4 March 2020

TO: SensibleHeatFlux Archive: workflow document
FROM: Al Cooper
SUBJECT: Workflow comments for the Tech Note re sensible-heat flux

1 Purpose

This workflow description provides additional details leading to the document generated by "SensibleHeatFluxTechNote.Rnw" and provides additional detail not in the report "SensibleHeatFluxTechNote.pdf." SensibleHeatFluxTechNote.Rnw contains both text (in LaTeX format) and R processing script for the analyses in the resulting report. The description of workflow provided here includes the process of collecting the observations and processing them to data files, the data archives used, the steps required to generate the plots and other results including the instances where manual intervention is required to identify appropriate subsets of the data, the relevant R code and LaTeX documents, and all the steps leading to the generation of the text in the technical note. "SensibleHeatFluxTechNote.Rnw" is the definitive reference, but this overview and these diagrams will help explain the workflow at a general level and so should substitute for reading the R and LaTeX code in most cases. The intent is to describe the workflow in sufficient detail to support replication of the analysis and figures presented in the report, to facilitate changes based on new data or new analysis approaches, and to make it practical to apply the proposed algorithms to data collected by research aircraft used in atmospheric studies.

The associated document described here was formatted for publication as an NCAR Technical Note, but will probably be retained as a web-posted technical not. For brevity it will be references as "the TN" here. Some parts of the TN may be suitable to accompany separate journal articles envisioned to augment this TN or perhaps to replace it if it is decided that publishing this in a series of papers is preferable. The alternative under consideration is to submit this series, all with a title like "Corrections for the Delayed Response of Airborne Thermometers:"

- Part 1: Determining the Characteristic Response Using Dynamic Heating
    1. Background and Differential Equations
    2. Solution and Transfer Function
    3. Fits to Forcing by Dynamic Heating
    4. Corrections to the Measurements
- Part 2: Errors Introduced By Conventional Data Processing
    1. Conventional Data Processing and the Source of Errors
    2. A Procedure for Removing These Errors

- Part 3: Correcting Measurements of Sensible-Heat Flux
    1. Illustrating Errors Arising from Response Characteristics
    2. Using the Transfer Function to Correct These Errors
    3. Simulation Examples for Illustration

2 Acquisition of the primary data

The measurements used in this report were collected using the NSF/NCAR research aircraft during the various research projects. The onboard data-acquisition program 'aeros' recorded the data in digital format, and those data files were then processed by the program 'nimbus' to produce an archive in NetCDF format. The software management group of NCAR/EOL maintains a version-controlled archive of these programs, so if they are of interest they can be obtained by contacting the data-management group of EOL (at this or this address). The data files available from NCAR/EOL can be found at links on this URL. The details of the processing algorithms including those for the calculation of wind are documented in this report on Processing Algorithms and in Lenschow and Spyers-Duran [1989]. These procedures as they pertain to the measurement of wind are also documented in Cooper et al. [2016]. The resulting data files contain measurements in scientific units and brief descriptions of each measurement, included as netCDF attributes for the files and for each variable.

3 The SensibleHeatFluxTechNote.Rnw file

The .Rnw file is basically LaTeX text, generated for simplicity using LyX and exported to .Rnw format and then processed in RStudio (RStudio [2009]). The .lyx file (SensibleHeatFluxTechNote.lyx) will run equivalently and produce a PDF-format version of the manuscript. The .Rnw file was generated from the .lyx file, but some subsequent editing of the .Rnw file makes that the reference version for the TN. Within the .Rnw file or within the .lyx file there are "chunks" of R code (R Core Team [2019]), delineated by a header having this format:

```
<<title, var=setting, ...>>=
...R code...
@
```

These chunks generate plots and other results of analyses that are incorporated into the manuscript using 'knitr' (Xie [2013, 2014]). In RStudio, the chunks appear as gray sections in the file when it is edited. Where tasks involve execution of R code, the chunk containing the code is referenced in the discussion below. Any results from the processing can be

incorporated into the LaTeX text via "\Sexpr{}" calls embedded in the LaTeX portion of the file.

Two "switches" serve to speed execution of the code: (1) "ReloadData," which when TRUE causes the original archive data files to be read; and the option "CACHE" which, when true, causes previously generated subsets of the results to be re-used as saved in a subdirectory "cache." Both are provided to speed execution in cases where small changes are made. When ReloadData is FALSE subsets of data files previously saved as "Rdata" files are restored to R data.frames, a process that is much faster than re-reading the original archive data file. If cache is FALSE all calculations are performed, but using "cache=TRUE" is usually much faster after the first run. To ensure a clean run, set CACHE to be FALSE and remove all entries from the "cache" subdirectory.

