Noise Subtraction in the gstlal Calibration Pipeline

Aaron Viets

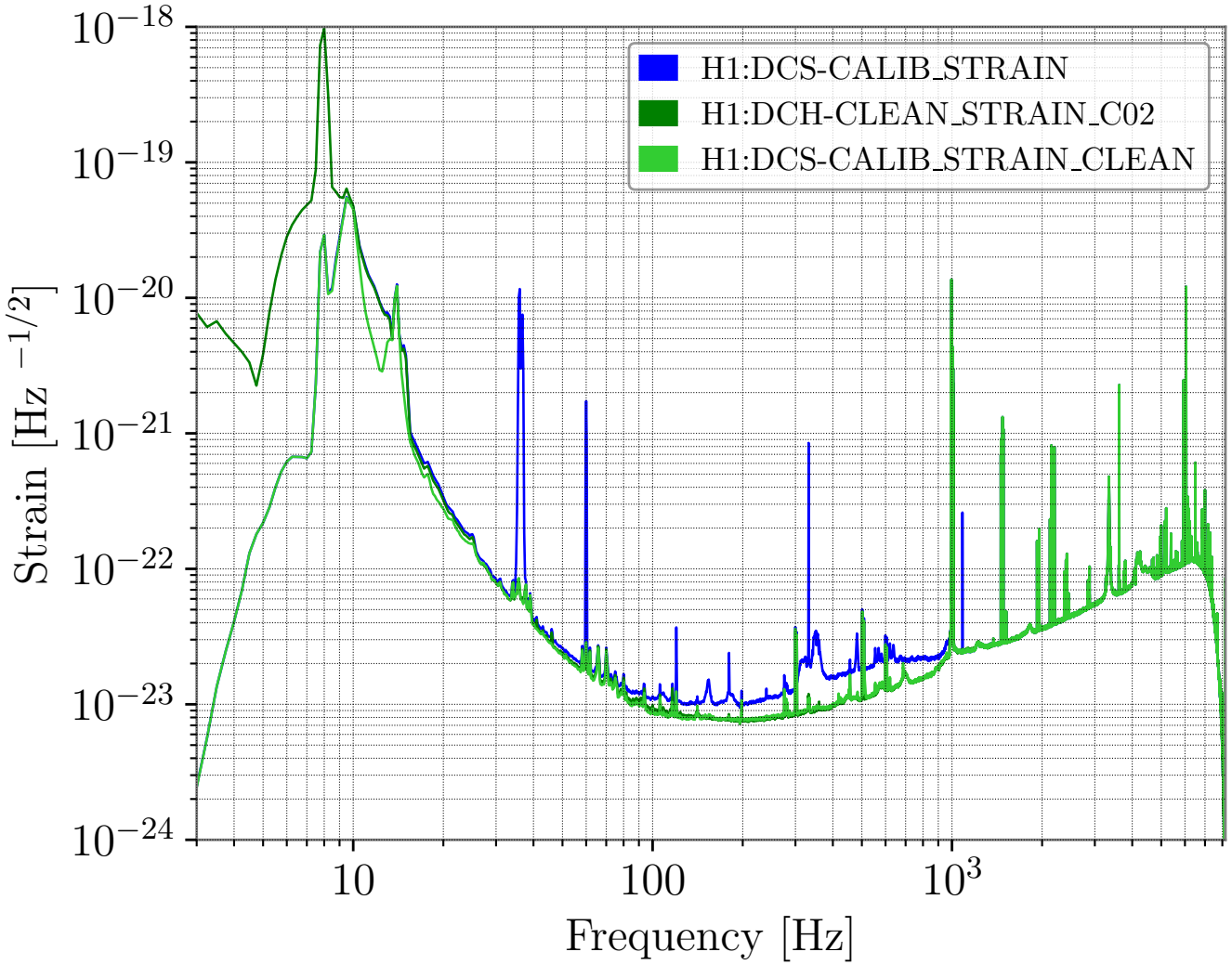
LIGO DCC Document G1801495-v9

Outline

I. Calibration line subtraction
   A. Methods
   B. Results

II. Power mains line subtraction
   A. Methods
   B. Results

III. Broadband noise subtraction
   A. Methods
   B. Results



Calibration Line Subtraction

- Option 1: Use the calibration model to compute lines in $h(t)$ using injection channels.
  - Provides a quick check of calibration accuracy relative to Pcal at those frequencies.
  - Confirms that the calibration (the kappas) does not change on very short timescales.
  - Negligible impact on latency
- Option 2: Compute transfer functions between $h(t)$ and appropriate witness channels.
  - Does not rely on calibration accuracy, and can therefore be done effectively even when calibration models are not reliable.
  - Negligible impact on latency

Calibration Line Frequencies

Line	Purpose	L1 Frequency (Hz)	H1 Frequency (Hz)	Subtracted?
$f_4^{(pc)}$	SRC detuning	N/A	7.93	No
f_{U}	κ_{U}	15.1	15.6	Yes
f_{P}	κ_{P}	15.7	16.4	Yes
$f_1^{(pc)}$	κ_{T}, κ_{P}, κ_{U}	16.3	17.1	Yes
f_{T}	κ_{T}	16.9	17.6	Yes
$f_2^{(pc)}$	κ_{C}, f_{cc}	434.9	410.3	Yes
$f_3^{(pc)}$	High-frequency check	1083.1	1083.7	Yes
$f_5^{(pc)}$	High-frequency check	roaming	roaming	No

CAL Line Subtraction Option A – Pcal Lines

Let $x_{pc}(t) = a_{pc}\cos(\omega t - \phi_{pc}) + n(t)$ be the Pcal injection channel, read in from the front end: CAL-PCALY_RX_PD_OUT_DQ

1. Multiply by a local oscillator to demodulate:
   $$e^{-i\omega t}\left[a_{pc}\cos(\omega t - \phi_{pc}) + n(t)\right] = \frac{a_{pc}}{2}\left[e^{-i\phi_{pc}} + e^{-i(2\omega t - \phi_{pc})}\right]$$

2. Low-pass filter, leaving only the amplitude and phase: $\frac{a_{pc}}{2}e^{-i\phi_{pc}}$

CAL Line Subtraction Option A – Pcal Lines

3. Multiply by the Pcal correction factor to convert from counts to meters:

$$pc_{corr} \times \frac{a_{pc}}{2} e^{-i\phi_{pc}} = \frac{a}{2} e^{-i\phi}$$

pc_{corr} includes $1 / f^2$ dependence and a time delay. See T1700106.

4. Reconstruct $h(t; \omega)$:

$$\mathrm{Re}\left[2e^{i\omega t} \times \left(\frac{a}{2} e^{-i\phi}\right)\right] = a\cos(\omega t - \phi)$$

5. Subtract: $h_{clean}(t) = h(t) - a\cos(\omega t - \phi)$

CAL Lines Option A – Actuation Lines

- Subtraction of the actuation lines is very similar to that of the Pcal lines, except for the conversion from counts to meters:

$$\kappa_i \times A_i \times \frac{a_{inj}}{2} e^{-i\phi_{inj}} = \frac{a}{2} e^{-i\phi},$$

  where a_{inj} is the amplitude of the line in the injection channel

- Note the application of the correction factor κ_i, done to ensure the best accuracy.

General Line Subtraction (CAL Lines B)

Witness and EPICS channels:

Channel Name	**Purpose**	**Current L1 Value**	**Current H1 Value**
CAL-PCALY_RX_PD_OUT_DQ	Subtract Pcal lines	N/A	N/A
SUS-ETMX_L3_CAL_LINE_OUT_DQ	Subtract TST/L3/ESD line	N/A	N/A
CAL-CS_TDEP_SUS_LINE3_REF_A_ TST_NOLOCK_ [REAL / IMAG]	Subtract TST/L3/ESD line	-1.11638141e-15 - 2.69608368e-16 *i*	5.31342527e-17 + 2.31051358e-16 *i*
SUS-ETMX_L2_CAL_LINE_OUT_DQ	Subtract PUM/L2 line	N/A	N/A
CAL-CS_TDEP_SUS_LINE2_REF_A_ PUM_NOLOCK_ [REAL / IMAG]	Subtract PUM/L2 line	2.04484822e-15 + 2.18105139e-16 *i*	-1.79675665e-18 + 9.40864182e-19 *i*
SUS-ETMX_L1_CAL_LINE_OUT_DQ	Subtract UIM/L1 line	N/A	N/A
CAL-CS_TDEP_SUS_LINE1_REF_A_ UIM_NOLOCK_ [REAL / IMAG]	Subtract UIM/L1 line	4.78670848e-18 + 1.0887087e-18 *i*	3.89767406e-18 + 9.09782576e-19 *i*
PEM-EY_MAINSMON_EBAY_1_DQ	Subtract Power Mains lines	N/A	N/A

General Line Subtraction (CAL Lines B)

1. Demodulate $h(t)$ and witness channel(s) at 60 Hz and harmonics, ω_i.
2. Solve $\begin{bmatrix} 1 & {}^{w_2}/_{w_1} & \cdots \\ {}^{w_1}/_{w_2} & 1 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{h}{w_1} \\ \frac{h}{w_2} \\ \vdots \end{bmatrix}$ for the T_i at frequency ω_i.
3. Compute a running median(currently set at 128 seconds).
4. $h_{\text{clean}}(t) = h(t) - \sum_i \sum_j T_j(\omega_i)\, w_j(\omega_i)$

Currently, only one channel is used: PEM-EY_MAINSMON_EBAY_1_DQ

This method can also be used to subtract the calibration lines instead of using the previous method.

Power Mains Line Subtraction

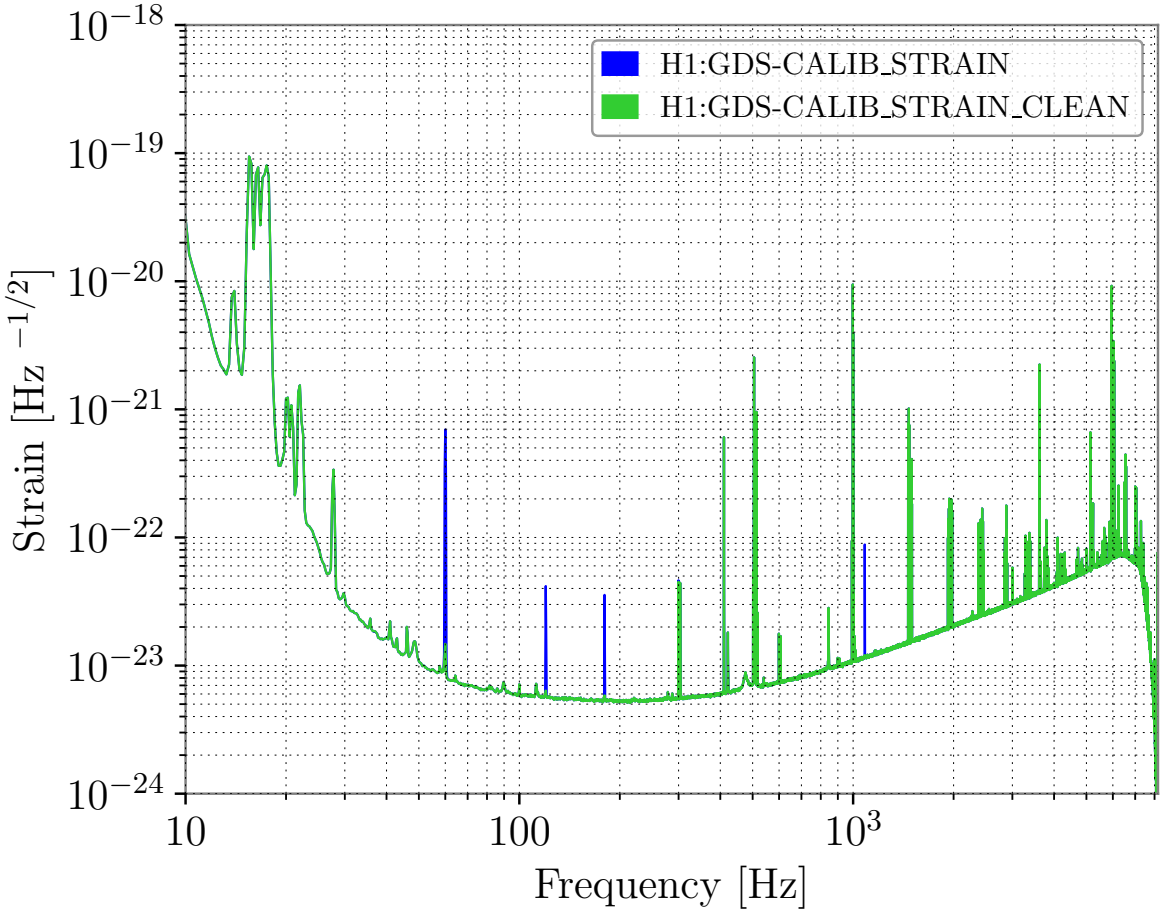
Problem: Power mains frequency drifts substantially, as much as ± 0.02 Hz intentionally (actually more based on measurements)

- High-latency solution:
  If the low-pass filter used in demodulation is centered in time, this is not a problem as the measured phase will just oscillate.

