Updates on edges software suite for EDGES-3 data analysis

Akshatha Vydula[1] and Steven Murray[1]

[1]School of Earth and Space Exploration, Arizona State University ,Tempe AZ USA 85281

EDGES-3 was deployed in Nov 2022 with significant changes from the previously operational EDGES-2 system. Notable changes are:

- Bigger ground plane 48 × 48 m
- in-situ calibration (instead of lab calibration)
- additional health data such as temperature and humidity

For details on deployment and hardware specification, refer the MIT EDGES memo series - Memo #291, #300, #303, #406. Due to certain hardware upgrades (mainly to accommodate in-situ calibration, the 3-position switch is replaced by an 8-position switch), and changes in file formats, the exiting `edges` software suite needed upgrades to be able to analyze EDGES-3 data. This memo is to document the software upgrade, and testing based on Alan's **c-code**.

File structure:

EDGES-3 primarily gives three kinds of measurements: (i) S11, (ii) calibration and sky spectra and (iii) temperature/humidity logs, totalling 12 S11 files, four calibration files[1], one sky spectra from the antenna and one temperature log file. The file structure is as follows:

- S11 files:

```
root_directory/
    YYYY_DOY_HH_<amb|ant|hot|open|short|lna|lna_S|lna_O|lna_L|L|O|S>.s1p
```

[1]Four calibration loads refer to `ambient, hot, long cable open & long cable short` loads used in noise wave calibration. Refer Monsalve et al. (2017) for formalism.

- Spectra:

  ```
  root_directory/mro/<amb|ant|hot|open|short>/<2022|2023>/
  YYYY_DOY_HH_MM_<amb|ant|hot|open|short>.acq
  ```

- Temperature logger:

  ```
  root_directory/temperature_logger/temperature.log
  ```

Calibration procedure

Since the calibration mechanism is slightly different, and the file naming is different from EDGES-2, we wrote additional software to the existing `edges-io, edges-cal` packages to be able to process EDGES-3 data. In addition, the structure of handling and processing the raw data is a little more involved to accommodate for different S11 calibration measurements. Figure 1 shows the flowchart detailing the processing pipeline employed. We follow the following procedure:

- Gather `S11 files` for each load and calibrate using `L, O, S` sources. In the pipeline, this is the `Load S11` object, for each load.
- Calibrate LNA S11 similarly using `L, O, S` sources, and apply cable loss correction to account for the extra path length. In the pipeline, this is the `Receiver S11` object.
- Gather calibration spectra (`.acq files`) for each load and apply three-position switch correction as described in Monsalve et al. (2017). To account for temperature dependent cable properties, additional loss correction is applied to the `hot` load.
- Parse the temperature logger using the time stamps in the spectra and calculate the mean temperature for each calibration load.
- Create `Load` object - an object that gives all the required information about a given load. This includes S11, spectra and the temperature.
- Calibration is performed in `CalibrationObservation` class, and it needs `Load` information of each load, `Receiver S11` and outputs the calibration co-efficients (C1, C2, T_{sin}, T_{cos}, T_{unc}) (Monsalve et al., 2017; Murray et al., 2022).
- Calibrated spectra can be obtained using antenna S11, sky spectra and the calibration co-efficients.

Note on Hot load loss correction:

EDGES-3 uses UT-141C-SP 50 ohm semi-rigid copper cable that is heated to $\sim$ 400 K to act as `hot` load. To account for the temperature dependent cable dielectric, cable loss properties and S11 properties, we apply an additional loss correction. For this analysis, we follow a similar approach as shown in Memo #392. Additional capabilities to handle different cables are also included, and can be changed in the input dictionary of `CalibrationObservation` using:

```
loss_models = {'hot_load': get_cable_loss_model(...)}
```

Comparison with Alan's pipeline

As a first order of checks of calibration pipeline, we run Alan's pipeline to calculate calibration solutions using the spectra recorded on Day `2022_316` and S11 recorded on Day `2022_319`. We replicate Alan's parameter choices as listed below. `edges-cal` related code block used for this analysis is given in Appendix A.

- c terms = 7
- w terms = 7
- S11 frequency range: 40-200 MHz
- S11 Model n-terms: 27
- Spectra frequency range: 48-198 MHz
- Calibration frequency range: 50-190 MHz
- Receiver Model n terms =10
- Cable length (for LNA path correction): 4.26 inch
- Cable dielectric percent (for LNA path correction): -1.24
- Cable loss percent: -91.5
- Hot load temperature: 393.22 K
- Ambient/Open/Short load temperature: 306.5 K

