HERA H1C Shadow Pipeline Validation Test #1

Matthew Kolopanis[1] and Daniel Jacobs[2]

[1,2]School Of Earth and Space Exploration, Arizona State University, Tempe, AZ

January 22, 2021

Abstract

A simpleDS analysis of full pipeline simulation; the effects of systematic removal; hidden value identification of injected signal

4 Nov 2020 Initial Draft

7 Dec 2020 Revision 1, after limited exchange with HERA validation group.

As part of a larger effort to build confidence in the HERA pipeline we are undertaking a parallel or "shadow" analysis. Here we limit our analysis to calculating a power spectrum from calibrated data. This parallels work by the HERA Power Spectrum and Statistics teams and uses outputs from the Analysis and Validation groups. Further discussion of this approach is described in a previous memo (D. Jacobs May 2020), attached below for your reference. That memo laid out the terms of this test. The point, to put it sharply, is rehearse the statistical and philosophical business of detecting a cosmological background when we don't know the answer.

The simulations were generated using the simulation pipeline developed by the HERA Validation team. These simulations include a noise model, a sky model, interference flagging (but not interference itself), and systematics all simulated to match the recorded 2017-2018 H1C observing season running to some dozen nights of data. This large dataset was then input into the HERA calibration and averaging pipeline to produce a single LST-binned dataset. Five of these simulations were produced for our purposes:

0.1 A known input power spectrum with no noise.

1.1 Foregrounds, noise, no EoR and systematic filtered

1.2 Foregrounds, noise, an unknown EoR and systematic filtered

1.3 Foregrounds, noise, no EoR

1.4 Foregrounds, noise, an unknown EoR

This memo details the results from the SIMPLEDS shadow pipeline test 1. Test 0 established that in the absence of any complications like foregrounds, noise, or flagging we are able to recover an eor power spectrum to within 50%, which is acceptable in comparison to our goal of 100ppm foreground removal. Each of the four sets is given a parallel analysis. We check our understanding of the expected systematics and noise level, compare theoretical and bootstrapped error bars, and make a power spectrum estimate.

Each of the four sets were analyzed in parallel. All four analysis notebooks are attached at the end of the document.

Though our aspiration was to maintain a strict veil around the shadow analysis, in practice some exchanges were required. We attempted to limit our discussion to specific points, like LST and frequency range selection, the weighting scheme used in the average and the meaning of flag and sample counts in the files. These are documented below by reporting the results of our initial round of analysis with minimal communication and a second one after resumption of talks.

As of this writing on 30 Nov 2020, we (DCJ and MK) maintain our ignorance of the final answer as to the true contents of the test sets and the level of cosmological signal injected therein. We have not read the relevant paper sections and have excused ourselves from discussion of those sections during paper discussion.

1 Initial Independent Analysis

In our first analysis we took delivery of four simulations generated by the Validation team according to the May 2020 memo. The files were scrubbed by changing the filenames and removing all history information. They are named Sets 1-4.

This section reports our first attempt at a hidden value analysis which was reported to the Validation team on 4 Nov 2020. Some figures have since been updated to reflect bug fixes motivated by later self consistency checks. These are noted with a **Bias Warning** like this. This is discussed more in subsection 2.1.

1.1 Input Data

The data consists of 10.7s integrations binned into 21.4s wide LST bins. We coherently average, using built-in pyuvdata tools, to generate a 5 min integration per sample.

1.2 Data selection

There are a number of analysis choices, most having to do with selection of data subsets. Our choices here are made to mimic the mainline HERA power spectrum pipeline as closely as possible, except where limited by the SIMPLEDS methods. Here are the selection ranges:

- No baseline with a projected length less than $14.6m$ are analyzed
- No same-baseline multiplications are used.
- Times are broken into two interleaved sets to mitigate noise bias.
- Lsts in the ranges [1, 2.7] and [4.5, 6.5] are analyzed
- Nominal spectral windows (in frequency channel): [175, 335] and [515, 695] are made "square" by trimming the higher range to [520, 680]

The spectral windows referenced above are illustrated in Figure 1. As mentioned above, nominal spectral windows for HERA power spectrum estimation are not "square" but SIMPLEDS requires "square" spectral windows for each redshift bin. This change is also made for simplicity when using

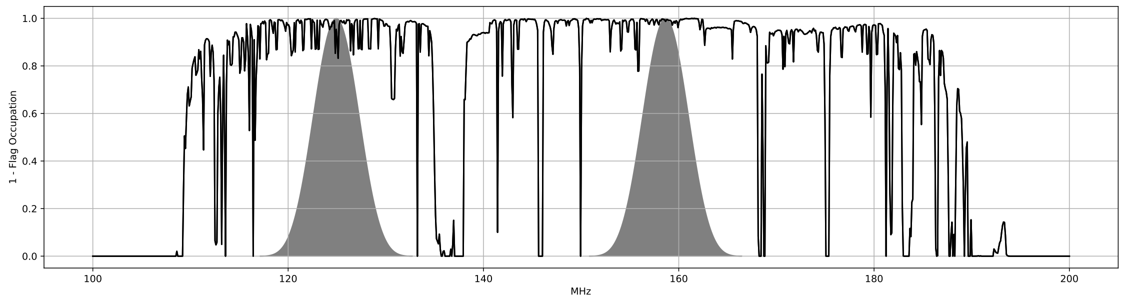


Figure 1: The two spectral windows used in the analysis.

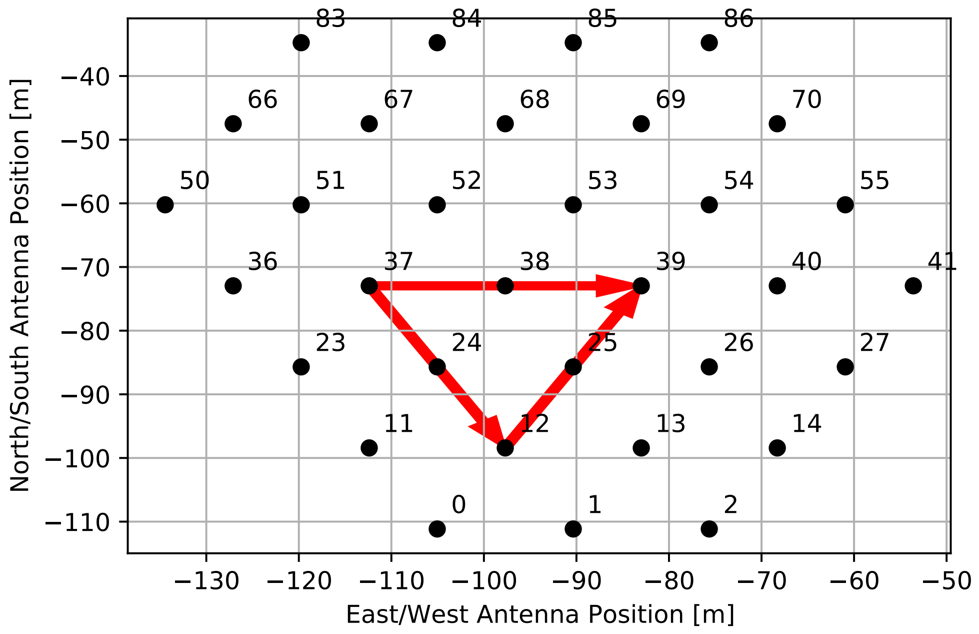

North/South Antenna Position [m]
East/West Antenna Position [m]

Figure 2: Three baselines representing the redundant groups used in this analysis

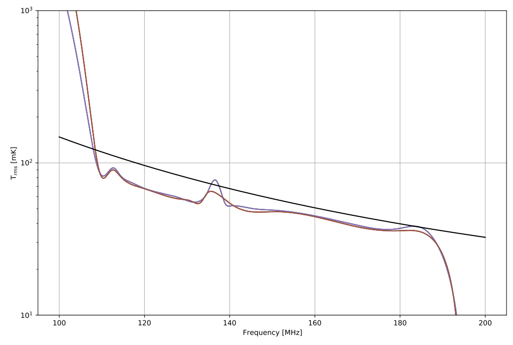
the SIMPLEDS object. They are selected to span regions with the most occupancy of data during LST binning and minimum flag occupancy.

The main limitation imposed by SIMPLEDS is in its simplistic approach to averaging in *uv*. SIMPLEDS is designed to operate on only one set of redundant baselines at a time. As a result, three separate analysis chains are analyzed in tandem to estimate the delay spectra from the baseline groups illustrated in Figure 2.

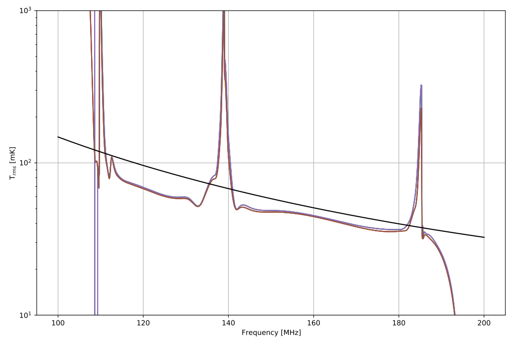
1.3 Error propagation

We estimate error bars in three ways, all of which discussed and developed in Kolopanis et al. (2019). First we ground our analysis by comparing to theoretical estimates of the type used to predict the sensitivity of the array. Second we give ourselves a reference with a noise data set which is processed in parallel with the data. This is useful as a check for our primary error bar estimation with a bootstrap sampling. Third, we estimate the actual variance in data samples with a bootstrap resampling.

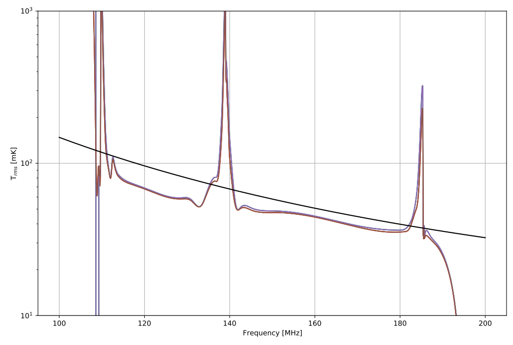
The noise simulation is based on the variance observed in the LST binned data. A difference is



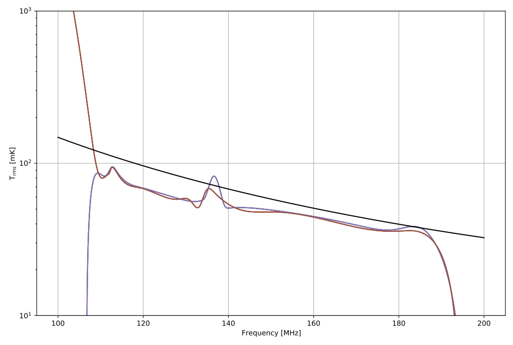
(a) Data set 1



(b) Data set 2



(c) Data set 3



(d) Data set 4

Figure 3: **Bias Warning** Figure updated in revision one to correct a bug where times did not match those included in analysis. T_{rms} estimated by differencing adjacent 20s time bins and averaged over baselines. Agrees within 25% of theory estimate (Equation 1) where we assume $\mathrm{T_{rcvr}} = 50$ K. We noted strong divergences in sets 2 and 3. Later we learned that delay spectrum reconstruction of missing channels (lately referred to as in-painting) was done in two bands separated by orbcomm at 137MHz. This step weights by a Blackman-Harris window which improves spectral dynamic range at the cost of increased uncertainty at the edges.

taken across interleaved time samples to provide a per time, per frequency noise realization. The root mean square of these estimations is taken over all baselines (in a redundant group) and time to estimate a per frequency noise power. noise powers, shown in Figure 3, are then used to create independent realizations of noise for every baseline.

These noise powers are compared to the predicted noise given the usual two component sky/thermal model Rogers & Bowman (2008)

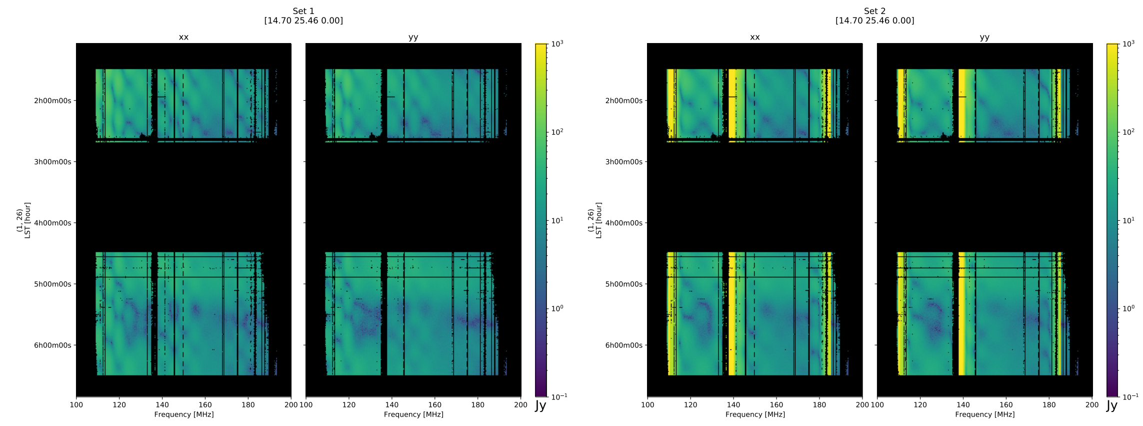
$$T_{\mathrm{sys}} = 180\ K \left(\frac{\nu}{180\mathrm{MHz}}\right)^{-2.55} + T_{\mathrm{rcvr}} \tag{1}$$

where we assume $T_{\mathrm{rcvr}} = 50$ K.

1.4 Identifying Systematic Removal

Bias Warning This section has been added for clarity after the HERA validation discussion and relies on knowledge of the systematic removal step gained after this discussion.

We have set up this test to answer two questions. First, can we detect the EoR in simulation of the complete HERA pipeline? However, fully aware that despite evidence to the contrary, we



(a) Data set 1 (b) Data set 2

Figure 4: Left: The visibility amplitude from a representative baseline from data set 1. Right: The visibility amplitude from a representative baseline from data set 2. Again we see high power regions at band edges due to per-band Fourier filtering.

worry about systematic filtering leading to false negatives or false positive we have attempted to answer an additional question: can we make a detection without the systematic filtering step? Our simulation properties are formally hidden from us, including the status of systematic removal, so as a first step we attempt to logically identify which Sets have had systematic removal applied.

Inspecting the amplitudes Figure 4 and variances Figure 3, a clear difference is immediately apparent. The systematic removal step is performed in delay space and as a result of the use of tapers during Fourier Transforms the final product will have large excesses at the band edges. Data sets 2 and 3 display multiple large noise excesses near the band edges selected for power spectrum estimation. These excess are consistent with the end result of a systematic removed dataset. Figure 4 shows the visual difference between data sets without systematic removal (data set 1 shown) and those with systematic removed (data set 2 shown). Visual inspection can easily bin the 4 data sets into groups with and without systematics removed.

The variance waterfalls for baselines in each of the four data sets also support this classification. Figure 5 compares the bootstrap variance for a baseline in each dataset to the bootstrap variance of the input noise simulation. Simulation sets 1 and 4, display excess variance near $\tau \sim 1000$ ns and from this we can gain conclude they have not had systematics like cable reflections removed. Similarly, the lack of these characteristics in simulation sets 2 and 3 indicate these two sets have undergone reflection removal. The higher variance relative to a noise simulation at these delay modes is consistent with high power from the $\tau = 0$ ns mode coupling to this mode through a cable reflection. This effect can cause high variance if the signal is exhibits similar amplitude on all baselines but with a varying phase. In other words, the amplitude of this signal can average out when cross multiplying many baselines, but the baseline to baseline variations will manifest in the uncertainty.

1.5 Calculating The Power Spectrum

SIMPLEDS estimates the power spectrum as a Fourier Transform over frequency to delay space and cross-multiplied across redundant baselines between every baseline and interleaved time pair. This provides both upper and lower triangles of the cross-multiplication matrix between baselines from time set 1 and baseline from time set 2. The diagonal of this matrix (same baseline but different time set) is discarded in order to reduce any chance of noise bias in the final estimate. The mean of this matrix yields a power spectrum at every time. Uncertainties are calculated, again per time step, by sampling with replacement over the baseline pair dimension and with each sample set calculating a mean. The standard deviation of the re-sampled means yields an error spectrum for each time.

These spectra are then incoherently averaged (folded) across the $\tau = 0$ ns line and then over time to produce a single spectrum as a function of $|\tau|$ for each baseline type, redshift, and polarization. **Bias Warning** The weighting used in the original analysis applied inverse variance using the bootstrap uncertainties. After discussion, this weighting scheme is changed to use $1/P_N^2$ weighting. P_N is constant across all delays but varies with time according to the number of LST samples. We estimate it as the delay averaged bootstrap uncertainties from the noise simulation which we've shown to agree with the more theoretical estimate.