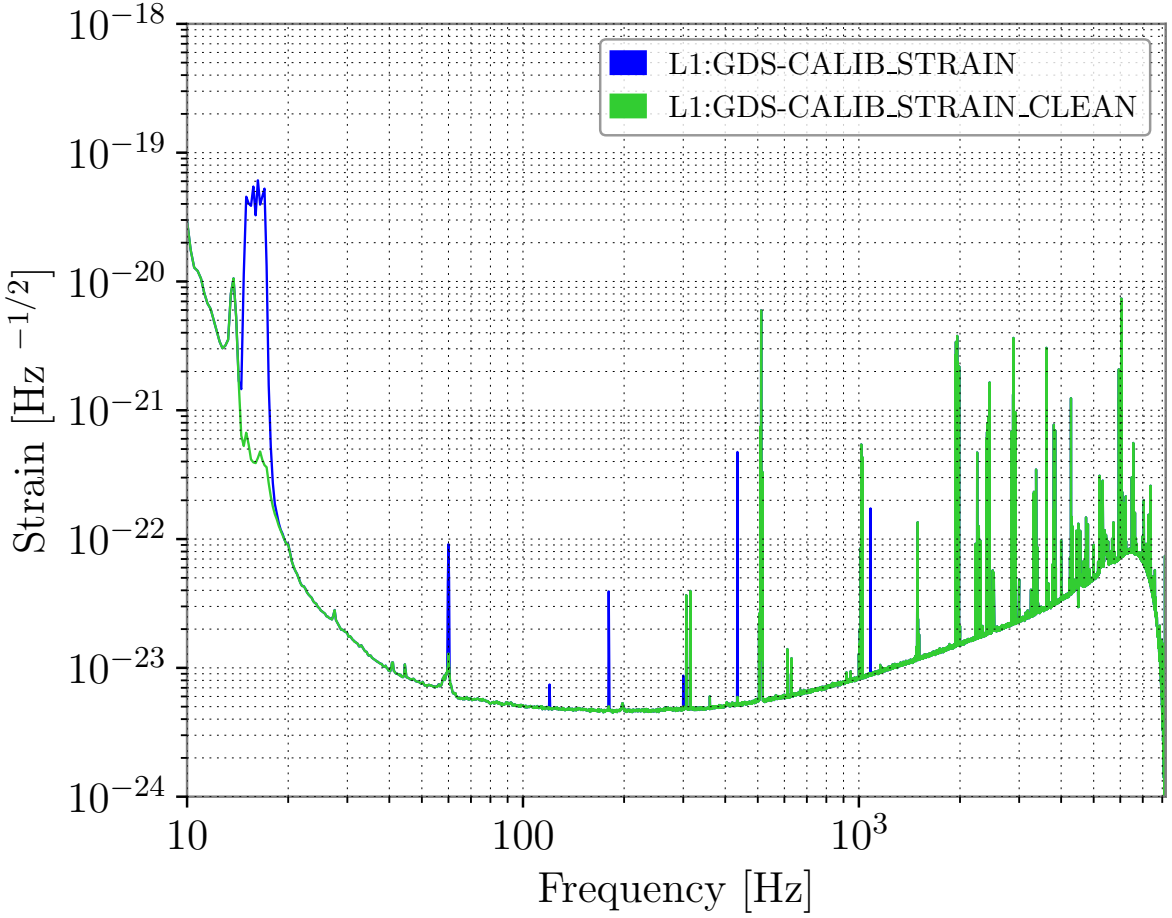
- Low-latency solution:
  The low-pass filter cannot be centered in time, so a corrective phase factor must be included: $h(\omega_i) = e^{i\phi_{corr}}\ T(\omega_i)\ w(\omega_i)$.
  This phase factor depends on the power mains frequency, which is constantly tracked using a gstlal-calibration element, lal_trackfrequency.

Calibration Line Subtraction Results

H1



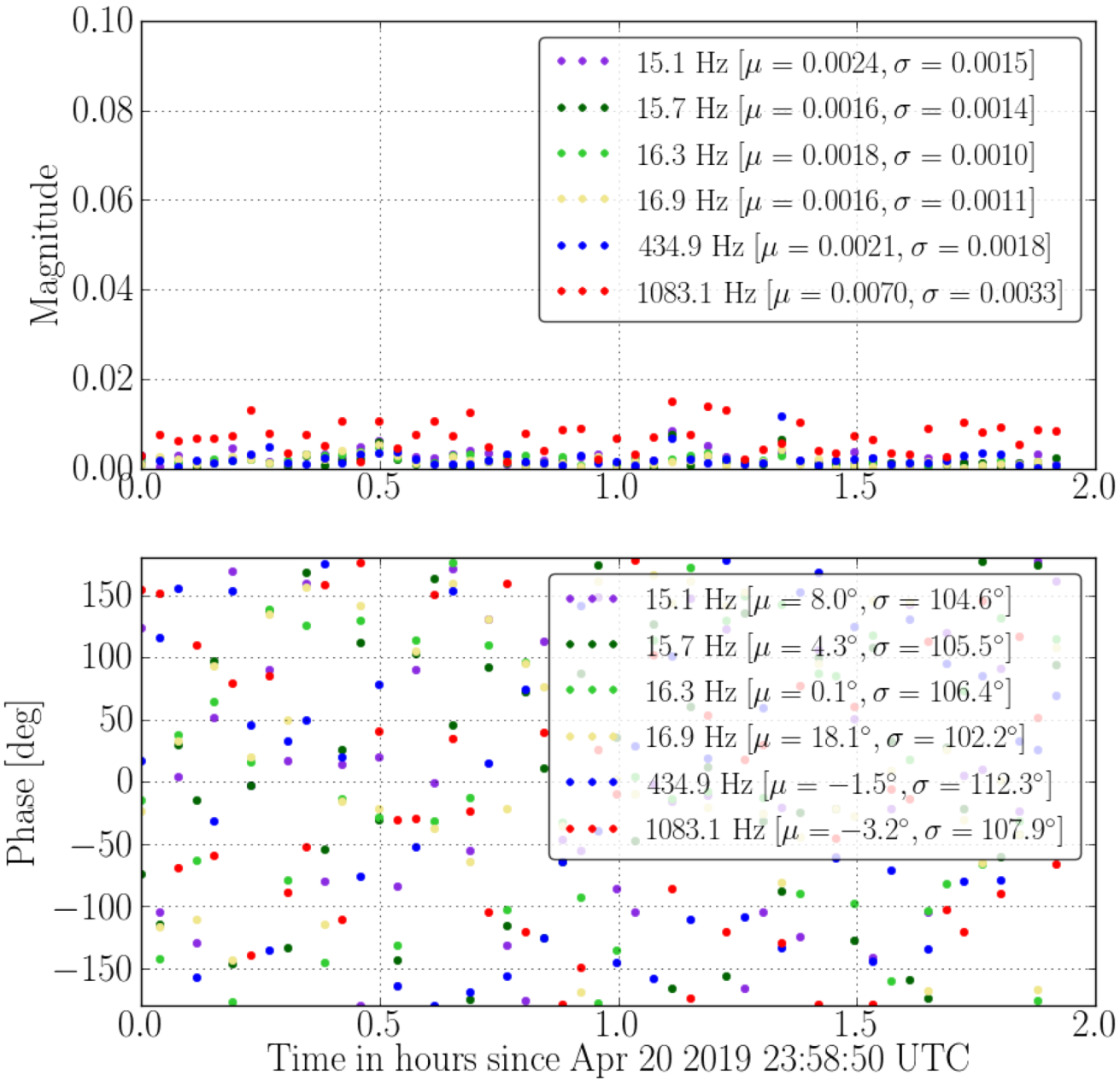
L1



(Mistake with line-frequency update last week!)

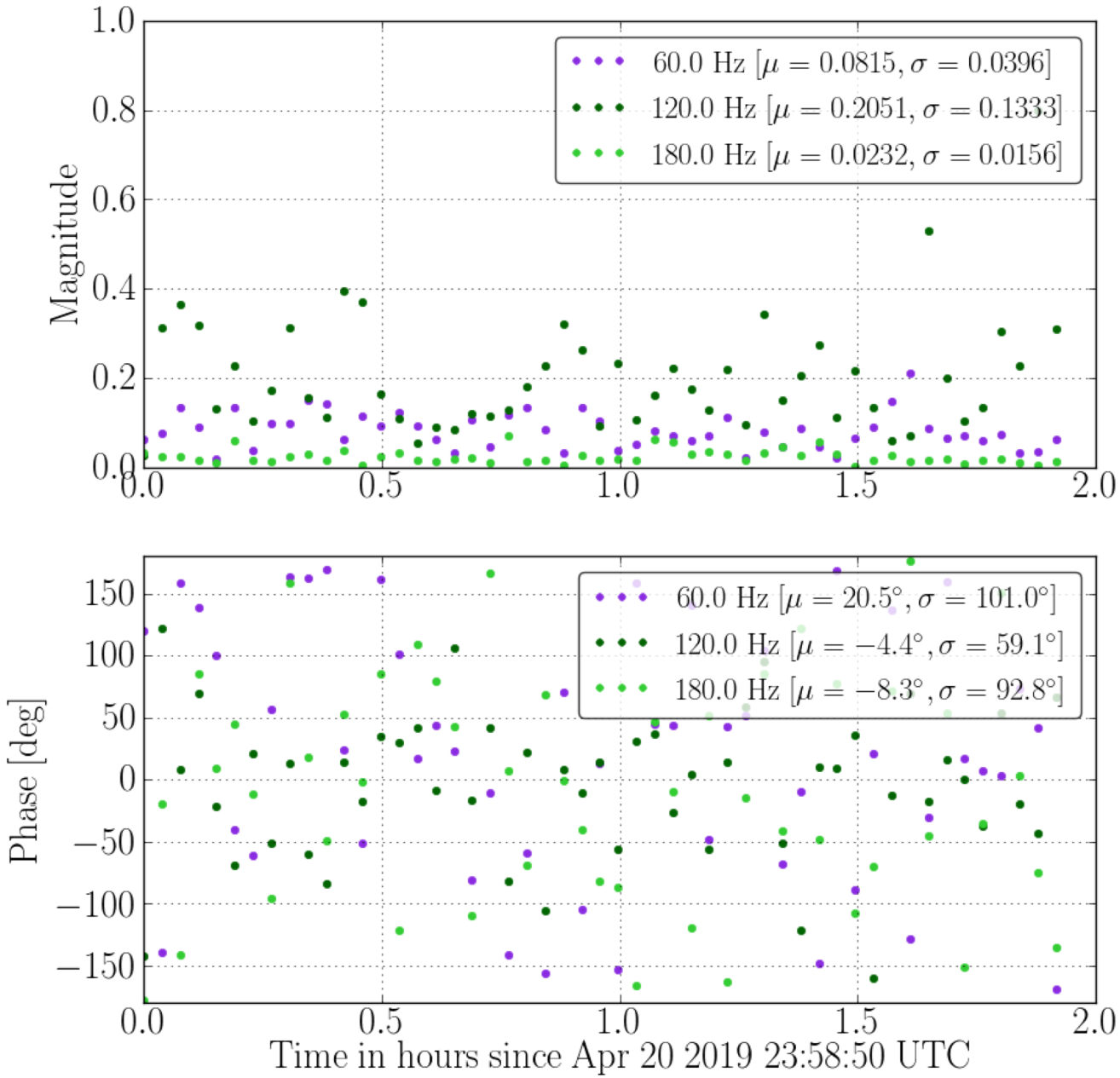
Calibration Line Subtraction Results (L1)

Transfer functions of $h_{\text{clean}}(t)$ / $h(t)$ are shown at each calibration line frequency for 2 hours.



Low-latency Power Mains Subtraction Results (L1)

60-Hz line has the highest SNR, so removing it has the most impact.



Line Subtraction – Alternative Method

Why not just compute a transfer function between witnesses (e.g. Pcal) and $h(t)$ and apply a filter at all frequencies?

- Computational cost – Filtering is expensive, and the calibration lines are already demodulated to compute the kappas.
- This method can only affect $h(t)$ at selected frequencies, reducing the risk of side-band noise.
- It may take longer to average over noise in transfer functions computed at all frequencies. Since line subtraction is done before broadband noise subtraction, this allows those transfer function to be computed with less delay.

Broadband Noise Subtraction

gstlal-calibration element lal_transferfunction computes transfer functions between witness channels and $h(t)$ and produces FIR filters.

1. Compute Hann-windowed, overlapped FFTs of $h(t)$ and witnesses.
2. Smooth off any troublesome frequencies, useful for violin modes.
3. Take ratios of FFTs: $h(t)$ / witness, and each possible pairing of witnesses.
4. Average these ratios to compute transfer functions and an autocorrelation matrix. Currently using 1800 x 4sec overlapped FFTs = 1 hr of data.
5. Resample the transfer functions to get desired FIR filter length, done with a sinc table. (This allows the user to average over time and/or frequency)
6. Solve $\begin{bmatrix} 1 & {}^{w_2}/_{w_1} & \cdots \\ {}^{w_1}/_{w_2} & 1 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{h}{w_1} \\ \frac{h}{w_2} \\ \vdots \end{bmatrix}$ for the T_i