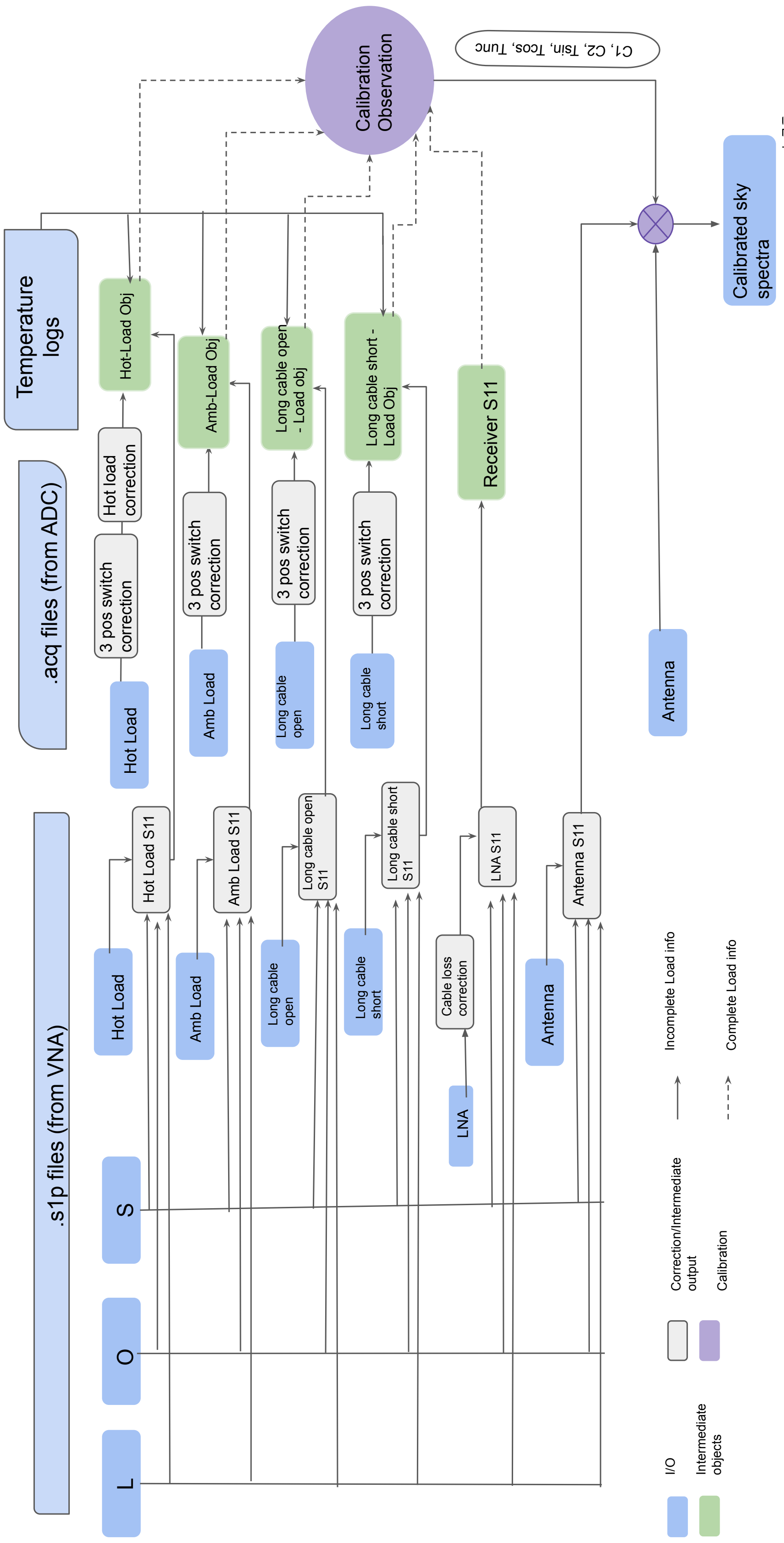

Figure 1: Flowchart showing the file structure and calibration procedure used in EDGES3 calibration.

With these parameters, we note that `edges-cal` performs comparable to that of Alan's pipeline. Following are the figure descriptions for the results:

- Figure 2 shows the comparison of calibrated S11 before modelling between Alan's output and that obtained from `edges-cal`.
- Figure 3 shows difference in calibrated S11 before modelling – order of $\sim 10^{-7}$.
- Figure 4 shows comparison of S11 models.
- Figure 5 difference in modeled S11 – order of $\sim 10^{-7}$ for `lna`, order of $\sim 10^{-6}$ for `amb, hot` loads, and order of $\sim 10^{-4}$ for long cable `open, short` loads.
- Figure 6 shows the comparison of calibration coefficients obtained from each pipeline
- Figure 7 shows the comparison of calibrated spectra

The obtained RMS of the calibrated spectra from `edges-cal` agree with those of Alan's for `amb, hot` loads, and are within 4% of Alan's output for `open` load and within 3% for `short` load. We consider this within the agreeable error at this stage of analysis.