Using unit conversion factors defined in Liu et al. (2014), Parsons et al. (2012b, 2014) the power spectrum is transformed from a function of delay $(P(|\tau|))$ to cosmological wavenumber $(P(|k_{\|}|))$.

Instead of performing averages over baselines in some annulus in k-space, we have simply averaged all three baseline types, which were pre-selected for this purpose.

These error bars can be compared to the theoretical sensitivity of the telescope calculated using the expected noise power derived in Parsons et al. (2012a) and applied in Pober et al. (2013, 2014), Cheng et al. (2018) and Kolopanis et al. (2019).

$$P_N(k) = \frac{X^2 Y \Omega_{eff} T_{sys}^2}{t_{int} N_{days} N_{bls} \sqrt{2 N_{lst}}} \tag{2}$$

where relevant definitions of each factor can be found in the attached analysis notebooks in Appendix A

1.6 Interpreting the results

Identifying the presence or lack of EoR signal in the simulations is not as trivially accomplished. Figure 6 and Figure 7 subdivide the simulations into sets with and without systematics removed for each redshift bin and polarization. As seen in the figures, an excess of power is distinguishable in the power spectrum of set 3 compared to 2 at this redshift in the range 300 ns $\leq \tau \leq$ 800 ns in the $z = 10.38$ bin.

In contrast, in the $z = 7.96$ bin, no discernible excess exists between the power spectra in each sub group with the exception of 2 data points in the range 500 ns $\leq \tau \leq$ 750 ns. However given the use of blackman-harris taper during the Fourier Transform, these two data points should be considered highly correlated and as a result would only be indicative of a single independent sample. Therefore there are no statistically significant differences between these two sets of simulations outside of the foreground dominated region. Inside the foreground dominated region there are statistically significant deviations at almost every point. Most notably that the largest power value is not observed at the $\tau = 0$ ns delay mode.

The statistical differences inside the foreground wedge lead to the conclusion that there is some over all large scale differences between simulation sets 3 and 2, and sets 1 and 4 however it is not possible to discern what these differences are in the $z = 7.96$ bin.

We note that the large bias in all sets looks suspiciously like the result of uneven spectral sampling of the type usually handled by a CLEAN-type reconstruction. Examining the sampling function in Figure 1 we see only modest variation in the spectral weights; some channels have more data points than others but all have something. Variation in sample counts will cause each channel to have slightly different error bars. This is included in the noise model which does not show any kind of offset. Nevertheless, we continue to harbor suspicions that some kind of window function is at work. **Bias Warning** This will turn out to be the case later.

2 Analysis 2: after limited discussion with Validation group

The outcome of the first analysis was, essentially a null result. Though we could easily tell which sets had their systematics removed, we saw a large bias swamping anything fainter.

Per the terms of the hidden value experiment, work up to this point had been done under a communication blackout. Though this was done to preserve the integrity of the test, it also limited our ability to align the details of the analysis choices with the primary group. Realizing this we all agreed to attempt correcting differences without contaminating us. We met with the Validation group and described the above analysis. This exchange revealed in a small number of differences between shadow and mainline analysis, none of which were perceived to be significant impact. Though we have taken precautions to avoid contamination we have marked any plots made after our exchange with a **Bias Warning** giving any details necessary like this.

2.1 Analysis choices different from mainline

The group noted several small differences in data selection or statistical measure. In this section we work through these changes and report any changes that result.

- LST ranges selected by the collaboration were slightly different than the ones we chose originally. First we adjusted to the mainline ranges of ([1.5, 2.7], [4.5, 6.5]) and then, as discussed later, these are again altered to ([1.5,2.5], [5.65,6.5]) to reduce a positive bias seen in the baseline type (37, 12).
- HERA official power spectrum uses a $\frac{1}{P_N^2}$ weighting to average which is independent of delay.
- Do not flag areas where the number of samples is zero. These zeroes represent no information, but the in-painting should not be flagged to allow for smooth Fourier Transforms. Flagging obviates the point of in-painting.

These differences are all small corrections in the pipeline. One decision must be made, however. SIMPLEDS provides multiple estimates of P_N: theoretical sensitivity, bootstrap error, time difference noise propagated as a noise simulation. We have elected to use the uncertainties of the noise simulation averaged across all delay modes to estimate P_N. Unlike the sensitivity theory in Equation 2, this estimator will reflect time variation in the thermal noise due to changing number of LST samples, but not additional variation due to non-redundancy. It strikes a middle of the road balance.

Updated results from these changes are shown in Figure 8 and Figure 9. The presence of many highly significant detections above the noise floor are obvious in all redshifts, and polarizations which are consistent across all four datasets. Since these detections are consistent across all four data sets, it is reasonable to conclude they are due to a non-EoR systematic signal present in the simulations.

2.2 Other tests: Slow Systematic

One possible explanation for the above bias is an error in simulation causing some added systematic to remain. We investigate the impact of a residual low-fringe rate (slowly time varying) offset by removing the time average (per frequency) from the data and performing the experiment over again. No significant difference was observed.

However other possible sources are not as obvious to identify and rule out. Suppressed by 6 orders of magnitude in the power spectrum equals to a systematic at the -30 dB level in visibility space and would appear below the per-baseline noise level. A follow up analysis using a statistical "drop-out" test could identify if individual baseline exhibit this systematic bias.

Two additional tests were conducted to identify possible sources of discrepancy but no discernible results were gleaned:

- Analyse each LST bin separately, instead of averaging all together
- Increasing the length of both spectral windows to be equal to the original length of the higher frequency band.

2.3 Other tests: systematic flagging

The biased systematic seems to be concentrated in the baseline type (37, 12) which is the southeasterly direction, see Figure 11. In considering unbiased ways to identify offsets like this we considered the SSINS (Wilensky et al., 2019) algorithm which identifies bias in visibilities below the noise of an individual baseline. We applied the SSINS algorithm which identified a significant increase in variance near the boundaries of the LST window. Revising these windows to ([1.5,2.5], [5.65,6.5]) resulted in a small improvement Figure 12 and Figure 13 but not enough to say which set contains an obvious EoR signal.

There is a statistical difference in the $z = 10.38$ redshift bin between data sets 2 and 3 Figure 10. However, data sets 1 and 4 are not statistically different in this redshift bin.

This all weakly implies data data set 3 has EoR simulation while 2 does not, but no conclusions can be drawn about data sets 1 and 4.

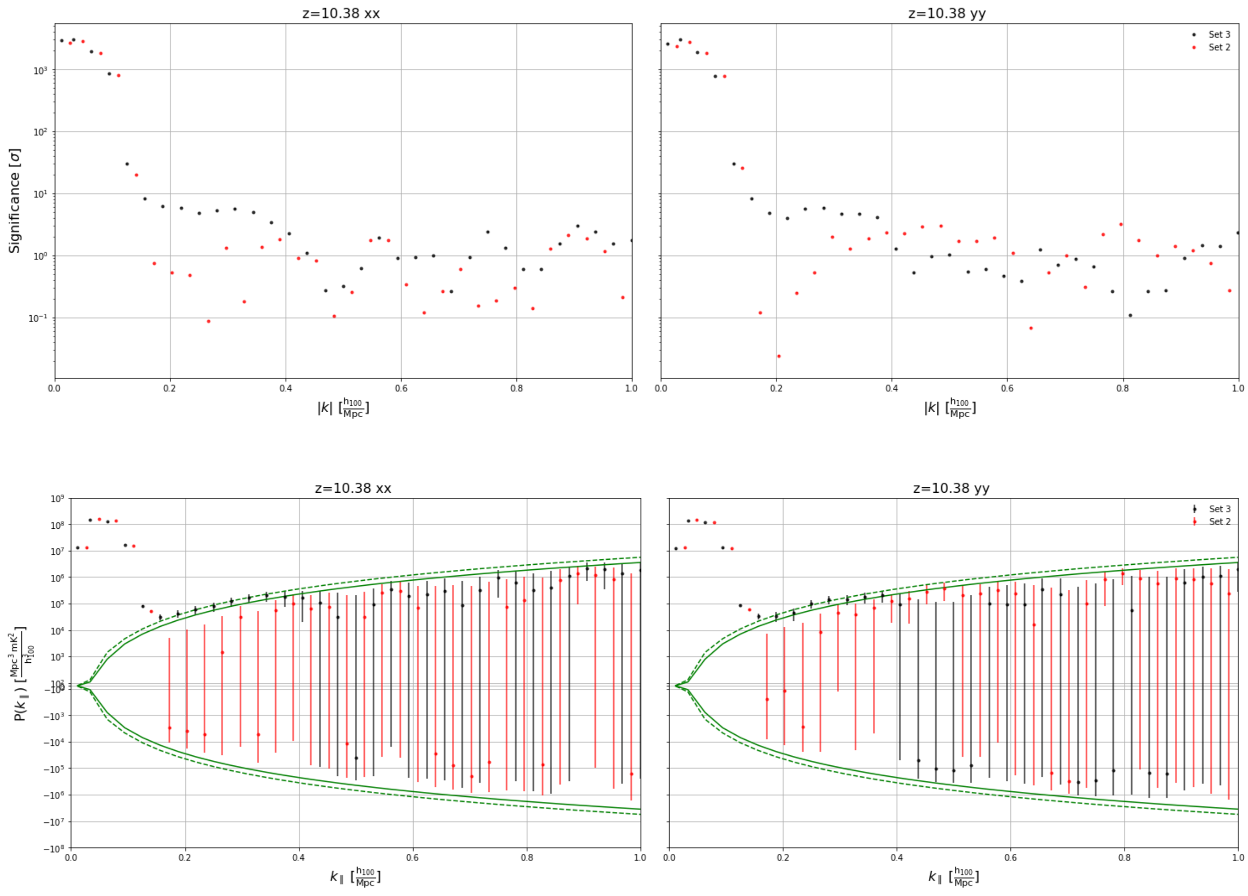


Figure 10: **Bias Warning** In set 3 we see 7 points between k of 0.2 and 0.35 which are substantially above the noise. In Set 2 these points are consistent with noise. These points have Δ^2 amplitudes of 2e4 rising to 2e5 mK^2/h^3

2.4 Revision Summary and Discussion

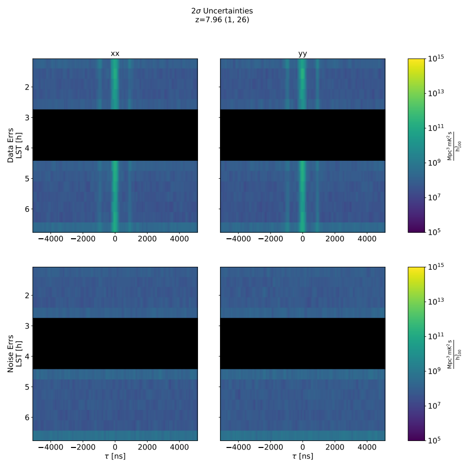
As the net result of these revisions seems to be a potentially conclusive improvement at redshift 10 Figure 9 but not Figure 8, it is important to pinpoint the significance of the various changes. We made an A/B study of each proposed change, examining the difference before and after the change and also noting any increased separation between sets 2 and 3. In most cases no significant improvement was noted. The most notable improvement came from restricting the input LST range; first narrowed from our initial uninformed selection to the selection used in the mainline analysis, and then, motivated by the SSINS variance, even slightly further.

HERA systematic removal is applied as a filter in the time and frequency domains. This filter is applied on individual spectral and temporal subsections defined by the LST and spectral windows. Each axis is weighted to minimize ringing resulting in large noise at the time and frequency edges. These have been accounted for in the HERA data selections but somehow that selection is a little too wide in this analysis. We speculate that either A) the settings of the filter applied to this simulation do not match those used to justify the mainline HERA LST cut B) the affected baseline type is not included in the HERA analysis, or C) something else.

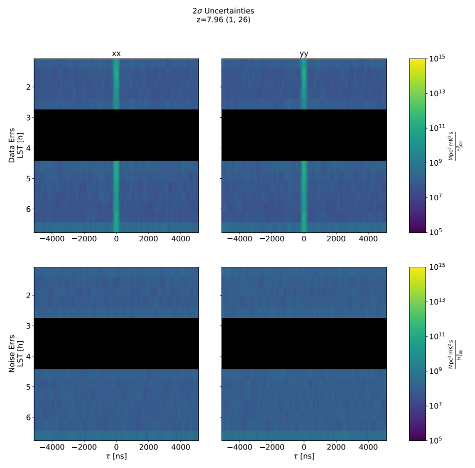
3 Conclusions

We offer the following conclusions based on the above analysis

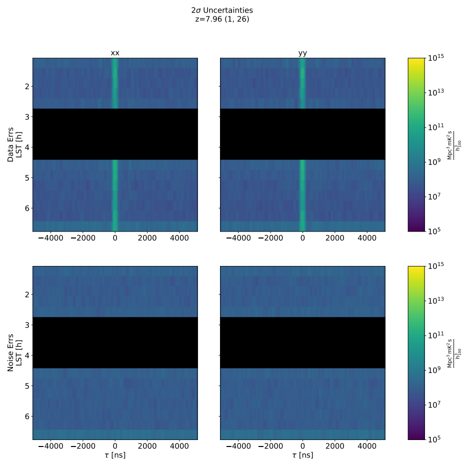
1. Data sets 1 and 4 have not been processed for systematic removal. Figure 3,Figure 4,Figure 5
2. Datasets 1 and 4 are indistinguishable under inspection but we have not offered a numerical confirmation of this conclusion.
3. In Data set 3 we observe a detection at redshift 11 above data set 2 in k values below 0.4 $hMpc^{-1}$ Figure 6, and **Bias Warning**
4. Dataset #2 is consistent with noise in this range. given the known parameters of the test, we conclude that dataset 3 contains an EoR simulation with an amplitude roughly 5x that of the noise.
5. We cannot offer a similar conclusion about EoR at redshift 7.



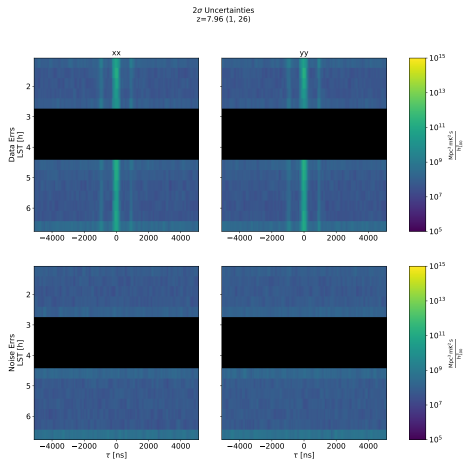
(a) Data set 1



(b) Data set 2

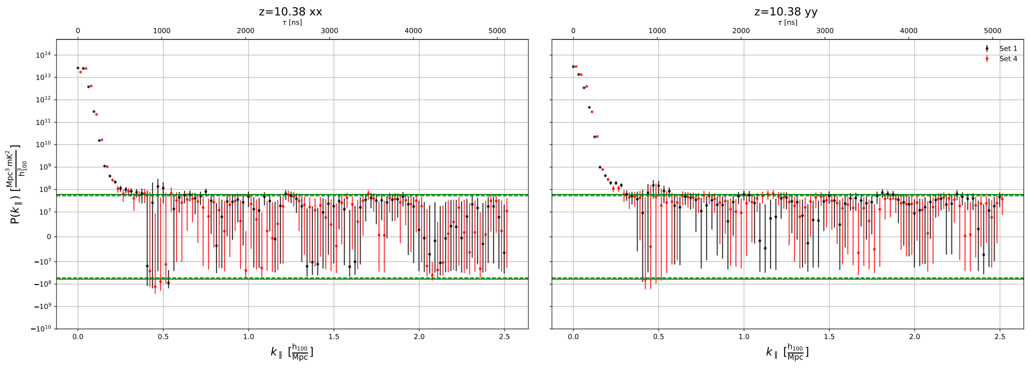


(c) Data set 3

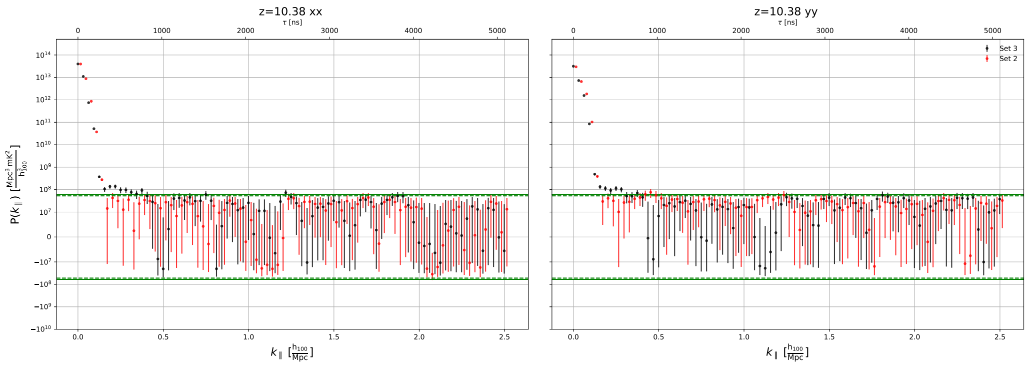


(d) Data set 4

Figure 5: Waterfalls of the bootstrap variance for each simulated data set and accompanying noise simulation. The excess variance near ±1000 ns in sets 1 and 4 indicate these simulations have not undergone systematic removal.

