22 October 2020

TO: Reprocessing File
FROM: Al Cooper
SUBJECT: A procedure and script to correct temperature measurements for sensor time response

# 1 The Governing Equations

The basis for this script is documented by Cooper et al. [2020]. The objective in this note is to provide specific information regarding how that can be implemented in a second-pass processing script. It is hoped that this note will be useful to anyone needing to use and maintain the processing script, which is called "CorrectTemperature.R".

## 1.1 Correction Method #1

The governing differential equations involve two unknowns, the actual recovery temperature $T_r(t)$, and the temperature of the supporting structure $T_s(t)$. As given inCooper et al. [2020], they are:

$$\frac{dT_s(t)}{dt} = \frac{T_r(t) - T_s(t)}{\tau_2} \tag{1}$$

$$\frac{dT_m(t)}{dt} = \frac{a(T_r(t) - T_m(t)) + (1-a)(T_s(t) - T_m(t))}{\tau_1} \tag{2}$$

where $T_s(t)$ is the temperature of the support, $T_m(t)$ the measured temperature of the sensing wire, and $T_r(t)$ the true recovery temperature that is the measurand. For heat transfer to or from the wire, the parameter $a$ then represents the fraction of the heat transferred by the air while $(1-a)$ is transferred to or from the support. The wire responds to the combined transfers of heat with characteristic time constant $\tau_1$ while the support structure responds to the air temperature more slowly, with time constant $\tau_2$.

It is possible to eliminate $T_r(t)$ from the simultaneous equations: The second equation, solved for $T_r(t)$, is

$$T_r(t) = \frac{1}{a}\left(\tau_1 \frac{dT_m(t)}{dt} + T_m(t) - (1-a)T_s(t)\right) \tag{3}$$

Substituting this into the first equation eliminates $T_r$:

$$\frac{dT_s(t)}{dt} = \frac{\frac{1}{a}\left\{\tau_1 \frac{dT_m(t)}{dt} + T_m(t) - (1-a)T_s(t)\right\} - T_s(t)}{\tau_2} \tag{4}$$

Because the measured temperature $T_m(t)$ is known, this can be integrated from an assumed initial value $T_s(0)$ to find the temperature of the support, $T_s(t)$. Then the true recovery temperature $T_r(t)$

can be obtained by solving (3) without further integration. The only choices needed are the numerical method used to find the derivative $dT_m/dt$ (here centered fourth-order) and the integration method applied to (4), here fourth-order Runge-Kutta integration with Cash-Karp (Cash and Karp [1990]) adjustment of the step size. If the values of the parameters ($a$, $\tau_1$, $\tau_2$) are known, this approach provides a method of correcting the measured recovery temperature $T_m(t)$ to retrieve (an estimate of) the true recovery temperature $T_r(t)$.

## 1.2 Correction Method #2

An alternate approach to this correction is to use the frequency-space transfer function as developed in Cooper et al. [2020], $H(\omega)$, which relates the Fourier representation of the measurement to that of the measurand via $\hat{T}_m(\omega) = H(\omega)\hat{T}_r(\omega)$. The measurand can be retrieved from the inverse Fourier transform of $\hat{T}_m(\omega)/H(\omega)$. This requires treating a block of measurements so that the Fourier components can be determined, so this is less straightforward than Method #1, especially if it is desirable to include variations in the parameters in the differential equations.

## 1.3 Application to the Heated Probe

For the HARCO heated sensor, both methods require modification. The best-fit parameters include $a = 0$, so (3) and (4) can't be used. for Method #1 Fortunately, there is an alternate solution to the equations for the case when $a = 0$:

$$T_r(t) = (\tau_1 + \tau_2)\frac{dT_m(t)}{dt} + T_m(t) + \tau_1\tau_2\frac{d^2T_m(t)}{dt^2} \tag{5}$$

This gives $T_r(t)$ without integration because finite-difference expressions can be used for the derivatives of the measurement $T_m(t)$. However, because the finite-difference estimates introduce high-frequency noise, the result needs to be smoothed using a low-pass filter with 2 Hz cutoff frequency. This is acceptable because the heated sensor has only insignificant response at such frequencies.

## 1.4 The Treatment of Dynamic Heating

As described by Cooper et al. [2020], the standard dynamic-heating correction should be modified so that only the part of that correction to which the sensor responds is subtracted. The above correction schemes apply to the recovery temperature, so further steps are needed if the measurement of air temperature is to be corrected. Several approaches to this problem are listed here:

1. Filter the dynamic-heating correction as required to find what part of that correction has influenced the recovery temperature, and subtract only that filtered correction. This produces a measurement of air temperature that is improved over the standard calculation because it

does not include the extraneous noise introduced by the unfiltered dynamic-heating correction. For this case, the resulting air temperature measurement is $T_a(t) = T_m(t) - Q_f(t)$ where $Q_f(t)$ is the dynamic-heating term filtered to represent how it will affect the measured recovery temperature: $Q_f(t) = \mathscr{F}^{-1}(H(\omega)\hat{Q}(\omega))$ where $\hat{Q}(\omega)$ denotes the Fourier transform of the dynamic-heating term $Q(t)$ and $\mathscr{F}^{-1}()$ denotes the inverse Fourier transform. However, the result then is not corrected for time response of the sensor.

2. Follow with application of Method #1 or Method #2 to the filtered result from item 1 above to obtain a revised measurement of air temperature. If the response of the sensor is linear, this will further correct the first estimate of the ambient temperature from approach #1 for the response of the sensor. In the frequency-space representation, the result can be written as:

$$T_a'(t) = \mathscr{F}^{-1}\left(\hat{T}_m(\omega)/H(\omega) - H(\omega)\hat{Q}(\omega)/H(\omega)\right) \tag{6}$$

The equivalent result is obtained by using the differential equations as in Method #1 but applied to the result from item 1 above.

3. An equivalent approach is to apply Method #1 or Method #2 to the measurement of recovery temperature, which will produce an estimate of what the sensor should have measured in the presence of normal dynamic heating, and then subtract the full unfiltered dynamic heating, which will have been restored to the recovery temperature by the correction procedure. In the frequency-space representation, because $H(\omega)\hat{Q}(\omega)/H(\omega) = \hat{Q}(\omega)$ the result is again (6).

4. For the heated sensor, the parameterized equations do not provide the best fit to the observations so the alternate fit using a polynomial might be preferable. That polynomial fit can only be applied using method #2. However, the limited examples in Cooper et al. [2020] seem to indicate that, despite the inferior fit to the transfer function, the parameterized transfer function provides a better reconstruction as judged relative to the unheated sensor.

# 2 Implementation: A Second-Pass Script

An R script "CorrectTemperature.R" adds the following corrected temperature variables to a netCDF file (where [vname] is an existing temperature variable):

1. [vname]F – A corrected variable improving on [vname] using approach #1 above, in which the dynamic-heating correction is filtered to represent the sensor response to dynamic heating and that filtered dynamic-heating correction is subtracted from the measured recovery temperature. This variable thus does not show the spurious noise present in the original variable but is not otherwise corrected for the time response of the sensor.

2. ]vname]C - A corrected variable improving on [vname] using approach #2 above, with method #1 (based on the differential equations). For measurements of air temperature, this

correction accounts for the time response of the sensor and also, for calculated air temperature, removes the spurious contributions from dynamic heating that the sensor does not detect. This is the favored result from this reprocessing. If [vname] is a recovery temperature, this result is the best estimate of the true recovery temperature and dynamic heating is not addressed.

3. [vname]C2 [optional] - A corrected variable like [vname]C but obtained by Fourier transformation. The process followed is to divide the portion of the flight from takeoff to landing into 10-min segments, each overlapping the previous segment by 5 min. Each segment is processed via Fourier transformation and the central 5-min segment is retained as the corrected variable. For the first and last segments, the leading or trailing portion of the segment is also retained so that the entire flight is covered. This optional processing is usually not an improvement over [vname]C but is available for testing.

Given a netCDF flight, the script identifies the air-temperature and recovery-temperature variables and processes each to obtain the new variables listed above. Because the recovery temperatures are not preserved in most production files, the script reconstructs the recovery temperature if necessary from the air temperature and the calculated dynamic heating, using the same recovery factor that was included as an attribute for the original air-temperature variable. The new variables are inserted into a copy of the original netCDF file, renamed to have a “T” suffix.

The processing code for this script is inserted into the “CorrectTemperature.Rnw” file as the “processor” code chunk. A processing example will be run when that file is compiled or when the associated .lyx file is exported to pdf via pdflatex. The script is available separately as “CorrectTemperature.R,” and that is included in the SensibleHeatFlux.zip file that is the project archive for the correction procedure developed to improve measurements of the flux density of sensible heat. That archive is preserved at this URL: https://github.com/WilliamCooper/SensibleHeatFlux .

The script will process files with either 25 Hz or 1 Hz time resolution. At 1 Hz, the corrected variables for the unheated probes show little change from the original although the changes are still significant for the heated probes, so by default the new variables for unheated probes are not included in the corrected files. A run-time argument can change that default if it is desired to produce corrected variables for all the sensors.