Note on S11 modelling:

S11 can be modeled either using Fourier model or using a large term polynomial model. To account for reflections of smaller order (over the dominant sin wave), we first calculate the delay corresponding to the dominant sin wave and subtract it from the raw S11. The residual is then fit with the chosen model, and the delay corresponding to the dominant sin wave is added back to obtain the S11 model. In the previous iteration of `edges-cal`, the delay corresponding to the dominant sin wave was directly taken from Alan's output. To make the pipeline agnostic of the delay choices, this delay is now calculated using `numpy minimization routine`.

Note on comparing outputs from Alan's pipeline

To make the comparison of outputs from Alan's pipeline from that of `edges-cal` modular, there's now an added functionality called `alanmode`. This module is meant to make it easy to work with and compare to Alan's C-code. It has a few functions adopted from Alan's pipeline including `reads1p1, corrcsv, acqplot7amoon & edges3cal`. Under the hood, they use `edges-cal`, and have two useful functionalities: (a) the input parameters to these functions are named exactly the same as in Alan's code. This will make it easier in the future if we want to run something to compare to Alan, and he gives us his script – we can easily just copy the parameters over. Note that not ALL parameters that are in his scripts are supported (yet). We can add them as we need them. (b) They force you to use the same algorithms and options that Alan does, which should make it easier in the future to compare to his code.

Along with these functions, there are few other `read` functions which read output files from Alan's pipeline. Additionally, there's also a new CLI command `alancal` which is meant to emulate Alan's `docal` script – i.e. it run the complete calibration procedure, given some input files. This CLI interface uses the `alanmode.py` module under the hood. The parameters to be given to this CLI match those of Alan's naming convention. This script also outputs files exactly in the same format as Alan's C-code, making it easy to compare the outputs.

Example CLI run command for `alancal`:

```
edges-cal alancal 2022_319_14 2022 316 -res 49.8 -ps 33
-cablen 4.26 -cabloss -91.5 -cabdiel -1.24 -fstart 48
-fstop 198 -smooth 8 -tload 300 -tcal 1000 -Lh -1
-wfstart 50.0 -wfstop 190.0 -tcold 306.5 -thot 393.22 -tcab 306.5
-cfit 7 -wfit 7 -nfit3 10 -nfit2 27 --redo-s11 --redo-cal
```