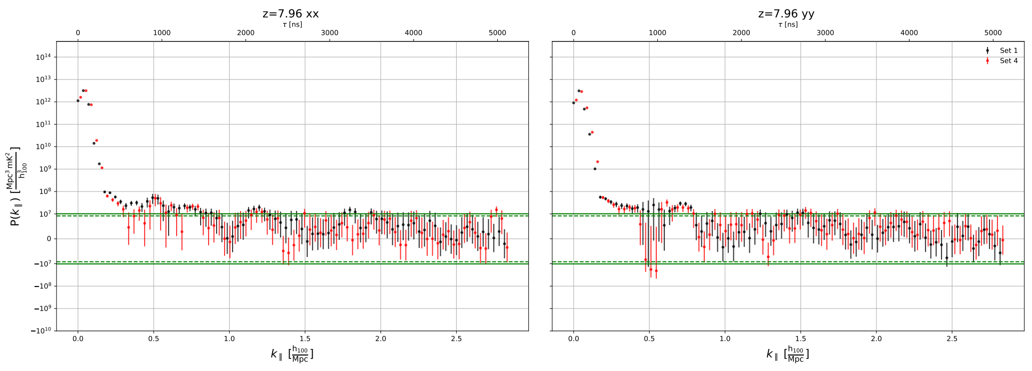


(a) Data sets 1 (black) and 4 (red) – systematic present



(b) Data sets 3 (black) and 2 (red) – systematic removed?

Figure 6: Top: the averaged power spectra for sets 1 (black) and 4 (red) at $z = 10.38$. Bottom: the averaged power spectra of sets 3 (black) and 2 (red) at $z = 10.38$. Power spectra are grouped together based on presence or absence of large variances seen in Figure 3. and Figure 5. An excess of power is distinguishable in the power spectrum of set 3 compared to 2 at this redshift in the range 300 ns $\leq \tau \leq$ 800 ns.


(a) Data sets 1 (black) and 4 (red) - systematic present

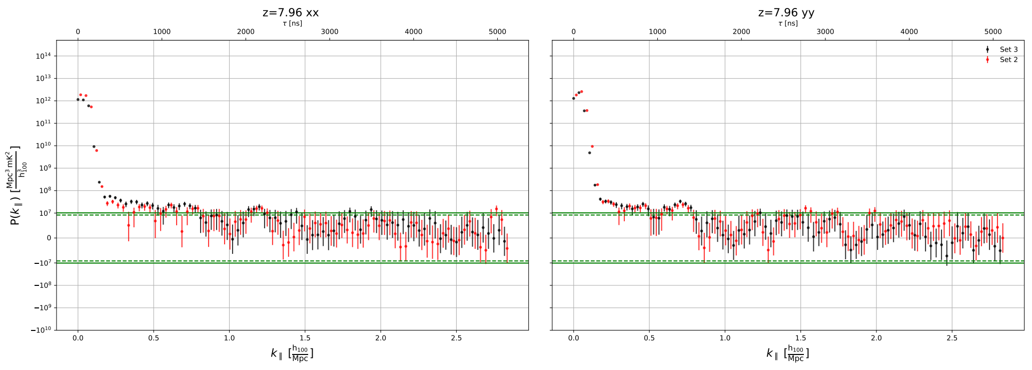
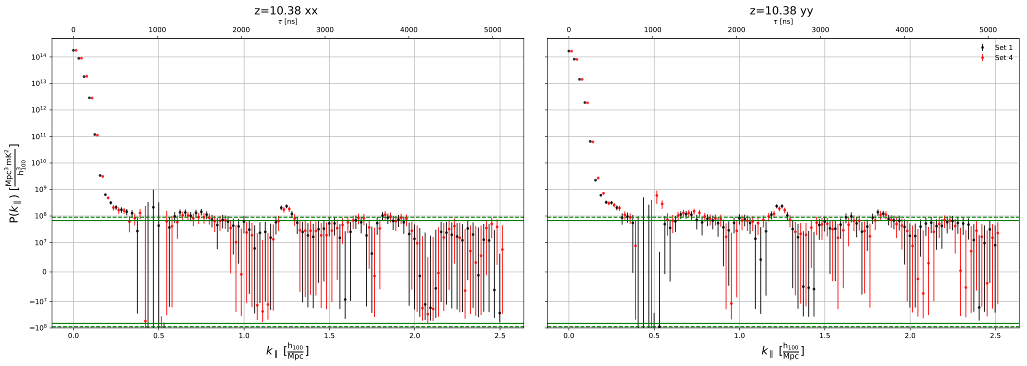
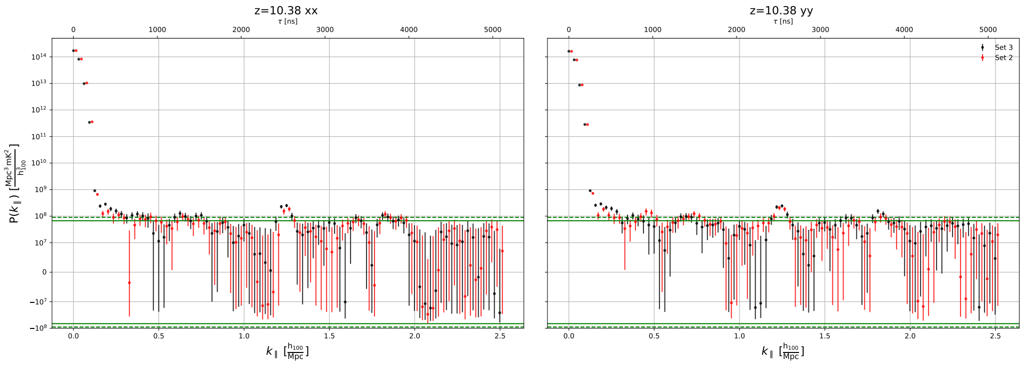

(b) Data sets 3 (black) and 2 (red) - systematic removed?

Figure 7: The same as Figure 6 except at redshift $z = 7.96$. No discernible excess exists between the power spectra in each sub group with the exception of 2 data points in the range 500 ns $\leq \tau \leq$ 750 ns. However given the use of Blackman-Harris taper during the Fourier Transform, these two data points should be considered highly correlated and effectively just one independent sample. Therefore there are no statistically significant differences between these two sets of simulations outside of the foreground dominated region. Inside the wedge there are statistically significant deviations at almost every point. Most notably that the largest power value is not observed at the $\tau = 0$ ns delay mode. This is predominantly caused by a single redundant baseline group dominating the average. The statistical differences inside the foreground wedge lead to the conclusion that there is some over all large scale differences between simulation sets 3 and 2, and sets 1 and 4.

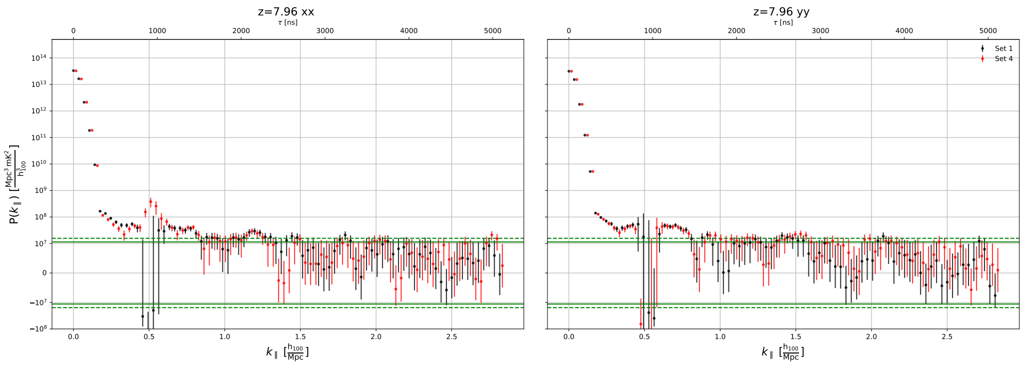


(a) Data sets 1 (black) and 4 (red) – systematic present

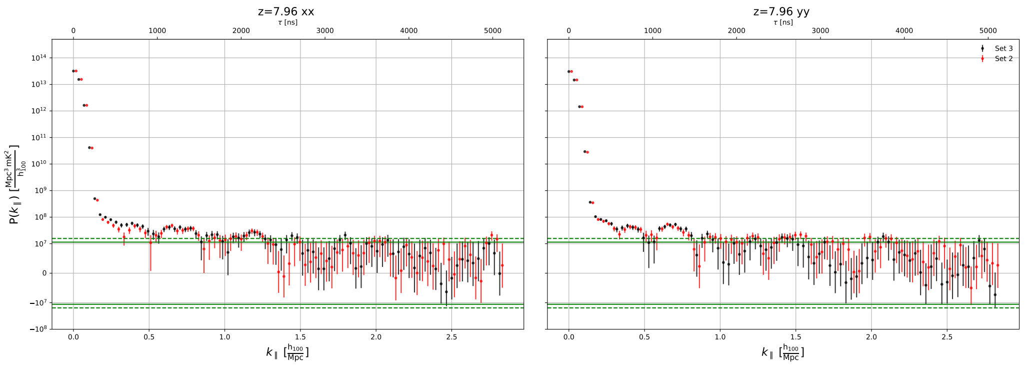


(b) Data sets 3 (black) and 2 (red) – systematic removed

Figure 8: **Bias Warning** Power spectra at $z = 10.38$ after changes described in subsection 2.1: LST range, weighting, and treatment of flags. The presence of highly significant detections in all sets of simulations, across many delay bins implies a systematic contaminating the power spectrum estimation. This systematic is below the per-baseline noise floor, but well above the noise floor of the integrated power spectrum.

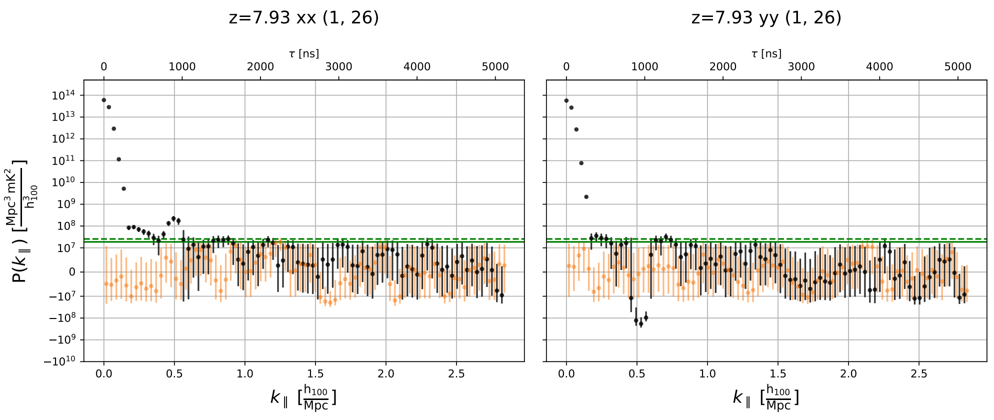


(a) Data sets 1 (black) and 4 (red)

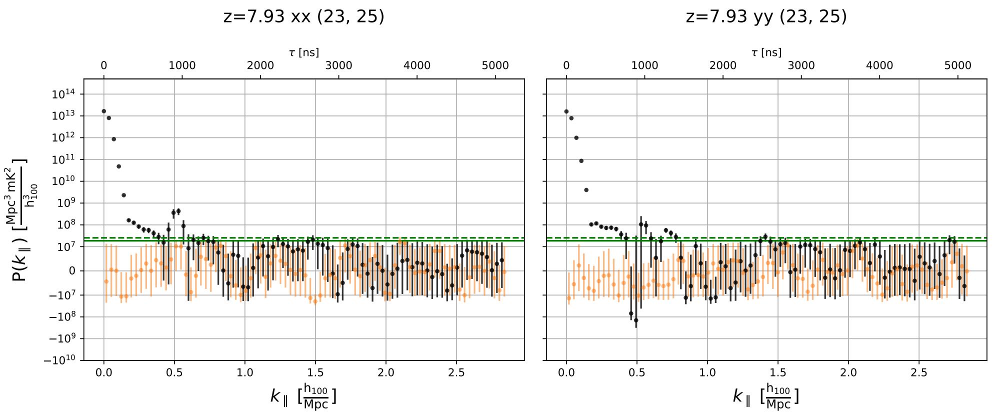


(b) Data sets 3 (black) and 2 (red) – systematic removed

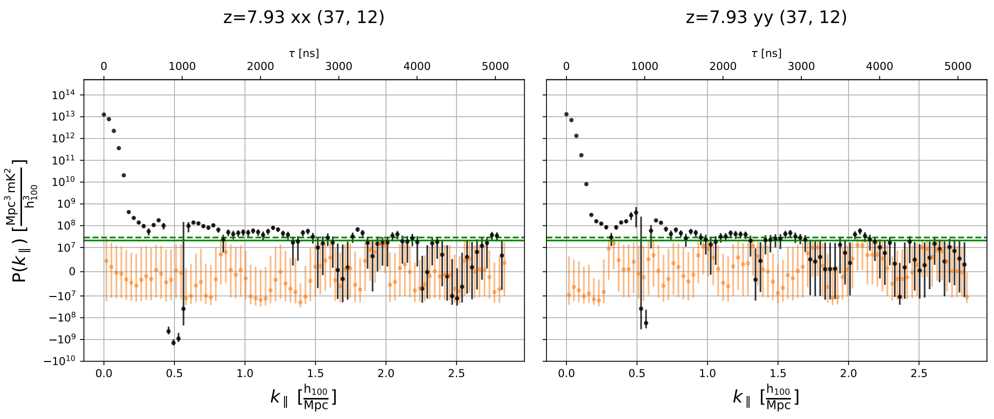
Figure 9: **Bias Warning** Power spectra after changes described in subsection 2.1: LST range, weighting, and treatment of flags. The same as Figure 8 except at redshift $z = 7.96$. The lower redshift bins still have no statistical differences providing evidence for the existence of EoR in any data sets.



(a) Redundant baseline group like (1, 26)

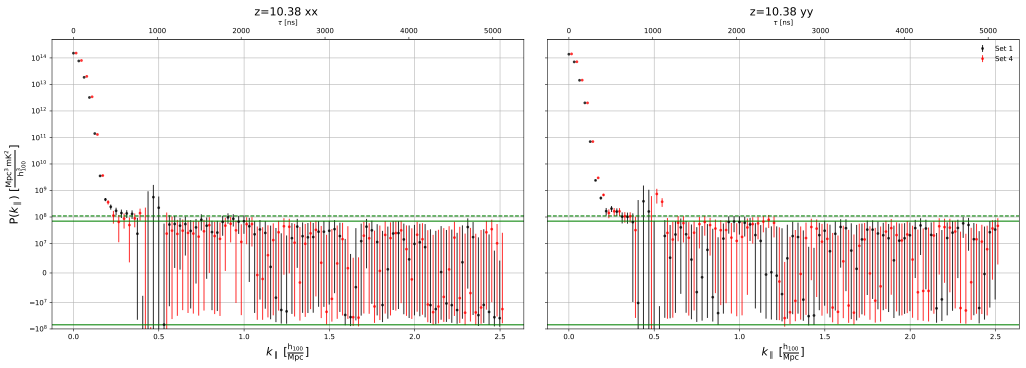


(b) Redundant baseline group like (12, 14)

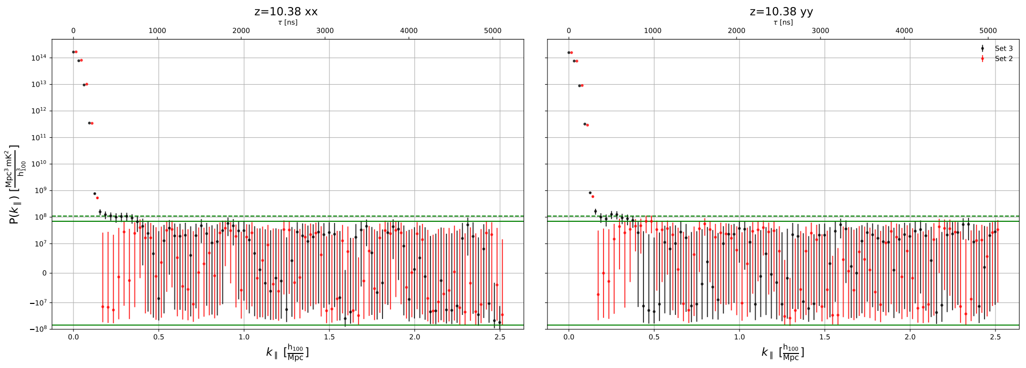


(c) Redundant baseline group like (37, 12)

Figure 11: **Bias Warning** Systematic bias seems strongest on one type of baseline, seen here in data set 1. The baseline type like (37, 12), the southeasterly baseline, has a much stronger bias. There is no other motivation for excluding this baseline in final averaging. Shrinking the included LST range has proven effective at reducing this bias.