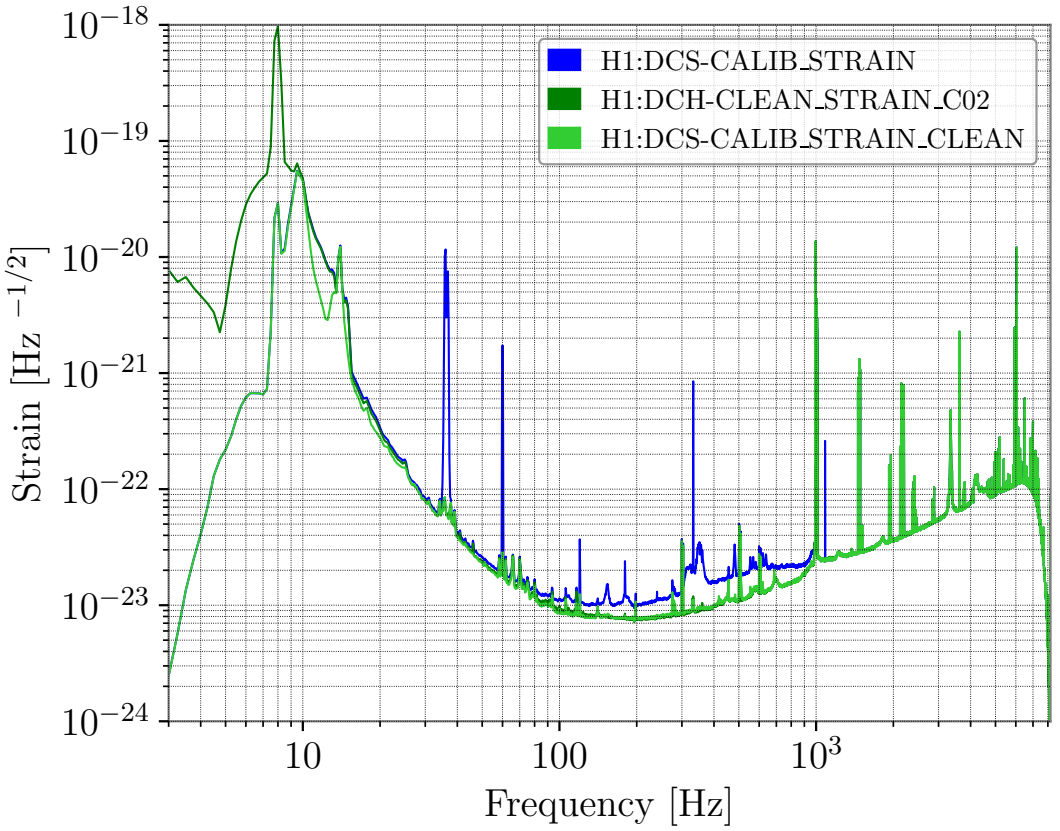
Broadband Noise Subtraction

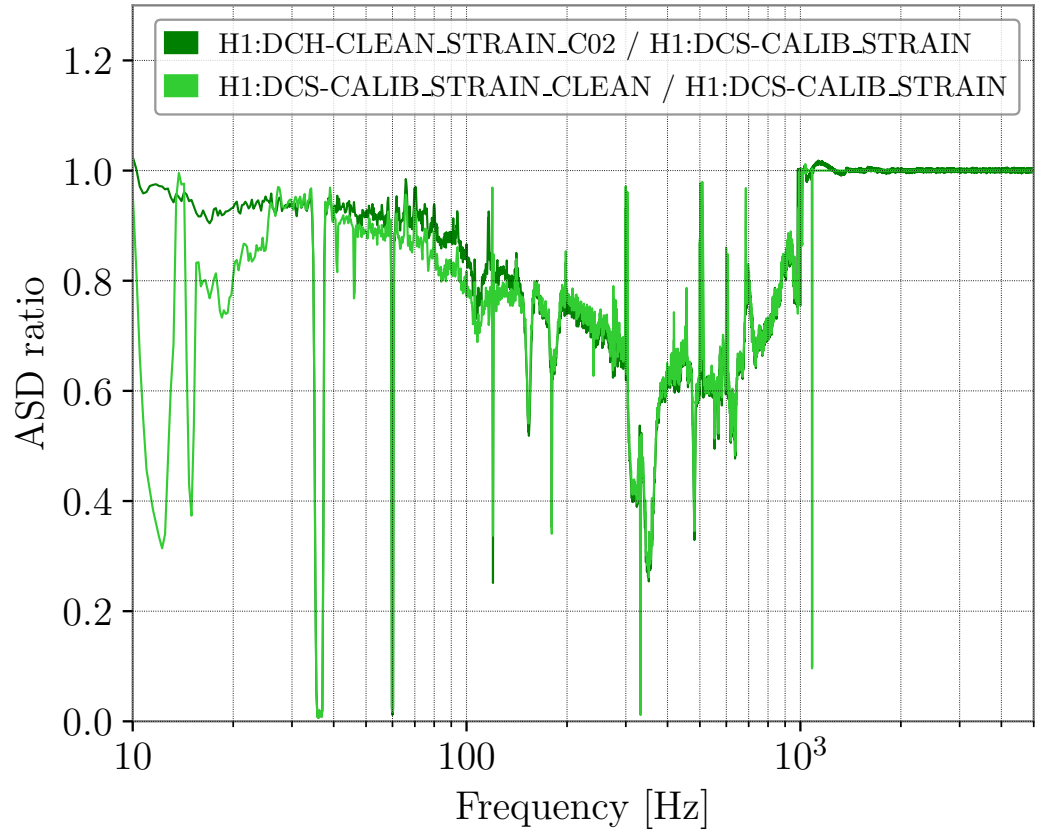
7. Multiply T_i by any high- or low-pass filters (currently a high-pass at 10 Hz).
8. Take an iFFT to make a FIR filter.
9. Smooth off the edges with a Tukey window.
10. Send new filter to lal_tdwhiten to apply to witness.
11. Subtract from $h(t)$
12. Repeat steps 1-11 with any additional sets of witnesses that should be handled separately. (Note: witness channels can be handled in parallel or in series.)

Broadband Noise Subtraction

- Low-latency:
  - Gated with GRD-ISC_LOCK_OK
  - Updates transfer functions and FIR filters every 1000 seconds (config-file option) and at the start of every lock stretch.
  - There will be 300 seconds (or our choice of FFT time) of data at the start of the first lock stretch after restarting the pipeline that will not be cleaned.
  - Testing indicates that this has negligible impact on latency (perhaps ~ 0.1 s).
- High-latency:
  - Gated with GRD-ISC_LOCK_OK
  - Updates transfer functions and FIR filters every 4096s (config-file option).
  - In order to clean as much data as possible, wait for transfer functions to be ready before producing frames.
  - To make output independent of start time, the data being subtracted from is the same data used to compute transfer function. This cannot be done in low latency.

Spectrum Comparison

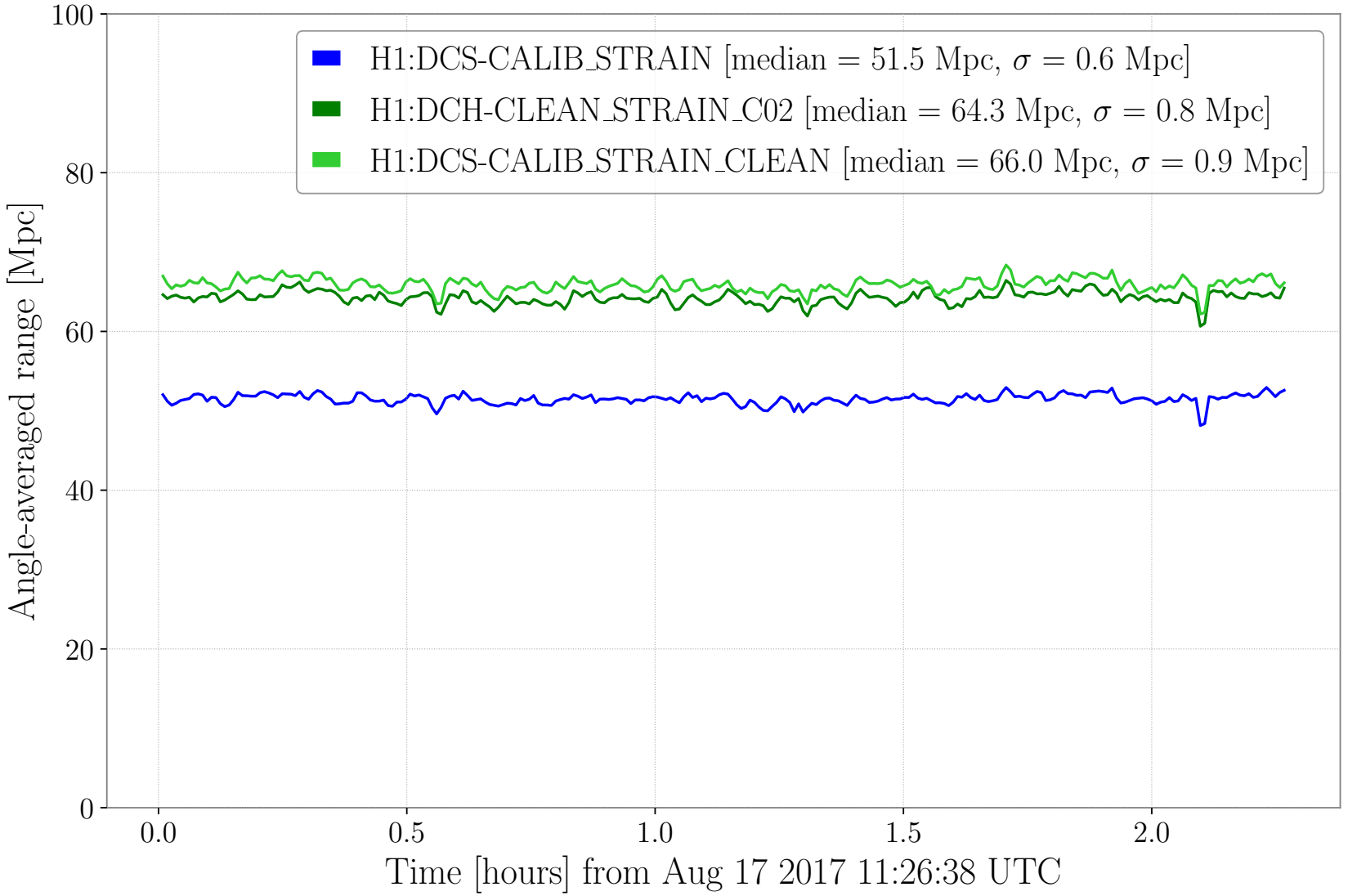




8192s of data is from around GW170817. TFs computed with 4096s of data and updated once every 4096s.

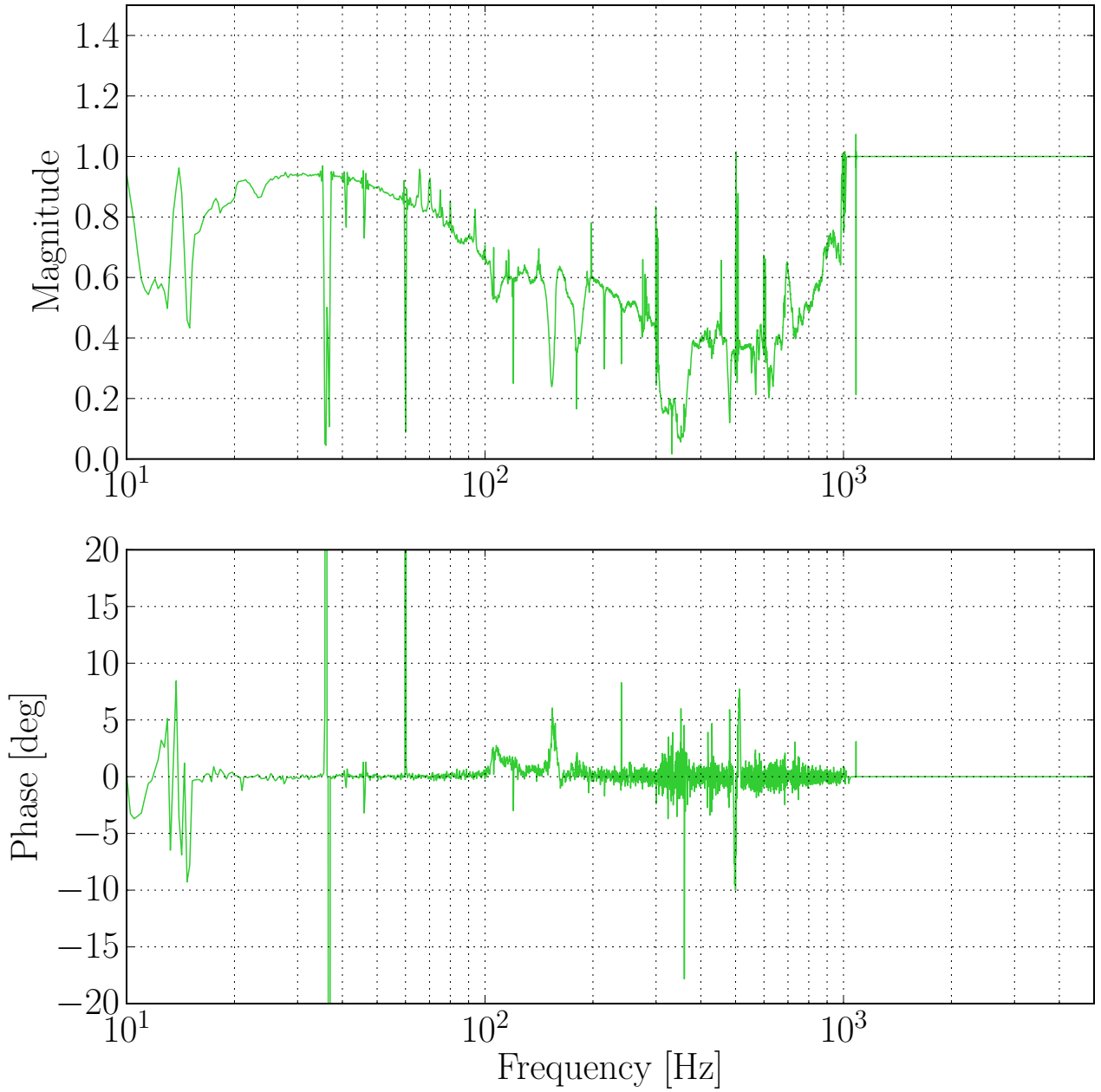
BNS Range

8192s of data from around GW170817. TFs computed with 4096s of data and updated once every 4096s.



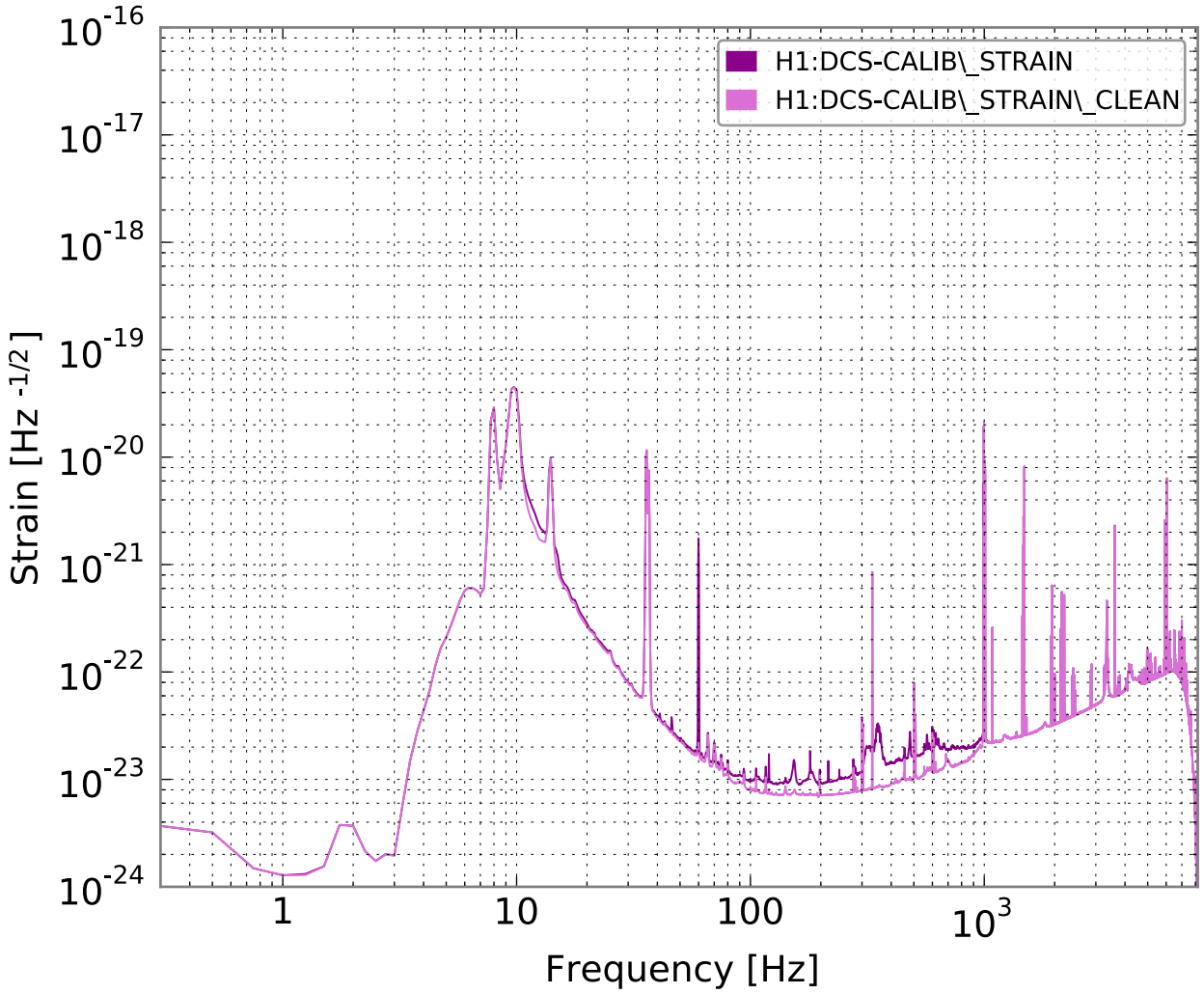
Noise Subtraction Transfer Function

- Averaged over ~22 hours of data
- Transfer functions update every 4096s.
- Phase difference is mostly consistent with zero, indicating updates are frequent enough.
- Updating every 10 hours resulted in significant deviation from zero.