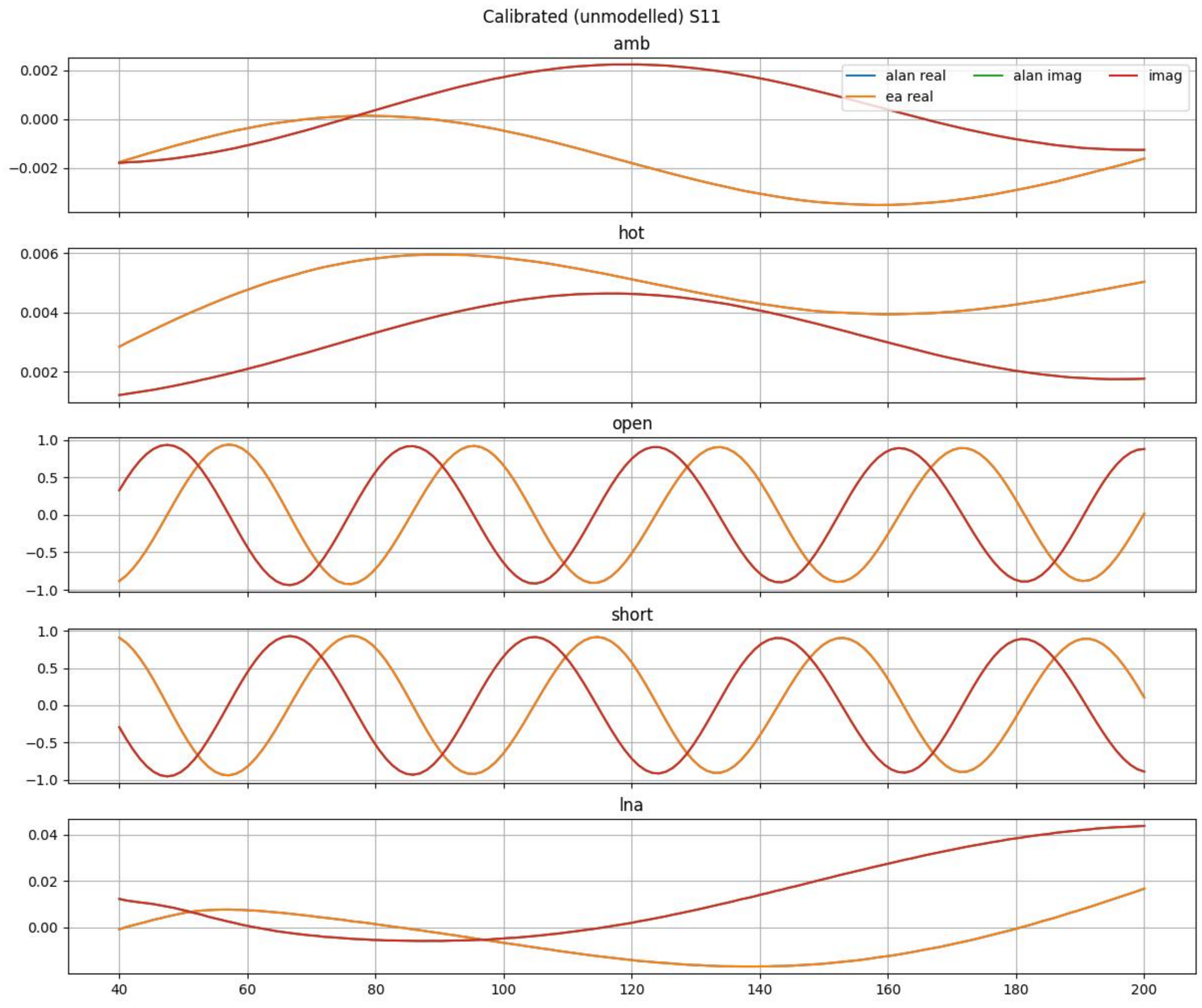

Figure 2: Comparison of unmodeled S11 obtained from Alan's pipeline and `edges-cal`.

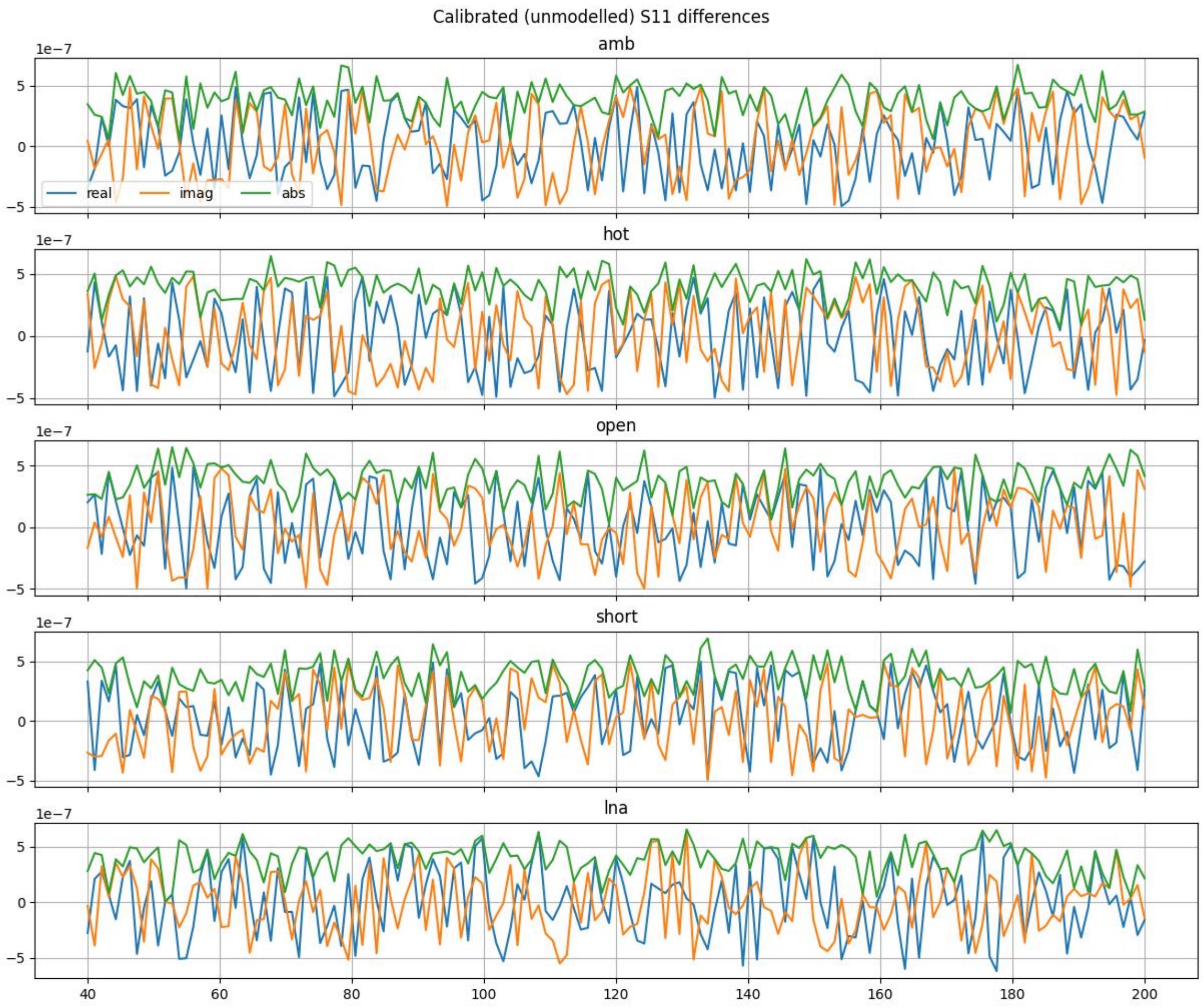

Figure 3: Difference in unmodeled S11 obtained from Alan's pipeline and `edges-cal`.