4 Required R packages including Ranadu

The R code used for analysis reported in this paper relies heavily on a package of routines for R called "Ranadu." This is a set of R scripts for working with the NetCDF archive data files produced by NCAR/EOL/RAF, and it includes some convenience routines for generating plots and performing other data-analysis tasks with the archived data. The Ranadu package is available at this GitHub address. To run the R code referenced here, that package should be included in the R installation. The SensibleHeatFluxTechNote.Rnw routine requires that package and also some others referenced in the file, including "knitr", "ggplot2", "grid", "ggthemes", "zoo", "signal" and "magrittr". In addition, Ranadu requires "ncdf4", "tcltk" and "stats". Some parts of "Ranadu" reference additional packages as needed, but they are not used in SensibleHeatFluxTechNote.Rnw so do not need to be available for this routine to run.

The data processing for this manuscript involved revising some parts of the Ranadu package, as listed below. The most significant change is the addition of a "flux()" function to the package. That function is used to generate and plot the cospectrum used to calculate the flux, and it can optionally apply the correction procedure developed in the TN. It is described in the following subsection.

It may be worthwhile to call attention to the "pipe" that is used frequently in the R code. This relies on the "magrittr" package for R. The general structure resembles the following:

```
DataFrame %>%
  select(Time, ATH1, ATF1, Ts) %>%
  selectTime(114500, 115000) %>%
  plotWAC()
```

At each occurrence of %>% the output from the previous command is piped to the next command as the first argument. This provides clear documentation of the sequences used to construct various plots.

The analysis uses R data.frames that are generated from the reference netCDF files by the function "Ranadu::getNetCDF()". Most of the plots are generated using "Ranadu::ggplotWAC()" or "Ranadu::plotWAC()", which are simple convenience routines that set various options preferred by the author before calling the R "ggplot()" or "plot()" routines.

There is a manual for the Ranadu package that includes information on how to install it. See this URL. However, to get the latest functions including "flux()" and some other modifications used here, download the "WAC" branch. The main branch will eventually be updated, so this is a temporary solution.

4.1 Ranadu::flux ()

This function calculates the appropriate cospectrum, generates a plot of that cospectrum, calculates the total flux and the partial flux associated with fluctuations smaller than a specified wavelength, and returns a data.frame containing the frequency, the smoothed cospectrum (not weighted by frequency), and the exceedance values. The data.frame also has attributes "Flux" and "FluxL" representing the total flux and the flux from wavelengths smaller than the wavelength "wavelengthLimit". The "wavelengthLimit" is also included as an attribute of the returned data.frame.

An optional argument to this function is "Par" which can be set to appropriate response parameters for a temperature sensor. In that case, a response function is generated using those parameters and the cospectrum is corrected using that response function.

This plot generated by this function is described in more detail in Section 5 of the Technical Note, fourth paragraph. The routine was used to generate figures like Fig. 28 in the TN. Within R, further information can be found using the R help facilities once the Ranadu package is available.

4.2 Ranadu::CohPhase ()

Another Ranadu function used for the calculations in the TN is "Ranadu::CohPhase()", which calculates and plots the squared coherence and phase between two time series. That function was recently modified for the needs of the analysis leading to the TN. Like all Ranadu functions, it can be obtained from the GitHub reference above and its documentation can be checked using R help facilities once the package is installed. It was used to generate the cospectra used in the determination of the phase between the measured recovery temperature and the expected effect of dynamic heating.

A modified version of this routine is included in the .Rnw file, named CohP( ). It was modified to return binned values of the phase and amplitude ratio labeled with bin numbers so that multiple cross-spectra covering the same frequency interval could be averaged to obtain these values. This special version was used to find the averaged values entering, for example, Figs. 5 and 6 in the TN. See the discussion that follows in Sect. 6.1.1.

4.3 Ranadu::VSpec ()

Plots of variance spectra, like that shown in Figs. 3 and 4, were generated using the Ranadu function "VSpec()". There is a brief discussion of the function in "RanaduManual.pdf" p. 39, available at this URL. A tutorial Shiny-app is also available here that discusses plotting spectral variance and describes some of the options available in "VSpec()". R help facilities also provide documentation once the "Ranadu" package is installed.

