Signal Loss due to Emperically Estimated Inverse Covariance Weighting

D. C. Jacobs

July 23, 2016

Here I investigate the signal loss one gets from estimating covariance empirically in data under conditions of low number statistics.

Lets keep things simple, our simulated data set is a set of Nchan channels, measured Ntimes on Nbaselines. Divide our Nbaselines into Ngroups. Within each group compute the weighted data and then measure the power spectrum cross-multiplying between independent groups. Calculate the power spectrum over many trials drawing different "eor" signals each time. [1] EoR is simulated by drawing random noise, filtering to provide a crude simulation of the way a true sky has only a few independent noise, and then adding this common signal to each baseline. Finally the data are filtered again (with the same filter) to reflect the final fringe rate filter performed on the data.

$$X = \mathrm{frf}(\mathrm{frf}(\mathrm{E}) + \mathrm{N}) \tag{1}$$

the dimensions of X, and the length scale of the filter are sized to match time and frequency in a typical PSA64 data set.

The power spectrum calculation is pretty basic

$$C = X \cdot X^{\mathsf{T}} \tag{2}$$

Apply the inverse to the data

$$Y = C^{-1} \cdot \mathrm{X} \tag{3}$$

Sum the weighted data over many draws of noise (equivalent to baseline groups in the paper pipeline). Also sum the inverse weight matrices.

$$Y_g = \sum_i^{N_b} Y_i \tag{4}$$

$$\mathrm{Cinvsum} = \sum_i^{N_b} C_i^{-1} \tag{5}$$

where i indexes redundant baselines within groups/different draws of independent noise and simulates the effect of having the input signal be below the noise on every individual sample.

Next, cross multiply between different baseline groups

$$P_{out} = \frac{\sum_{i \neq j} Y_i \cdot Y_j}{\sum_{i \neq j} W_i W_j} \tag{6}$$

where i and j index different groups and W is a weight vector formed by summing across the rows of Cinvsum.

Using this method we calculate a number of power spectra for each trial,

- injected power spectrum P_{in}

[1] "Visibilities" are simulated here as real numbers to minimize complexity.

- weighted power spectrum output with inject P_{out}
- weighted power spectrum output *without* inject P_v
- power spectrum before weighting or injection (in this simulation this is usually just noise) P_{noise}
- "foreground" power spectrum (noise realization that is the same on every baseline) P_{fg}

The output for each inject value is a large number of trial output power spectrums, over which there is a wide range of losses (near 0 to 100%).

Loss Estimators

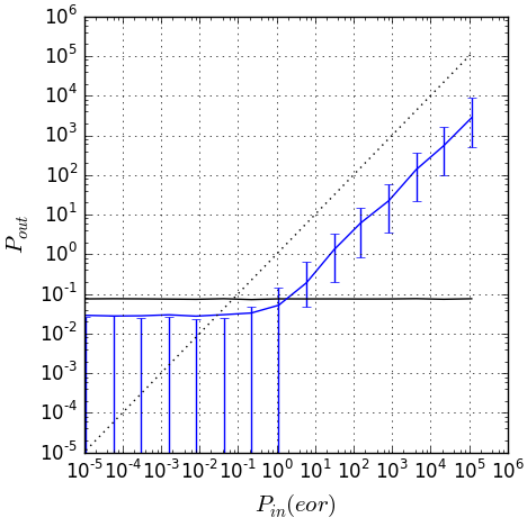
Given a lossy estimator we can formulate multiple slightly different questions:

First estimator: Total Power

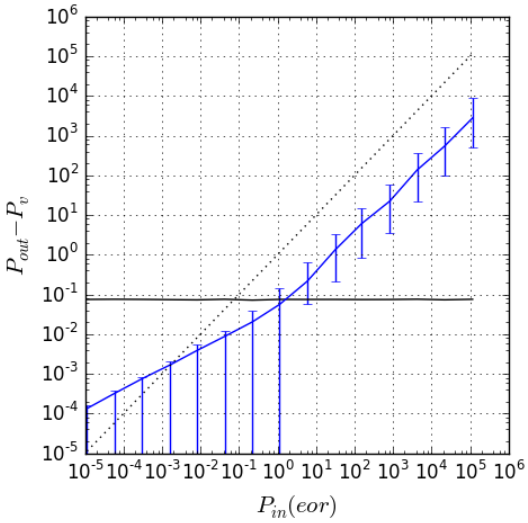
In this measure we ask: "What is the output power spectrum for a range of 21cm signal power levels¿' In this measure one sets the weighted power spectrum P_{meas} equal to the total output power spectrum,

$$P_v = P_{out} \tag{7}$$

P_{out} includes whatever is in the original data (noise, foregrounds, 21cm signal) as well as the injected 21cm simulation. The result is plotted in Figure 1a



(a) P_{out} vs P_{in}: Weighted net output power spectrum as a function of injected power spectrum. Error bars enclose 50% of the run outputs.



(b) $P_{out} - P_v$ vs P_{in}: Here we make an estimate of the recovered injected power spectrum by subtracting the weighted power spectrum of the data before the eor simulation is injected.

Second estimator: Recovered Power

Here we ask: "What is the recovered 21cm power spectrum level for a range 21cm signal power levels" We estimate the recovered 21cm signal by subtracting the power spectrum without the injected signal (ie the original measured power spectrum P_v)

$$P_v = P_{out} - P_v \tag{8}$$

This is plotted in Figure 1b

Effect of extra frf

In the loss calculation by C. Cheng we are injecting the simulated EoR signal *after* the application of the final frf filter. When we do the same thing in this simulation, we see that the signal loss decreases fairly substantially

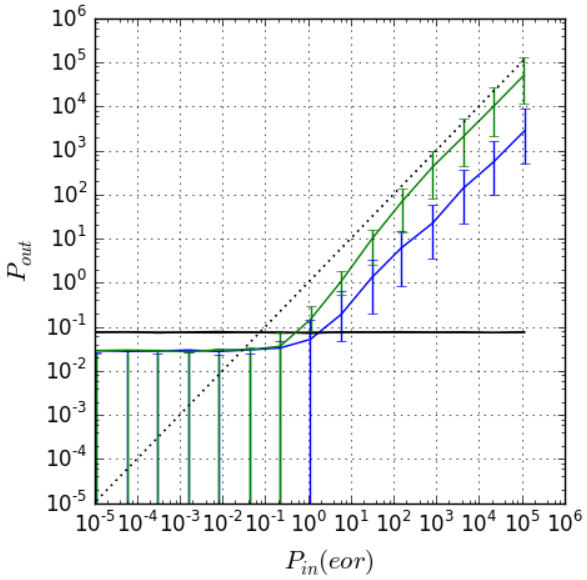


Figure 2: Injecting before (blue) and after (green) the fringe rate filter