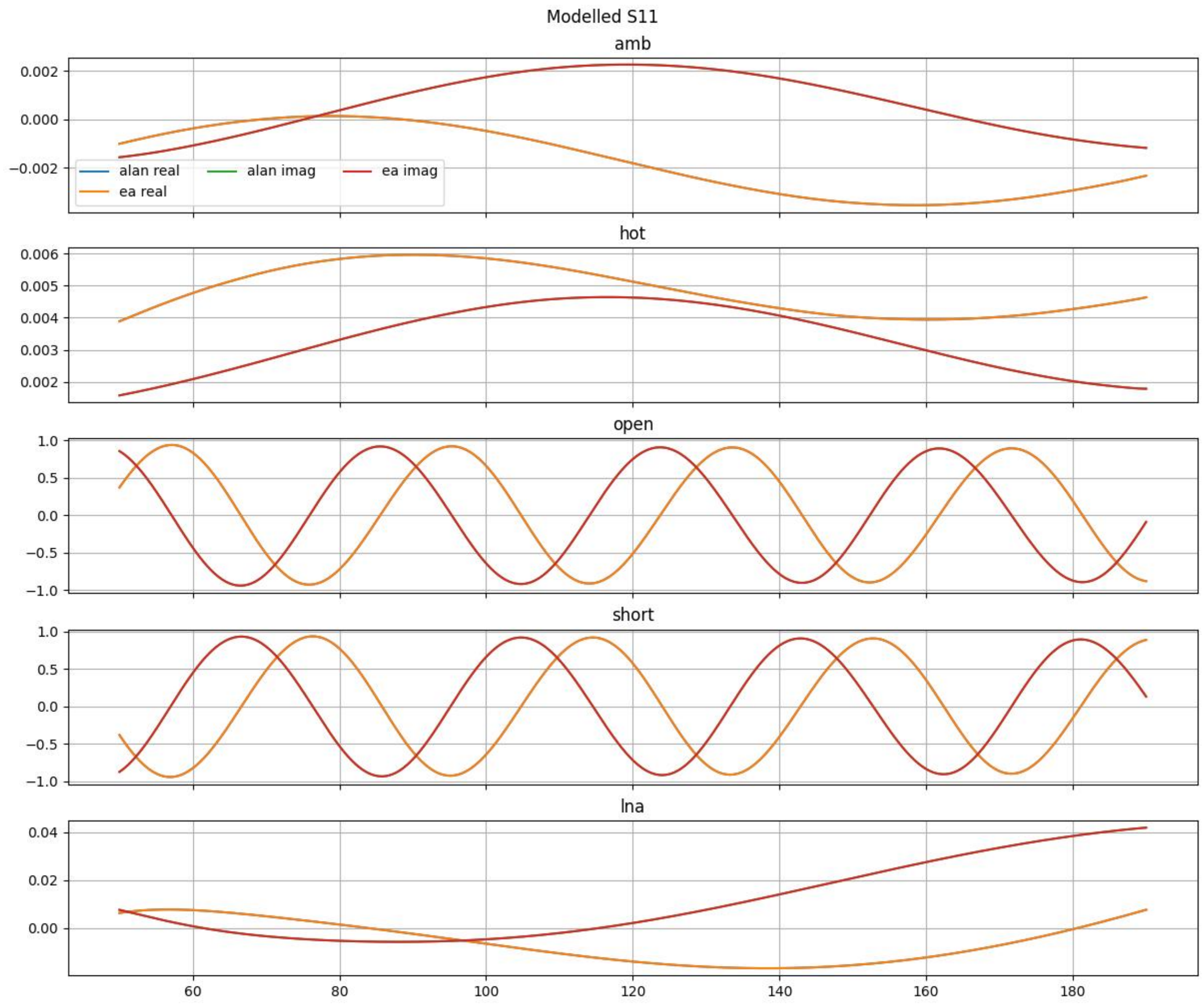

Figure 4: Comparison of modeled S11 obtained from Alan's pipeline and `edges-cal`.

