# compare-alanmode-output

August 8, 2024

## 1 Compare AlanMode to C code

This memo notebook summarizes the results of careful matching of the EDGES-3 calibration with `edges-cal` to the output of Alan's C-code. The main result of this memo is that `edges-cal` **can now match the C-code to better than** $10^{-6}$ **precision** in the calibration parameters ($C_1$, $C_2$, $T_{unc}$, $T_{cos}$ and $T_{sin}$).

### 1.1 The Data

The data for this comparison are:

- Calibration load S11s measured on `2022_319_14`,
- and Spectra measured on `2022_316`,

which are calibration observations chosen by Alan to calibrate observations on days 2022:300-310.

### 1.2 The C-Code

The C-code used to produce Alan's results here was originally provided by Alan directly via email. Though several `.c` files are in the overall package, only some of these are used in the calibration here:

- `reads1p1.c` – for reading raw VNA output, performing calibration, and writing back out at raw frequency resolution
- `acqplot7amoon.c` – for reading spectra (ACQ files), averaging them over time, and writing out the smoothed, averaged, uncalibrated temperatures, $T^*_{src}$.
- `edges3c.c` – for reading the calibrated S11's, modelling them with smooth functions and evaluating them on the finer frequency resolution of the spectra, then doing the iterative calibration routine to find the calibration parameters.
- `docal_316test` – a C-shell script that puts all of those operations together to do the calibration.

Some modifications were made to the code that was directly provided. In almost all cases, these were simply to add outputs at various stages of the processing to make it easier to perform the comparison. We summarize these changes here:

- In `reads1p1.c` the output format of the S11 `.csv` files was modified simply to increase the precision: all values were changed from `%f` to `%1.16e`. Since very small differences in the raw S11 values can affect their subsequent modelling, this was found necessary in order to properly compare the final results, since in `edges-cal`, the output values from the step equivalent to `reads1p1` are passed *in-memory* to the modelling step, which means they were much higher precision than the `%f` format would allow for the C-code.

- In `acqplot7amoon.c` a new file output was added: the `flagfile.txt` which simply tracks which integrations are flagged. In the case of this calibration, no integrations are flagged, so this is somewhat unnecessary (but it was used to make this check).
- In `edges3c.c`:
  - We added the `s11_modelled.txt` output, which simply writes all the calibration load S11's (plus the LNA) *after* modelling, at the resolution of the spectra. These are written out with format `%1.12e` in order to get good precision for their comparison.
  - Added a similar `ant_s11_modelled.txt` output.
  - Modified the output precision of the `specal.txt` to be `%1.12e`. This was not because we think we need that precision for calibration, but because it aids in comparison here.
  - The value of `PI` defined in the code was updated to add 5 more digits of precision. This was done since the S11 models use `PI` in their basis functions, and small differences here can propagate to larger differences down the line. We show the effect of this change in this memo. XXXXX.
  - **Fixed a small bug** in the `fittp()` function, in which the lowest frequency that contributes to the fit was being taken as the *second* lowest un-flagged datum, rather than the lowest. We note that all previous results from Alan's code will have this bug present in them, and we talk about this further in this memo.

The full code actually used in this comparison is available on github at https://github.com/edges-collab/edges3-day300-310-test.

## 1.3 The Python Code

In this comparison we use the new `alancal` CLI command in the `edges-cal` package. In particular we use the `6.2.6.dev166+ge2c5fbb` version of `edges-cal`. After showing that the basic function compares well to the C-code, we will also discuss some of the changes that were required to get there, and show comparisons with and without these changes. For these, some ad-hoc changes were required to the code, but these will be discussed explicitly.

The basic command we used for this comparison is:

```
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 --no-plot
```

## 1.4 Comparison of Outputs from the C-code with Fixes

Before we continue, we discuss the two fixes made to the C-code, and the impact they make. Both of these fixes were made to the `edges3c.c` module. One of them (the `fittp` fix) exclusively affects modelling, which in the case of calibration, only affects the modelled S11s. The increase of precision in `PI` affects several components of `edges3c.c`, but only to a low level, and probably the S11's most of all.

More explicitly, the `fittp()` function in `edges3.c` performs the model-fitting for the S11's (for all four loads and the LNA). Prior to actually fitting the data, the function determines the first and last frequency bin to use in the fit – taking into account the weights (which are primarily determined by the arguments `-wfstart` and `wfstop`). There was a small bug in which the *second* bin was used as the lowest bin instead of the first. All plots below labeled "With `fittp()` fix" have this fixed in the C-code. All runs with `edges-cal` throughout the memo have the correct behaviour.

[63]:
```
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
```

[2]:
```
from edges_cal.alanmode import read_s11_csv, read_spec_txt, read_specal
```

[16]:
```
loads = ['amb', 'hot', 'open', 'short']
```

[12]:
```
def read_s11m(pth):
    _s11m = np.genfromtxt(pth, comments="#", names=True)
    s11m = {}
    for load in loads + ['lna']:
        s11m[load] = _s11m[f"{load}_real"] + 1j*_s11m[f"{load}_imag"]
    s11m['freq'] = _s11m['freq']
    return s11m
```

First, let's have a look at the differences made to the S11 models themselves. We ran `edges3c` for all four combinations of the fixes, and saved the outputs.