The results for 1-Hz files are better, however, if the 25-Hz file is processed first and the corrected variables are averaged to 1-Hz. The script has provision to use these values if they are available. For a 1-Hz file, the script checks if a previous run at 25-Hz produced a 1-Hz data.frame containing the averaged values, and if so those values are used instead for the corrected temperature.

```
system('Rscript CorrectTemperature.R SOCRATES rf15h')
```

## 2.1 Running the Script

The processing script starts by determining the flight or series of flights to process. This information can be provided in either of two ways:

- If run interactively (e.g., from RStudio or an R console window), the script will ask the user to provide the project and flight. The project name should follow standard archiving conventions for the netCDF files; an example is "SOCRATES" or "CSET". The flight can be provided in several different ways:
  - "NEXT" will process the next high-rate file in the project directory with a standard name like "SOCRATESrf01h.nc". This can be used sequentially to process an entire set of flights.
  - "NEXTLR" will process the next low-rate file.
  - "ALL" will process all high-rate files, skipping any that have already been processed. This may be unacceptably slow; tests indicate it may take as much as an hour.
  - "ALLLR" will process all low-rate files, skipping any that have been already processed.
  - For a specific individual flight, provide the flight name following one of these examples: "rf15h" or "rf15". An entry like "15" will process the low-rate file "[ProjectName]rf15.nc".
- The script can also be run from a terminal, after changing to the working directory containing the script. The entry at a terminal window should follow this example: "Rscript CorrectTemperature.R SOCRATES rf15h" (without the quotation marks). Three additional optional arguments, "RT", "FFT" and "UH1", can be appended, as discussed below. Interactive use will also query to determine those arguments.

The result will be a new netCDF file with "T" appended to the name, for example "SOCRATESrf15hT.nc" if the original was "SOCRATESrf15h.nc". The new file will have these added variables:

- [ATyy]C if the original file contained a variable named [ATyy] – the corrected result for air temperature that includes modifications for the sensor response and the filtered dynamic heating. For 1-Hz files, these variables are omitted for unheated sensors unless the "UH1" argument is provided either in the run-time arguments or interactively.
- [ATyy]F – the original measurement with only the addition of the usual dynamic-heating term and then subtraction of the filtered dynami-heating term. This removes the excess noise arising from the failure of the sensor to respond fully to dynamic heating, but it does not otherwise correct the measurement for sensor response.
- [RTyy]C – the corrected result for the recovery temperature. This correction addresses the sensor response but does not involve dynamic heating. It is the best estimate of the true recovery temperature that is the measurand.

For 1-Hz files, these variables are omitted for unheated sensors unless the "UH1" argument is provided either in the run-time arguments or interactively. At 1 Hz, the changes to measurements from the unheated sensors are so minor that they are probably not useful in most cases, so excluding them is the default.

In addition, these variables are optionally added:

- [RTyy] – Often archived files do not include the recovery temperature, so in the course of processing it must then be recalculated from the air temperature and the dynamic-heating term. If [RTyy] is not present in the original file, the recalculated value can be added if it is desired. These variables can be included, for cases where they are not present in the original file, by adding the argument "RT" to the Rscript call or answering the appropriate question during interactive use.[1]

- [ATyy]C2 – like [ATyy]C but calculated using the transfer function and FFT processing instead of solution of the differential equations. The variable [ATyy]C2 is calculated by subtracting the full unfiltered dynamic-heating term from [RTyy]C2, listed next. This is thought to be equivalent to [ATyy]C but can suffer from end-of-sequence effects when splicing together the FFT results, so it is excluded by default. This and the following variable are only included if "FFT" is appended to the arguments of the Rscript call or if they are requested in response to an interactive prompt.

- ]RTyy]C2 – like [RTyy]C but calculated using the transfer function and FFT processing.

## 2.2 Special Considerations for 1-Hz Files

The processing script will work for 1-Hz files, but better results can be obtained by using the corresponding 25-Hz file. For this reason, when a 25-Hz file is processed, a special data.frame is saved with the name ProjectFlightLRT.Rdata (for example, VOCALSrf05hLRT.Rdata). If such a file is available, the script will instead use the values of variables in that file when processing a 1-Hz file.

The preferable approach is to process all 25-Hz files first and then process the 1-Hz files. The new entries in the 1-Hz file will then be 1-s-average values of the corresponding variables in the 25-Hz file. See the "Examples" section for an illustration of the benefit of this processing sequence.

## 2.3 Description of Key Functions

1. *correctDynamicHeating(D, AT):* AT is a variable in the original data-frame D. The routine uses attributes of AT to select the right filter. The filters were generated in the SensibleHeatFluxAMT.Rmd file and were saved in "ARF.Rdata", which is loaded at the start of this CorrectTemperature.R script. This function retrieves the recovery factor from the attribute

---

[1]

– A special case is that where the recovery temperature is included in a 25-Hz netCDF file but only at 1 Hz. In that case the recovery temperature is recalculated using the 25-Hz air temperature. If it is desired to keep the reconstructed recovery temperature for this case, the old 1-Hz variable is renamed by appending an 'x' to the name; e.g., RTH1x.

saved with the variable, as needed if the recovery temperature is absent and should be recalculated. It then uses that recovery factor to calculate the dynamic-heating term. Application of the signal::filter() function will fail is there are missing values in the time series, so Ranadu::SmoothInterp() is used to replace missing values by linear interpolation before the filter is applied. Then the appropriate filter from the reference paper is applied to the time series via "signal::filter(AF, Q)" where "AF" is the appropriate filter (dependent on the sensor and the data rate) and "Q" is the dynamic-heating term, including the recovery factor. The file "ARF.Rdata" also specifies the appropriate time shift used for each filter. The function Ranadu::ShiftInTime()" is used after filtering to restore proper timing to the time series after filtering. The value returned from this function is the air temperature corrected by the term $(Q-QF)$ where $Q$ is the usual dynamic-heating term $\alpha_r V^2/(2C_p)$ and $QF$ is the result of applying the filter developed in the reference paper.

2. *correctTemperature(D, RT, responsePar):* *RT* is the recovery temperature, either contained in the data.frame *D* or recalculated if it is absent, and *responsePar* contains the transfer-function parameters for the particular sensor providing *RT*. The function has two branches, depending on whether to sensor is heated or unheated:

   (a) For an unheated sensor, (3) and (4) provide the basis for the correction. For the derivative $dT_m/dt$, needed to integrate (4), a centered fourth-order expression is used:

   $$\frac{dT_m[i]}{dt}=\frac{-T[i+2\}+8T_m[i+1]-8T_m[i-1]+T_m[i-2]}{12/R} \tag{7}$$

   where $R$ is the sample rate (usually 25 or 1). The expression used for this in R is

   ```
   DTMDT <- (c(0, 8 * diff(RTF1, 2), 0) -
   c(0, 0, diff(RTF1, 4), 0, 0)) * Rate / 12
   ```

   It was found that this works significantly better than a second-order centered finite-difference expression (i.e., DTMDT <- c(0, diff(RTF1, 2), 0) * Rate / 2). The integration then proceeds using a 4th-order Runge-Kutta integration with Cash-Karp adjustment of the time step, as implemented in the function *Ranadu::rk4.integrate()*, after which (3) is used to find the corrected recovery temperature. The integration routine, coded for this usage after it was found that the standard R-provided Runge-Kutta scheme had numerical problems, is not coded very well and should be used with attention to how the function being integrated is supplied. In this implementation, the function needs not only the current state of the integral but also the derivative as expressed above and the original measurement, as well as the transfer-function parameters. An example of how this is implemented is as follows, where the support temperature is calculated from (4):

   ```
   fS <- function(y, i) {
         ((tau1 * D$DTMDT[i] + D$TTRR[i] -
         (1 - a) * y) / a - y) / (Rate * tau2)
       }
   D$Ts <- rk4.integrate (fS, D$Ts[1], 1:nrow(D))
   ```

The function needs the indexed vectors DTMDT and TTRR but they are not provided explicitly as arguments to the function. Instead, it is assumed that they are present in the calling environment, after for example calculating DTMDT as in the earlier box, and only the index $i$ is provided to the function. If this script continues to be useful in the future, this weakness should be cleaned up by providing additional arguments to the function as needed.

(b) For the heated sensor, the corrected temperature is found from (5), with appropriate finite-difference formulas for the derivatives. Equation (7) can be used for the first derivative, but this equation also involves the second derivative $\frac{d^2T_m(t)}{dt^2}$. The fourth-order centered finite difference expression for this second derivative is:

$$\frac{d^2T_m([i]}{dt^2} = \frac{-T_m[i+2]+16T_m[i+1]-30T_m[i]+16T_m[i-1]-T_m[i-2]}{12/R^2} \tag{8}$$

which is calculated by the following R code:

```
DTM2DT2 <- (-c(0,0, RTH1)[1:nrow(DSX)] + 16*c(0,RTH1)[1:nrow(DSX)]
      - 30 * RTH1 + 16 * c(RTH1[2:nrow(DSX)], 0)
      - c(RTH1[3:nrow(DSX)], 0, 0)) * (Rate^2) / 12
```

Because the finite-difference formulas introduce unacceptable high-frequency noise where the heated sensor has negligible response, a third-order Butterworth low-pass filter with cutoff frequency of 2 Hz is then applied to the corrected temperature.

3. *addFFTsoln(D, RV, responsePar)* – For the recovery temperature RV, this function calculates and returns the corrected recovery temperature. It uses the transfer function provided by "*responsePar"* and uses sequential segments with Fourier transforms. These calculations are only performed, and the variables are only added, when "FFT" is an argument to the script. The solution for the unheated probe is hard to distinguish from that from the differential equations, and the solution for the heated probe is usually inferier to that provided by the differential equations, so this is usually not a useful option. It is kept here because the processing procedure might be a model for other processing and because there might be uses for this routine during testing or debugging.

4. *processFile(ProjectDir, Project, Flight):* This is the main processing function, called after determining the run parameters via user interaction or the provided run arguments. The next section describes the processing governed by this function.

## 2.4 The Flow of Processing

Once run-time parameters like the project and flight are determined, the main routine calls *processFile()* where the processing is performed by calling the functions in the preceding subsection and then constructing the new output file. A flow chart for the script is shown in Fig. 1.