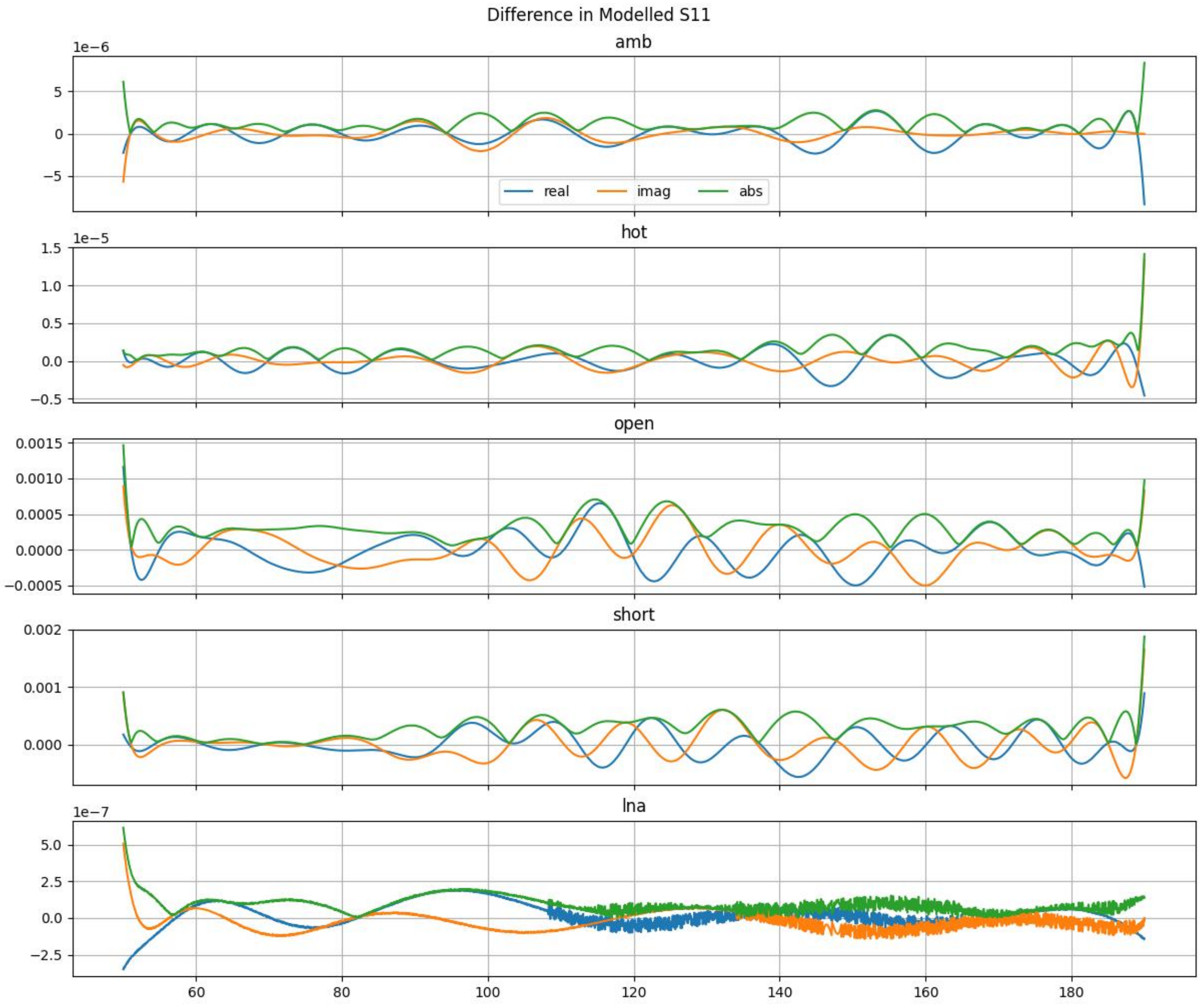

Figure 5: Difference in modeled S11 obtained from Alan's pipeline and `edges-cal`.