Noise Subtraction with Calibration Lines

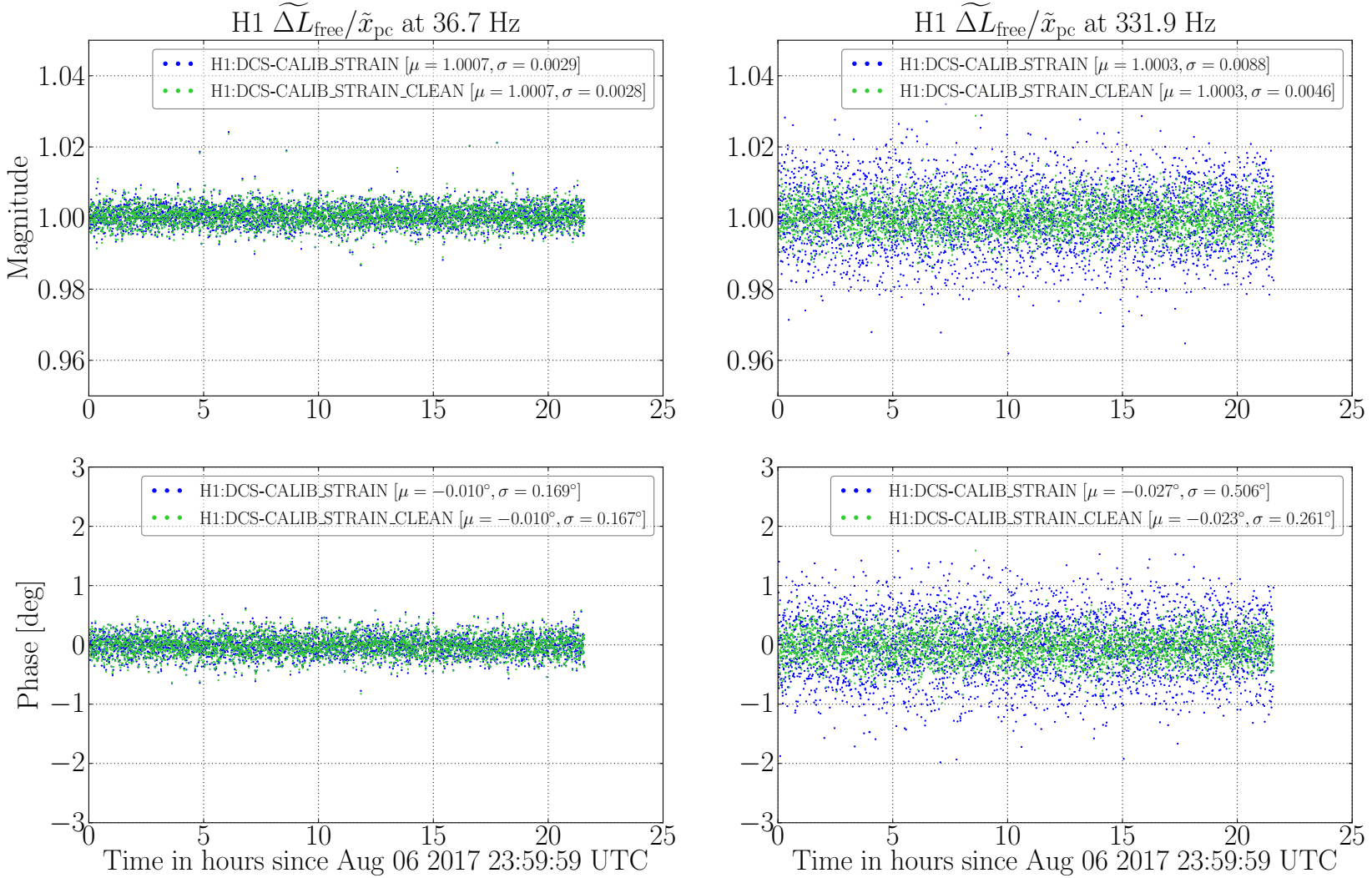
- Same 20-hr dataset was used here, without calibration line subtraction.
- Purpose is to see impact, if any, on calibration accuracy.
- Noise subtraction still works with calibration lines. Line frequencies were smoothed over in the transfer functions.



Pcal-to-DARM ratio comparison

- If the calibration is accurate, we expect unity magnitude and zero phase.
- Noise subtraction should preserve average values while reducing noise.
- Little impact at 36.7 Hz, as not much noise is subtracted there.
- Significant improvement at 331.9 Hz, as expected.
- Pcal and ΔL_{free} were demodulated using 20 s chunks of data.
- Plots shown on next slide

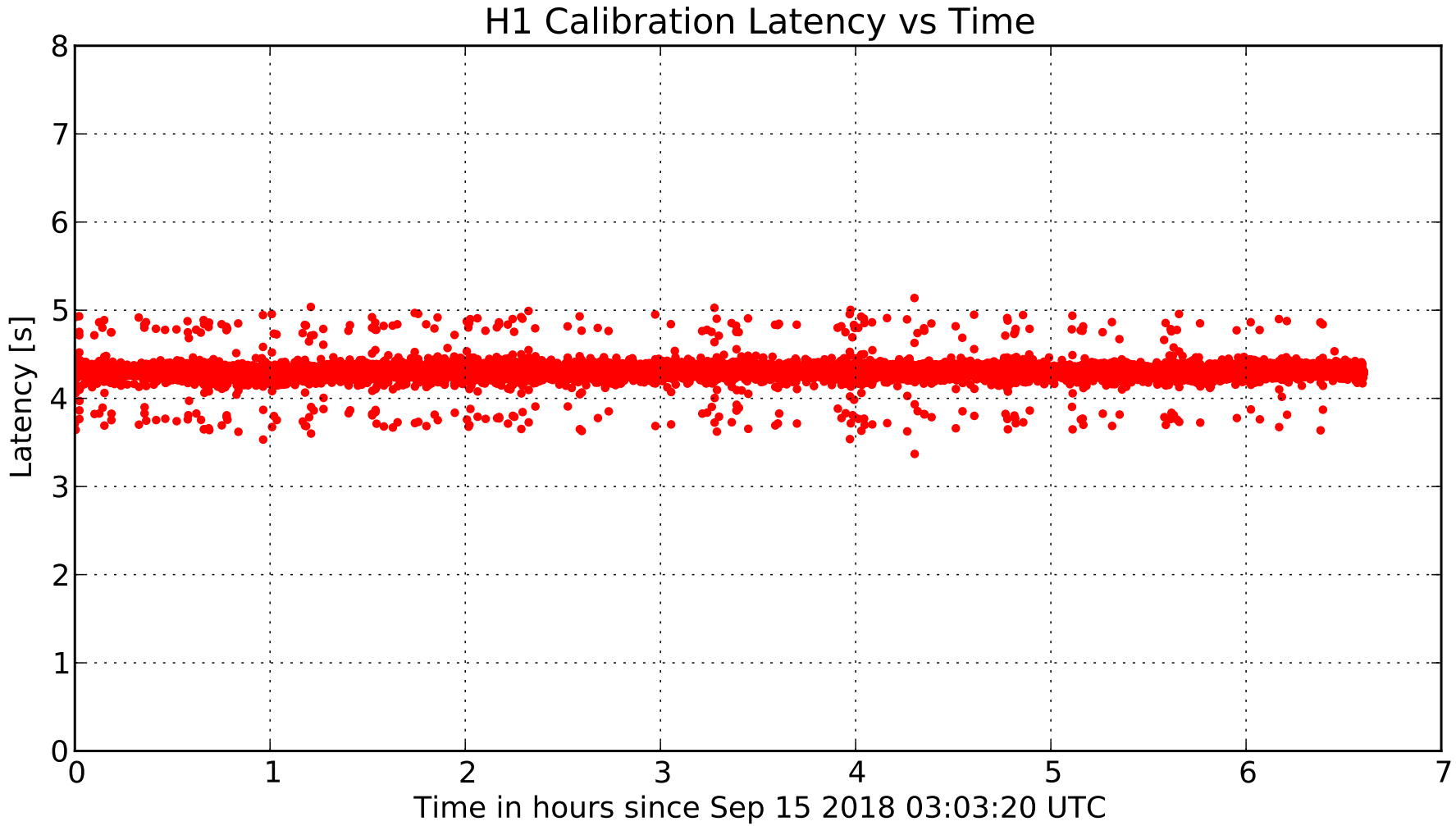
Pcal-to-DARM ratio comparison



Effect on Latency

Latency was computed by measuring the difference in real time between when data enters and leaves the gstlal calibration pipeline. 1-second frames were used for this test.

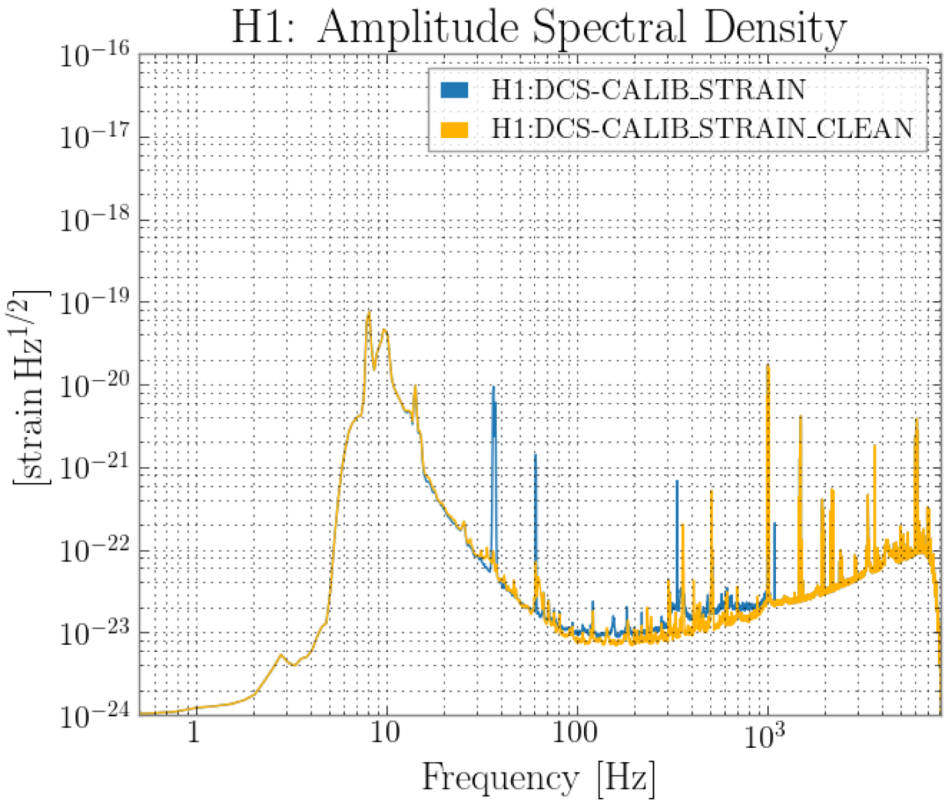
Six cleaning filter updates occurred during the time plotted.



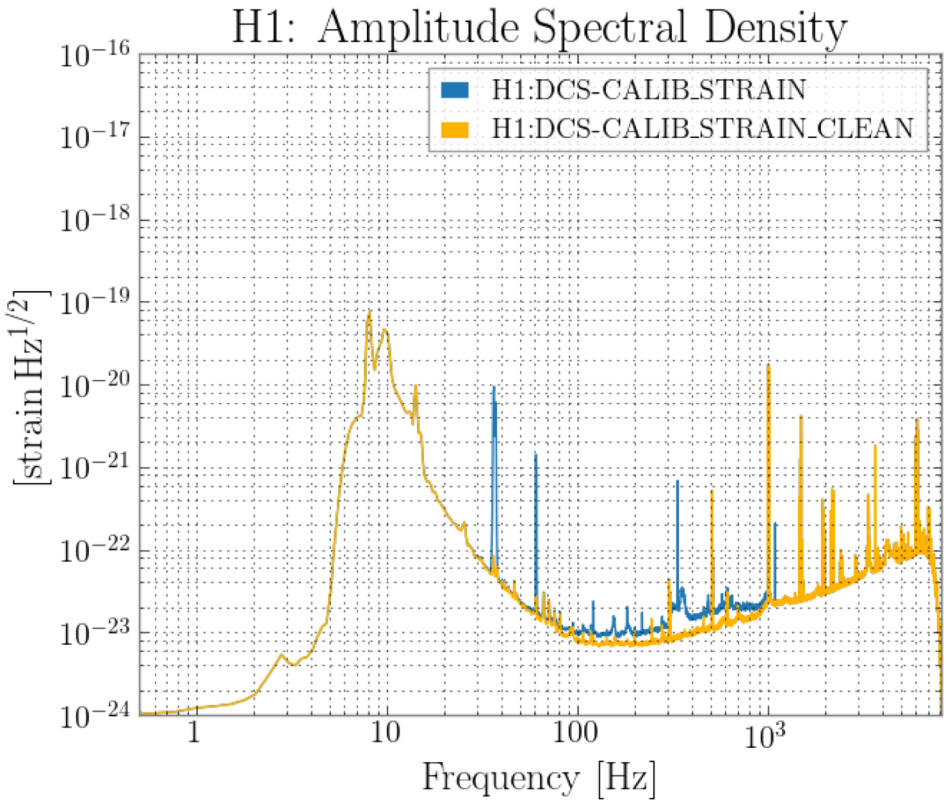
What about glitches?

A loud glitch could corrupt transfer functions. Taking a median instead of a mean minimizes the effect of large outliers. A loud glitch in H1 data on 2017-08-23 was used for this comparison, where transfer functions were computed from ~13 minutes of data.

Mean



Median



New CALIB_STATE_VECTOR Bits

- 26: LINE_SUBTRACTION_OK
  - Bandpass $h(t)$ and cleaned $h(t)$ in a chosen frequency band, and compute the RMS. If RMS[clean $h(t)$] < RMS[$h(t)$] in that band, bit is on, else off.
  - Main purpose is to check calibration line subtraction.