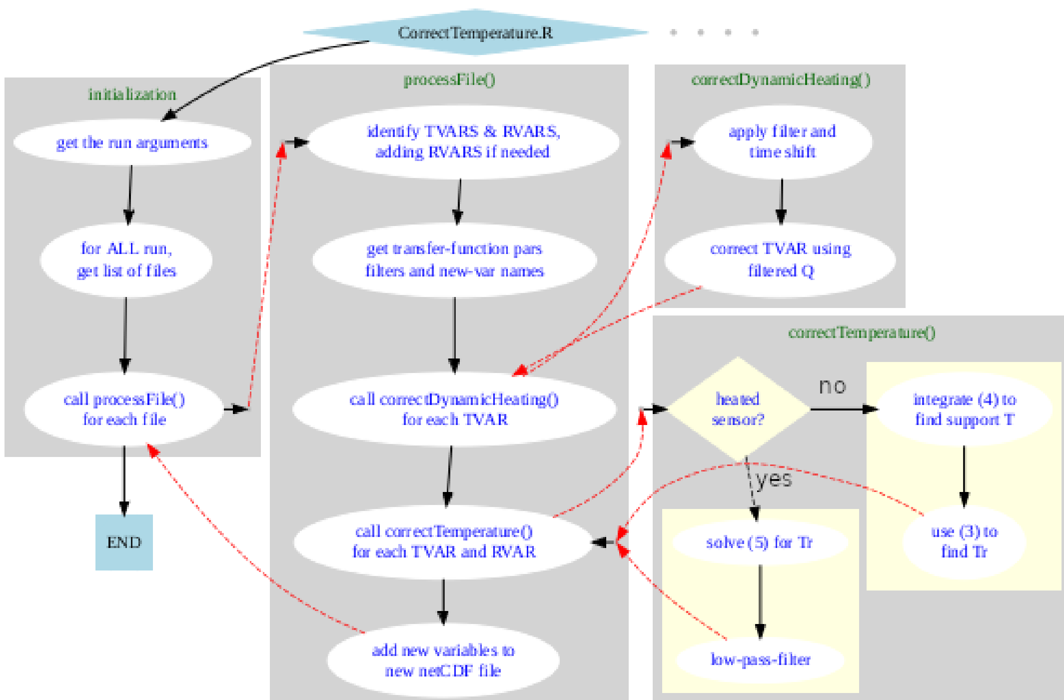


Figure 1: Flow diagram for the script CorrectTemperature.R. The individual functions are described in the preceding subsection.

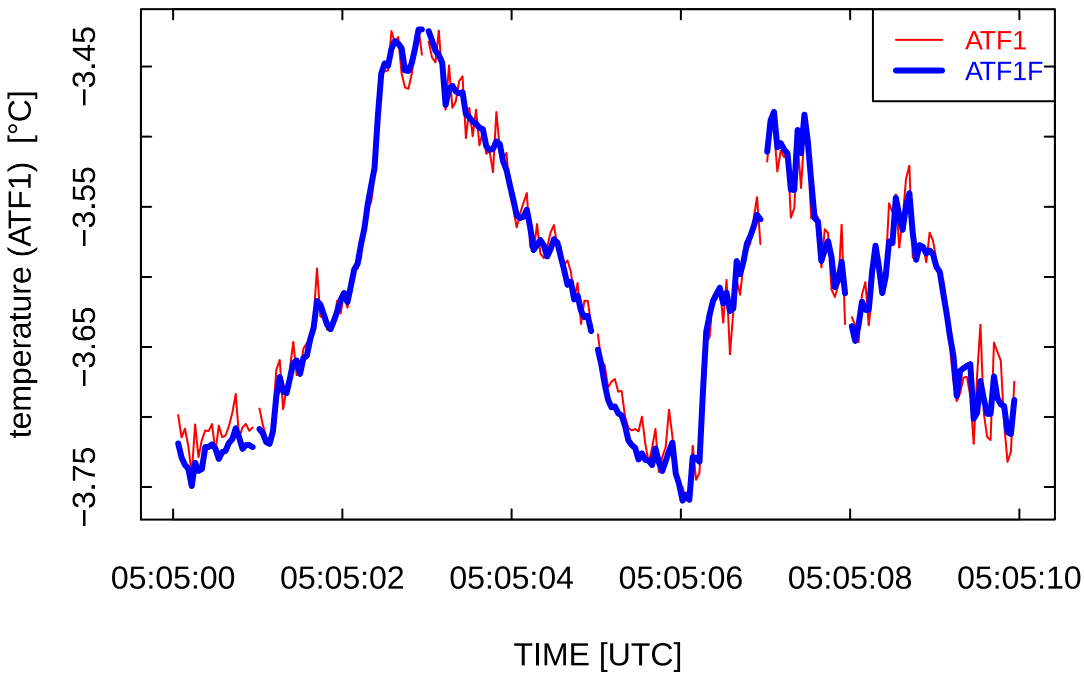


Figure 2: The measurements from an unheated sensor (ATF1) and the same measurements with the dynamic-heating correction filtered to match the response of the sensor (ATF1F). Data at 25-Hz from SOCRATES flight 15.

# 3 Examples

## 3.1 [vname]F - corrected dynamic heating

Examples were shown in Cooper et al. [2020], Fig.9. It is expected that the filtering will remove variations in dynamic heating to which the sensor does not respond, so the result will be smoother than the original variable but will track it fairly well. A significant contribution to high-frequency noise should also be removed, and this will appear in the variance spectra.

For a 25-Hz file and for the unheated sensor, Fig. 2 shows the expected smoothing and removal of false high-frequency fluctuaions, and Fig. 3 shows that the high-frequency noise in the variance spectrum is reduced substantially.

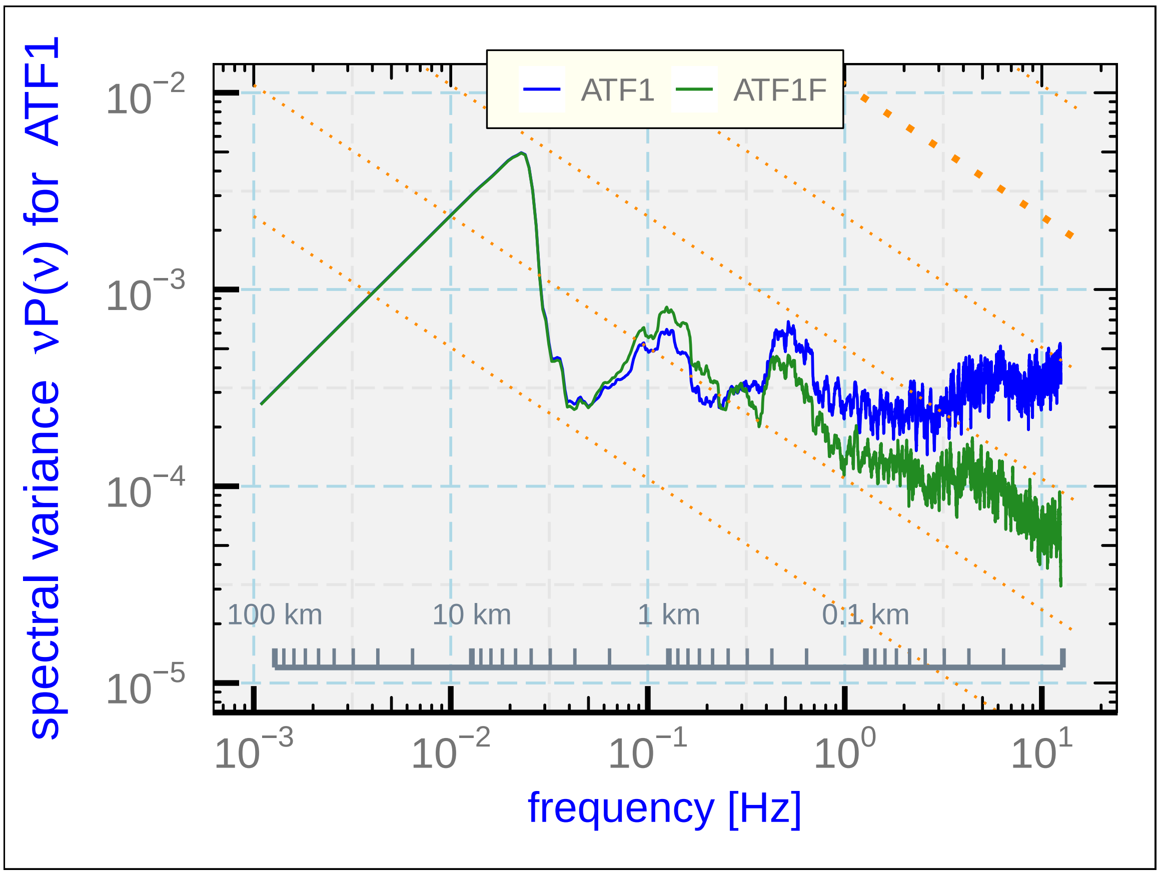


Figure 3: The variance spectra for measurements from an unheated sensor (ATF1) and for the measurements after filtering the dynamic-heating correction to match the response of the sensor (ATF1F). Data at 25 Hz from SOCRATES flight 15, 5:55:00 to 6:10:00 UTC, a flight segment at low level in the boundary layer.