[13]:
```
s11m_cases = {
    'With fittp() fix; default PI': read_s11m('alans-code/
↪with_fittp_fix_no_pi_fix/s11_modelled.txt'),
    'With fittp() fix; hi-prec PI': read_s11m('alans-code/
↪with_fittp_fix_and_pi_fix/s11_modelled.txt'),
    'No fittp() fix; default PI': read_s11m('alans-code/no_fittp_fix_no_pi_fix/
↪s11_modelled.txt'),
    'No fittp() fix; hi-prec PI': read_s11m('alans-code/
↪no_fittp_fix_with_pi_fix/s11_modelled.txt')
}
```

[66]:
```
default_case = s11m_cases['No fittp() fix; default PI']

fig, ax = plt.subplots(len(loads) + 1, 1, sharex=True, constrained_layout=True,␣
↪figsize=(12, 10))

for j, (case, s11) in enumerate(s11m_cases.items()):
    if case == 'No fittp() fix; default PI':
        continue

    for i, load in enumerate(loads + ['lna']):
        ax[i].plot(s11['freq'], np.abs(s11[load] - default_case[load]),␣
 ↪label=case, ls=['-', '--', ':', '-.'][j])
        ax[i].set_title(load)
        ax[i].set_yscale('log')
        ax[i].set_ylabel(r"$|S_{11} - S_{11}^{\rm original}|$")
ax[0].legend(ncols=4);

ax[-1].set_xlabel("Frequency [MHz]");
fig.suptitle("Comparison of S11 between changes to Alan's Code")
```

```
[66]: Text(0.5, 0.98, "Comparison of S11 between changes to Alan's Code")
```

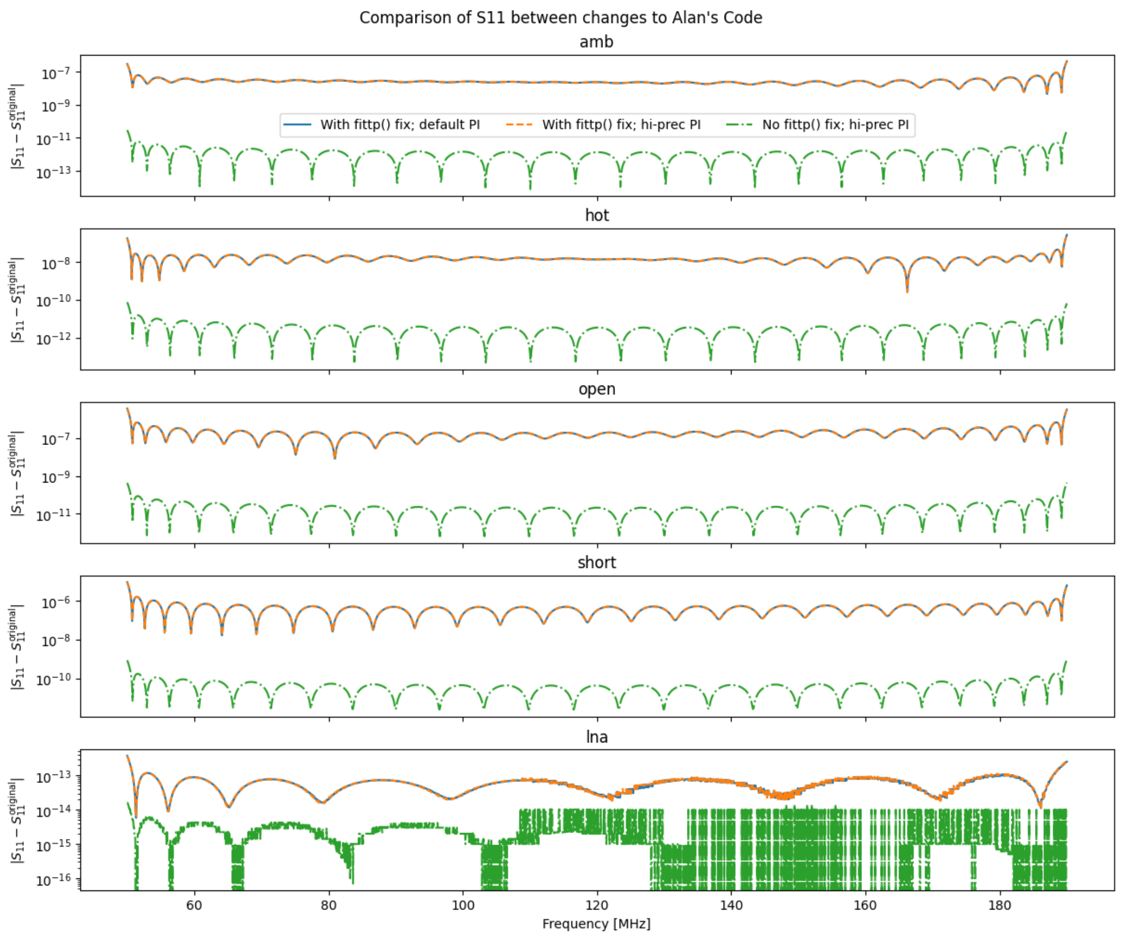

**Figure 1** | Comparison of calibrated S11's from Alan's pipeline after smoothing with a linear model. Each panel represents a different input source, including the LNA. Four cases were run: with and without the fix to `fittp()` (see text), and with and without an increase in precision of

PI. The values shown are the absolute differences of the $S_{11}$ in each case compared to the original code, i.e. without the fix to fittp() and with the original precision of PI. The figure indicates that the change to PI (blue and orange lines) is much *less* significant than the change to fittp() (green line). The files were written with the format %1.12e and therefore 'spiky' changes in the green line for the LNA (bottom panel) are simply precision error on top of the inherent (very small) error.

From this, we see that making the fittp() fix is reasonably important – it has an effect at the $\sim 10^{-7}$ level for the source S11s. The increase in precision of PI is not very important. To confirm this, let us look at the final calibration solutions.