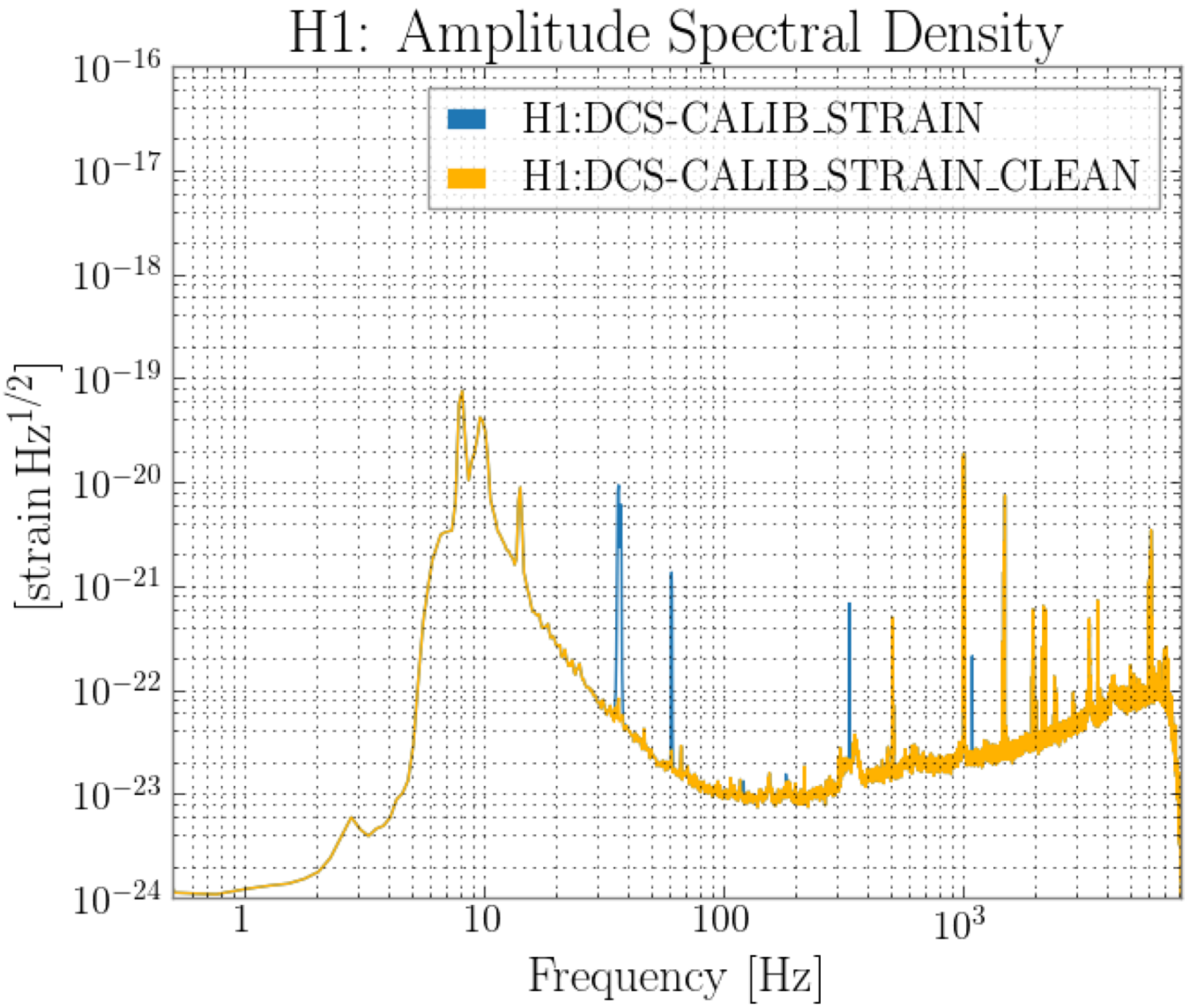
- 27: NOISE_SUBTRACTION_OK
  - Bandpass $h(t)$ and cleaned $h(t)$ in a chosen frequency band, and compute the RMS. If RMS[clean $h(t)$] < RMS[$h(t)$] in that band, bit is on, else off.
  - Main purpose is to check noise subtraction in most effective band.

- 28: NOISE_SUBTRACTION_GATE
  - Checks if the calculation of transfer functions is being gated

What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

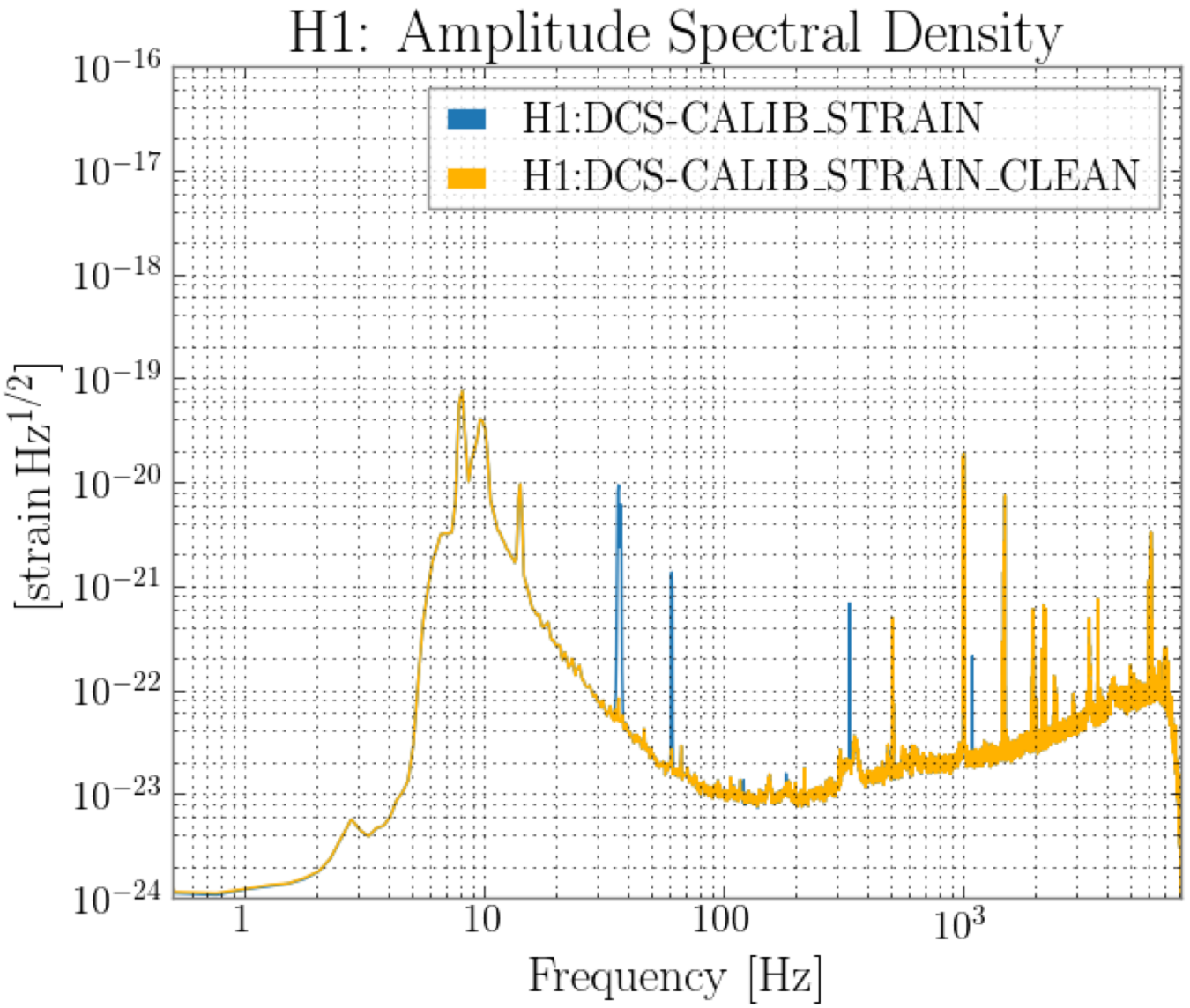
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

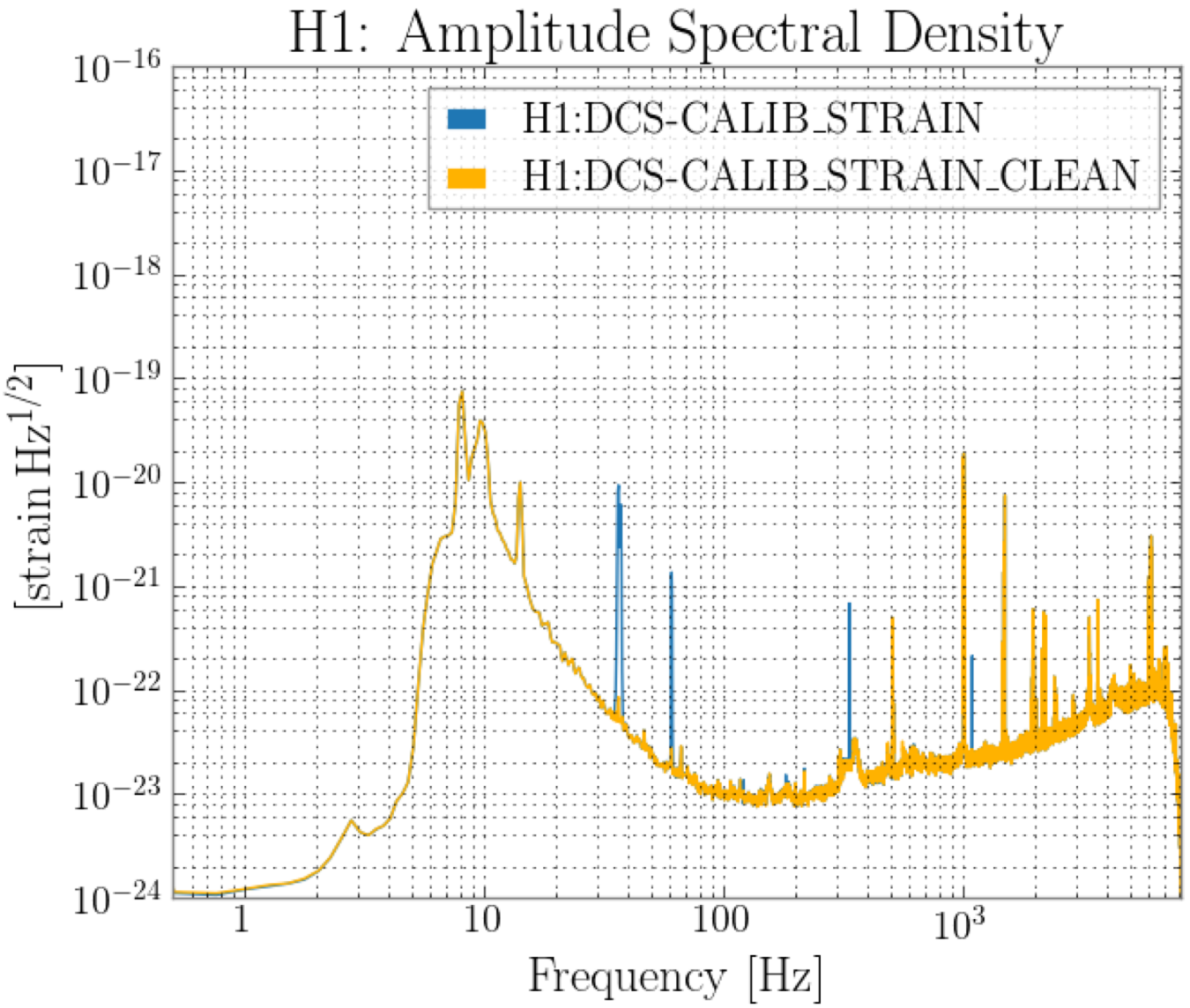
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

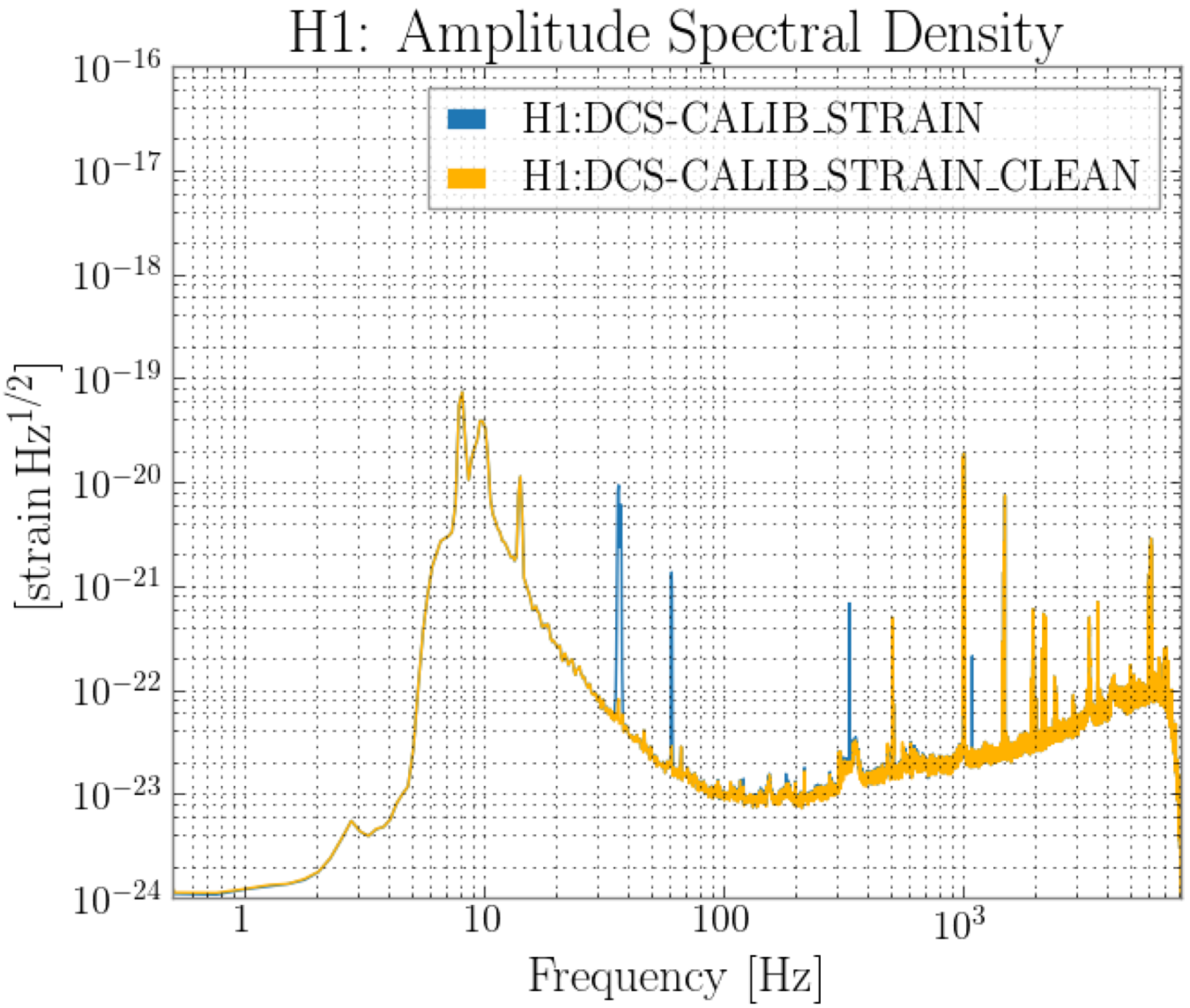
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