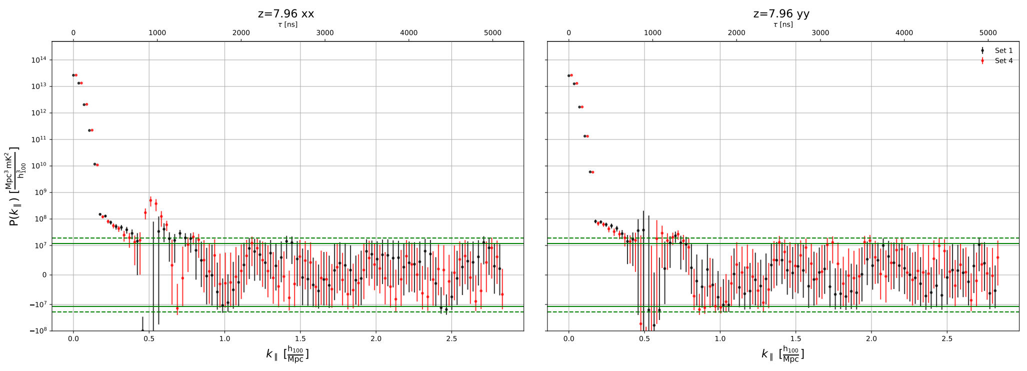


(a) Data sets 1 (black) and 4 (red)

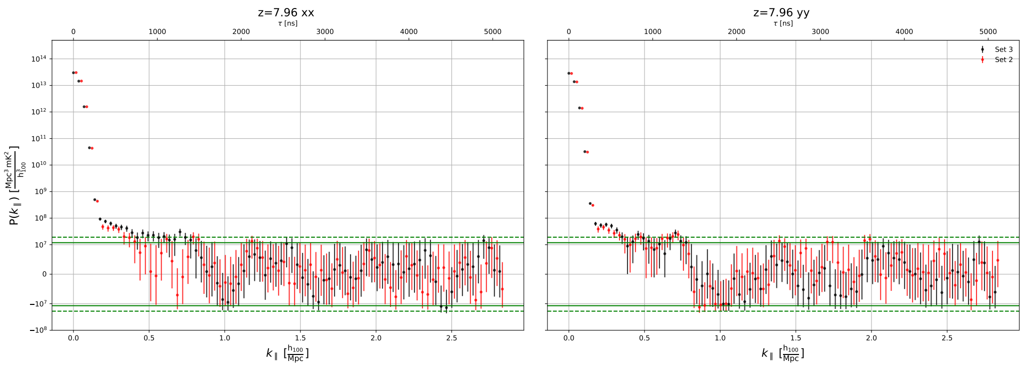


(b) Data sets 3 (black) and 2 (red) – systematic removed

Figure 12: **Bias Warning** Top: the averaged power spectra for sets 1 (black) and 4 (red) at $z = 10.38$. Bottom: the averaged power spectra of sets 3 (black) and 2 (red) at $z = 10.38$. Power spectra are grouped together based on presence or removal of a reflection systematic near $\tau \sim 1000$ ns. There is no direct evidence for the existence of EoR signal in the data sets 1 and 4 comparison, however data set 3 has a highly significant excess compared to set 2 below $k = .5$ h^{-1}Mpc



(a) Data sets 1 (black) and 4 (red)



(b) Data sets 3 (black) and 2 (red) – systematic removed

Figure 13: **Bias Warning** The same as Figure 12 except at redshift $z = 7.96$. The lower redshift bins still have no statistical differences providing evidence for the existence of EoR in any data sets.

References

Cheng, C., Parsons, A. R., Kolopanis, M., et al. 2018, ApJ, 868, 26, doi: 10.3847/1538-4357/aae833

Kolopanis, M., Jacobs, D. C., Cheng, C., et al. 2019, ApJ, 883, 133, doi: 10.3847/1538-4357/ab3e3a

Liu, A., Parsons, A. R., & Trott, C. M. 2014, Phys. Rev. D, 90, 023018, doi: 10.1103/PhysRevD.90.023018

Parsons, A., Pober, J., McQuinn, M., Jacobs, D., & Aguirre, J. 2012a, ApJ, 753, 81, doi: 10.1088/0004-637X/753/1/81

Parsons, A. R., Pober, J. C., Aguirre, J. E., et al. 2012b, ApJ, 756, 165, doi: 10.1088/0004-637X/756/2/165

Parsons, A. R., Liu, A., Aguirre, J. E., et al. 2014, ApJ, 788, 106, doi: 10.1088/0004-637X/788/2/106

Pober, J. C., Parsons, A. R., Aguirre, J. E., et al. 2013, ApJ, 768, L36, doi: 10.1088/2041-8205/768/2/L36

Pober, J. C., Liu, A., Dillon, J. S., et al. 2014, ApJ, 782, 66, doi: 10.1088/0004-637X/782/2/66

Rogers, A. E. E., & Bowman, J. D. 2008, AJ, 136, 641, doi: 10.1088/0004-6256/136/2/641

Wilensky, M. J., Morales, M. F., Hazelton, B. J., et al. 2019, PASP, 131, 114507, doi: 10.1088/1538-3873/ab3cad

Appendices

Shadow Pipeline Plan

HERA
Internal Memo

Plan for a hidden answer test of HERA simpleDS shadow pipeline
12 May 2020
D. Jacobs

A Narrative
This is a plan for a hidden-answer test of two power spectrum pipelines. The broader goal of the parallel pipeline test is to add another way of guarding against experimenter bias in pipeline development. Ideally while not hobbling ourselves in the face of limited understanding of systematics. A shadow power spectrum pipeline using simpleDS has been part of the project plan for some time, but has not been discussed much until recently. **We are now faced with a desire to know how closely the two power spectrum codes get, but don't want to test on the real data.** The memo summarizes discussions had over several meetings of the hera_pspec group and the broader collaboration during datacons.

We propose a test which is distinct from the mainline HERA Validation effort but uses simulations generated by that group. Validation efforts are focused on testing the primary pipeline. Here we will use a few of the Validation tests to make a hidden answer test of the primary power spectrum estimator (hera_pspec) and the "shadow" simulator (simpleDS).

We want several things from this test. First we want to see that at least two different implementations of similar algorithms get close to the expected answer, recognizing that we might not have a 100% analytical prediction for the expectation answer given the inputs. Second we want to see that both register the effect of the systematic removal portion of our calibration pipeline. Third, we want an exercise in reporting a result when we don't know the answer.

The simplest way to incur experimenter bias is to draw conclusions from exploration data. The solution is to build simulations that model the data and use this to test the pipeline. This is the goal of the HERA Validation team. However another kind of bias can still creep in if there is tight feedback between the simulations and the analysis pipelines or between two parallel pipelines and they begin to influence each other. This has been seen for example in the development of the penultimate PAPER pipeline where noise simulations and analytic error were developed in tight coupling, ending with two codes outputting error bars that were erroneously small. The goal here is to make checks between analysis steps with minimal development interactions.

A Plan
Using simulations generated by the validation group which has been operating largely independently from the development of simpleDS we will have avoided tight coupling. We will use some **known answer** simulations for a limited validation check and then switch to **unknown answer** simulations.

In the case of the unknown answer we will attempt to place limits on the EoR using only the information available to us in the case of real data. Given that one tool at our disposal is secrecy, we will observe the

Shadow Pipeline Plan

HERA
Internal Memo

principle that conclusions about any hidden answer will provide an interesting data point at small extra cost in resources. In other words, **lets keep as much as possible secret!**

Known Answer Test

We will use the **Noise + flat Pk** validation test

1. Thermal noise is reference simulation 0.0 <insert NRAO path here>
2. flat gaussian Pk is ref sim 0.1 (no noise) /lustre/aoc/projects/hera/alanman/eor_sky_sim/

Question: are we returning an answer consistent with analytic predictions?

Unknown Answer

The hidden value test should be run sparingly. This will minimize feedback loops. This pushes us to think each run out carefully, taking advantage of possible variations. At the present time it is thought that the systematic removal step in the pipeline will present the most uncertainty in a final analysis. Also the receiver temperature is usually poorly constrained and estimated as part of a power spectrum analysis. This motivates the following variations which are listed here and summarized in Table 1.

Hidden Variation 1:
Whether the simulated high delay systematic has been filtered.
Hidden Variation 2:
EoR power spectrum level and shape. In half the simulations EoR will be excluded, in the other half it will be included at an unknown level and spectrum shape.
Hidden Parameter 1:
All simulations will include noise. We will constrain receiver temperature variation to being within 50% of a flat 100K+-50K. Trcvr will not vary between simulations.

Filenames should be cleansed of all hidden values. LST binned data products will be integrated into a single dataset (not split into even/odd or whatever)

Table 1: First HERA Hidden Value test. Values in Red are known to the simulator but hidden from the analyst.

Sky	**Noise**	**EoR**	**Systematic filtering**
GSM+Gleam + gain errors + calibration	Trcvr = 100K+-50K	Some level	No
GSM+Gleam + calibration + gain errors + calibration	Trcvr = 100K+-50K	None	No
GSM+Gleam + calibration + gain errors + calibration	Trcvr = 100K+-50K	Some level	Yes
GSM+Gleam + calibration + gain errors + calibration	Trcvr = 100K+-50K	None	Yes

Shadow Pipeline Plan

HERA
Internal Memo

🟩 Known to Analysis	🟥 Hidden from Analysis

Organization
Simulations
Simulations will be generated by the Validation group. Noise and flat Pk available now. Results Simulations from the Hidden Value expected in a week or two May 20 - 27. Form of results LST binned (10days, 6 hours of LST) simulated data.

Analysis
SimpleDS will be carried out by M. Kolopanis and D. Jacobs. Report on Noise+Pl to be delivered by May 22. Initial reports on hidden analysis will be discussed within the ASU group before reporting out to the group. Final report comparing all results and reporting out guesses for model components and EoR levels will be made two weeks (TBD) after simulation delivery.

Parameters
We'd like the power spectrum parameters to match up between the prime and shadow analyses.
Spectral window--channels 515:695
LST ranges
Type of polarization products (I vs xx)
Form of power spectrum (cross baselines, t_i * t_{i+1})

- bl_1_time_even x bl_2_time_odd and bl_2_time_even x bl_1_time_odd

All LSTs, integrated in time, and integrated in baseline.
Beams: /lustre/aoc/projects/hera/Validation/HERA_beams

- NF_HERA_dipole_linpol_power_healpix128.fits
- NF_HERA_dipole_IQ_power_healpix128.fits

Shadow Pipeline Pspec/stage0.1 validation

SimpleDS Shadow Pipeline Test 0.1

Matthew Kolopanis

October 28, 2020

- Initial Memo: May 29, 2020
- Revision 1: June 5, 2020

SimpleDS is a python packaged developed to perform FFTs along frequency axis of radio interferometric data and perform all possible baseline cross multiplications, while accounting for units and cosmological conversion factors. In order to provide a credible alternative power spectrum estimate to the hera_pspec power spectrum pipeline, additional tests have been designed to evaluate the outputs of this python package against known and unknown inputs.

For a complete list of a tests involved in the shadow pipeline evaluation and more information on the parameters of each test, refer to the document attached at the end of this memo. I will provide a brief summary here. The shadow pipeline test suite involves five total tests characterized by the following simulation parameters:

0.1 A known input power spectrum with no noise.

1.1 Foregrounds, noise, no EoR and no systematic

1.2 Foregrounds, noise, an unknown EoR and no systematic

1.3 Foregrounds, noise, no EoR and an unknown systematic

1.4 Foregrounds, noise, an unknown EoR and an unknown systematic

This memo details the results from the HERA SimpleDS shadow pipeline test 0.1 "Flat $P(k)$". The data product analyzed consists of simulated HERA data containing one realization of three unique baseline groups. The goal is to estimate the input power spectrum the simulator used which is expected to be a flat in power spectrum space as a function of cosmological wavenumber, k. While not a necessary parameter of this test, we have elected to perform this test with the input power spectrum level unknown.

After initial analysis and results are reported, the input power spectrum value is compared with the results of this memo. Additional analysis is then appended to the original memo and any changes made to previous figures and tables are noted in line.

For completeness, the entire analysis notebook is also included.

[31]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm, SymLogNorm
```

Shadow Pipeline Pspec/stage0.1 validation

```
import os
from itertools import cycle
from IPython.display import display, Latex

from pyuvdata import UVData, UVBeam, utils as uvutils
import pyuvdata
import glob

from scipy import integrate
from scipy.signal import windows
from scipy.interpolate import interp1d, UnivariateSpline as USpline

from itertools import product

import simpleDS as sds
from simpleDS import  DelaySpectrum, utils, cosmo

from astropy import units, constants as const
from astropy.coordinates import Angle
from astropy.visualization import quantity_support
from astropy.cosmology import LambdaCDM, WMAP9, Planck15
```

[2]:
```
quantity_support();
```

[70]:
```
def sci_notation(num, err=None, decimal_digits=1, precision=None, exponent=None):
    """
    Returns a string representation of the scientific
    notation of the given number formatted for use with
    LaTeX or Mathtext, with specified number of significant
    decimal digits and precision (number of decimal digits
    to show). The exponent to be used can also be specified
    explicitly.
    """
    if isinstance(num, units.Quantity):
        unit = num.unit
        num = num.value

    else:
        unit = units.dimensionless_unscaled
    if exponent is None:
        exponent = np.int(np.floor(np.log10(np.abs(num))))
    coeff = np.round(num / np.float(10**exponent), decimal_digits)
    if precision is None:
        precision = decimal_digits
    if err is not None:
        err = err.to_value(unit)
        uncert = np.round(err / np.float(10**exponent), decimal_digits)
```

2

```
        return r"(${0:.{3}f} \pm {1:.{3}f}) \cdot10^{{{2:d}}}$ {4}".
↪format(coeff, uncert, exponent, precision, unit.to_string("latex"))
    else:
        return r"${0:.{2}f}\cdot10^{{{1:d}}}$ {3}".format(coeff, exponent,␣
↪precision, unit.to_string("latex"))
```

[4]:
```
# print version info
for m in [np, sds, pyuvdata]:
    print("{} verison {}".format(m.__name__, m.__version__))
```

```
numpy verison 1.18.4
simpleDS verison 2.0.1.dev14+gc154036
pyuvdata verison 2.0.3.dev149+gf7497fa2
```

[5]:
```
DATA_PATH="/lustre/aoc/projects/hera/alanman/eor_sky_sim/"
BEAM_PATH = "/lustre/aoc/projects/hera/Validation/HERA_beams"
data_files = [os.path.join(DATA_PATH, "eorsky_3.00hours_Nside128_sigma0.
↪03_fwhm12.13_uv.uvh5")]
beam_file = os.path.join(BEAM_PATH, "NF_HERA_dipole_IQ_power_healpix128.fits")
outfile = "/lustre/aoc/projects/hera/mkolopan/shadow_pipeline/flat_pk_sds.hdf5"
spws = [[0, 383]]
```

[6]:
```
uvb = UVBeam()
uvb.read_beamfits(beam_file)
```

[7]:
```
uv = UVData()
uv.read(data_files[:], read_data=False, file_type='uvh5')
uv.read(
    data_files[:],
    file_type='uvh5',
    polarizations=np.intersect1d(
        uvb.polarization_array, uv.polarization_array
    )
)
```