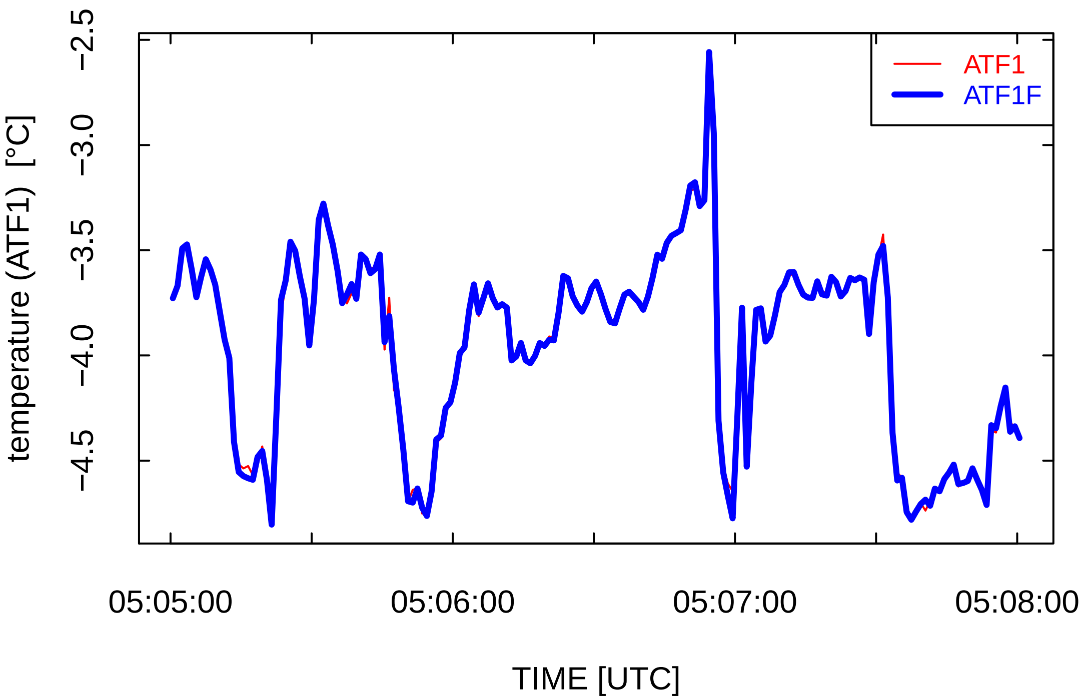


Figure 4: The measurements from an unheated sensor (ATF1) and the same measurements with the dynamic-heating correction filtered to match the response of the sensor (ATF1F). Data at 1 Hz from SOCRATES flight 15.

The corresponding results for a 1-Hz file, also for the unheated sensor, are shown in Figs. 4 and 5. At 1 Hz, filtering the dynamic-heating term has negligible effect, so this processing is not useful for the unheated sensor. It will be omitted by default unless the run-time variable "UH1" is provided either as an argument to the script or interactively.

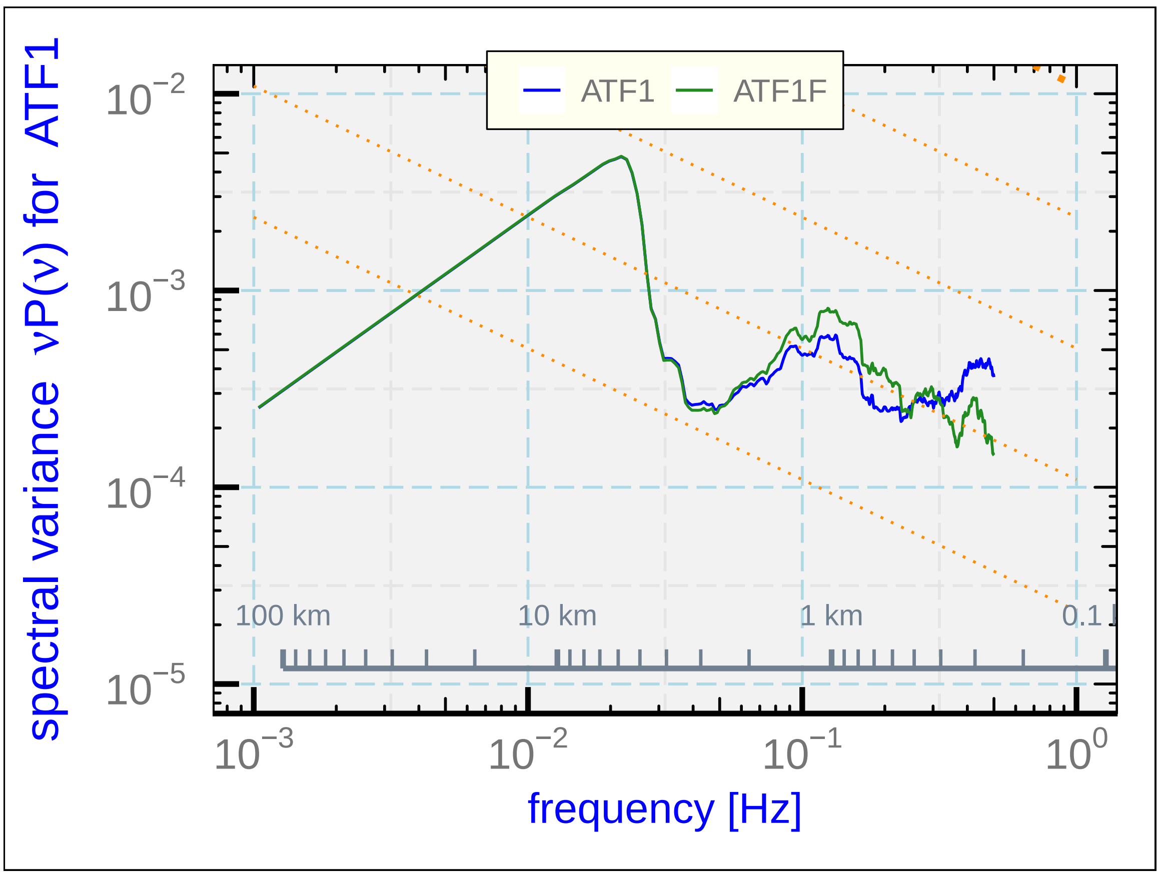


Figure 5: The variance spectra for measurements from an unheated sensor (ATF1) and for the measurements after filtering the dynamic-heating correction to match the response of the sensor (ATF1F). Data at 1 Hz from SOCRATES flight 15, 5:55:00 to 6:10:00 UTC, a flight segment at low level in the boundary layer.

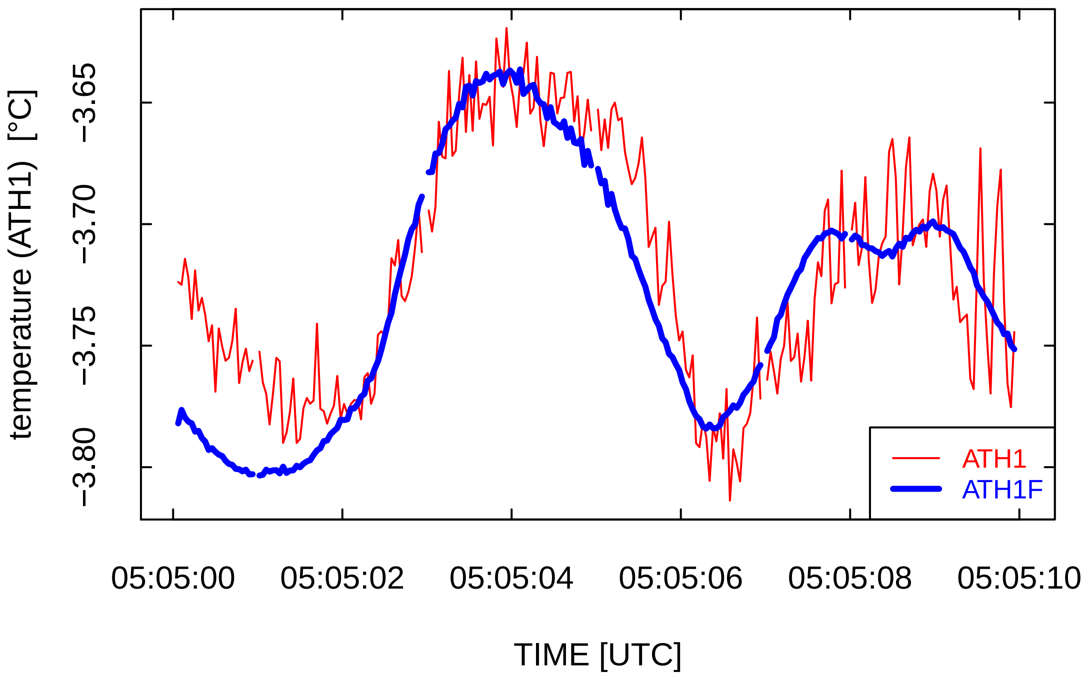


Figure 6: The measurements from a heated sensor (ATH1) and the same measurements with the dynamic-heating correction filtered to match the response of the sensor (ATH1F). Data at 25-Hz from SOCRATES flight 15.

For the heated HARCO sensor, the corresponding results for 25-Hz measurements are shown in Figs. 6 and 7. In this case, the change is quite pronounced, and the variance spectra emphasize how much of the original high-frequency variance is the result of erroneous subtraction of dynamic-heating fluctuations to which the sensor does not respond.