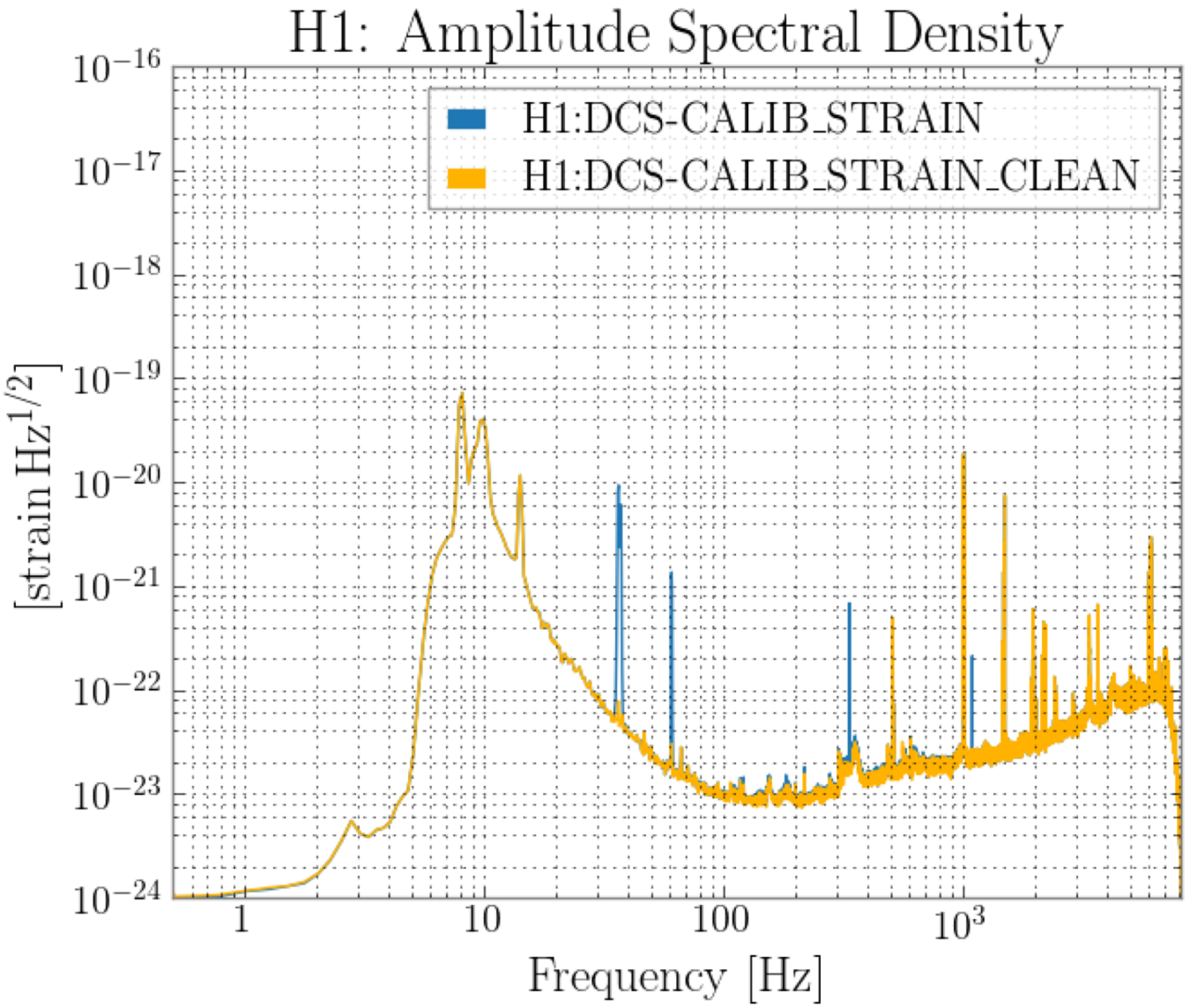
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

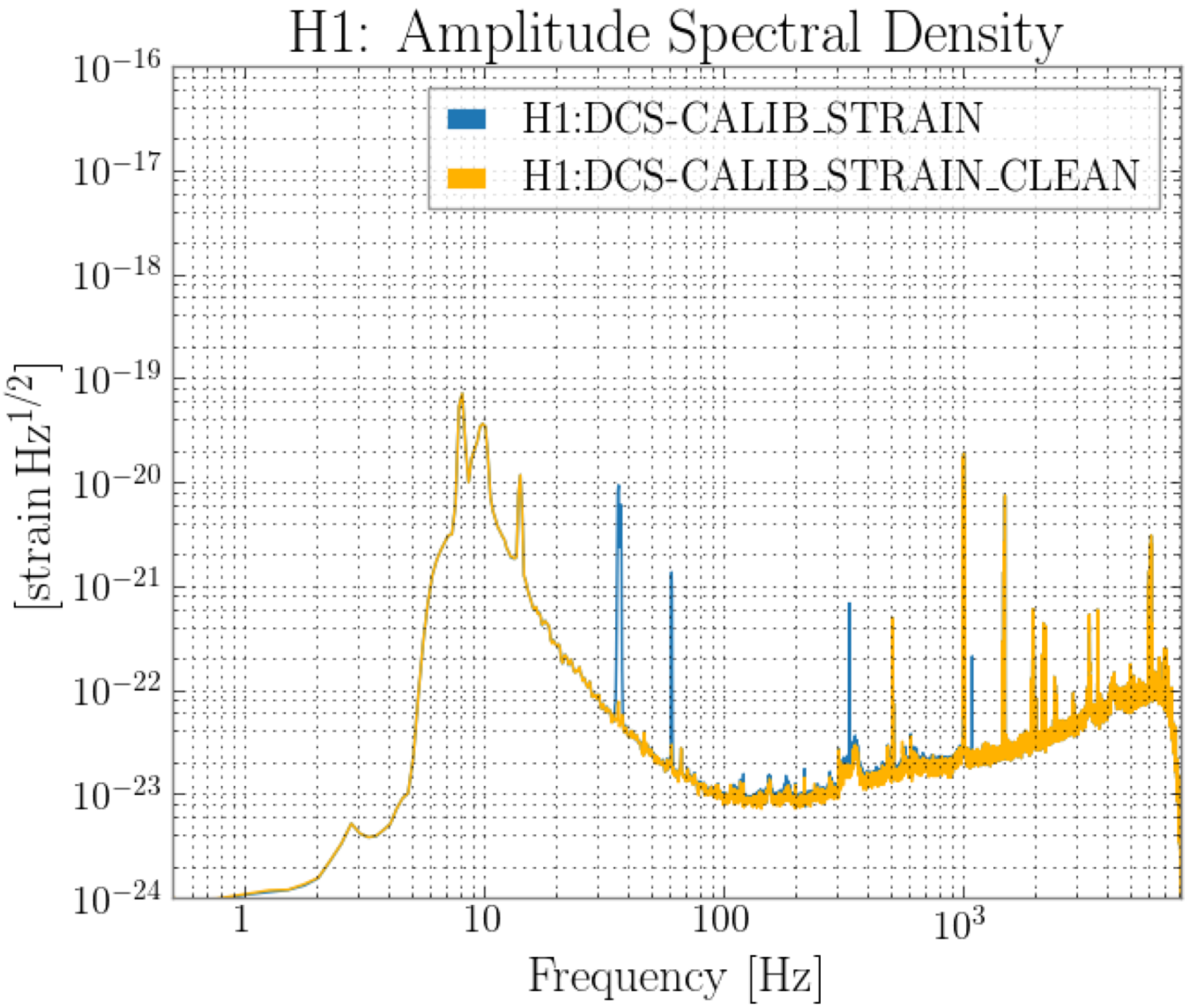
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

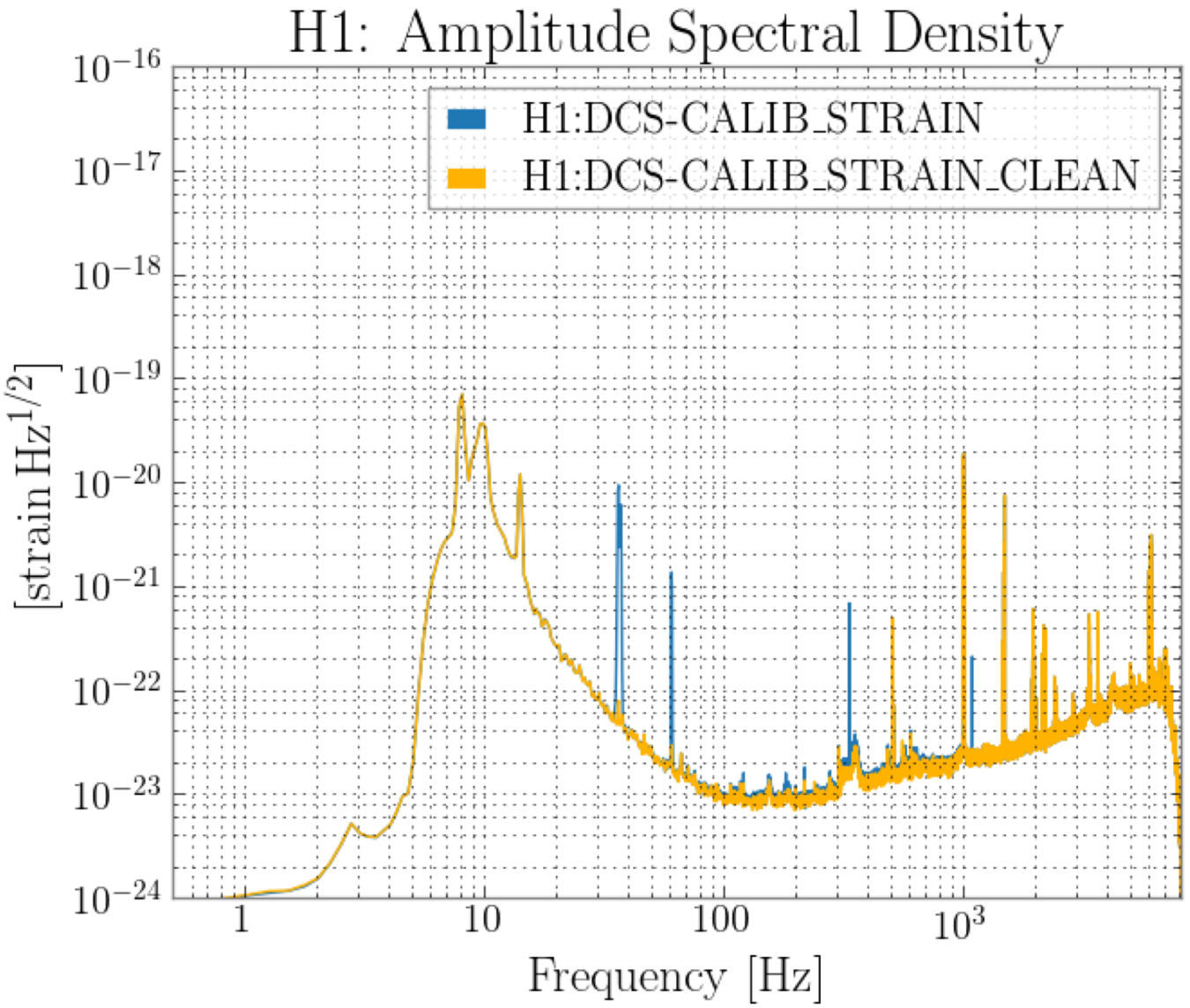
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

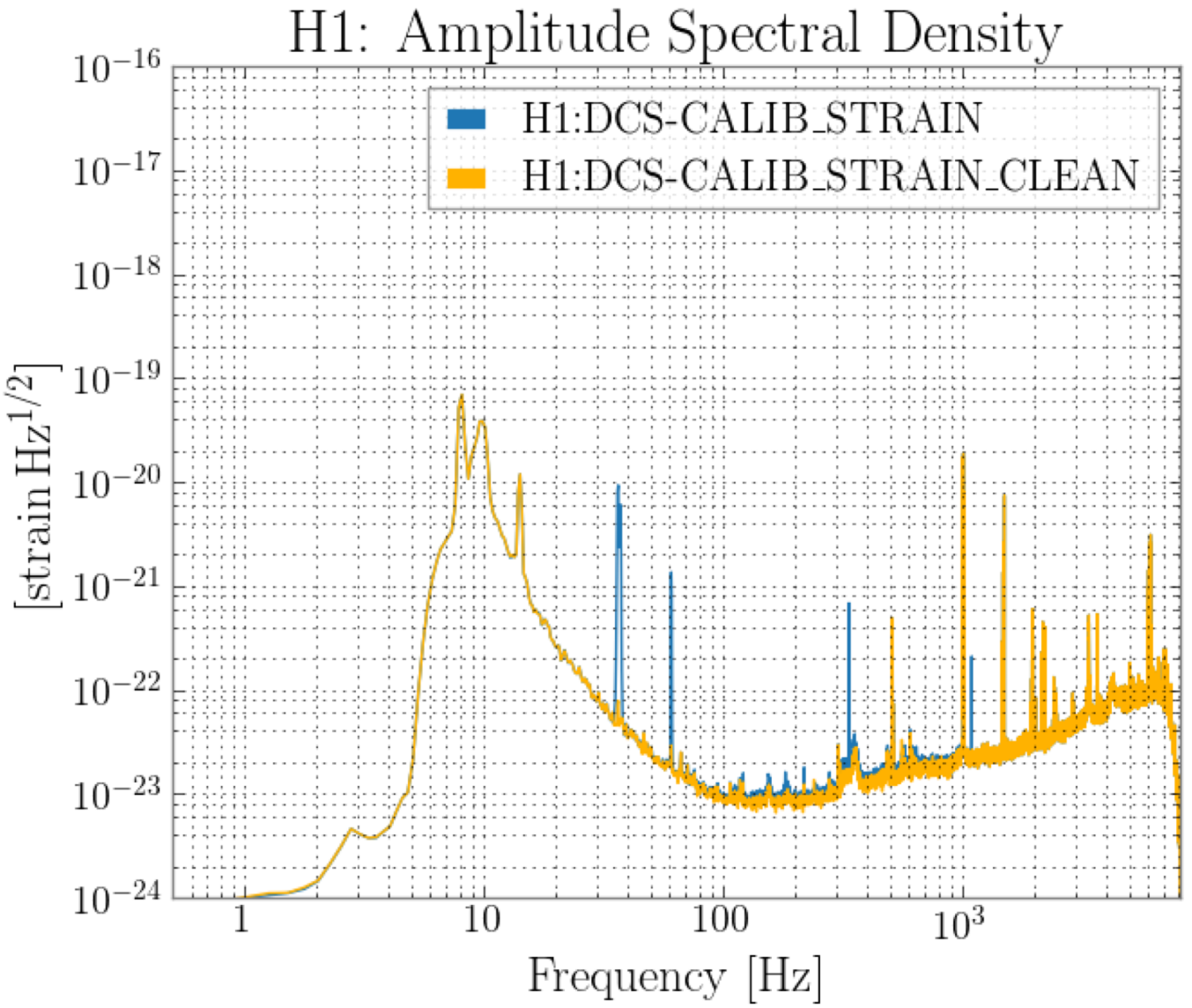
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