```
Telescope eorsky is not in known_telescopes.
```

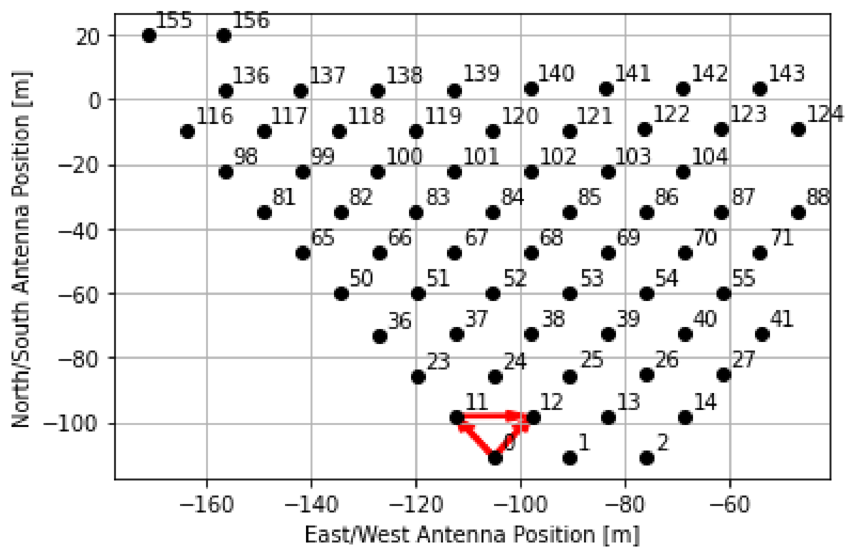
[8]:
```
uv.extra_keywords
```

[8]:
```
{'bm_fwhm': 12.129942517040208,
 'bsq_int': 0.02535452255352503,
 'nside': 128,
 'skysig': 0.031,
 'slurm_id': '1321259'}
```

Shadow Pipeline Pspec/stage0.1 validation

[12]:
```
antpos, ants = uv.get_ENU_antpos(pick_data_ants=False)
x, y, z = antpos.T
fig, ax = plt.subplots(1, facecolor='white')
plt.plot(x,y, 'ko');

# ax1.set_xlim([50, 175])
# ax1.set_ylim([200,300])
for _ant, (_x,_y) in zip(ants, zip(x,y)):
    xlims = ax.get_xlim()
    ylims = ax.get_ylim()
    if np.logical_and(
        _x >= np.min(xlims), _x < np.max(xlims)
    ) and  np.logical_and(
        _y >= np.min(ylims), _y < np.max(ylims)
    ):
        ax.text(_x +1.5, _y + 2.5, _ant)

for bl, bl_ind in zip(*np.unique(uv.baseline_array, return_index=True)):
    a1, a2 = uv.baseline_to_antnums(bl)
    ind1 = np.argwhere(ants == a1)
    ax.arrow(
        x[ind1].squeeze(),
        y[ind1].squeeze(),
        uv.uvw_array[bl_ind, 0],
        uv.uvw_array[bl_ind, 1],
        length_includes_head=True,
        color='red',
        width=1.0625,
    )

ax.grid()
ax.set_xlabel("East/West Antenna Position [m]");
ax.set_ylabel("North/South Antenna Position [m]");
```

Shadow Pipeline Pspec/stage0.1 validation

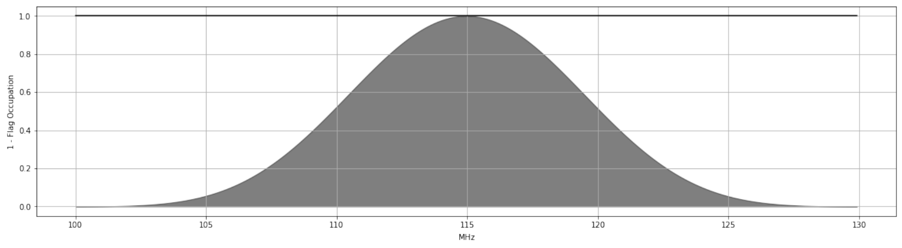


The simulation consists of the three baselines (0, 11), (0, 12), and (11, 12) to represent the three shortest redundant groups of the HERA hex show in the figure and covers the spectral band of 100-130 MHz.

The default spectral taper for a SimpleDS object is a Blackman-Harris shown in the figure below along with the inverse flag occupation (total fraction of unflagged samples). For a simulation this is expected to be identically 1.

```
[13]: fig, ax = plt.subplots(1, figsize=(20,5))
freqs = uv.freq_array[0] * units.Hz
flags = uv.nsample_array.astype(float).squeeze()
ax.plot(
    freqs.to("MHz"),
    flags.sum(0)/flags.sum(0).max(),
    'k-'
);

for chan in spws:
    mid = np.mean(chan)
    _w = windows.blackmanharris(chan[1] - chan[0] + 1)
    ax.fill_between(freqs[chan[0] : chan[1] + 1].to('MHz'), 0, _w,␣
 ↪color='black',alpha=.5 )
ax.set_ylabel("1 - Flag Occupation")
ax.grid()
```

5



Since a SimpleDS object can only perform power spectrum estimation on a single baseline group, three objects are used to estimate the power spectrum of all redundant groups in this simulations.

[14]:
```
bl_cross = [(11, 12), (0, 11), (0, 12)]
times = np.unique(uv.time_array)

uv1 = uv.select(bls=bl_cross[0], times=times[::2], inplace=False)
uv2 = uv.select(bls=bl_cross[0], times=times[1::2], inplace=False)
ds = DelaySpectrum(uv=[uv1, uv2])
ds.add_uvbeam(uvb)
ds.cosmology = Planck15
# ds.set_taper(windows.boxcar)

uv1 = uv.select(bls=bl_cross[1], times=times[::2], inplace=False)
uv2 = uv.select(bls=bl_cross[1], times=times[1::2], inplace=False)
ds2 = DelaySpectrum(uv=[uv1, uv2])
ds2.add_uvbeam(uvb)
ds2.cosmology = Planck15

uv1 = uv.select(bls=bl_cross[2], times=times[::2], inplace=False)
uv2 = uv.select(bls=bl_cross[2], times=times[1::2], inplace=False)
ds3 = DelaySpectrum(uv=[uv1, uv2])
ds3.add_uvbeam(uvb)
ds3.cosmology = Planck15
```

```
lst_array parameter value is array, values are not close

Input LST arrays differ on average by 0.18383526862692248 min. Keeping LST array
stored from the first data set read.

lst_array parameter value is array, values are not close
lst_array parameter value is array, values are not close
```

[15]:
```
ds_list = [ds, ds2, ds3]
for _d in ds_list:
```

6

```
    ds.set_taper(windows.blackmanharris)
```

```
[18]: fft_freq = lambda x: np.fft.fftshift(np.fft.fft(x * ds.taper(x.shape[1]).
       ↪reshape(1,-1), axis=1), axes=1)
      fft_fringe = lambda x: np.fft.fftshift(np.fft.fft(x * ds.taper(x.shape[0]).
       ↪reshape(-1,1), axis=0), axes=0)
      fringe_rates = (np.fft.fftshift(np.fft.fftfreq(ds.Ntimes, d=np.diff(ds.lst_array.
       ↪to('hourangle'))[0].value) )/units.hour).to('mHz')
```

The Data

A quick look at the data in the simulations illustrates a noise-like variation in the amplitude over frequency and time. This is consistent with the expected underlying flat $P(k)$ input.

```
[19]: ncols = ds.Nuv + 1
      nrows = ds.Nbls - np.all(ds.flag_array, axis=(0,1,2,4,5)).sum()
      cmap = plt.cm.viridis
      cmap.set_bad('black')
      vmin = -5
      vmax = 5
      norm = LogNorm(vmin=1e-5, vmax=1e-1)
      for _d in ds_list:
          fig, ax = plt.subplots(
                  figsize=(12, 3 * nrows + 1),
                  nrows=nrows,
                  ncols=ncols,
                  sharex=False,
                  sharey=False,
                  facecolor='white',
                  gridspec_kw={"width_ratios":[15] * ds.Nuv + [1]},
                  squeeze=False,
                  )
          fig.subplots_adjust(hspace=.35, wspace=0.1, top=.9)
          fig.suptitle(
              f"z={_d.redshift.item(0):.2f} "
              f"{uvutils.polnum2str(_d.polarization_array.item(0))} "
              f"{uvutils.baseline_to_antnums(_d.baseline_array[0], _d.
       ↪Nants_telescope)}"
          )
          for day_cnt, day in enumerate(["even", "odd"]):
              for bl_cnt, bl in enumerate(_d.baseline_array):

                  if np.all(ds.flag_array[:, :, :, bl_cnt]):
                      continue
                  sharedy = ax[bl_cnt, 0].get_shared_y_axes()

                  sharedx = ax[bl_cnt, 0].get_shared_x_axes()
```

7

```
            ax[bl_cnt, 0].set_ylabel("LST [hour]")

            ax[bl_cnt, 0].set_xlabel(r"Frequency [MHz]")
            for pol_cnt, pol in enumerate(uvutils.polnum2str(ds.
↪polarization_array)):
                sharedx.join(ax[bl_cnt, 0], ax[bl_cnt, day_cnt])
                sharedy.join(ax[bl_cnt, 0], ax[bl_cnt, day_cnt])

                masked_data = np.ma.masked_array(
                    _d.data_array[0, day_cnt, pol_cnt, bl_cnt].value,
                    mask=_d.flag_array[0, day_cnt, pol_cnt, bl_cnt]
                )

                im = ax[bl_cnt, day_cnt].pcolormesh(
                    _d.freq_array[0].to('MHz'),
                    _d.lst_array.to("hourangle"),
                    np.abs(masked_data),
                    cmap=cmap,
                    norm=norm,
                )

                ax[bl_cnt, day_cnt].set_title(f"{day}")
                _ylim = ax[bl_cnt, day_cnt].get_ylim()
                ax[bl_cnt, day_cnt].set_ylim([np.max(_ylim), np.min(_ylim)])


                tick_labels =  ax[bl_cnt, day_cnt].get_yticks()
                tick_labels = [Angle(t*units.hourangle).to_string() for t in␣
↪tick_labels]
                ax[bl_cnt, day_cnt].set_yticklabels(tick_labels);

                if  0 < day_cnt < ncols - 1:
                    plt.setp(ax[bl_cnt, day_cnt].get_yticklabels(),␣
↪visible=False)
                    ax[bl_cnt, day_cnt].set_ylabel("")


            cbar = fig.colorbar(im, cax=ax[bl_cnt, -1], label="Jy")#
            cbar.ax.set_xlabel(
                cbar.ax.get_ylabel(),
                fontsize=18,
            )
            cbar.ax.set_ylabel(None)
```

Transforming to Fringe-Rate space illustrates the slight evolution of the data over frequency with respect to Fringe-Rate for each baseline orientation. Again this small evolution is consistent

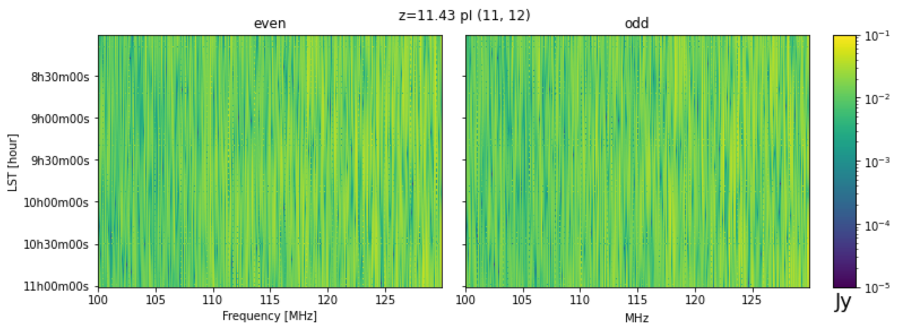
8

with expectations over the 30MHz band. Positive and negative fringe-rates may be a result of conjugation conventions of the data.

[20]:
```
ncols = ds.Nuv + 1
nrows = ds.Nbls - np.all(ds.flag_array, axis=(0,1,2,4,5)).sum()
cmap = plt.cm.viridis
cmap.set_bad('black')
vmin = -5
vmax = 5
norm = LogNorm(vmin=1e-5, vmax=1e-1)
for _d in ds_list:
    fig, ax = plt.subplots(
            figsize=(12, 3 * nrows + 1),
            nrows=nrows,
            ncols=ncols,
            sharex=False,
            sharey=False,
            facecolor='white',
            gridspec_kw={"width_ratios":[15] * ds.Nuv + [1]},
            squeeze=False,
            )
    fig.subplots_adjust(hspace=.35, wspace=0.1, top=.9)
    fig.suptitle(
        f"z={_d.redshift.item(0):.2f} "
        f"{uvutils.polnum2str(_d.polarization_array.item(0))} "
        f"{uvutils.baseline_to_antnums(_d.baseline_array[0], _d.
↪Nants_telescope)}"
    )
    for day_cnt, day in enumerate(["even", "odd"]):
        for bl_cnt, bl in enumerate(_d.baseline_array):

            if np.all(ds.flag_array[:, :, :, bl_cnt]):
                continue
            sharedy = ax[bl_cnt, 0].get_shared_y_axes()

            sharedx = ax[bl_cnt, 0].get_shared_x_axes()

            ax[bl_cnt, 0].set_ylabel("Fringe-Rate [mHz]")

            ax[bl_cnt, 0].set_xlabel(r"Frequency [MHz]")
            for pol_cnt, pol in enumerate(uvutils.polnum2str(ds.
↪polarization_array)):
                sharedx.join(ax[bl_cnt, 0], ax[bl_cnt, day_cnt])
                sharedy.join(ax[bl_cnt, 0], ax[bl_cnt, day_cnt])

                masked_data = np.ma.masked_array(
                    _d.data_array[0, day_cnt, pol_cnt, bl_cnt].value,
                    mask=_d.flag_array[0, day_cnt, pol_cnt, bl_cnt]
```

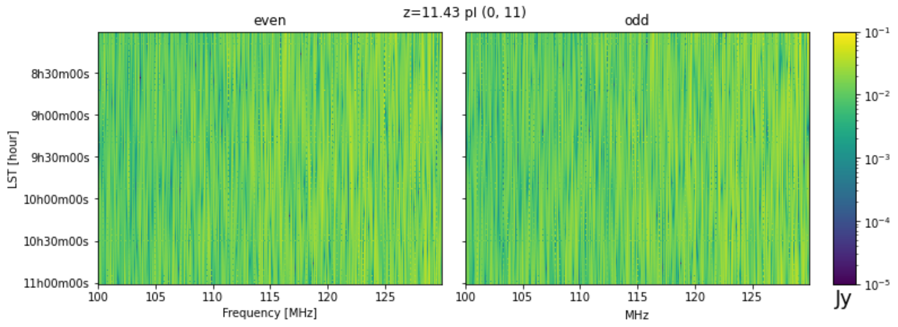
9

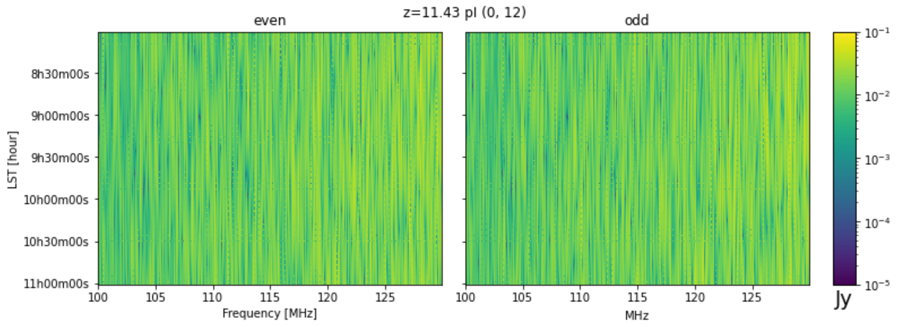
```
                )

                im = ax[bl_cnt, day_cnt].pcolormesh(
                    _d.freq_array[0].to('MHz'),
                    fringe_rates,
                    np.abs(fft_fringe(masked_data)),
                    cmap=cmap,
                    norm=norm,
                )

                ax[bl_cnt, day_cnt].set_title(f"{day}")
                _ylim = ax[bl_cnt, day_cnt].get_ylim()
                ax[bl_cnt, day_cnt].set_ylim([np.max(_ylim), np.min(_ylim)])