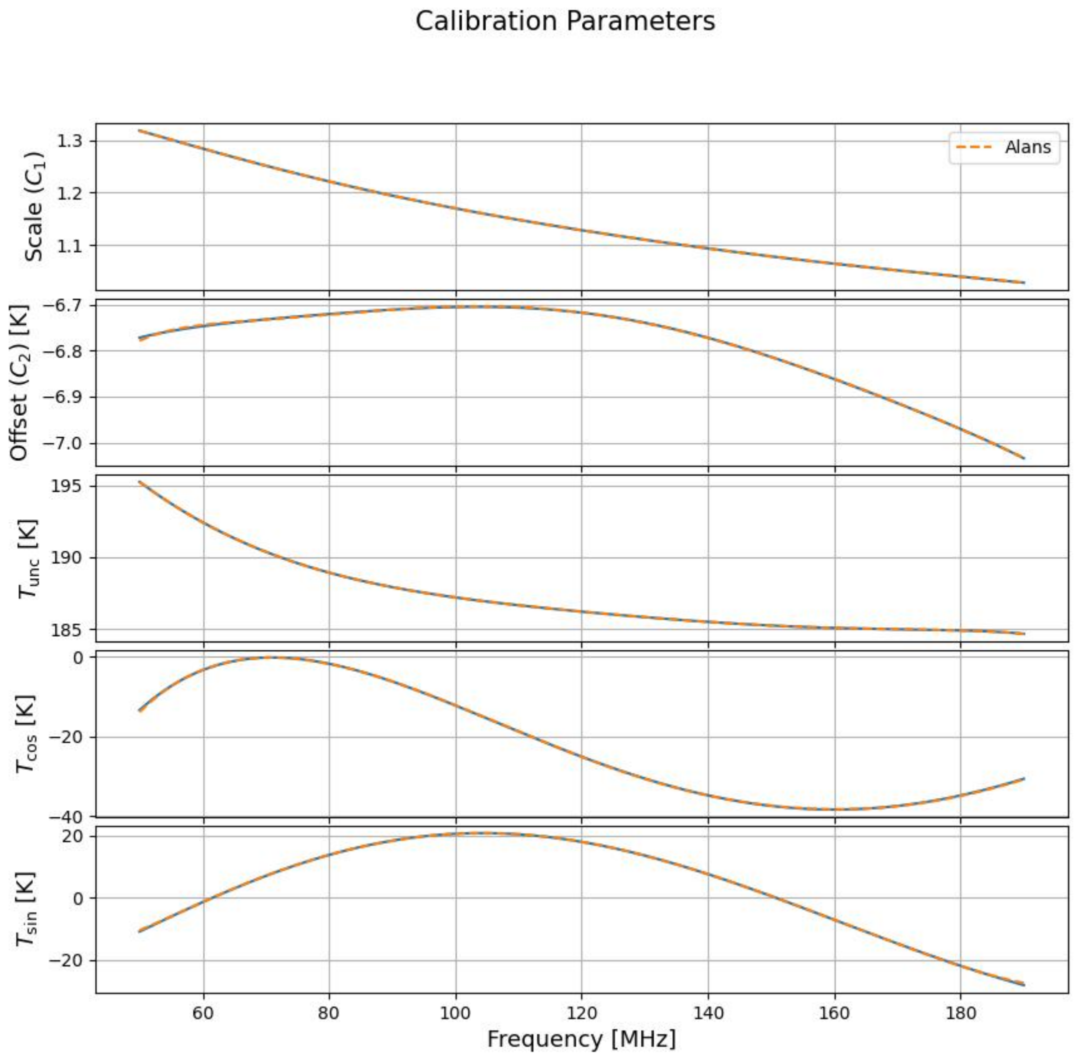

Figure 6: Comparison of calibration co-efficients obtained from Alan's pipeline and `edges-cal`.