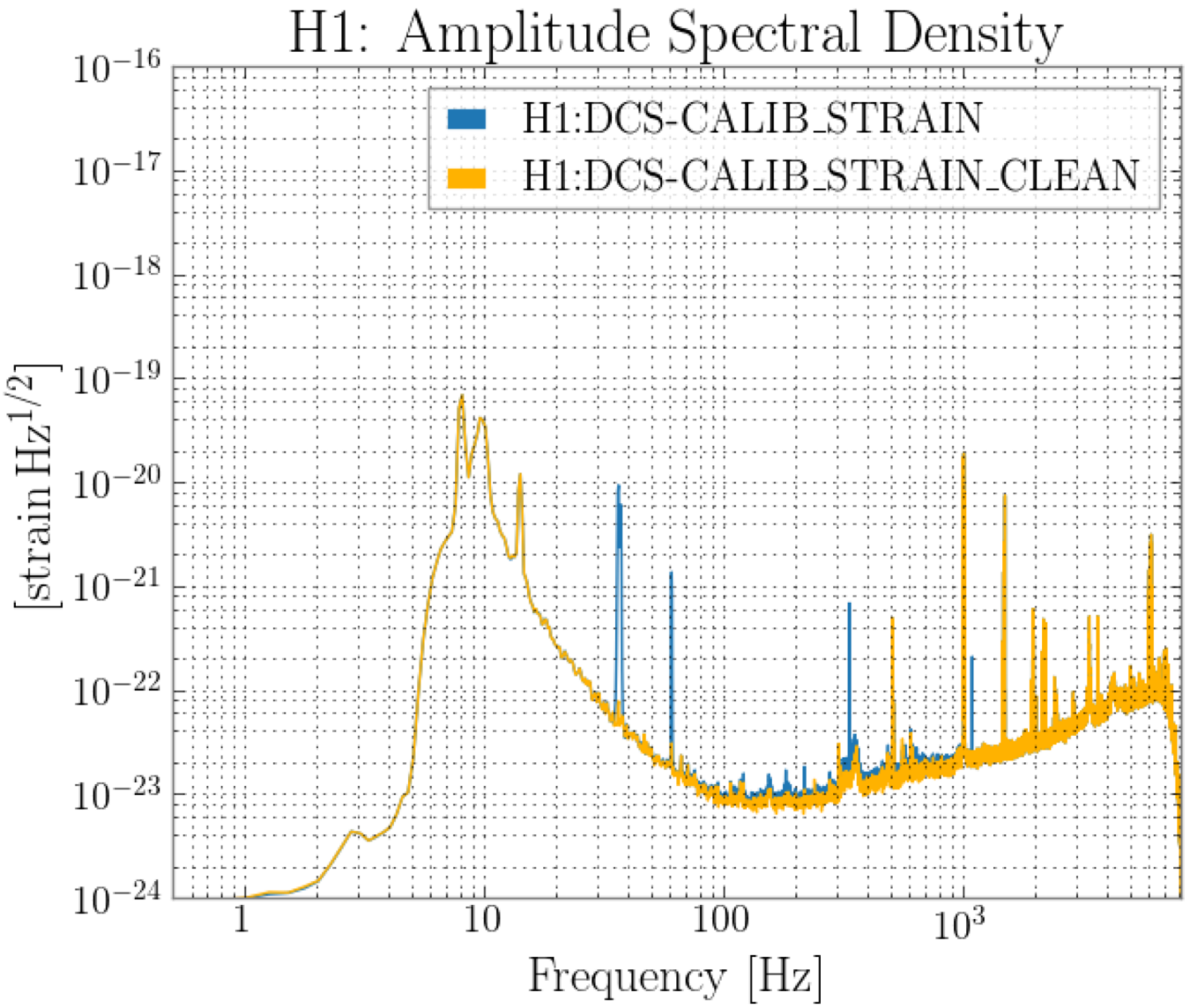
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

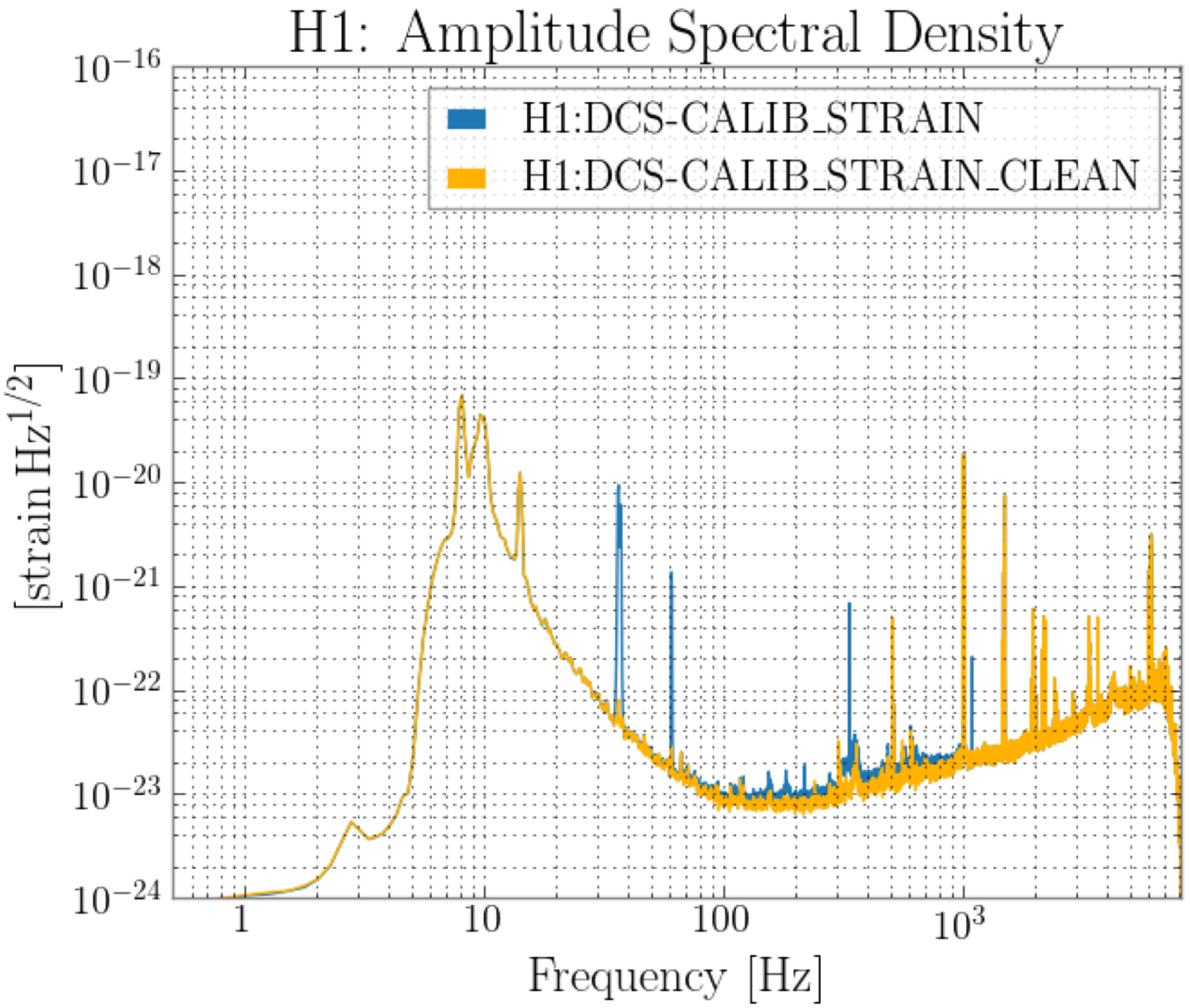
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

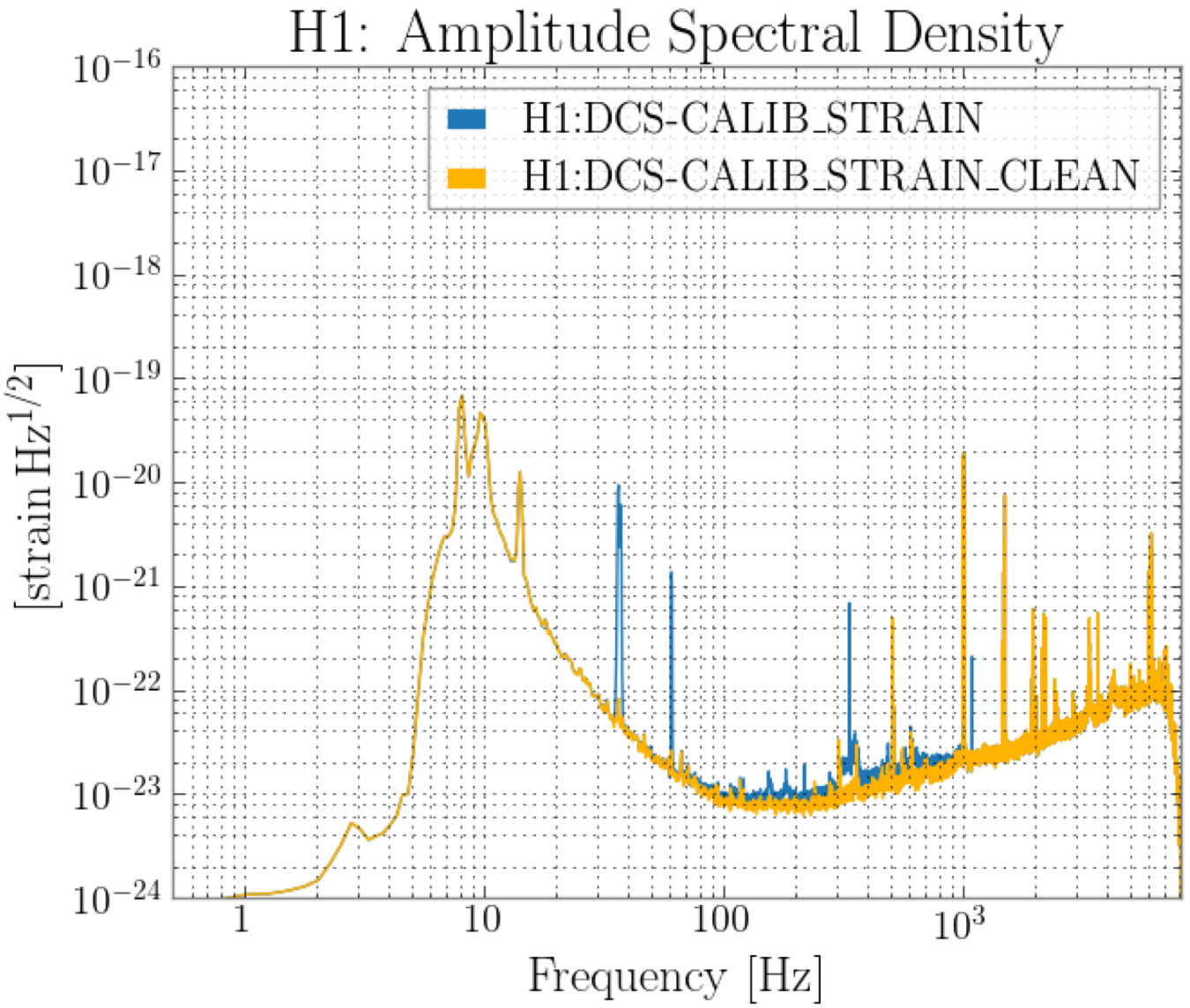
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

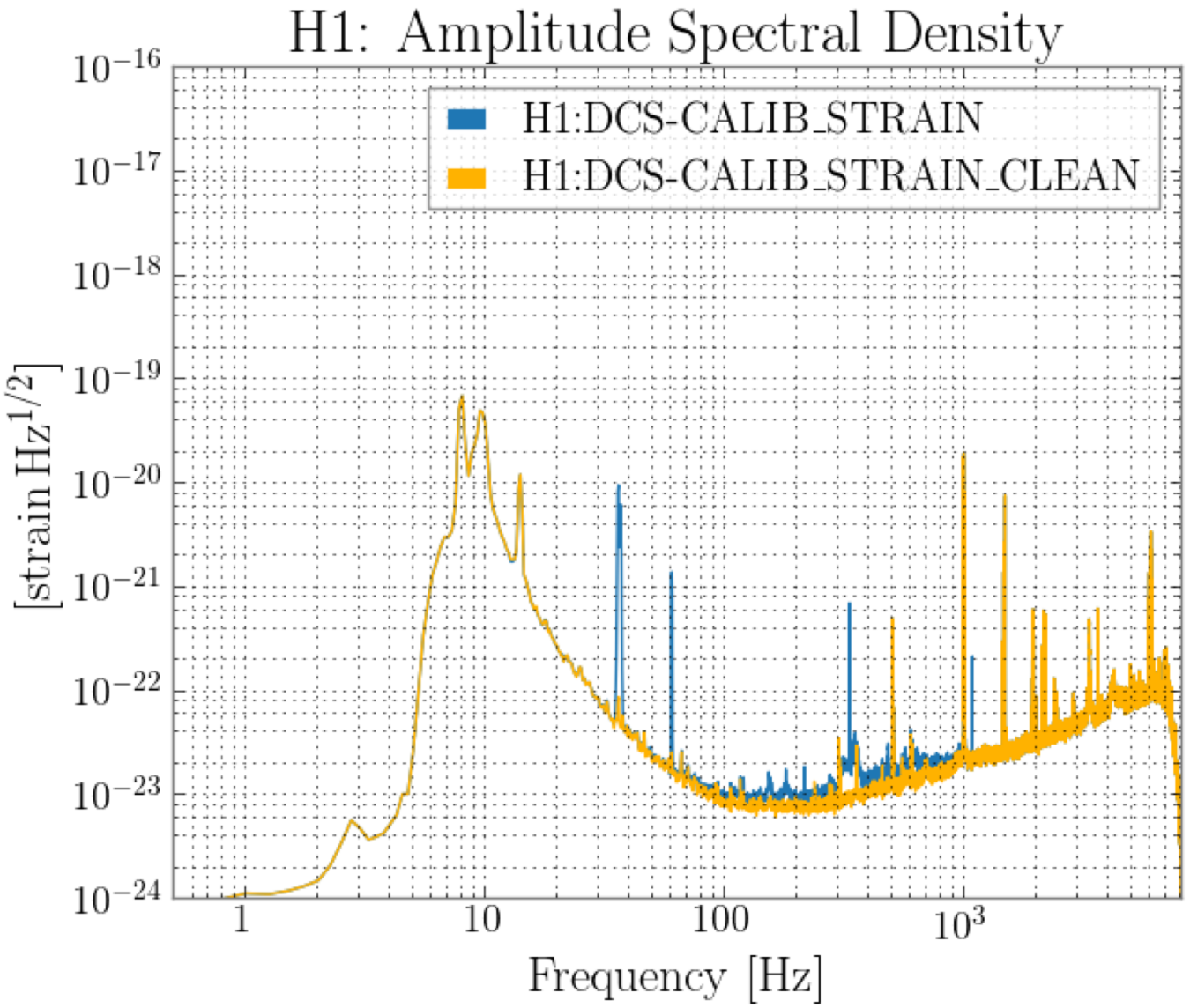
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