                if  0 < day_cnt < ncols - 1:
                    plt.setp(ax[bl_cnt, day_cnt].get_yticklabels(),␣
↪visible=False)
                    ax[bl_cnt, day_cnt].set_ylabel("")
            ax[bl_cnt, 0].set_ylim([1, -1])
            cbar = fig.colorbar(im, cax=ax[bl_cnt, -1], label="Jys")#
            cbar.ax.set_xlabel(
                cbar.ax.get_ylabel(),
                fontsize=12,
            )
            cbar.ax.set_ylabel(None)
```

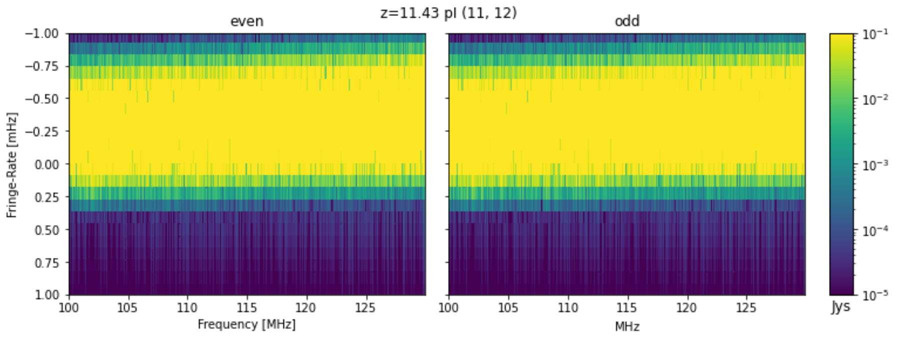


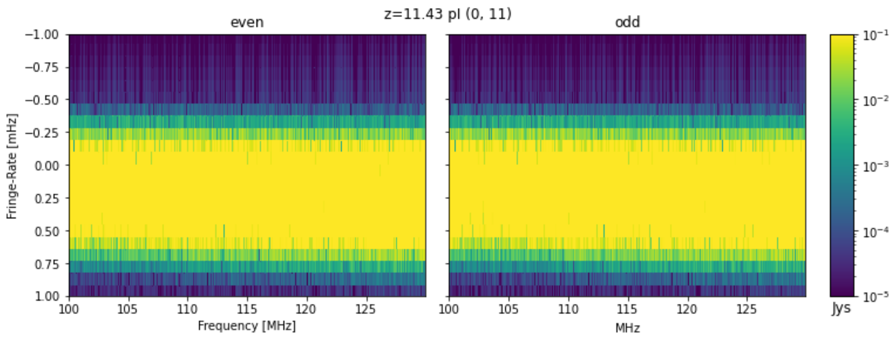
10

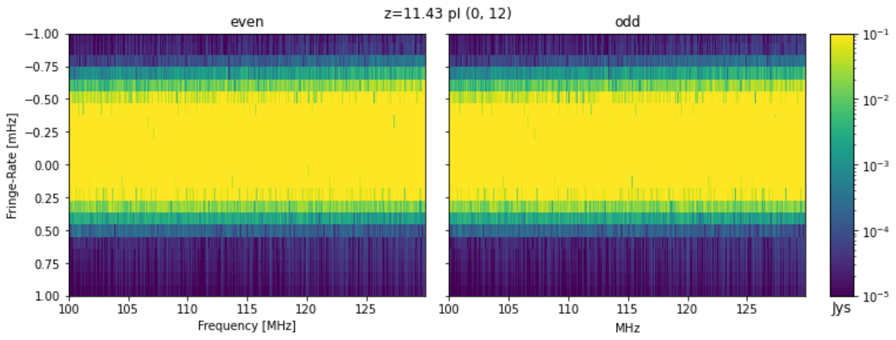
Shadow Pipeline Pspec/stage0.1 validation











```
[21]: for _d in ds_list:
          _d.select_spectral_windows(spws)
          _d.check()
```

```
[22]: for _d in ds_list:
          _d.calculate_delay_spectrum(littleh_units=True)
```

```
[23]: ncols = np.int(np.ceil(np.sqrt(ds.Npols)))
      nrows = np.int(np.ceil(ds.Npols/float(ncols)))
```

The Power Spectrum

In power spectrum space the data again looks noise-like in amplitude, this is consistent with the Fourier transform of a flat spectrum input.

```
[24]: cmap = plt.cm.viridis
      cmap.set_bad('black')
```

Shadow Pipeline Pspec/stage0.1 validation

```
vmin = -5
vmax = 5
norm = LogNorm(vmin=1e0, vmax=1e10)
for _d in ds_list:
    for z_cnt, z in enumerate(_d.redshift):
        ncols = _d.Npols + 1
        nrows = _d.Nbls - np.all(_d.flag_array, axis=(0, 1, 2, 4, 5)).sum()
        fig, ax = plt.subplots(
            figsize=(12, 5 * nrows + 1),
            nrows=nrows,
            ncols=ncols,
            sharex=False,
            sharey=False,
            facecolor='white',
            squeeze=False,
            gridspec_kw={"width_ratios":[10] * _d.Npols + [1]},
            )
        fig.subplots_adjust(hspace=.35, wspace=0.1, top=.95)
    #         fig.suptitle(f"{np.around(np.linalg.norm(_d.uvw.value),1)} {day:
 ↪s}")
        fig_count = 0
        for bl_cnt, bl in enumerate(_d.baseline_array):

            if np.all(ds.flag_array[:, :, :, bl_cnt]):
                continue
            sharedy = ax[fig_count,0].get_shared_y_axes()

            sharedx = ax[fig_count,0].get_shared_x_axes()

            ax[fig_count, 0].set_ylabel("LST [hour]")

            ax[fig_count, 0].set_xlabel(r"$\tau$ [ns]")
            for pol_cnt, pol in enumerate(uvutils.polnum2str(_d.
 ↪polarization_array)):
                sharedy.join(ax[fig_count,0], ax[fig_count, pol_cnt])
                sharedx.join(ax[fig_count,0], ax[fig_count, pol_cnt])
                masked_data = np.ma.masked_array(
                    ds.power_array[z_cnt, pol_cnt, 0, 0].value,
                    mask=ds.flag_array[z_cnt, pol_cnt, 0, 0],
                )
                im = ax[fig_count, pol_cnt].pcolormesh(
                    _d.delay_array.to('ns'),
                    _d.lst_array.to('hourangle'),
                    np.real(masked_data),
                    cmap=cmap,
                    norm=norm,
                )
```

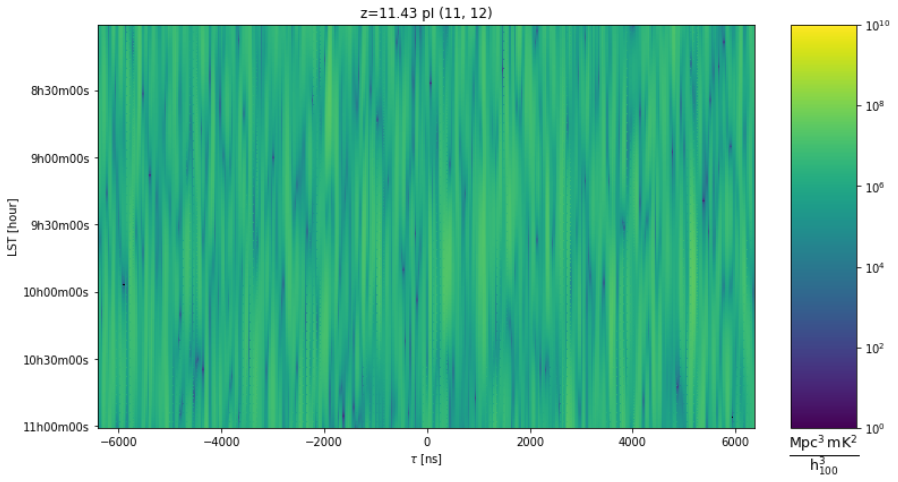
13

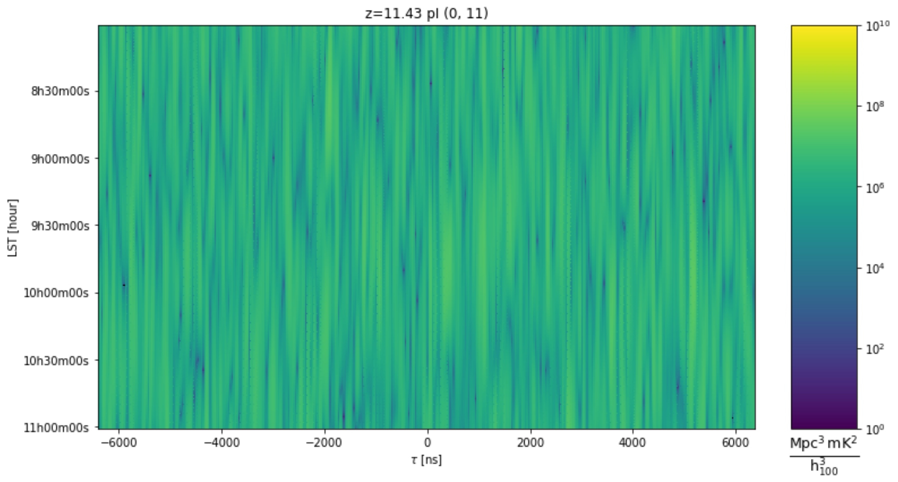
Shadow Pipeline Pspec/stage0.1 validation

```
                ax[fig_count, pol_cnt].set_title(
                    f"z={z:.2f} "
                    f"{uvutils.polnum2str(_d.polarization_array[pol_cnt])} "
                    f"{uvutils.baseline_to_antnums(_d.baseline_array[0], _d.
↪Nants_telescope)}"
                )
                _ylim = ax[fig_count, pol_cnt].get_ylim()
                ax[fig_count, pol_cnt].set_ylim([np.max(_ylim), np.min(_ylim)])

                tick_labels =  ax[fig_count, pol_cnt].get_yticks()
                tick_labels = [Angle(t*units.hourangle).to_string() for t in␣
↪tick_labels]
                ax[fig_count, pol_cnt].set_yticklabels(tick_labels);
                if  0 < pol_cnt < ncols - 1:
                    plt.setp(ax[fig_count, pol_cnt].get_yticklabels(),␣
↪visible=False)
                    ax[fig_count, pol_cnt].set_ylabel("")

            cbar = fig.colorbar(im, cax=ax[fig_count, -1], label=ds.power_array.
↪unit.to_string("latex"))#
            cbar.ax.set_xlabel(
                cbar.ax.get_ylabel(),
                fontsize=18,
            )
            cbar.ax.set_ylabel(None)
            fig_count += 1
```

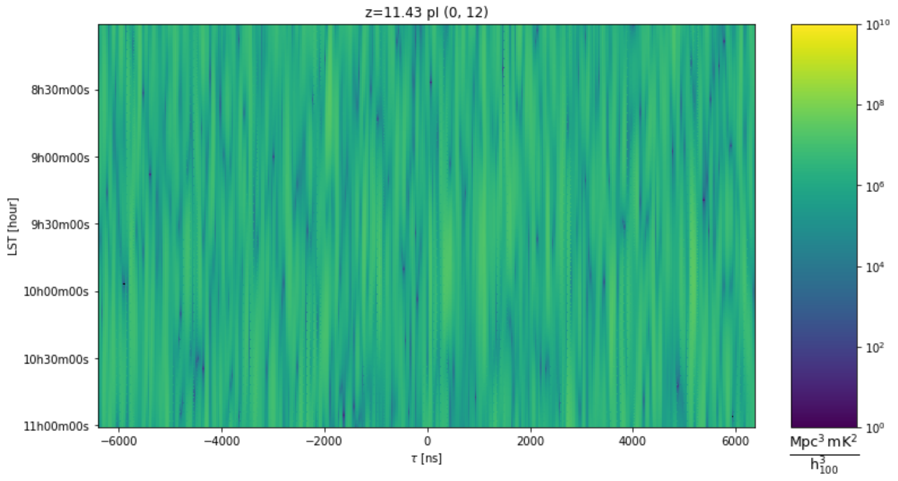
[25]:
```
ncols = np.int(np.ceil(np.sqrt(ds.Npols)))
nrows = np.int(np.ceil(ds.Npols/float(ncols)))
```

Shadow Pipeline Pspec/stage0.1 validation





Shadow Pipeline Pspec/stage0.1 validation



Estimating Input $P(k)$

To properly estimate the variance of the power spectrum as a function in time, it is necessary to estimate the number of unique LST bins based on the FWHM of the input beam. To perform this estimation I use the following relations

$$\theta = \frac{\lambda}{D}$$

The beam crossing time in LSTs is then calculated as

$$LST_{\Omega} = \theta \frac{12 \text{ hours}}{\pi \; cos(l)}$$

where l is the latitude of the telescope location. Finally the number of unique LST bins is

$$N_{LSTS} = \frac{T_{obs}}{LST_{\Omega}}$$

where T_{obs} is the total length of the input simulation. **Revision 1: The form of this equation has changed slightly to more clearly indicate that the numerator should reflect the entire observation time of the input data.**

The uncertainty of the power spectrum is then estimated as the standard deviation over the LST axes, divided by $\sqrt{N_{LST}}$, or the standard deviation of the mean accounting for the total number of unique elements along this axis.

When fitting for $\hat{\mu}$ (input power spectrum level), the effective degrees of freedom must account for the number of unique delay bins after performing a spectral taper. Since the Blackman-Harris taper has a noise equivalent bandwidth (NEBW) of 2, the number of unique bins is then $\frac{N_{delays}}{2}$ where N_{delays} is the total number of delay bins. The total number of degrees of freedom of the fit is then $dof = \frac{N_{delays}}{2} - 1$.

16

Shadow Pipeline Pspec/stage0.1 validation

All three baseline orientations fit an input power spectrum are not consistent with a single input $P(k)$ but average to $\hat{\mu} = 1.55 \times 10^6 \frac{\mathrm{Mpc}^3\,\mathrm{mK}^2}{\mathrm{h}^3_{100}}$. The exact fits, uncertainties and $\frac{\chi^2}{dof}$ are outlined in the table below.

	Unweighted Average			Inverse Variance Average		
Ant Pair	$\hat{\mu}_u$	$\hat{\sigma}_u$	$\frac{\chi^2}{dof}$	$\hat{\mu}_{iv}$	$\hat{\sigma}_{iv}$	$\frac{\chi^2}{dof}$
(11, 12)	2.70	0.13	2.706	1.56	0.06	1.077
∣(0, 11)	2.69	0.11	2.616	1.64	0.06	1.193
∣(0 ,12)	2.68	0.12	3.592	1.46	0.06	1.462

Table 1: The values of the $\hat{\mu}$ and $\hat{\sigma}$ are presented in the units $10^6 \frac{\mathrm{Mpc}^3\,\mathrm{mK}^2}{\mathrm{h}^3_{100}}$. **Revision 1: This table has been altered to also include the unweighted mean and uncertainty to compare against values computed during revision 1.**

[55]:

```

```

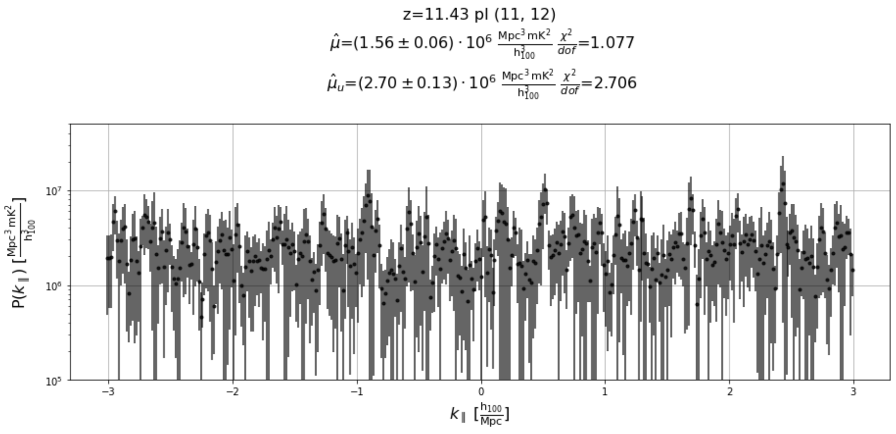
[77]:

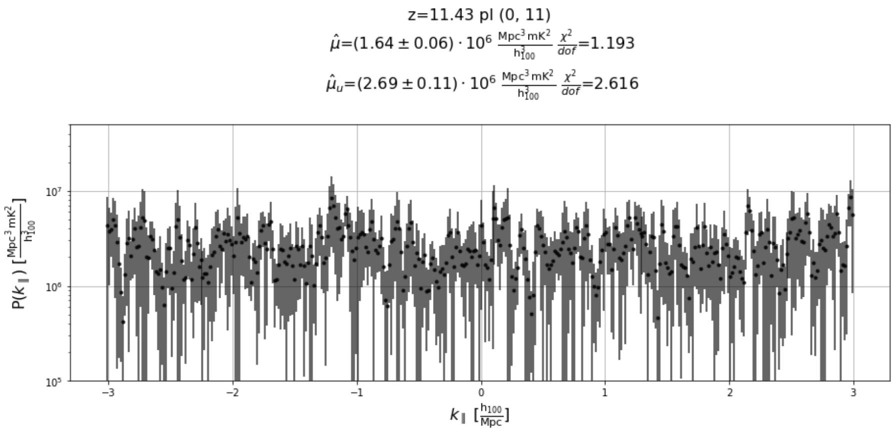
```
for _d in ds_list:
    # Trying to estimate the number of independent time samples to properly␣
↪account for the number of samples when taking the variance.
    beam_crossing = ((const.c / _d.freq_array.mean()) / (14.4*units.m) * 12 *␣
↪units.h / np.pi / np.cos(uv.telescope_location_lat_lon_alt[0])).to(units.min)

    for z_cnt, redshift in enumerate(ds.redshift):
        fig, ax = plt.subplots(
            ncols=ncols,
            nrows=nrows,
            figsize=(15,5),
            facecolor='white',
            sharex=True,
            sharey=True,
            squeeze=False
        )
        ax = ax.ravel()
        for pol_ind in range(_d.polarization_array.size):
            vals = _d.power_array[z_cnt,pol_ind].mean((0,1,2)).real
            errs = 2 * _d.power_array[z_cnt,pol_ind].mean((0,1)).real.std(0) /␣
↪np.sqrt((_d.Ntimes * _d.integration_time.item(0) / beam_crossing).si)