4.4 Ranadu::SmoothInterp( )

Many functions including the standard spectral analysis functions and stats::fft will fail if there are missing values in the processed time series. To avoid this problem, "SmoothInterp( )" has been used extensively in the code for this TN. It calls "zoo::na.approx( )" to substitute interpolated values for missing values, and it uses the "rule = 2" argument to that function to extend values to beginning and ending sequences with missing values. ("zoo:na.spline( )" has been used in this function, but it tends to introduce extraneous variance.) By default, "SmoothInterp( )" also applies Savitzky-Golay smoothing also, so this needs to be suppressed by the ".Length=0" argument to "SmoothInterp( )" when smoothing is not desired (which applies to all uses in this TN). "SmoothInterp( )" is called internally from within some of the functions that deal with variance spectra, including "VSpec( )" and "CohPhase( )".

The following construction can be used to remove all the missing values in a data.frame (as in code chunk "SOC11x") and substitute interpolated values:

```
removeNA <- function (D) {D <- SmoothInterp(D, .Length=0)}
DSW <- as.data.frame(lapply(DS11, removeNA))
DSW <- transferAttributes(DS11, DSW)
```

The last line is needed because the "lapply( )" function doesn't carry forward the variable attributes so this transfers them from the original data.frame.

4.5 rk4.integrate()

For use in this study, a new function was defined that implements the Cash-Karp method of step adjustment (Cash and Karp [1990]) for a fourth-order Runge-Kutta integration. This was done when it became apparent that the standard R function runge.kutta() in the "rmutil" package (Swihart and Lindsey [2019]) did not perform adequately for step sizes of 0.04 s (as required for processing 25-Hz files) because the shorter time constant for the unheated Rosemount sensor is smaller than this step size. This new integration routine may be added eventually to the Ranadu package but for now is defined in a separate code "chunk" near the beginning of the "Rnw" file. It is based on the description of the method in Press et al. [1992] but has been coded independently in R for this TN. Unlike the standard method,

it does not ever increase the step size above the time increment in the file, but if the error estimate for a particular step does not meet the specified tolerance then the step is divided into multiple smaller steps. If the tolerance is still not met, further subdivision occurs.

One aspect of how this routine is used makes it problematic for general use: The integration takes as an argument a function that provides the derivative to be integrated, but that function takes a single argument. As applied in this TN, however, additional variables are needed to calculate the derivative. Here is an example of a function definition used to find the temperature of the support that contacts the wire:

```
fS <- function(y, i) {  # Eq. Ts3
    ((1/a) * (tau1 * DS11$DTMDT[i] + DS11$RTF1[i] - (1-a) * y) - y) / (Rate * tau2)
}
```

This function only works by using quantities that are in the function environment (here, DS11, a, tau1, Rate, tau2). A better structure would provide those as additional arguments to the function, but that will require restructuring the "rk4.integrate( )" function as well. The specific function that provides the derivative is defined each time an integration is performed, usually just before the integration, but the practice of referencing the calling environment might lead to problems for someone wanting to extend or revise the program embedded with this document in the ".Rnw" file.

5 Comments on the differential equations and their solution

Section 2 of the TN presents the steady-state solution to the differential equations (TN Eq. (3) and (4)) for a sinusoidal input, as TN Eq. (5)–(8), and determines the frequency-dependent "transfer function" (ratio of output to input for sinusoidal input) from that solution. The solution was obtained using Laplace transforms, which may not be familiar to some potential users of this work, so extensive documentation of the approach is included in this section of the workflow document. Also included is a direct verification of the solution obtained by substituting into the governing differential equations and showing that those equations are satisfied. This detail did not seem appropriate to include in the TN but it may be of interest to someone wanting to replicate or extend this work. In particular, it may be useful if additional terms are needed to describe the time response.

5.1 Re: Section 2.1

Consider a sine-wave input $T(t) = \sin \omega t$ representing the normalized actual history of the air temperature. The temperature of the sensor and that of the support must both also be sine waves, possibly of different amplitudes and phases: $T_m(t) = c \sin(\omega t + \phi)$ and $T_s(t) = b \sin(\omega t + \zeta)$.

The equation for the support temperature$T_s(t)$, given by TN Eq. (3) and repeated here, is

$$\frac{dT_s(t)}{dt} = \frac{(T(t) - T_s(t)}{\tau_2}$$

Taking Laplace transforms gives

$$s\tau_2\mathcal{L}(T_s) - \tau_2 T_s(0) = \frac{\omega}{s^2+\omega^2} - \mathcal{L}(T_s) \tag{1}$$

$$\begin{aligned}\mathcal{L}(T_s) = &\frac{T_s(0) + \frac{1}{\tau_2}\frac{\omega}{s^2+\omega^2}}{\frac{1}{\tau_2}+s} \\ =&\frac{T_s(0)}{\frac{1}{\tau_2}+s} + \frac{\omega}{\tau_2}\frac{1}{\left(\frac{1}{\tau_2}+s\right)(s+i\omega)(s-i\omega)}\end{aligned}$$