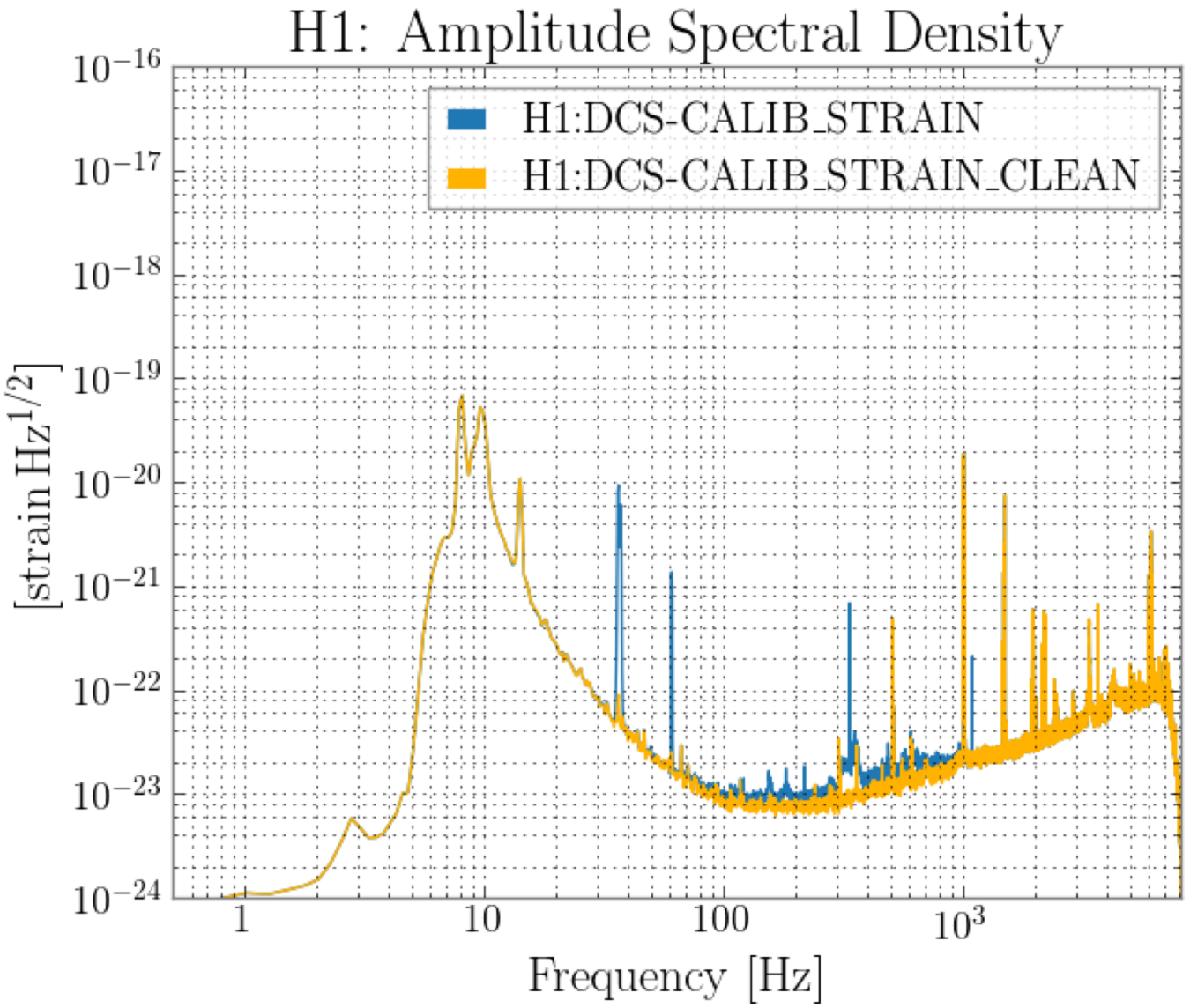
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



What about filter transitions?

Filters are tapered in using half of a Hann window over 10 seconds of transition time.

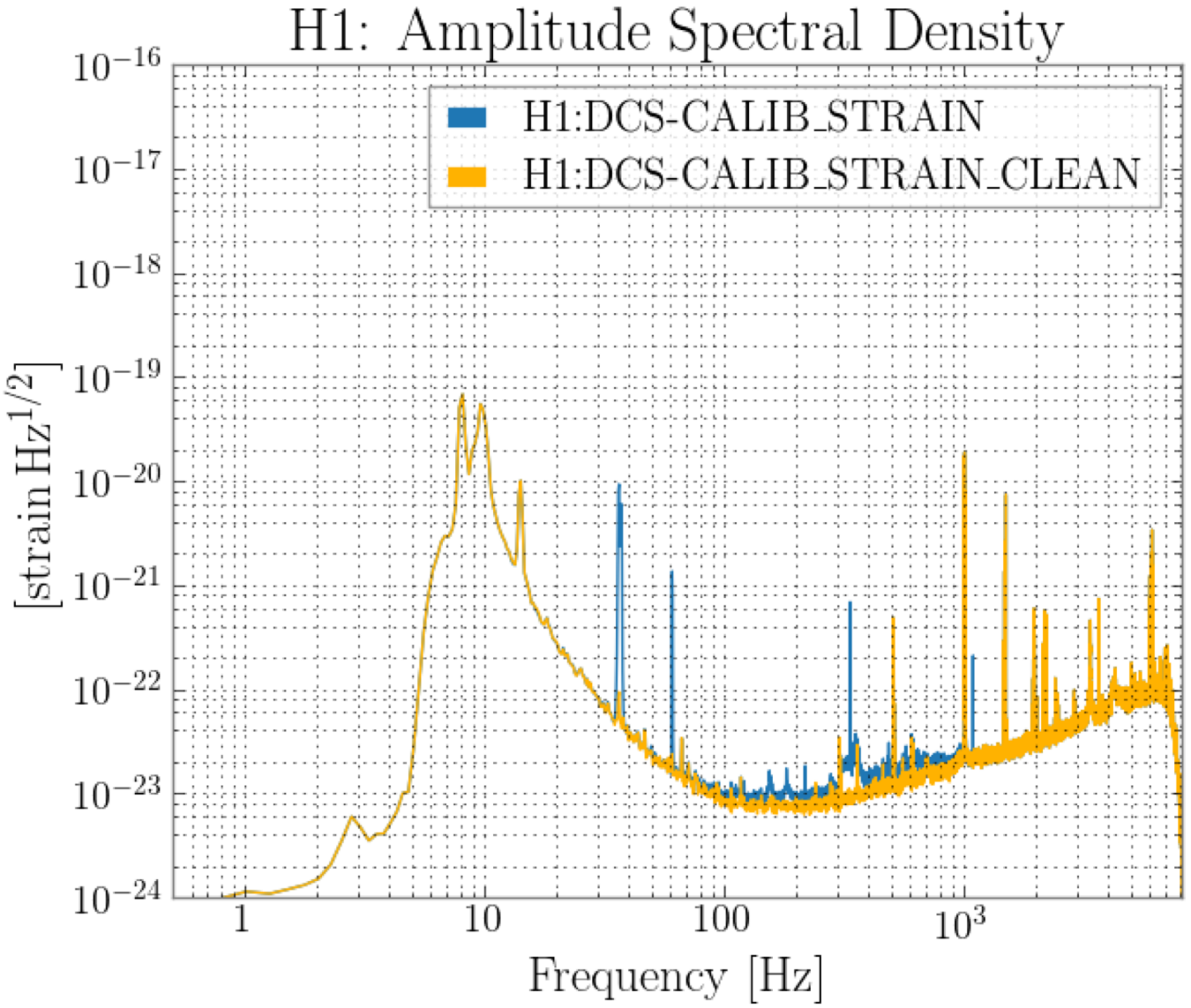
The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



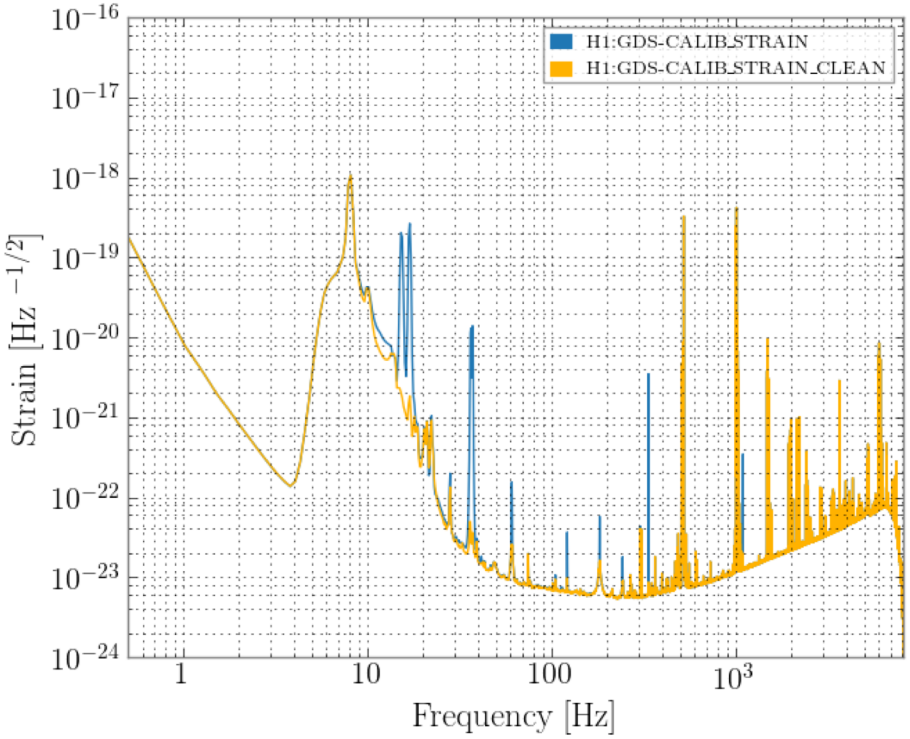
What about filter transitions?

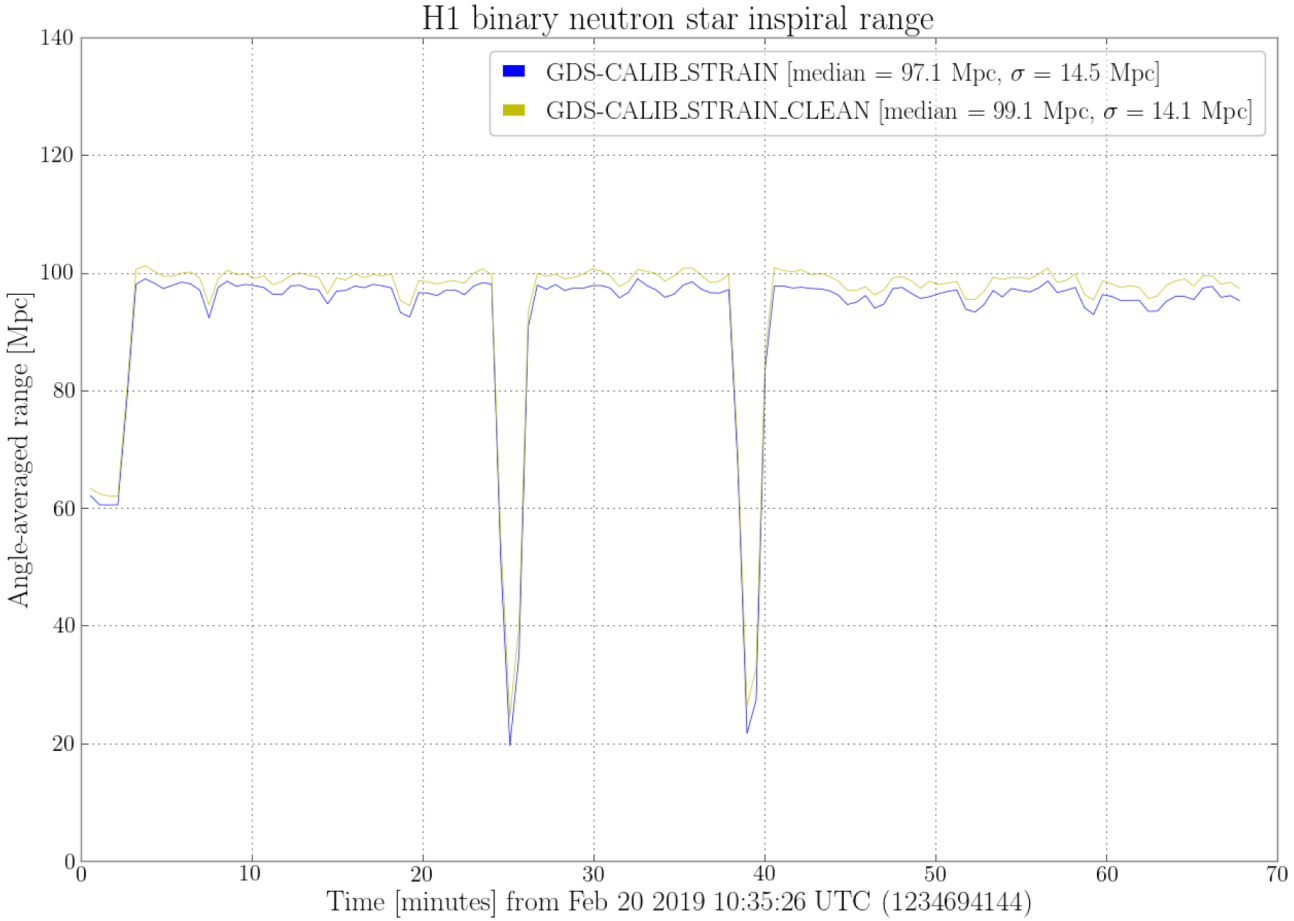
Filters are tapered in using half of a Hann window over 10 seconds of transition time.

The plot on the right shows an ASD comparison using 100s of data starting before filters are computed. Each plot in the next slides is shifted 10s later until the transition is over.



So far in ER13/ER14…





Smaller impact, 6.3% increase in detectable volume in this instance.
Spectrum comparisons coming soon on calibration testing tab of summary pages.

References

- J. C. Driggers *et al.* “Improving astrophysical parameter estimation via offline noise subtraction for Advanced LIGO.” arxiv: 1806.00532 (2018).
- D. Davis *et al.* “Improving the Sensitivity of Advanced LIGO Using Noise Subtraction.” arxiv: 1809:05348 (2018).
- L. Tsukada *et al.* “Application of a zero-latency whitening filter to compact binary coalescence gravitational-wave searches.” Phys. Rev. D **97**, 103009 (2018).