[79]:
```
specal_cases = {
    'With fittp() fix; default PI': read_specal('alans-code/
↪with_fittp_fix_no_pi_fix/specal_316test.txt'),
    'With fittp() fix; hi-prec PI': read_specal('alans-code/
↪with_fittp_fix_and_pi_fix/specal_316test.txt'),
    'No fittp() fix; default PI': read_specal('alans-code/
↪no_fittp_fix_no_pi_fix/specal_316test.txt'),
    'No fittp() fix; hi-prec PI': read_specal('alans-code/
↪no_fittp_fix_with_pi_fix/specal_316test.txt')
}
```

[82]:
```
def plot_calcoeff_cases(cases, comp_case, fig=None, ax=None):
    if fig is None:
        fig, ax = plt.subplots(len(comp_case.dtype.names)-4, 1, sharex=True,␣
↪constrained_layout=True, figsize=(12, 15))

    for ic, (case, cal) in enumerate(cases.items()):
        j = 0

        for i, name in enumerate(comp_case.dtype.names):
            if name in ('freq', 'weight') or name.startswith("s11lna"):
                continue
            ax[j].plot(comp_case['freq'], comp_case[name] - cal[name],␣
↪label=case, ls=(ic, (5, 4)))
            ax[j].set_title(name)
            ax[j].set_yscale('symlog', linthresh=1e-6)
            ax[j].set_ylabel(f"Diff. w.r.t original" + ("[K]" if name.
↪startswith("T") else ""))

            j += 1

    ax[0].legend()
    ax[-1].set_xlabel("Frequency [MHz]")
    fig.suptitle("Comparison of Receiver Calibration Coefficients between␣
↪C-Code changes")
    return fig, ax
```

```
[83]: plot_calcoeff_cases({k: v for k,v in specal_cases.items() if k!='No fittp() fix;
       ↪ default PI'}, specal_cases['No fittp() fix; default PI']);
```

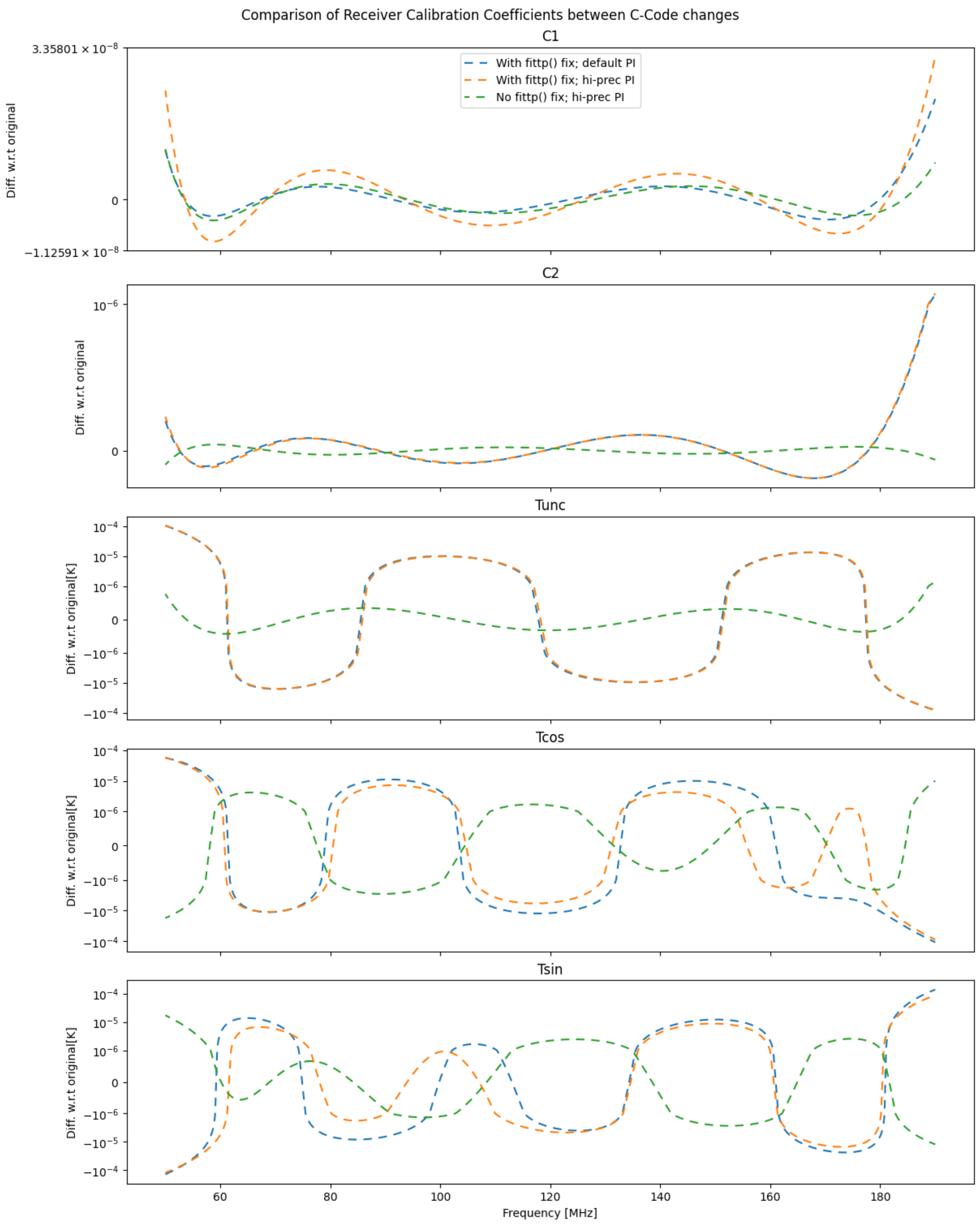

**Figure 2 |** Comparison of calibration coefficients produced by the C-code between the changes in the C-code we tested in this memo. See **Fig. 1** for details on the changes, and note that the colors in this plot are the same as **Fig 1**. Note that fixing `fittp()` has a relatively larger effect overall, but none of the changes makes a very big impact.