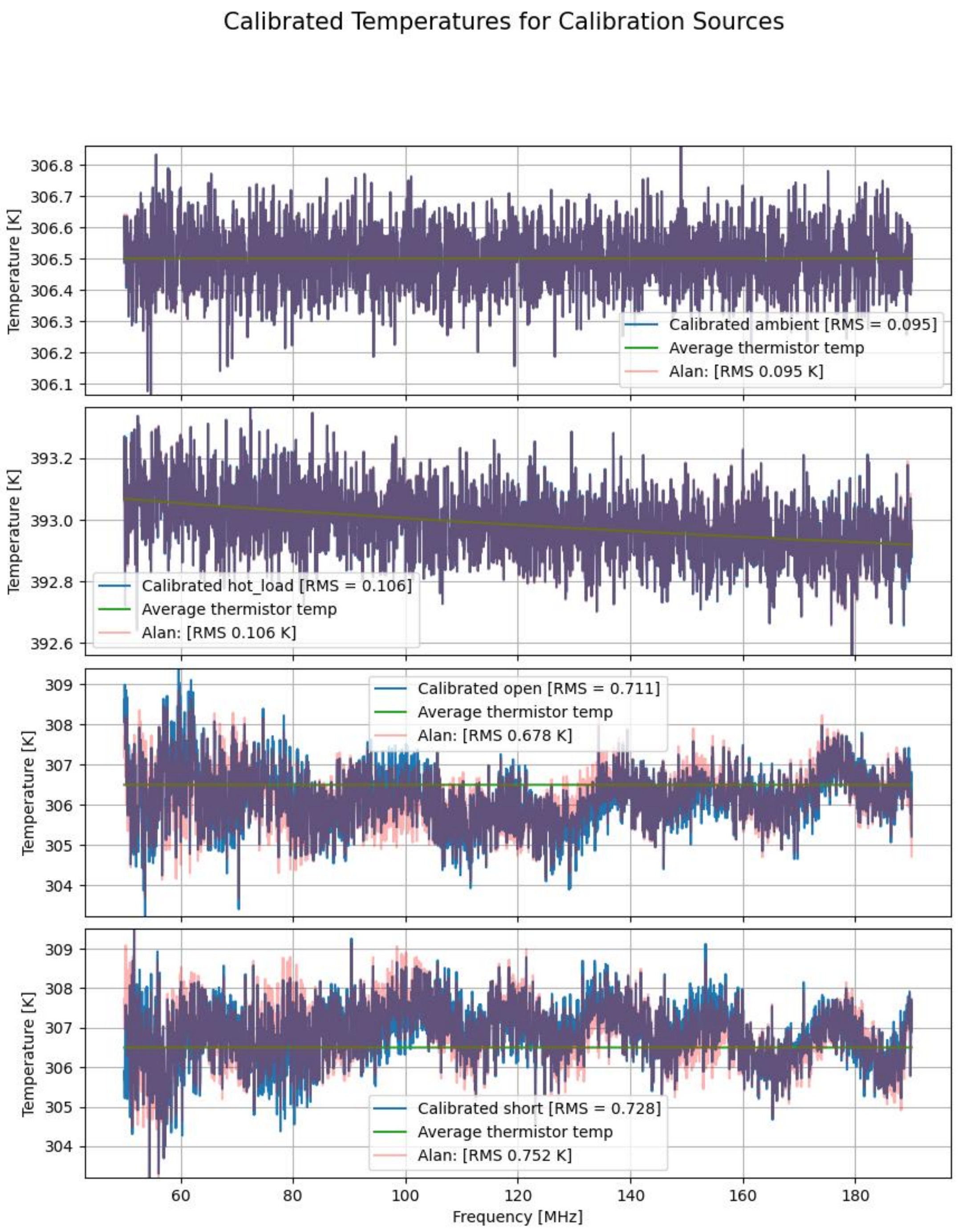

Figure 7: Comparison of calibrated spectra obtained from Alan's pipeline and `edges-cal`.

References

Monsalve R. A., Rogers A. E., Bowman J. D., Mozdzen T. J., 2017, The Astrophysical Journal, 835, 49

Murray S. G., Bowman J. D., Sims P. H., Mahesh N., Rogers A. E., Monsalve R. A., Samson T., Vydula A. K., 2022, Monthly Notices of the Royal Astronomical Society, 517, 2264

A edges-cal usage

The following code block was used to run the `edges-cal` to calculate calibration solutions using the spectra recorded on Day `2022_316` and S11 recorded on Day `2022_319`.

```
#Create an io object
calio = io3.CalibrationObservation.from_date(
    root_dir=root_dir,
    year= spec_year,
    day=spec_day,
    s11_day = s11_day
)
#calibration standards
calkit = get_calkit(AGILENT_ALAN, resistance_of_match=49.8*un.Ohm)

#calibration block:

with warnings.catch_warnings():
    warnings.simplefilter('ignore')
    # ignore all the warnings in reading temperature log

    # Note that Alan gets his hot_load temperature directly from the file,
    but not the other load temps. When using temperature logs, do not define


    calobs = CalibrationObservation.from_edges3(
        io_obj=calio,
        f_low=50*un.MHz,
        f_high=190*un.MHz,
        freq_bin_size=8,
        spectrum_kwargs={
            "default": {
```

```
            "t_load_ns": 1000,
            "t_load": 300,
            "temperature": 306.5,
            'cache_dir': 'spec-cache',
            'frequency_smoothing': 'gauss',
            'allow_closest_time': True,
            'f_low': 48*un.MHz,
            'f_high': 198*un.MHz,
        },
        'hot_load': {'temperature': 393.22}
    },
    s11_kwargs={
        "default": {
            'model_type': mdl.Fourier,
            # Use Fourier model for all loads.
            'complex_model_type': mdl.ComplexRealImagModel,
            # Fit on real/imag instead of abs/phase
            'model_transform': mdl.ZerotooneTransform(range=(40, 200)),
            # Alan uses (0, 1) range of freq.
            'model_kwargs': {'period': 1.5},
            # Alan uses 2pi/1.5 in his cos/sin terms
            'n_terms': 55,
            # corresponds to nfit2 in alans pipeline (Alan uses 27)
        },

    },
    receiver_kwargs = {
        'n_terms': 10,          # Use 11 terms
        'model_type': 'polynomial',
        # Alan used Fourier series on receiver
        'model_transform': mdl.Log10Transform(scale=120),
        'complex_model_type': mdl.ComplexRealImagModel,
        'calkit': calkit,
        'cable_length': (4.26*un.imperial.inch).to("m"),
        'cable_dielectric_percent': -1.24,
        'cable_loss_percent': -91.5,
    },
    restrict_s11_model_freqs=True,
    cterms=7,
    wterms=7
)
```