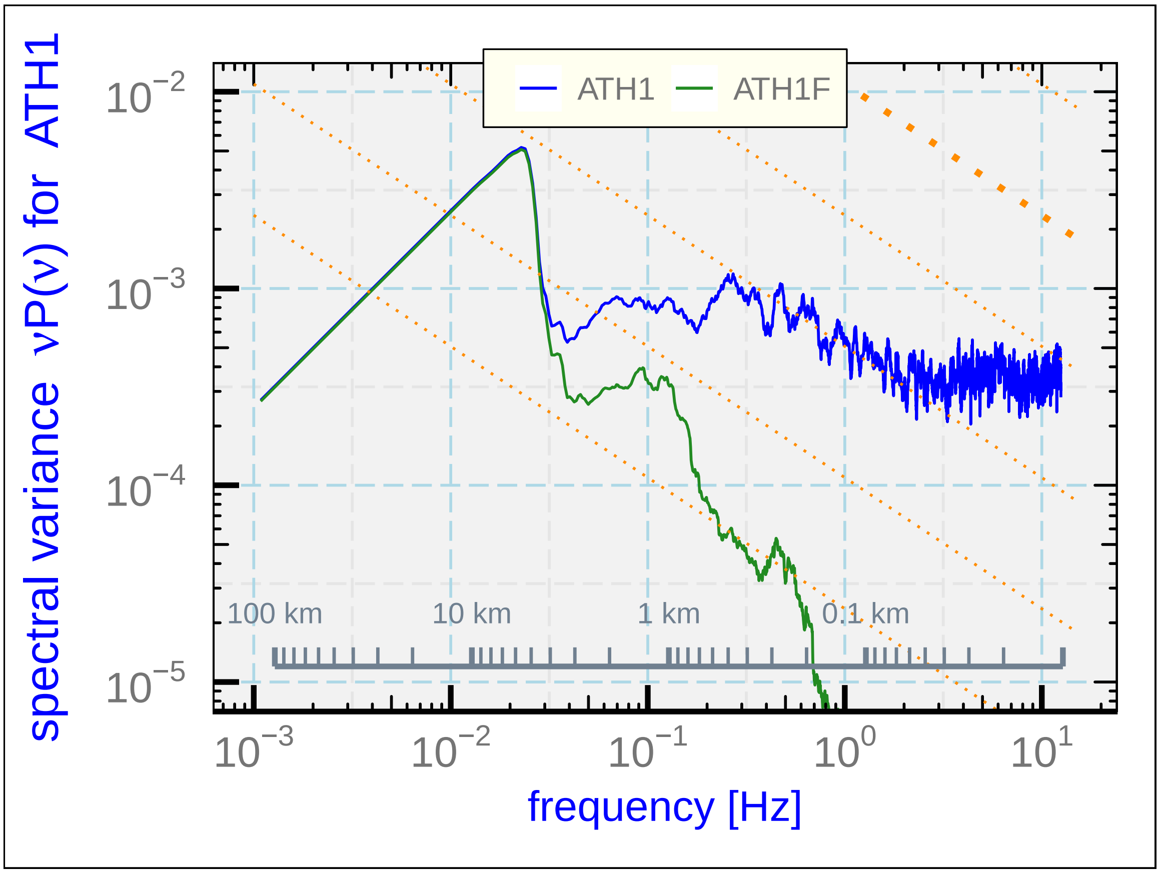


Figure 7: The variance spectra for measurements from a heated sensor (ATH1) and for the measurements after filtering the dynamic-heating correction to match the response of the sensor (ATH1F). Data at 25 Hz from SOCRATES flight 15, 5:55:00 to 6:10:00 UTC, a flight segment at low level in the boundary layer.

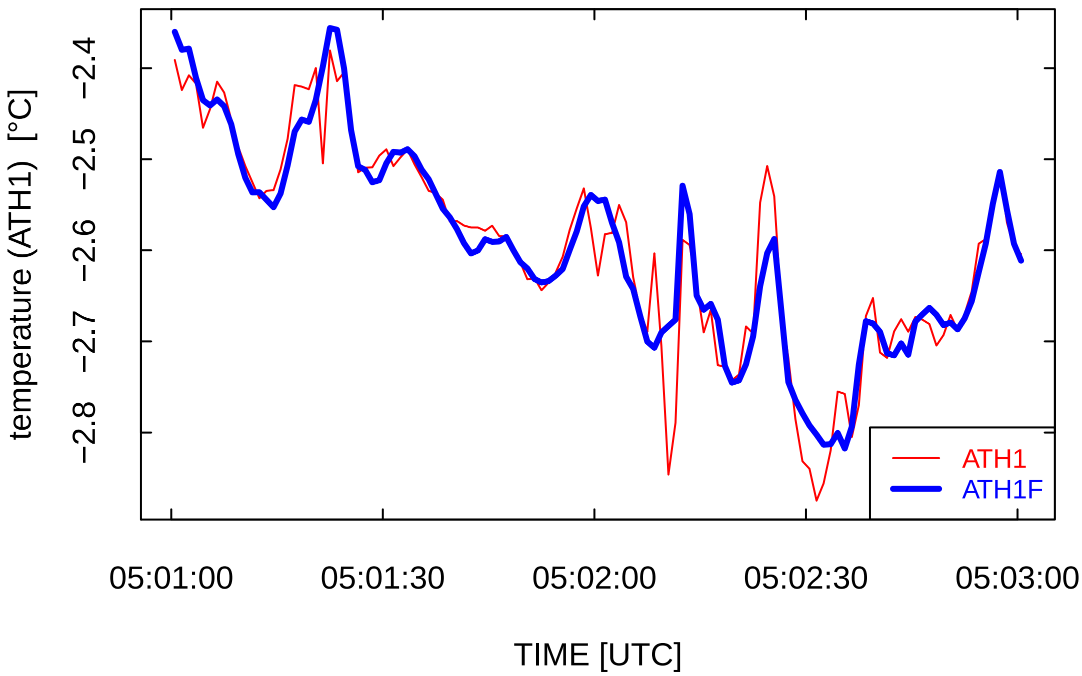


Figure 8: The measurements from a heated sensor (ATH1) and the same measurements with the dynamic-heating correction filtered to match the response of the sensor (ATH1F). Data at 1-Hz from SOCRATES flight 15.

Unlike the unheated sensor, for 1-Hz measurements filtering dynamic heating removes some significant erroneous fluctuations from measurements from the heated sensor, as shown in Fig. 8. This is also apparent in the plot of spectral variance, as shown in Fig. 9.

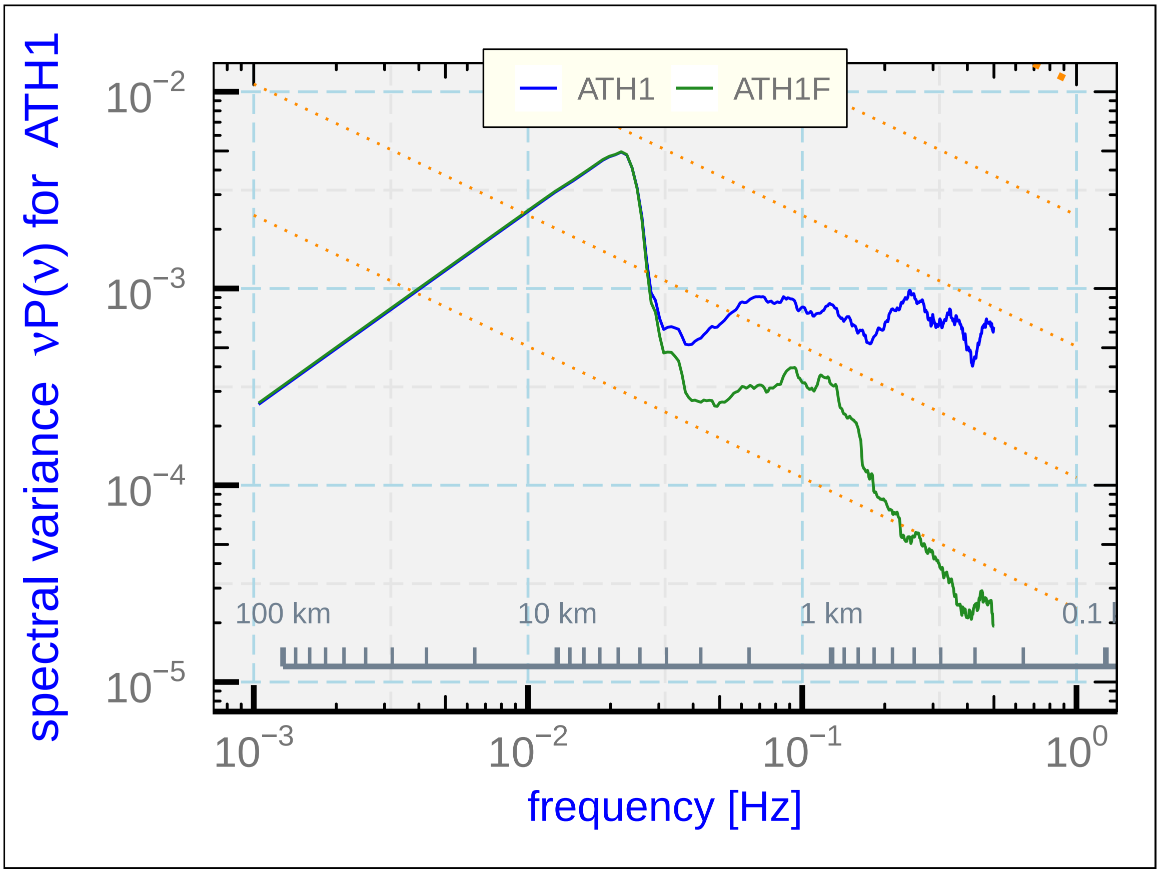


Figure 9: The variance spectra for measurements from a heated sensor (ATH1) and for the measurements after filtering the dynamic-heating correction to match the response of the sensor (ATH1F). Data at 1 Hz from SOCRATES flight 15, 5:55:00 to 6:10:00 UTC, a flight segment at low level in the boundary layer.