Here, we see that actually neither of the changes will have a very large impact – the scaling, $C_1$ stays the same to less than $10^{-8}$, and while the `fittp` fix does change the noise wave parameters to a much larger level ($\sim 10^{-4}$), these are additive and therefore unlikely to be of much effect in the final results.

Nevertheless, it is good to keep these things in mind for the rest of the analysis. For the rest, we will use the version *with* the `fittp` fix (since we cannot and should not emulate this with the `edges-cal` package) and *without* the `PI` enhancement (since this is not a big effect, and is unlikely to be updated in future C-code versions).

## 1.5 Demonstration of Matching Calibration

Now we show that `edges-cal` and the C-code match to very good precision, when the parameters are chosen accordingly. This comparison uses the above call to `edges-cal alancal` precisely.

### 1.5.1 Calibrated (unmodelled) S11s

[94]:
```
alans11 = {}
ours11  = {}
for load in loads + ['lna']:
    s11freq, alans11[load] = read_s11_csv(f"alans-code/s11{load}.csv")
    _, ours11[load] = read_s11_csv(f"alanmode/s11{load}.csv")
```

[95]:
```
fig, ax = plt.subplots(len(alans11), 1, sharex=True, constrained_layout=True,␣
 ↪figsize=(12, 10))
for i, load in enumerate(alans11):
    ax[i].plot(s11freq, (alans11[load].real - ours11[load].real)/np.
 ↪abs(alans11[load]), label='real')
    ax[i].plot(s11freq, (alans11[load].imag - ours11[load].imag)/np.
 ↪abs(alans11[load]), label='imag')
    ax[i].plot(s11freq, np.abs(alans11[load] - ours11[load])/np.
 ↪abs(alans11[load]), label='abs')
    ax[i].set_title(load)
    ax[i].set_ylabel("Difference/|S11 Alan|")

ax[0].legend(ncols=3)
ax[-1].set_xlabel("Frequency [MHz]")

fig.suptitle("Calibrated (unmodelled) S11 differences");
```

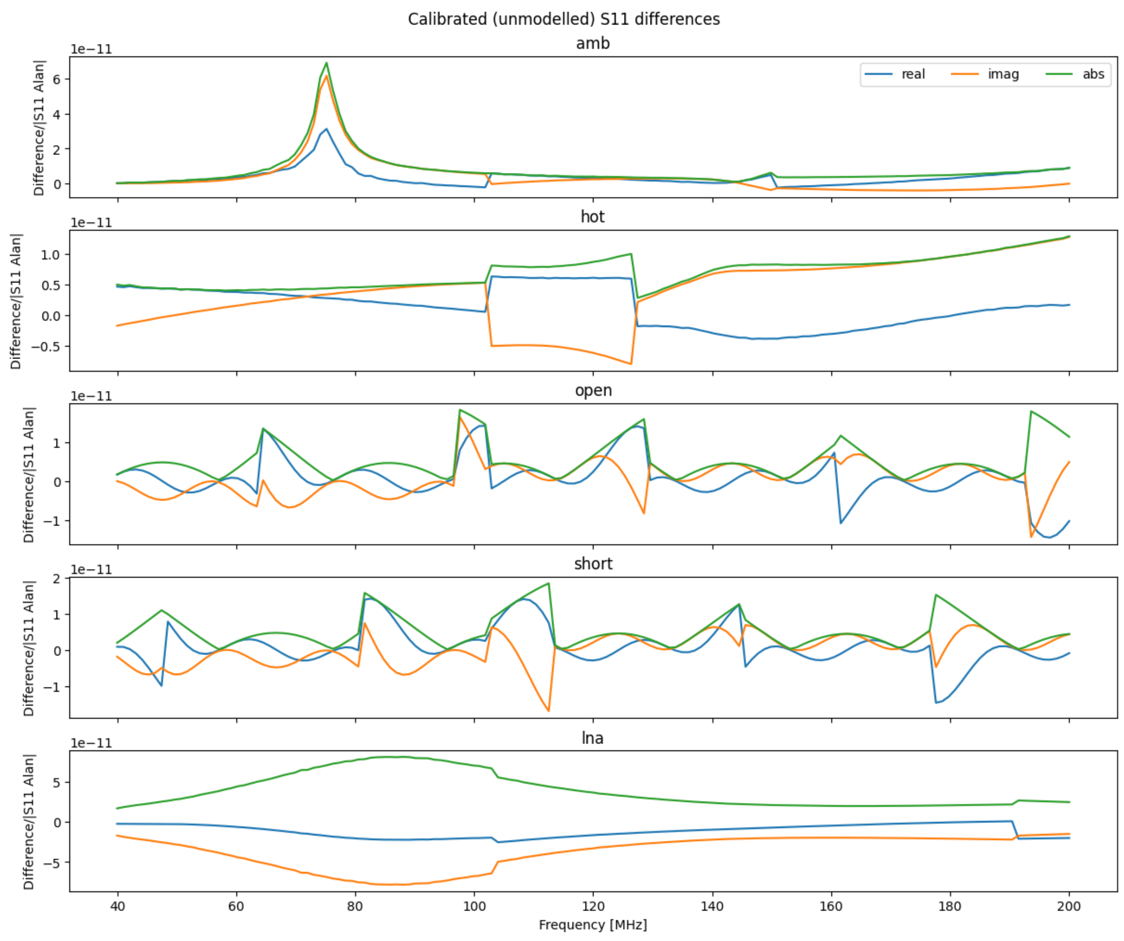

**Figure 3** | Comparison of calibrated $S_{11}$ for different loads, at raw frequency resolution. These $S_{11}$'s have been calibrated via OSL standards, but they have not been smoothed. The comparison is between the calibration done in `edges-cal` vs. the C-code (using `reads1p.c`). The values plotted are the *fractional* differences, as compared to the absolute value of the $S_{11}$ calibrated with the C-code. While the fractional residuals are small, they are not consistent with precision error in the output files, which is at the level of $10^{-16}$. This may be because of differing values of constants such as the speed of light in the calibration codes. It is also unclear why the residuals have "step-like" behaviours.