            _mean =  np.average(vals, weights=1. / (errs) ** 2)
            _err = np.sqrt(1./np.sum(1. / (errs)**2))
            _chi2 = np.sum((_mean - vals) ** 2 / ((errs) ** 2))


            dof = (ds.Ndelays * utils.noise_equivalent_bandwidth(ds.taper(ds.
↪Nfreqs)) - 1)
```

Shadow Pipeline Pspec/stage0.1 validation

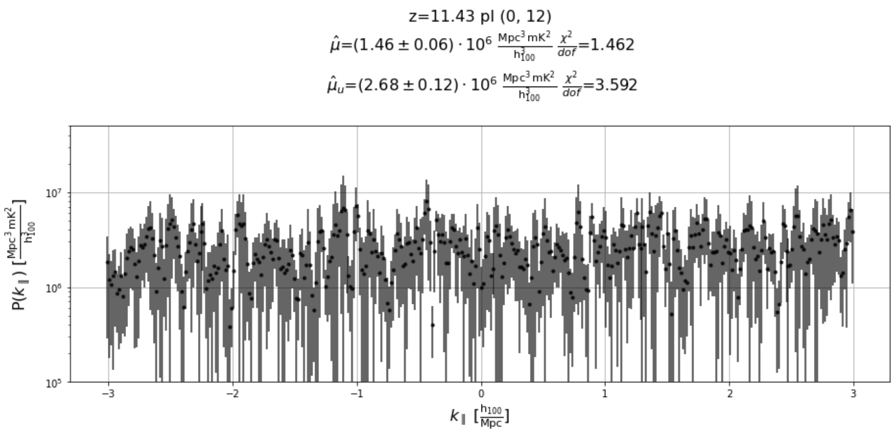
```
            ax[pol_ind].errorbar(
                _d.k_parallel[0],
                vals,
                errs,
                fmt='k.', alpha=.8,
                label="Data" if pol_ind==0 else ''
            );

            ax[pol_ind].grid()
            ax[pol_ind].set_ylabel(r'P($k_{{\parallel}}$) [{0}]'.format((_d.
↪power_array.unit).to_string('latex')), fontsize=16)
            ax[pol_ind].set_xlabel(r'$k_{{\parallel}}$ [{0}]'.format(_d.
↪k_parallel.unit.to_string('latex')), fontsize=16)
            ax[pol_ind].set_title(
                f"z={_d.redshift[z_cnt]:.2f} "
                f"{uvutils.polnum2str(_d.polarization_array[pol_ind])} "
                f"{uvutils.baseline_to_antnums(_d.baseline_array[0], _d.
↪Nants_telescope)}\n"
                r" $\hat{\mu}$="
                f"{sci_notation(_mean, _err, decimal_digits=2)} "
                r"$\frac{\chi^{2}}{dof}$="
                f"{_chi2/dof:.3f}\n"
                , fontsize=16)

        sharex = ax[0].get_shared_x_axes()
        for pol_cnt in range(_d.polarization_array.size):
            sharex.join(ax[0], ax[pol_cnt])

        sharey = ax[0].get_shared_y_axes()
        for pol_cnt in range(ds.polarization_array.size):
            sharey.join(ax[0], ax[pol_cnt])
        ax[0].set_yscale('log')
        ax[0].set_ylim(1e5, 5e7)
        fig.subplots_adjust(wspace=.05, hspace=.175, top=.85)
        fig.suptitle(
            f" "

        );
#          ax[0].axhline(P0mk, color='red', linestyle='dashed')
#          fig.legend(frameon=False, ncol=3, numpoints=1,bbox_to_anchor=[.5, .
↪06]);
```

Shadow Pipeline Pspec/stage0.1 validation





Shadow Pipeline Pspec/stage0.1 validation



Revision 1: These figures have been altered to also include the mean and uncertainty obtained by performing an unweighted average.

Hidden Value Conclusions

This analysis produces estimates for the input flat power spectrum most consistent with $\hat{\mu} = 1.55 \times 10^6 \frac{\mathrm{Mpc^3\,mK^2}}{\mathrm{h^3_{100}}}$ for all three baseline groups.

- **Beginning of Revision 1: June 5, 2020**

Comparison with Input

After the initial results are reach without knowledge of the input $P(k)$, the parameters in the input UVData objects are read to compute the expected value of $P(k)$.

[59]:
```
# get the parameters needed to compute the expected amplitude of the power␣
 ↪spectrum
nside = uv.extra_keywords[u'nside']
dOmega = 4 * np.pi/(12 * nside ** 2)  * units.sr                # sr
skysig = uv.extra_keywords[u'skysig'] * 1e3  * units.mK
df = np.diff(np.unique(uv.freq_array)).mean() * units.Hz # Hz
z = cosmo.calc_z(uv.freq_array.mean() * units.Hz)
X2Y = cosmo.X2Y(z) # (h^-1 Mpc)^3 sr^-1 Hz^-1

# calculate expected power spectrum amplitude
P0 = skysig ** 2 * df.si * dOmega * X2Y
#
P0mk=P0.to(ds.power_array.unit, units.with_H0(_d.cosmology.H0))
```

Shadow Pipeline Pspec/stage0.1 validation

[72]:
```
display(Latex(" The initial test results were inconsistent with the expected␣
↪$P(k)$= " +sci_notation(P0.to("mK^2Mpc^3/littleh^3", units.with_H0(cosmo.
↪Planck15.H0)), decimal_digits=2)))
```

The initial test results are inconsistent with the expected $P(k)$= $2.89 \cdot 10^6\ \frac{\mathrm{Mpc}^3\,\mathrm{mK}^2}{\mathrm{h}_{100}^3}$

Analysis Differences

After reviewing validation test 0.1 notebook the following areas were identified to be potential causes of discrepancies

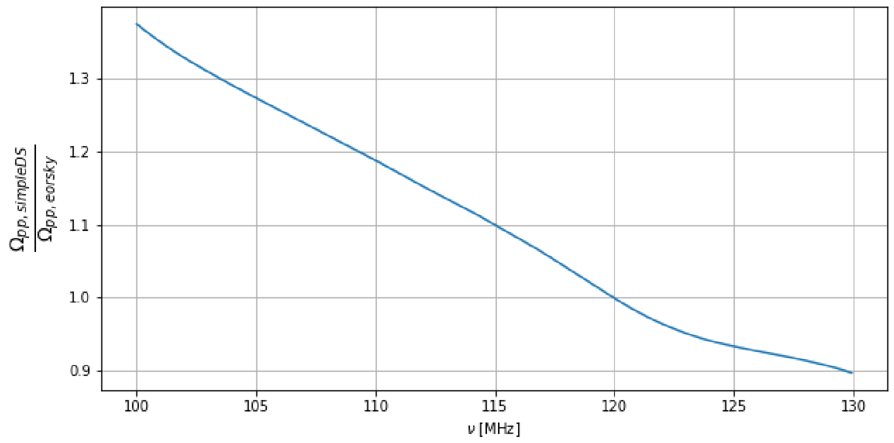
- Inverse Variance vs Unweighted averaging
- Power spectrum normalization via the beam square integral

This memo now reports values for fits using both weighting schemes to determine whether or not the choice of weighting scheme affects the fit value,

Power spectrum normalization and unit conversion factors can contribute to significant discrepancies between expected inputs and fit values.

To evaluate whether difference can cause discrepancies in the final fit, a ratio of the beam square area computed by simpleDS from the HERA beam input file ($\Omega_{pp,simpleDS}$), and the reported beam square integral in the provided data file ($\Omega_{pp,eorsky}$) is plotted below. Since all three baseline types are being transformed over the same contiguous band, only one beam squared integral is plotted in the figure.

[16]:
```
fig, ax = plt.subplots(1, figsize=(10,5))
ax.set_ylabel(r"$\frac{\Omega_{pp,simpleDS}}{\Omega_{pp,eorsky}}$", fontsize=20)
ax.set_xlabel(r"$\nu$ [MHz]")
ax.plot(ds.freq_array[0].to("MHz"), ds.beam_sq_area[0,0].value/uv.
↪extra_keywords["bsq_int"])
ax.grid()
```



21

This difference in the beam squared integral Ω_{pp} can result in normalization difference in final power spectrum.

The ratio of the $\int \frac{d\nu}{\Omega_{pp}(\nu)}$ for each beam square integral will provide an overall normalization factor, B_Ω, which may explain small difference in the fit values of the input power spectrum.

$$B_\Omega = \frac{\int \frac{d\nu}{\Omega_{pp,simpleDS}}}{\int \frac{d\nu}{\Omega_{pp,eorsky}}}$$

```
[17]: beam_sq_int_sds = integrate.trapz(1./ds.beam_sq_area[0,0], x=ds.freq_array[0])
      beam_sq_int_input = integrate.trapz(np.ones_like(ds.beam_sq_area[0,0].value)/(uv.
       ↪extra_keywords["bsq_int"] * units.sr), x=ds.freq_array[0])
      b_omega = beam_sq_int_sds/beam_sq_int_input
      print(b_omega, f"Percent difference: {100*(1-b_omega):.3f}")
```

```
0.9216614501215799 Percent difference: 7.834
```

Indicating up to $\sim$ 8% difference may be expected between the input $P(k)$ and the fit power spectrum values.

This normalization factor B_Ω is not a value normally applied to data. It is a necessary factor to convert between the integrated spectrally varying beam square area of the beam provided for the shadow pipeline analysis and the spectrally constant beam reported by the simulation file.

Refined Results

The fitting is repeated but this time accounting for the B_Ω normalization discrepancy, this is equivalent to multiplying each $\hat{\mu}$ and the uncertainties $\hat{\sigma}$ by the scale factor $\frac{1}{B_\Omega}$.

Two fits are also now performed, an average ($\hat{\mu}_u$) to be consistent with previous test 0.1 notebook averaging scheme and the inverse variance average ($\hat{\mu}_{iv}$) to compare how results change with the analysis previously computed in this memo.

All three baseline orientations fit an input power spectrum consistent with an unweighted fit consistent with $\hat{\mu}_u = 2.92 \times 10^6 \frac{\mathrm{Mpc}^3\,\mathrm{mK}^2}{\mathrm{h}^3_{100}}$ and when fit with inverse variance weighting are not all consistent with a single input $P(k)$ but will average to $\hat{\mu}_{iv} = 1.69 \times 10^6 \frac{\mathrm{Mpc}^3\,\mathrm{mK}^2}{\mathrm{h}^3_{100}}$. The exact fits, uncertainties and $\frac{\chi^2}{dof}$ are outlined in the table below.

	Unweighted Average			Inverse Variance Average		
Ant Pair	$\hat{\mu}_u$	$\hat{\sigma}_u$	$\frac{\chi^2}{dof}$	$\hat{\mu}_{iv}$	$\hat{\sigma}_{iv}$	$\frac{\chi^2}{dof}$
(11, 12)	2.92	0.14	2.706	1.70	0.07	1.077
(0, 11)	2.92	0.12	2.616	1.78	0.07	1.193
(0 ,12)	2.91	0.13	3.592	1.58	0.07	1.462

Table 2: The values of the $\hat{\mu}$ and $\hat{\sigma}$ are presented in the units $10^6 \frac{\mathrm{Mpc}^3\,\mathrm{mK}^2}{\mathrm{h}^3_{100}}$.

22

Shadow Pipeline Pspec/stage0.1 validation

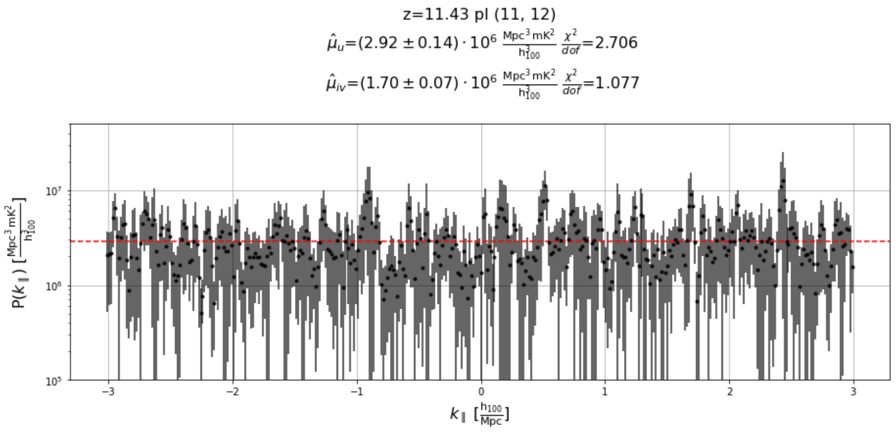
```
[76]: for _d in ds_list:
          # Trying to estimate the number of independent time samples to properly␣
      ↪account for the number of samples when taking the variance.
          beam_crossing = ((const.c / _d.freq_array.mean()) / (14.4*units.m) * 12 *␣
      ↪units.h / np.pi / np.cos(uv.telescope_location_lat_lon_alt[0])).to(units.min)

          for z_cnt, redshift in enumerate(ds.redshift):
              fig, ax = plt.subplots(
                  ncols=ncols,
                  nrows=nrows,
                  figsize=(15,5),
                  facecolor='white',
                  sharex=True,
                  sharey=True,
                  squeeze=False
              )
              ax = ax.ravel()
              for pol_ind in range(_d.polarization_array.size):
                  vals = _d.power_array[z_cnt,pol_ind].mean((0,1,2)).real/ b_omega
                  errs = 2 * _d.power_array[z_cnt,pol_ind].mean((0,1)).real.std(0) /␣
      ↪np.sqrt((_d.Ntimes * _d.integration_time.item(0) / beam_crossing).si) / b_omega

                  u_mean = np.average(vals)
                  u_err = np.sqrt(np.sum(errs ** 2)/np.abs(errs.size) **2 )
                  u_chi2 = np.sum((u_mean - vals) ** 2 / ((errs) ** 2))

                  iv_mean =  np.average(vals, weights=1. / (errs) ** 2)
                  iv_err = np.sqrt(1./np.sum(1. / (errs)**2))
                  iv_chi2 = np.sum((iv_mean - vals) ** 2 / ((errs) ** 2))


                  dof = (ds.Ndelays * utils.noise_equivalent_bandwidth(ds.taper(ds.
      ↪Nfreqs)) - 1)


                  ax[pol_ind].errorbar(
                      _d.k_parallel[0],
                      vals,
                      errs,
                      fmt='k.', alpha=.8,
                      label="Data" if pol_ind==0 else ''
                  );

                  ax[pol_ind].grid()
                  ax[pol_ind].set_ylabel(r'P($k_{{\parallel}}$) [{0}]'.format((_d.
      ↪power_array.unit).to_string('latex')), fontsize=16)
```

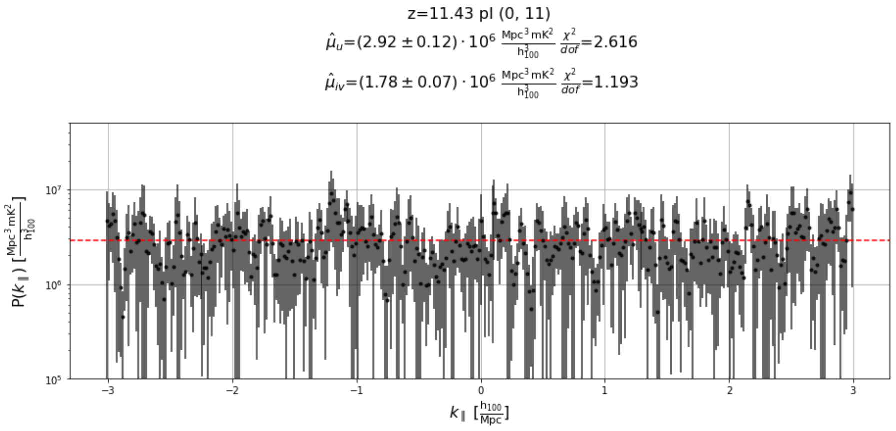
23