Expanding the last term in partial fractions leads to:

$$\frac{T_s(0)}{\frac{1}{\tau_2}+s} + \frac{\omega}{\tau_2}\left(\frac{c_1}{\frac{1}{\tau_2}+s} + \frac{c_2}{s+i\omega} + \frac{c_3}{s-i\omega}\right)$$

where, from Kreyszig [1962] p. 219, and with $\tau = \tau_2$ so the equations can be reused later:

$$c_1 = \frac{1}{\omega^2 + (1/\tau)^2} = \frac{\tau^2}{1+\omega^2\tau^2} \tag{2}$$

$$c_2 = -\frac{1}{2i\omega(1/\tau - i\omega)} = \frac{\tau(i-\omega\tau)}{2\omega(1+\omega^2\tau^2)} \tag{3}$$

$$c_3 = \frac{\tau(-i-\omega\tau)}{2\omega(1+\omega^2\tau^2)} \tag{4}$$

The solution is then obtained by taking the inverse Laplace transform:

$$T_s(t) = T_s(0)e^{-t/\tau_2} + \left(\frac{\omega\tau_2}{1+\omega^2\tau_2^2}e^{-t/\tau_2} + \frac{(i-\omega\tau_2)}{2(1+\omega^2\tau_2^2)}e^{-i\omega t} + \frac{(-i-\omega\tau_2)}{2(1+\omega^2\tau_2^2)}e^{i\omega t}\right) \tag{5}$$

After initial transient response decays, the long-time solution is then

$$T_s(t) = \frac{1}{(1+\omega^2\tau_2^2)}\left(-\omega\tau_2\cos\omega t + \sin\omega t\right) = b\sin(\omega t+\zeta) = b(\cos\zeta\sin\omega t + \sin\zeta\cos\omega t) \tag{6}$$

which leads to the solution for b and ζ:

$$b\cos\zeta = -\frac{1}{1+\omega^2\tau_2^2}$$

$$b\sin\zeta = \frac{-\omega\tau_2}{1+\omega^2\tau_2^2}$$

$$\tan\zeta = -\omega\tau_2 \tag{7}$$

$$b = \frac{1}{1+\omega^2\tau_2^2}\sqrt{(1+\omega^2\tau_2^2)} = \frac{1}{\sqrt{1+\omega^2\tau_2^2}} \tag{8}$$

This illustrates the use of the Laplace-transform solution that will next be applied to (4) from the TN. A specific input, $T(t) = \sin\omega t$, has been used, but a more general solution would replace the Laplace transform of this input ($\mathcal{L}(T) = \omega/(s^2+\omega^2)$) with the general form $\mathcal{L}(T)$ in (1). The transfer function in Laplace-transform notation then is the ratio of the Laplace transform of the output to that of the input, with initial-condition-transients neglected; in this case the transfer function is $G(s) = 1/(1+s\tau_2)$. For a general input function, this leads to the Laplace transform of the response via $\mathcal{L}(T_s) = G(s)\mathcal{L}(T)$, so the inverse transform of this equation will specify the solution for a general input. For example, for $T = A\cos\omega t$ with Laplace transform $\mathcal{L}(T) = As/(s^2+\omega^2)$, $\mathcal{L}(T_s) = As/\left((1+s\tau_2)(s+i\omega)(s-i\omega)\right)$ or

$$\frac{\tau_2}{A}\mathcal{L}(T_s) = s\left(\frac{c_1}{s+1/\tau_2} + \frac{c_2}{s+i\omega} + \frac{c_3}{s-i\omega}\right)$$

with coefficients c_i as found above. Apart from an initial decaying exponential, the solution is obtained from the inverse Laplace transform:

$$\begin{aligned} T_s(t) &= A\left(-\frac{i\omega(i-\omega\tau_2)}{2\omega(1+\omega^2\tau_2^2)}e^{-i\omega t} + \frac{i\omega(-i-\omega\tau_2)}{2\omega(1+\omega^2\tau_2^2)}e^{i\omega t}\right) = \left(\frac{A}{1+\omega^2\tau_2^2}\right)(\cos\omega t - \omega\tau_2\sin\omega t) \\ &= \frac{A}{1+\omega^2\tau_2^2}\cos(\omega t+\zeta) \end{aligned}$$

where the gain is $1/(1+\omega^2\tau_2^2)$ and the phase shift is $\tan\zeta = \omega t$.