### 1.5.2 Averaged Spectra

[34]:
```
alanspec = {}
ourspec = {}
for load in loads:
    data = read_spec_txt(f"alans-code/sp{load}.txt")
    spfreq = data['freq']
    alanspec[load] = data['spectra']
```

```python
    data = read_spec_txt(f"alanmode/sp{load}.txt")
    ourspec[load] = data['spectra']
    assert np.allclose(spfreq, data['freq'])
```

[97]:
```
!head alans-code/spamb.txt
```

```
   48.004150   299.973456    1 450 // temp.acq 2022:316:08:59:25  11.205  24.000
12.674  14.143  11.205 fmpwr   0.000
   48.052979   299.869072    1
   48.101807   299.745895    1
   48.150635   299.712125    1
   48.199463   299.803907    1
   48.248291   299.831184    1
   48.297119   299.893957    1
   48.345947   299.810161    1
   48.394775   299.896787    1
   48.443604   299.956277    1
```

[96]:
```python
fig, ax = plt.subplots(len(alanspec), 1, sharex=True, constrained_layout=True,␣
 ↪figsize=(12, 10))
for i, load in enumerate(alanspec):
    ax[i].plot(spfreq[10:-10], alanspec[load][10:-10] - ourspec[load][10:-10])
    ax[i].set_title(load)
    ax[i].set_ylabel("Difference [K]")

fig.suptitle("Raw spectra differences")
ax[-1].set_xlabel("Frequency [MHz]");
```

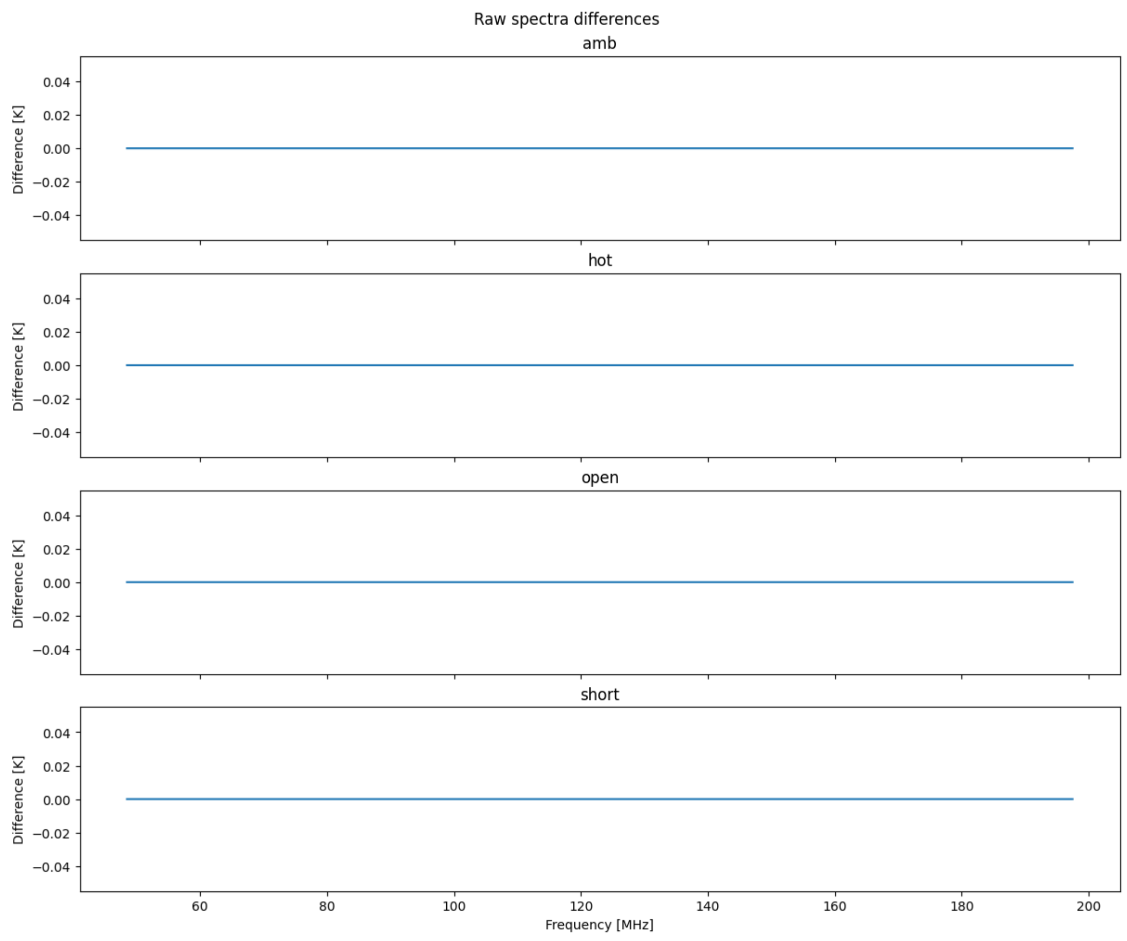

**Figure 4** | Comparison of averaged spectra between the `edges-cal` and the C-code. Each panel is a different input source. The difference is exactly zero to the precision of the output files (which is $10^{-6}$ K). Note that since these files are written out and then read back in at the next stage of processing, any potential differences between the codes below this level are removed here.