```
            ax[pol_ind].set_xlabel(r'$k_{{\parallel}}$ [{0}]'.format(_d.
↪k_parallel.unit.to_string('latex')), fontsize=16)
            ax[pol_ind].set_title(
                f"z={_d.redshift[z_cnt]:.2f} "
                f"{uvutils.polnum2str(_d.polarization_array[pol_ind])} "
                f"{uvutils.baseline_to_antnums(_d.baseline_array[0], _d.
↪Nants_telescope)}\n"
                r" $\hat{\mu}_{u}$="
                f"{sci_notation(u_mean, u_err, decimal_digits=2)} "
                r"$\frac{\chi^{2}}{dof}$="
                f"{u_chi2/dof:.3f}\n"
                r" $\hat{\mu}_{iv}$="
                f"{sci_notation(iv_mean, iv_err, decimal_digits=2)} "
                r"$\frac{\chi^{2}}{dof}$="
                f"{iv_chi2/dof:.3f}\n"
                , fontsize=16)

        sharex = ax[0].get_shared_x_axes()
        for pol_cnt in range(_d.polarization_array.size):
            sharex.join(ax[0], ax[pol_cnt])

        sharey = ax[0].get_shared_y_axes()
        for pol_cnt in range(ds.polarization_array.size):
            sharey.join(ax[0], ax[pol_cnt])
        ax[0].set_yscale('log')
        ax[0].set_ylim(1e5, 5e7)
        fig.subplots_adjust(wspace=.05, hspace=.175, top=.85)
        fig.suptitle(
            f" "

        );
        ax[0].axhline(P0mk, color='red', linestyle='dashed')
#          fig.legend(frameon=False, ncol=3, numpoints=1,bbox_to_anchor=[.5, .
↪06]);
```

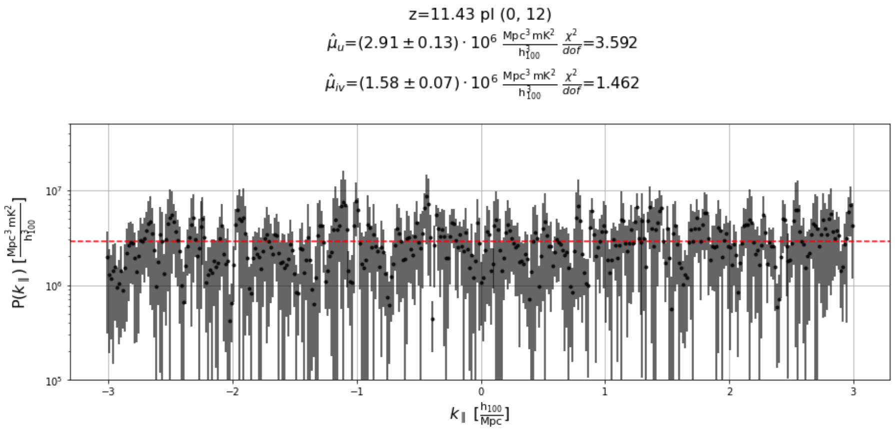
24

Shadow Pipeline Pspec/stage0.1 validation





Shadow Pipeline Pspec/stage0.1 validation



Conclusion

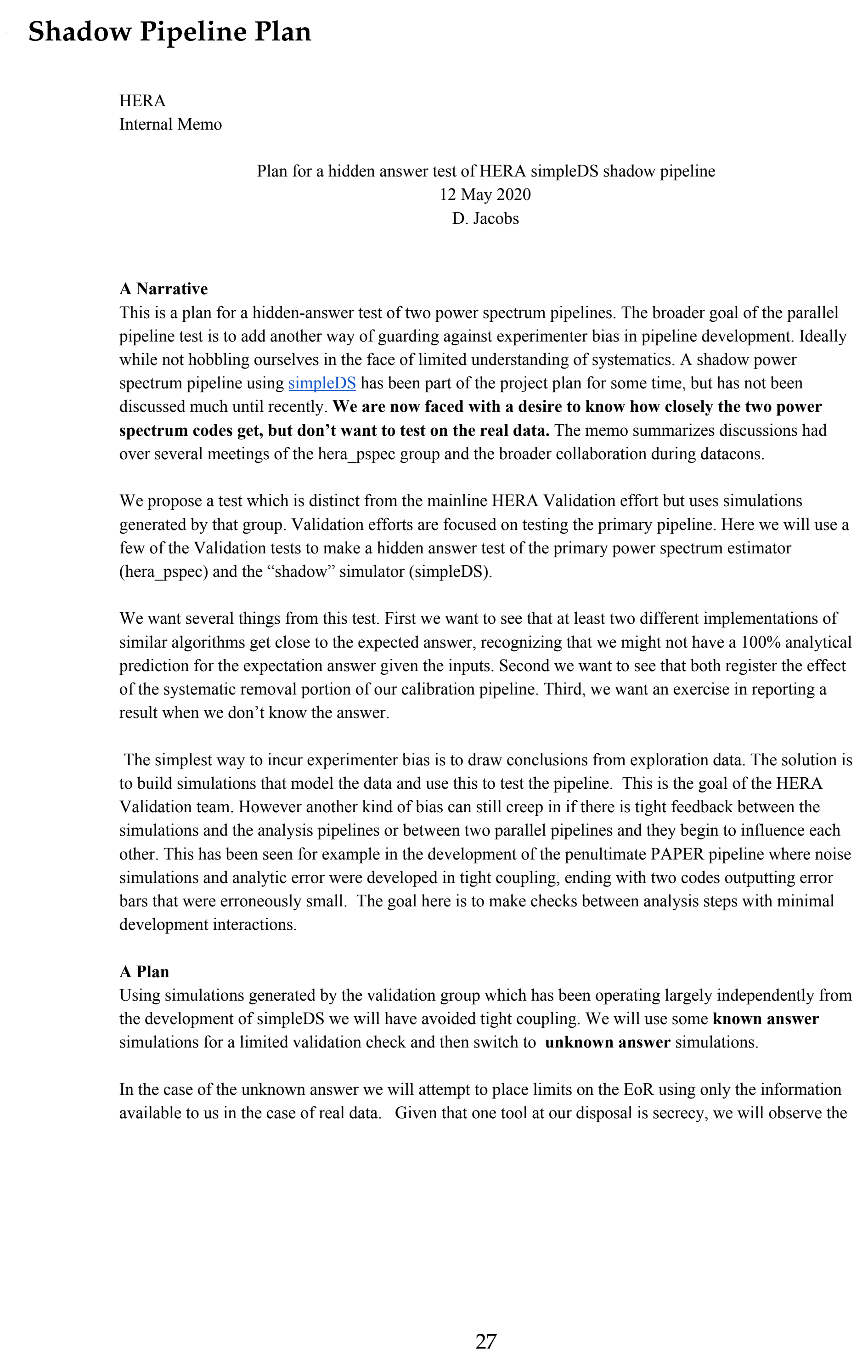
Comparing the fits before and after applying the normalization constant indicates the averaging scheme is the largest contributing factor to the discrepancy between expectation and fitted value.

When accounting for differences in averaging schemes and power spectrum normalization, the fit $\hat{\mu}$ is consistent with the expected value of $2.89 \cdot 10^6 \frac{\mathrm{Mpc^3\,mK^2}}{\mathrm{h_{100}^3}}$ however these fits are accompanied by a $\frac{\chi^2}{dof} \sim 3$ may indicate this is not the optimal fit for the computed power spectra.

Outstanding Questions

When this analysis is repeated with a boxcar spectral taper, numerical results are consistent with fits obtained above. However, the computed $\frac{\chi^2}{dof}$ indicates that the unweighted average is a better fit to the data than the inverse variance weighted average. This poses the question of whether the number of degrees of freedom for the fit are being computed correctly.

Shadow Pipeline Pspec/stage0.1 validation



Shadow Pipeline Plan

HERA
Internal Memo

Plan for a hidden answer test of HERA simpleDS shadow pipeline
12 May 2020
D. Jacobs

A Narrative
This is a plan for a hidden-answer test of two power spectrum pipelines. The broader goal of the parallel pipeline test is to add another way of guarding against experimenter bias in pipeline development. Ideally while not hobbling ourselves in the face of limited understanding of systematics. A shadow power spectrum pipeline using simpleDS has been part of the project plan for some time, but has not been discussed much until recently. **We are now faced with a desire to know how closely the two power spectrum codes get, but don't want to test on the real data.** The memo summarizes discussions had over several meetings of the hera_pspec group and the broader collaboration during datacons.

We propose a test which is distinct from the mainline HERA Validation effort but uses simulations generated by that group. Validation efforts are focused on testing the primary pipeline. Here we will use a few of the Validation tests to make a hidden answer test of the primary power spectrum estimator (hera_pspec) and the "shadow" simulator (simpleDS).

We want several things from this test. First we want to see that at least two different implementations of similar algorithms get close to the expected answer, recognizing that we might not have a 100% analytical prediction for the expectation answer given the inputs. Second we want to see that both register the effect of the systematic removal portion of our calibration pipeline. Third, we want an exercise in reporting a result when we don't know the answer.

The simplest way to incur experimenter bias is to draw conclusions from exploration data. The solution is to build simulations that model the data and use this to test the pipeline. This is the goal of the HERA Validation team. However another kind of bias can still creep in if there is tight feedback between the simulations and the analysis pipelines or between two parallel pipelines and they begin to influence each other. This has been seen for example in the development of the penultimate PAPER pipeline where noise simulations and analytic error were developed in tight coupling, ending with two codes outputting error bars that were erroneously small. The goal here is to make checks between analysis steps with minimal development interactions.

A Plan
Using simulations generated by the validation group which has been operating largely independently from the development of simpleDS we will have avoided tight coupling. We will use some **known answer** simulations for a limited validation check and then switch to **unknown answer** simulations.

In the case of the unknown answer we will attempt to place limits on the EoR using only the information available to us in the case of real data. Given that one tool at our disposal is secrecy, we will observe the

27

Shadow Pipeline Pspec/stage0.1 validation

Shadow Pipeline Plan

HERA
Internal Memo

principle that conclusions about any hidden answer will provide an interesting data point at small extra cost in resources. In other words, **lets keep as much as possible secret!**

Known Answer Test

We will use the **Noise + flat Pk** validation test

1. Thermal noise is reference simulation 0.0 <insert NRAO path here>
2. flat gaussian Pk is ref sim 0.1 (no noise) /lustre/aoc/projects/hera/alanman/eor_sky_sim/

Question: are we returning an answer consistent with analytic predictions?

Unknown Answer

The hidden value test should be run sparingly. This will minimize feedback loops. This pushes us to think each run out carefully, taking advantage of possible variations. At the present time it is thought that the systematic removal step in the pipeline will present the most uncertainty in a final analysis. Also the receiver temperature is usually poorly constrained and estimated as part of a power spectrum analysis. This motivates the following variations which are listed here and summarized in Table 1.

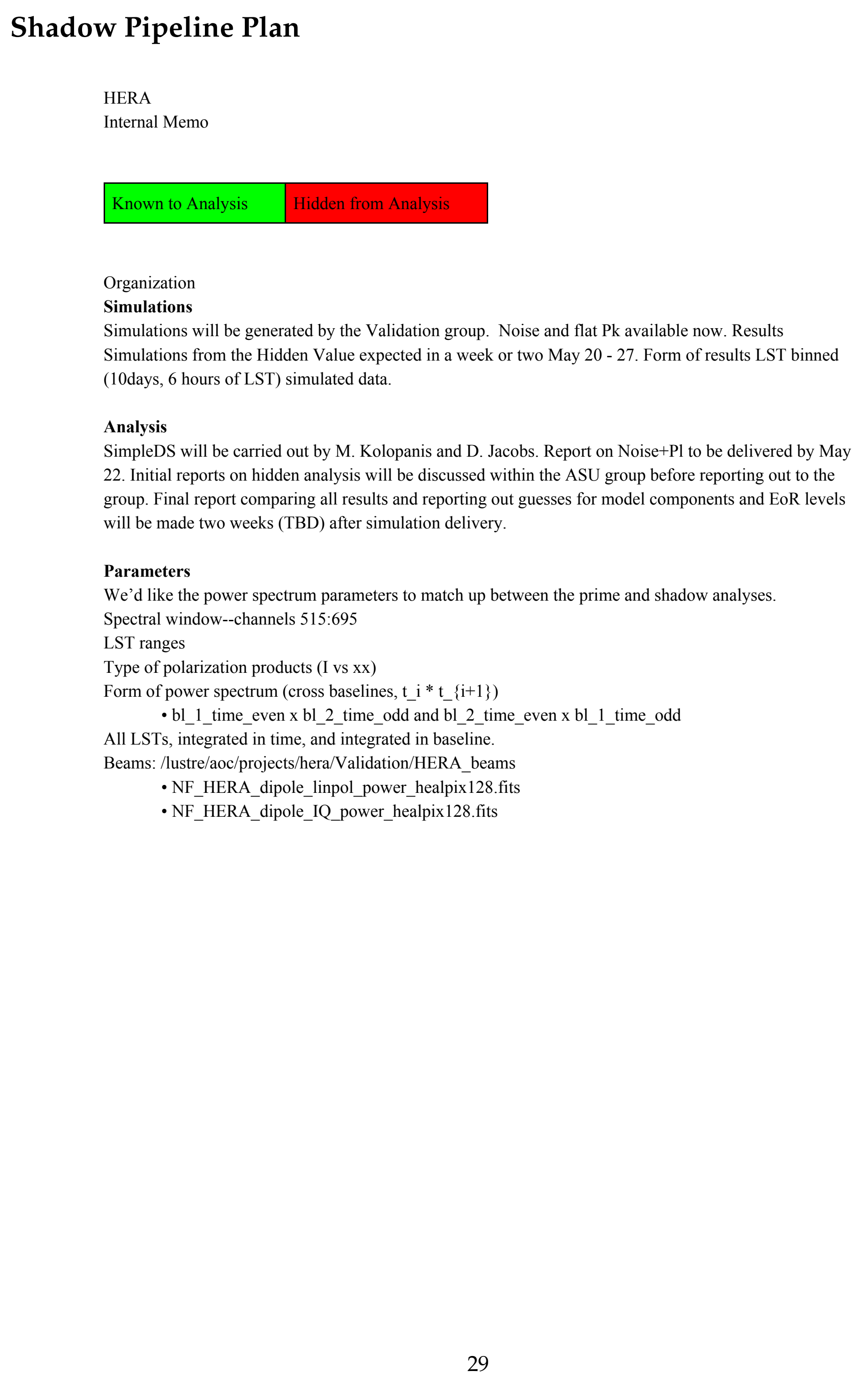
Hidden Variation 1:
Whether the simulated high delay systematic has been filtered.
Hidden Variation 2:
EoR power spectrum level and shape. In half the simulations EoR will be excluded, in the other half it will be included at an unknown level and spectrum shape.
Hidden Parameter 1:
All simulations will include noise. We will constrain receiver temperature variation to being within 50% of a flat 100K+-50K. Trcvr will not vary between simulations.

Filenames should be cleansed of all hidden values. LST binned data products will be integrated into a single dataset (not split into even/odd or whatever)

Table 1: First HERA Hidden Value test. Values in Red are known to the simulator but hidden from the analyst.

Sky	**Noise**	**EoR**	**Systematic filtering**
GSM+Gleam + gain errors + calibration	Trcvr = 100K+-50K	Some level	No
GSM+Gleam + calibration + gain errors + calibration	Trcvr = 100K+-50K	None	No
GSM+Gleam + calibration + gain errors + calibration	Trcvr = 100K+-50K	Some level	Yes
GSM+Gleam + calibration + gain errors + calibration	Trcvr = 100K+-50K	None	Yes

Shadow Pipeline Pspec/stage0.1 validation



Shadow Pipeline Plan

HERA
Internal Memo

🟩 Known to Analysis	🟥 Hidden from Analysis

Organization
Simulations
Simulations will be generated by the Validation group. Noise and flat Pk available now. Results Simulations from the Hidden Value expected in a week or two May 20 - 27. Form of results LST binned (10days, 6 hours of LST) simulated data.

Analysis
SimpleDS will be carried out by M. Kolopanis and D. Jacobs. Report on Noise+Pl to be delivered by May 22. Initial reports on hidden analysis will be discussed within the ASU group before reporting out to the group. Final report comparing all results and reporting out guesses for model components and EoR levels will be made two weeks (TBD) after simulation delivery.

Parameters
We'd like the power spectrum parameters to match up between the prime and shadow analyses.
Spectral window--channels 515:695
LST ranges
Type of polarization products (I vs xx)
Form of power spectrum (cross baselines, t_i * t_{i+1})
- bl_1_time_even x bl_2_time_odd and bl_2_time_even x bl_1_time_odd

All LSTs, integrated in time, and integrated in baseline.
Beams: /lustre/aoc/projects/hera/Validation/HERA_beams
- NF_HERA_dipole_linpol_power_healpix128.fits
- NF_HERA_dipole_IQ_power_healpix128.fits

29

A Analysis Notebooks

All notebooks used in this analysis can be found at https://github.com/mkolopanis/HERA_shadow_pipeline_notebooks