLY),
```

```
                '.', label='HERA')
        plt.title('Jy to mK multiplier')
        plt.ylim(1e-1,1e2)
        plt.legend(); plt.grid()
        plt.show()
```



In [7]:
```
WINDOW = 'none'
fq0 = .150 # GHz
dfq = .01 # GHz
z = capo.pspec.f2z(fq0)
B = dfq * capo.pfb.NOISE_EQUIV_BW[WINDOW] #proper normalization
bm_PAPER = np.polyval(capo.pspec.DEFAULT_BEAM_POLY, fq0) * 2.35
bm_HERA = np.polyval(HERA_BEAM_POLY, fq0) * 2.175
scalar_PAPER = capo.pspec.X2Y(z) * bm_PAPER * B
scalar_HERA = capo.pspec.X2Y(z) * bm_HERA * B
print 'Scalar PAPER', scalar_PAPER, ' [h^-3 Mpc^3] / [sr * GHz]'
print 'Scalar  HERA ', scalar_HERA, ' [h^-3 Mpc^3] / [sr * GHz]'
```

```
Scalar PAPER 8783524591.59  [h^-3 Mpc^3] / [sr * GHz]
Scalar  HERA  450523765.952  [h^-3 Mpc^3] / [sr * GHz]
```

3 Gain Scale

Finally, we have from Beardsley's Memo 16 (updated April 2017) that the estimated antenna (voltage) gain scale for HERA is $g \sim 0.03$, while the scale for PAPER was ~ 0.005. The input signal levels of these antennas were tuned to be approximately the same, suggesting that the conversion factor from raw signal input to power spectrum amplitude differs by a factor of

```
In [9]: g_PAPER, g_HERA = .005, .03
        raw2mK2_PAPER = (g_PAPER**-2 * capo.pspec.jy2T(fq0))**2 \
                * scalar_PAPER
        raw2mK2_HERA = (g_HERA**-2 * capo.pspec.jy2T(fq0, HERA_BEAM_POLY))**2 \
                * scalar_HERA
        print raw2mK2_PAPER / raw2mK2_HERA
```

```
77.601673619
```