### 1.5.3 Modelled S11s

[41]:
```
ours11m = read_s11m("alanmode/correct/s11_modelled.txt")
```

[101]:
```
default_case = ours11m

fig, ax = plt.subplots(len(loads) + 1, 1, sharex=True, constrained_layout=True,␣
  ↪figsize=(12, 10))

for j, (case, s11) in enumerate(s11m_cases.items()):
    for i, load in enumerate(loads + ['lna']):
        ax[i].plot(s11['freq'], np.abs(s11[load] - default_case[load])/np.
↪abs(default_case[load]), label=case, ls=['-', '--'][j%2])
        ax[i].set_title(load)
        ax[i].set_yscale('log')
        ax[i].set_ylabel("| Diff | / |S11 edges-cal|")

ax[0].legend(ncols=4)
ax[-1].set_xlabel("Frequency [MHz]")
fig.suptitle("Absolute Difference in edges-cal Modelled S11 versus C-code")
```

```
[101]: Text(0.5, 0.98, 'Absolute Difference in edges-cal Modelled S11 versus C-code')
```

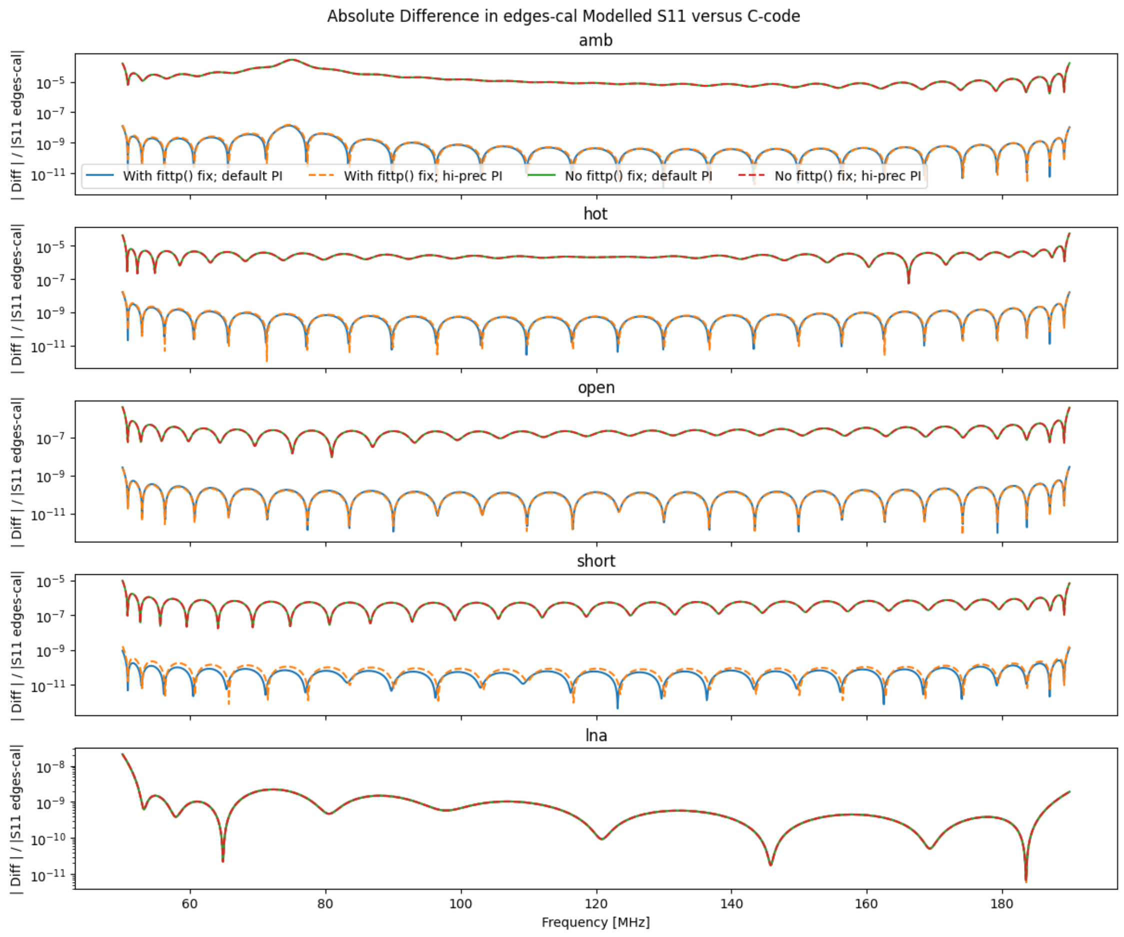

**Figure 5** | Comparison of $S_{11}$ for each input source (and receiver) after linear modelling. Each color represents a different modification to the C-code (described above). All cases are compared to `edges-cal` which has high-precision `PI` and corresponds to `fittp()` being corrected (i.e. we expect it to be closest to the orange line). It is indeed closest to the orange and blue, for which it has an accuracy of about 1 part in $10^9$.