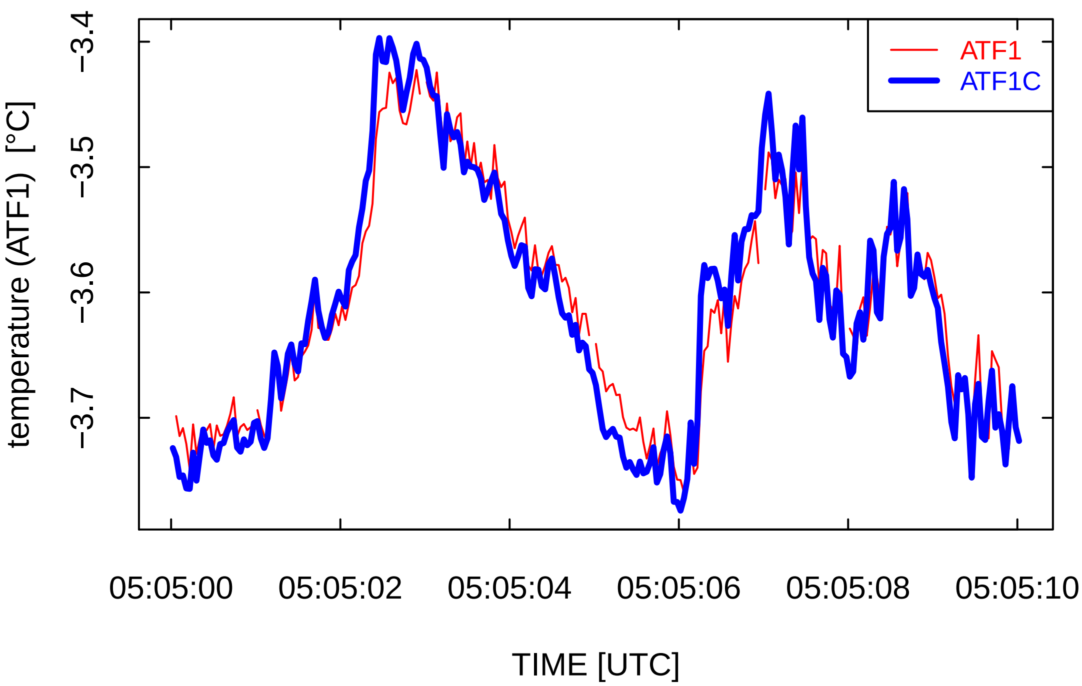


Figure 10: Measurements at 25 Hz from an unheated sensor before (ATF1) and after (ATF1C) correction. Data from SOCRATES flight rf15.

## 3.2 [vname]C - correction for time response

The plots in this subsection show the effect of correcting for the time response of the sensor. The preceding section only filtered the dynamic-heating correction and did not make any additional correction for the time response as represented by the sensor transfer function. The results in this section were obtained by correcting the measurement of recovery temperature for sensor response and then subtracting the unfiltered dynamic-heating correction. This is appropriate because the transfer function restores a best estime of the true recovery temperature, which includes the effect of dynamic heating without filtering. These then are the best measurements of the corrected air temperature.

Figure 10 shows the uncorrected measurement and the measurement after correction (respectively ATF1 and ATF1C) for 25-Hz measurements, and Fig. 11 shows an example of the variance spectra for the same variables. The correction improves the sensor response and makes significant changes to the measurements, particularly giving improved response at times like 5:05:02 or 5:05:06 where there are rapid changes in the temperature.

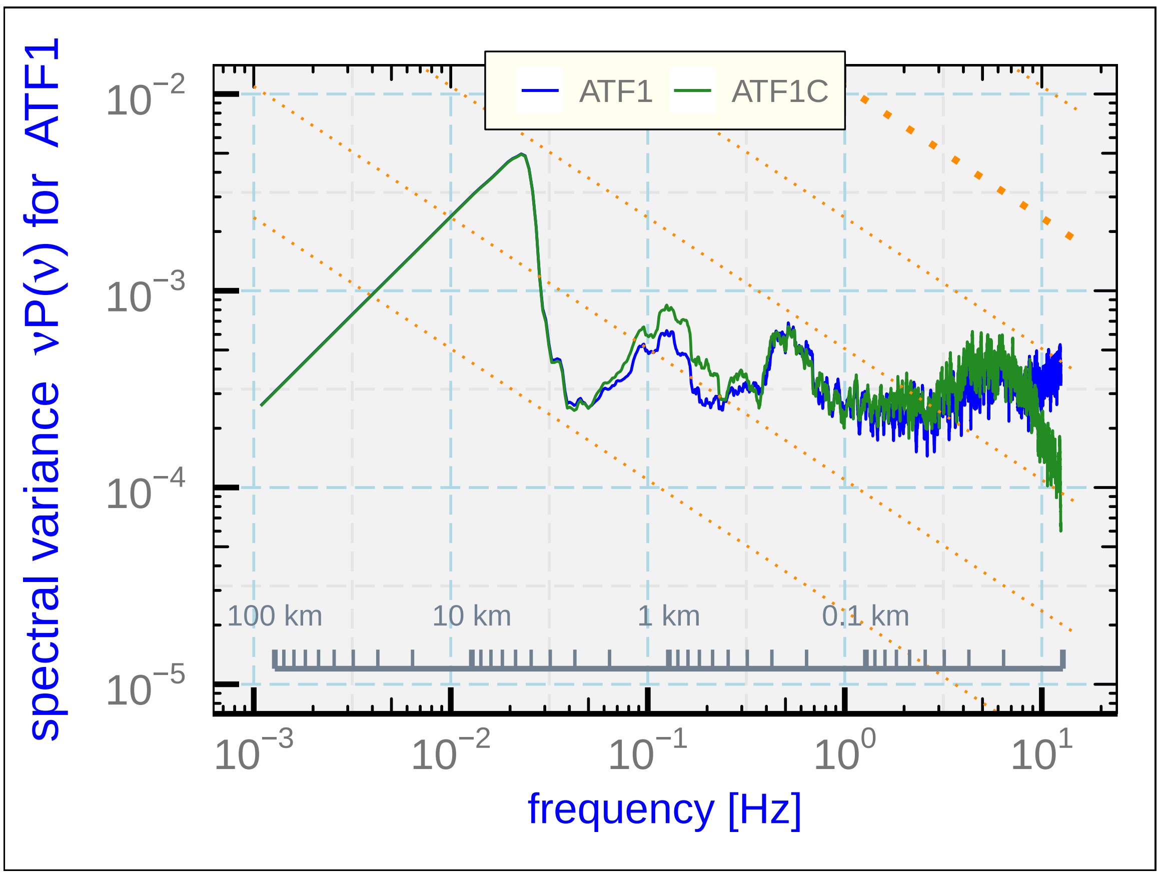


Figure 11: Variance spectra for measurements at 25 Hz from an unheated sensor before (ATF1) and after (ATF1C) correction. Data from SOCRATES flight rf15, 5:55:00 to 6:10:00 UTC, a flight segment in the marine boundary layer.

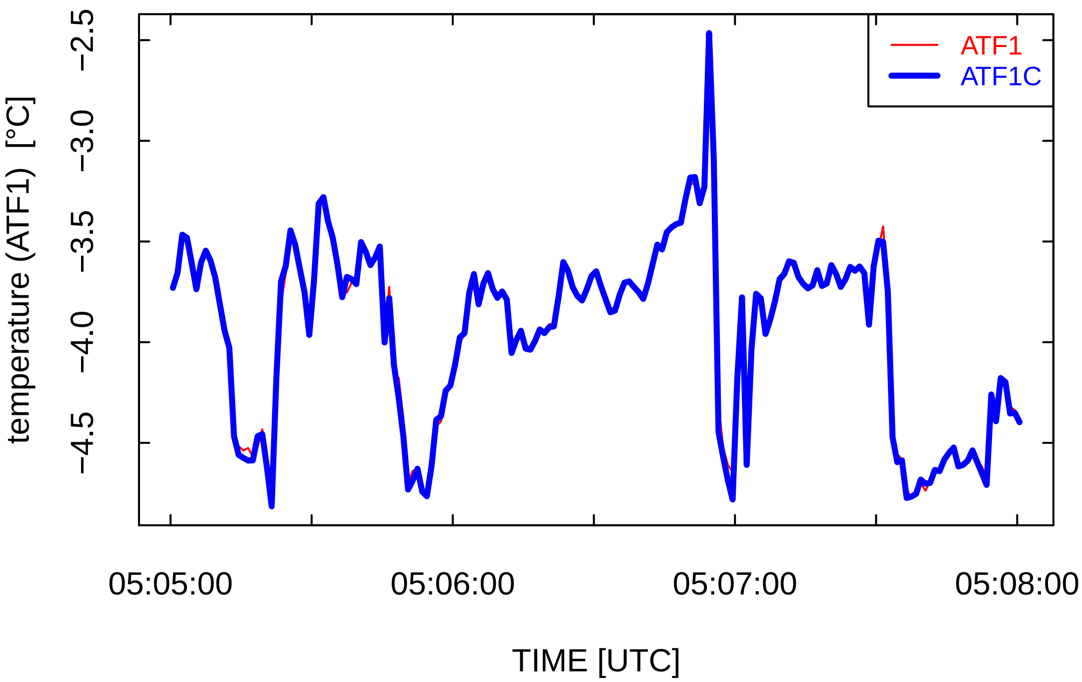


Figure 12: Measurements at 1 Hz from an unheated sensor before (ATF1) and after (ATF1C) correction. Data from SOCRATES flight rf15.

The corresponding results for a 1-Hz file, also for the unheated sensor, are shown in Figs. 12 and 13. At 1 Hz, filtering the dynamic-heating term has negligible effect, so this processing is not useful for the unheated sensor. For this reason, variables like ATF1C are omitted from the output by default, unless they are requested specifically by the runtime argument “UH1”.