### 1.5.4 Calibration Coefficients

```
[105]: ourcal = read_specal("alanmode/correct/specal.txt")
```

```
[107]: plot_calcoeff_cases(specal_cases, ourcal);
```

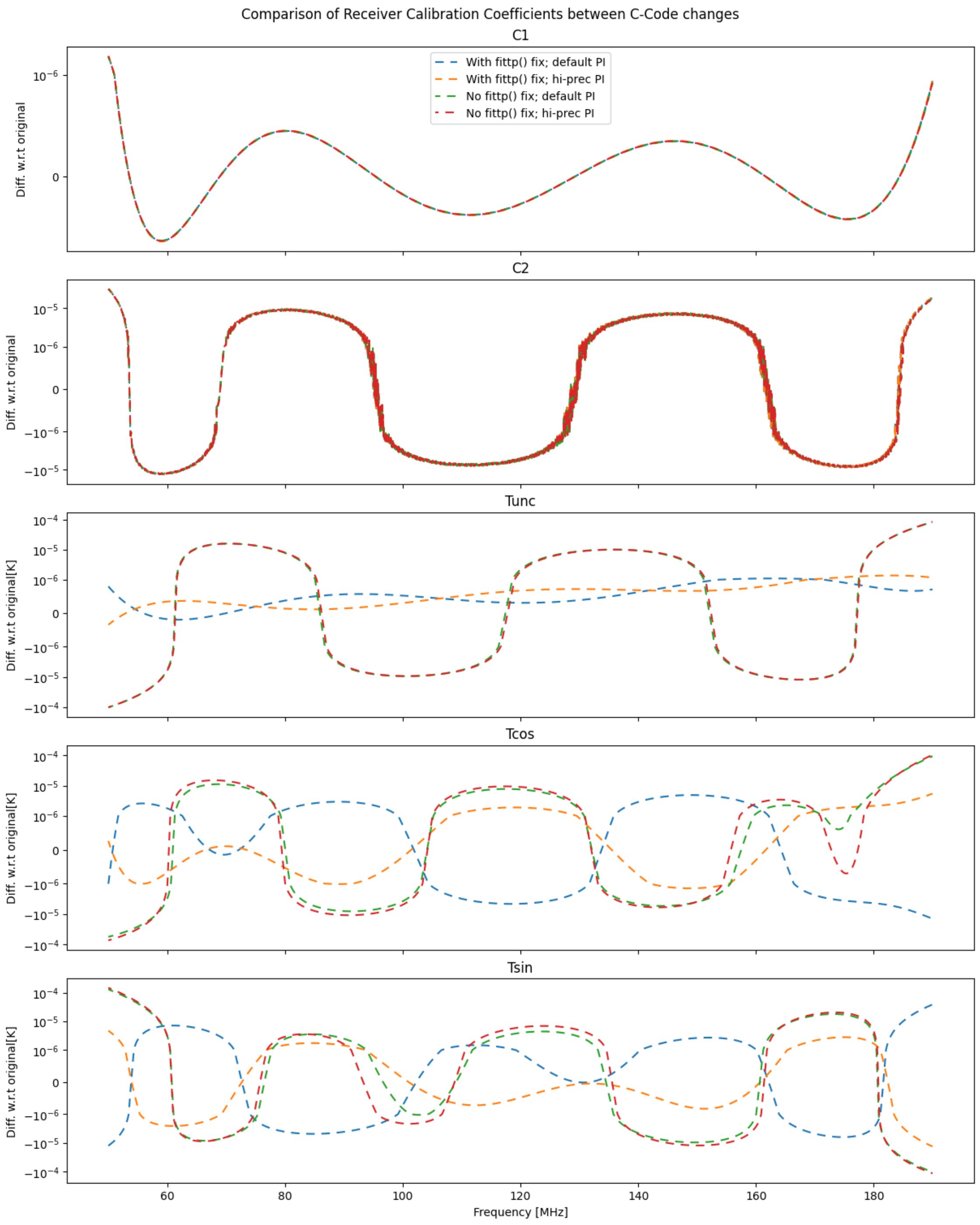

**Figure 6** | The same as **Fig 2** except that here the comparison calibration is done with `edges-cal` and the different colors are the different C-code cases. We expect the *orange* to be closest to `edges-cal`, and we find that to be true. All coefficients are very close to the C-code in all cases, however $C_1$ is the most important, as it is multiplicative in the final calibration – it is accurate to better than $10^{-6}$.

### 1.6 Noise-Wave Calibration Loop Improvements

In the process of this careful comparison, several updates were made to the *calibration loop* in which the scale, offset and noise-wave parameters are determined within `edges-cal`. In particular, these were (labels of following plots precede the description):

- `lstsq fit`: Changing the default `lstsq` matrix-inverse solving method used to solve for the linear parameters to a custom-writting method `_alan_qrd`, which is a port of the C function in Alan's code.
- `hot load loss on temp`: Changing the hot-load loss from being applied to the thermistor temperature, to instead be applied to the spectrum (in reverse).
- `no delay subtraction`: Adding a substraction of a fitted delay to the noise-wave correlated data.
- `4 iterations (not 8)`: Using 8 iterations instead of the default 4.
- `smooth sca/off in loop`: Switching from modelling and smoothing $C_1$ and $C_2$ within the calibration loop to instead only modelling them at the end of the loop.
- `standard poly`: Switching from using a polynomial with integer-order terms, to half-integer order terms for $C_1$ and $C_2$.

In particular, for the `standard poly` case, we *were* using the polynomial

$$C = \sum_i a_i \left(\frac{\nu}{\nu_c}\right)^i,$$

but have switched, in line with the C-code, to

$$C = \sum_i a_i \left(\frac{\nu}{\nu_c}\right)^{i/2}.$$

We show the effect of each of these changes in this section. Since we are focusing on the changes arising from these aspects of the *calibration loop*, rather than the S11 modelling (which feeds into that loop), here we show results of *injecting* the exact S11 from the C-code into `edges-cal` to perform the calibration loop, showing the difference with respect to the results from the loop in the C-code.