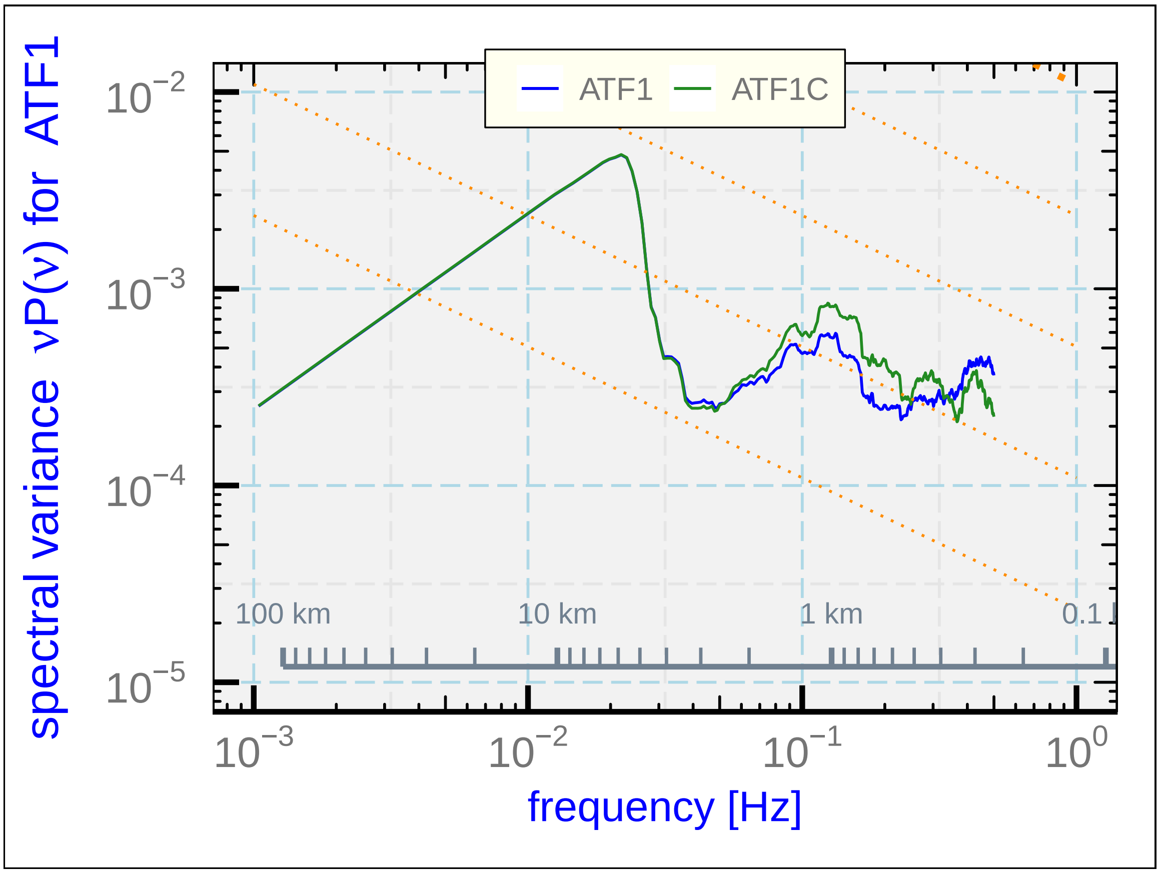


Figure 13: Variance spectra for measurements at 1 Hz from an unheated sensor before (ATF1) and after (ATF1C) correction. Data from SOCRATES flight rf15, 5:55:00 to 6:10:00 UTC, a flight segment in the marine boundary layer.

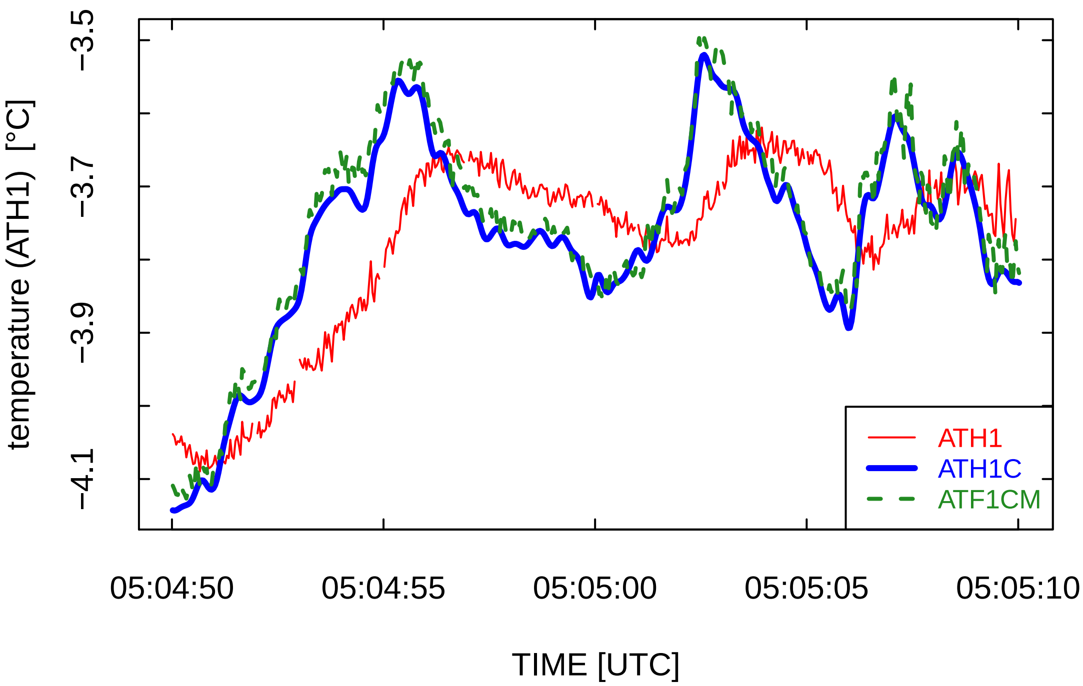


Figure 14: The corrected variable ATH1C compared to the original measurement ATH1, for a heated sensor and for 25-Hz measurements. Data from SOCRATES flight 15. The corrected measurements from the unheated sensor (ATF1CM) are also shown after an imposed shift of -0.1 C to facilitate comparison.

For the heated sensor and for 25-Hz measurements, the effect of the correction is to produce a measurement in reasonable agreement with the much faster unheated sensor. This is shown in Fig. 14, where the corrected result from the unheated sensor (ATF1C) is shown as the dashed green line after adjustment by -0.1°C (and labelled instead ATF1CM) to facilitate comparison to the corrected measurement from the unheated sensor. The time-response correction is quite important for this sensor and improves the evident time response of the measurement significantly, leading to a measurement almost as good as that from the unheated sensor.

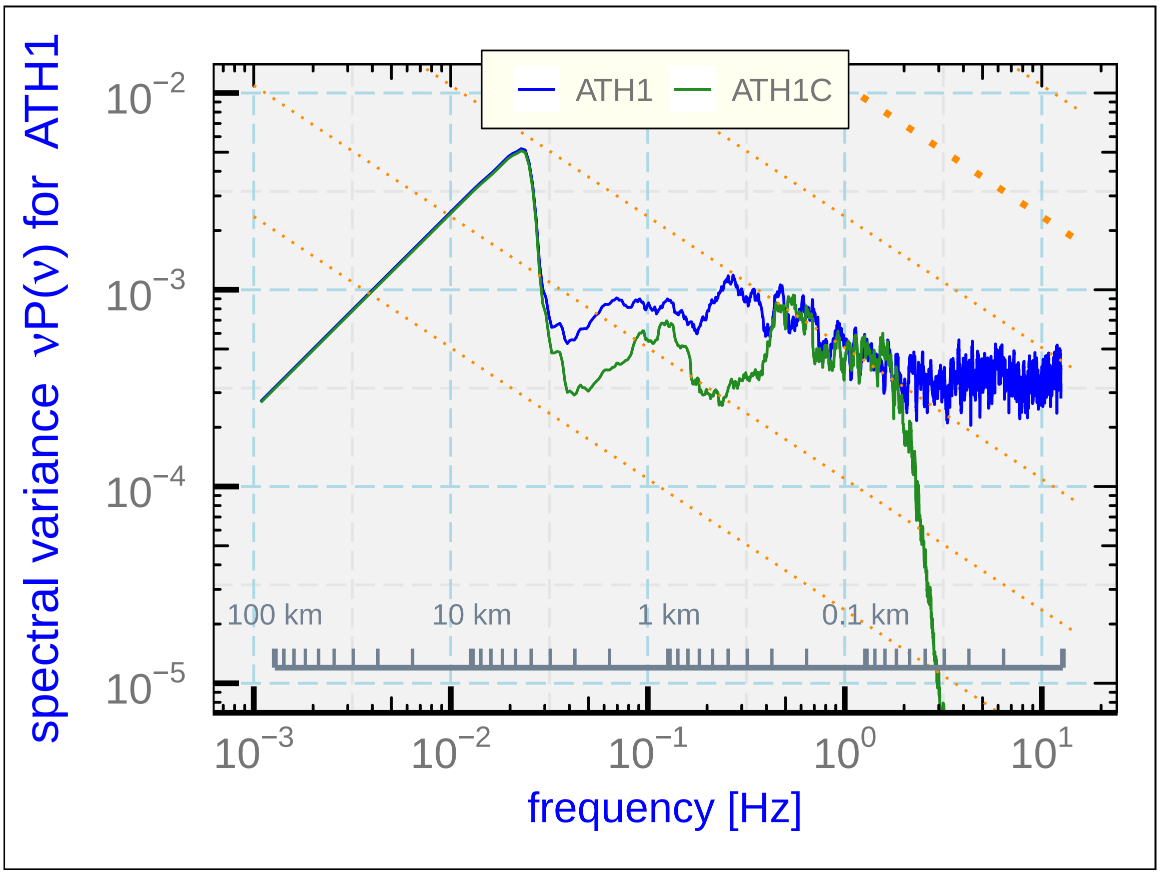


Figure 15: Variance spectra for measurements at 25 Hz from a heated sensor before (ATH1) and after (ATH1C) correction. Data from SOCRATES flight rf15, 5:55:00 to 6:10:00 UTC, a flight segment in the marine boundary layer.

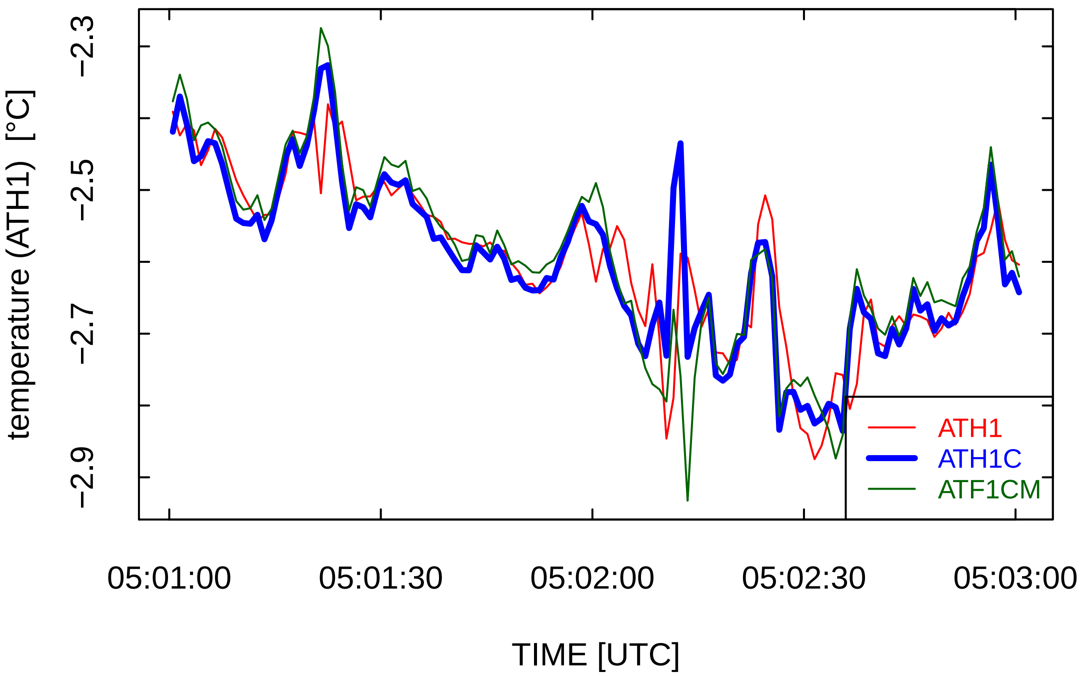


Figure 16: The corrected variable ATH1C compared to the original measurement ATH1, for a heated sensor and for 1-Hz measurements. Data from SOCRATES flight 15. The corrected measurements from the unheated sensor (ATF1CM) are also shown after an imposed shift of $-0.1$C to facilitate comparison.

Figures 16 and 17 show corresponding results for the heated sensor for 1-Hz measurements. The corrections move the measurements in the direction of those from the unheated sensor (shown as the green line), but the improvement does not appear to be as significant as in the case of 25-Hz measurements. This measurement still appears to be useful and is therefore included in the output files.

A still better way of applying this correction is to resample 25-Hz measurements at 1 Hz, because the 25-Hz correction appears to be significantly better than this one.

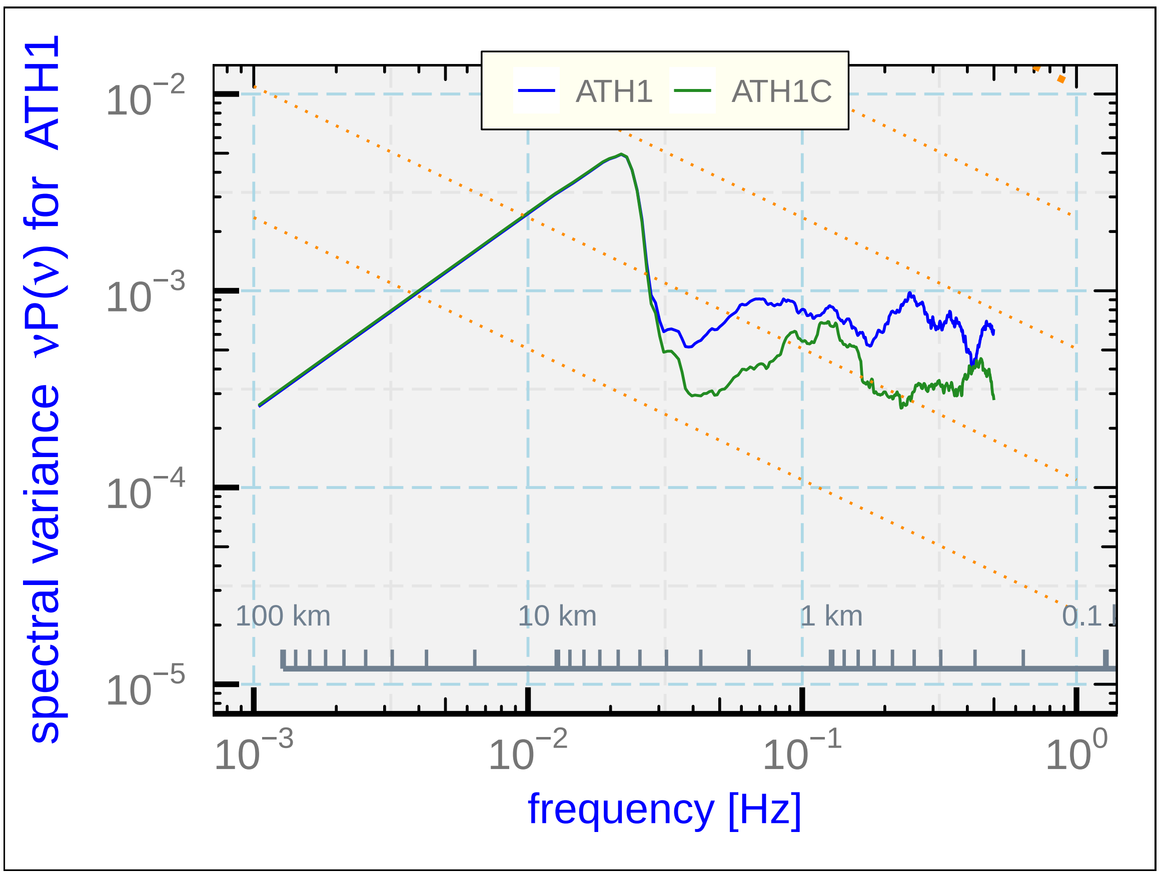


Figure 17: Variance spectra for measurements at 1 Hz from a heated sensor before (ATH1) and after (ATH1C) correction. Data from SOCRATES flight rf15, 5:55:00 to 6:10:00 UTC, a flight segment in the marine boundary layer.

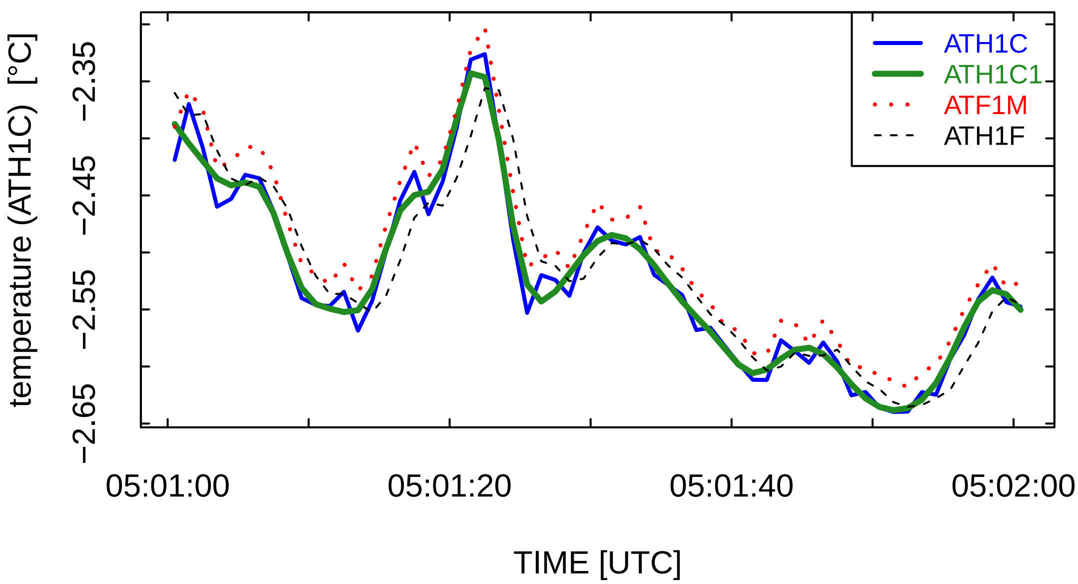


Figure 18: Corrected measurements for an unheated sensor (ATH1) from a 1-Hz data file from SOCRATES flight 15. The original measurement corrected by filtering the dynamic-heating contribution is shown as the dashed thin black line. It is evidently delayed and smoothed relative to the corrected measurement obtained from the unheated sensor, shown for reference as the dotted red line (with adjustment by $-0.1$°C to aid comparison to the heated sensor). The green line labeled "ATH1C1" is the corrected measurement obtained from the heated sensor using processing applied to the 1-Hz file, which the blue line labeled "ATH1C" is the result of averaging the 25-Hz measurements to 1 Hz. Both move closer to the reference "ATF1M" line, but "ATH1C" appears to provide a superior correction.

# References

J. R. Cash and A. H. Karp. A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software (TOMS)*, 16(3): 201–222, 1990.

W. A. Cooper, A. Bailey, and J. Carnes. TO BE SUBMITTED: On Measuring Sensible-Heat Flux With Airborne Thermometers. *Atmospheric Measurement Techniques*, 2020. URL `https://github.com/WilliamCooper/SensibleHeatFlux/SensibleHeatFluxAMT.pdf`.

– End of Memo –

Reproducibility:

| | |
|---|---|
| PROJECT: | CorrectTemperature |
| ARCHIVE PACKAGE: | CorrectTemperature.zip in the Git directory below |
| CONTAINS: | attachment list below |
| PROGRAM: | CorrectTemperature.Rnw |
| ORIGINAL DATA: | /scr/raf_data/SOCRATES/rf15h.nc for the example |
| GIT: | https://github.com/WilliamCooper/SensibleHeatFlux/CorrectTemperature.zip |

Attachments: CorrectTemperature.Rnw
CorrectTemperature.pdf
CorrectTemperature.R
CorrectTemperature.dot
SessionInfo