Note that we performed this injection with the version of the C-code *without* the `fittp` fix and `PI` fix, nevertheless these differences remain qualitatively the same.

[108]:
```
cases = {
    "lstsq fit": 'default-fitmethod',
    "hot load loss on temp": 'loss-on-temp',
    "no delay subtraction": "nodelay",
    "4 iterations (not 8)": 'nter4',
    "smooth sca/off in loop": "smooth-in-loop",
    "standard poly": "wrong-poly",
}

allspec = {case: read_specal(f"alanmode/{fld}/specal.txt") for case, fld in␣
↪cases.items()}
```

[111]:
```
fig, ax = plot_calcoeff_cases(allspec, specal_cases["With fittp() fix; hi-prec␣
↪PI"])
plot_calcoeff_cases({"correct": ourcal}, specal_cases["With fittp() fix;␣
↪hi-prec PI"], fig=fig, ax=ax);
```

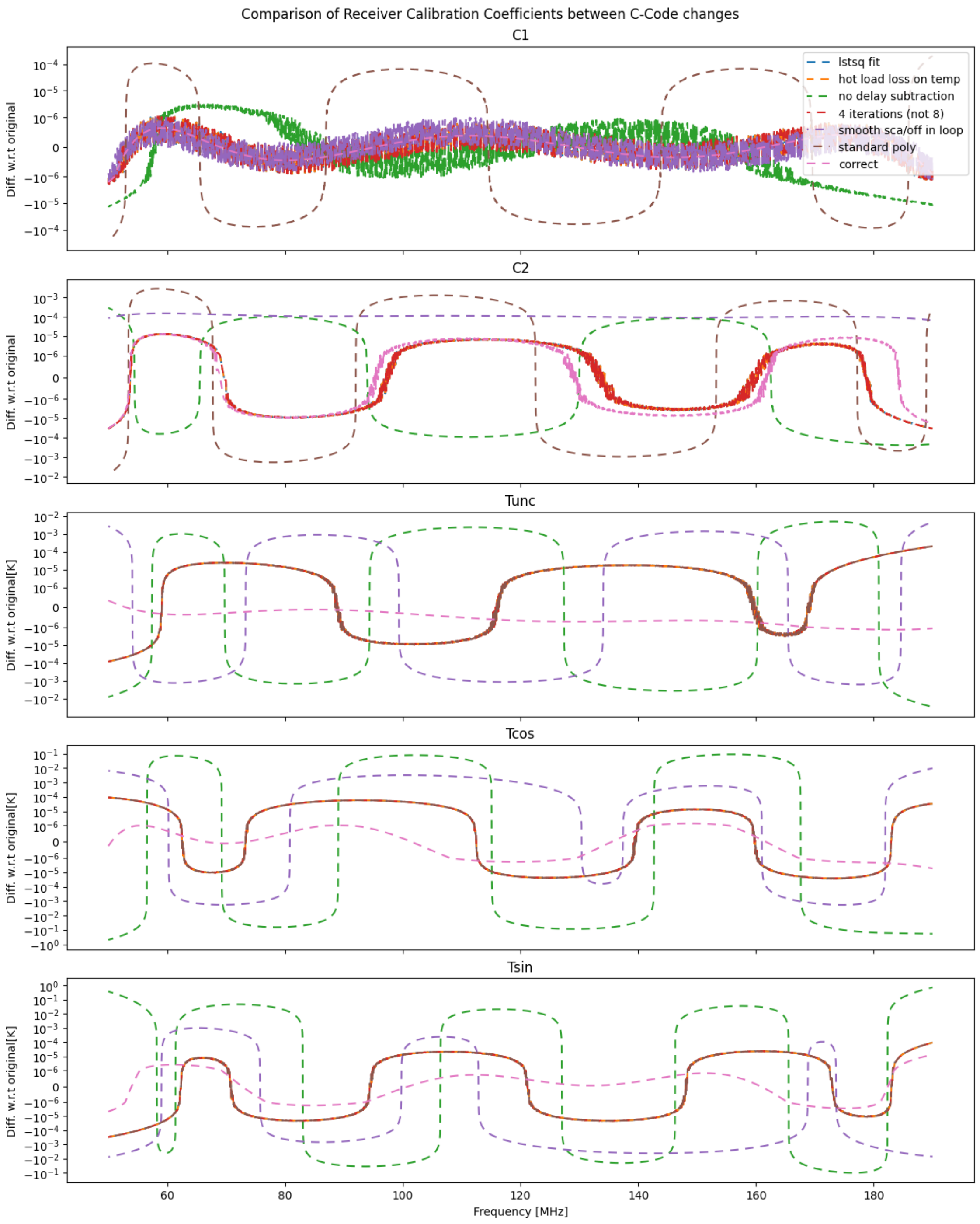

**Figure 7 |** Comparison of calibration coefficients from different iterations of `edges-cal` with that from the C-code. Different colors represent calibrations performed with `edges-cal` with various slight differences from the C-code (see above for details). The case they are compared to is the C-code with the fix to `fittp()` and high-precision $\pi$.

The key takeaways from this plot are:

1. The biggest differences to the NW params are from not subtractiving a delay, and secondarily from smoothing the scale/offset within the loop.
2. The biggest difference to the scale (C1) is fitting with a different polynomial.
3. This is the same for the offset (C2), except that also the smoothing inside the loop and delay subtraction play a notable role.
4. Three modifications have almost no effect:
    1. the number of iterations (4 instead of 8 )
    2. the method of fitting (lstsq instead of alan-qrd)
    3. using the loss on the temperature instead of the spectrum