As a check, the solution can also be obtained as follows:

$$b\omega\cos(\omega t+\zeta) = \frac{\sin\omega t - b\sin(\omega t+\zeta)}{\tau_2}$$

$$b\omega\tau_2[\cos\omega t\cos\zeta - \sin\omega t\sin\zeta] = \sin\omega t - b[\sin\omega t\cos\zeta + \cos\omega t\sin\zeta]$$

Rearranging:

$$[1 - b\cos\zeta + b\omega\tau_2\sin\zeta]\sin\omega t = b[\sin\zeta + \omega\tau_2\cos\zeta]\cos\omega t$$

This can only be valid if each term in square brackets is zero. The right-side term leads to a simple result for the phase:

$$\tan\zeta = -\omega\tau_2 \tag{9}$$

The amplitude-ratio b then follows from the left-side term:

$$b = \frac{1}{(\tan\zeta\sin\zeta + \cos\zeta)} = \cos\zeta = 1/\sqrt{1+\omega^2\tau_2^2} \tag{10}$$

The support temperature also can be written as follows: $T_s(t) = b(\cos\zeta \sin\omega t + \sin\zeta \cos\omega t)$.

Once b and ζ are known functions of frequency, they can be used in (4) from the TN to find the amplitude c and phase ϕ of the response $T_m(t)$.

The same approach can then be applied to the equation for the sensor response $T_m(t)$ in response to a real temperature $T(t) = \sin\omega t$. The Laplace transform of TN Eq. (4) leads to

$$\tau_1(s\mathcal{L}(T_m) - T_m(0)) = a\mathcal{L}(T) + (1-a)b(\cos\zeta\frac{\omega}{s^2+\omega^2} + \sin\zeta\frac{s}{s^2+\omega^2}) - \mathcal{L}(T_m) \tag{11}$$

$$\tau_1(s\mathcal{L}(T_m) - T_m(0)) = \mathcal{L}(T)\left(a + (1-a)\frac{\omega}{\tau_2}\frac{1}{\left(\frac{1}{\tau_2}+s\right)(s+i\omega)(s-i\omega)}\right) - \mathcal{L}(T_m)$$

$$\mathcal{L}(T_m) = \frac{T_m(0) + \left(a + (1-a)\frac{\omega}{\tau_2}\frac{1}{\left(\frac{1}{\tau_2}+s\right)(s+i\omega)(s-i\omega)}\right)\mathcal{L}(T)}{\tau_1(s+1/\tau_1)} \tag{12}$$

The ratio of the output to input transform, if the constant term is dropped, is

$$\mathcal{H} = \frac{a(s+\frac{1}{\tau_2})(s+i\omega)(s-i\omega) + (1-a)\omega/\tau_2}{\tau_1(s+1/\tau_1)(s+1/\tau_2)(s+i\omega)(s-i\omega)} = \frac{a}{\tau_1}\frac{1}{(s+\frac{1}{\tau_1})} + \frac{(1-a)\omega}{\tau_1\tau_2(s+1/\tau_1)(s+1/\tau_2)(s+i\omega)(s-i\omega)}$$

The inverse Laplace transform then leads to the general solution for $T_m(T)$ for sine-wave input:[1]

$$T_m(t) = \frac{T_m(0)}{\tau_1}e^{-t/\tau_1} \tag{13}$$

$$+\mathcal{L}^{-1}\left(\frac{a+(1-a)b\cos\zeta}{\tau_1}\frac{\omega}{(s+1/\tau_1)(s^2+\omega^2)}\right) \tag{14}$$

$$+\mathcal{L}^{-1}\left(\frac{(1-a)b\sin\zeta}{\tau_1}\frac{s}{(s+1/\tau_1)(s^2+\omega^2)}\right) \tag{15}$$

Use partial fractions to write, following Kreyszig [1962] p. 219,

$$\frac{1}{(s+1/\tau_1)(s^2+\omega^2)} = \frac{c_1}{s+1/\tau_1} + \frac{c_2}{s+i\omega} + \frac{c_3}{s-i\omega} \tag{16}$$

The coefficients c_i are the same as found previously, (2) to (4), except now $\tau = \tau_1$. The Laplace-transform solutions for the three terms in (16), omitting the coefficients, are respectively e^{-t/τ_1}, $e^{-i\omega t}$ and $e^{i\omega t}$. The three terms arising from representing the last term in (13) by partial fractions are the same as the previous term but multiplied by s instead of ω. Because

[1]Here the notation $\mathcal{L}^{-1}()$ denotes the inverse Laplace transform.

multiplying the Laplace transform by s corresponds to differentiation, differentiating the three terms gives $-(1/\tau_1)e^{-t/\tau_1}$, $-i\omega e^{-i\omega t}$ and $i\omega e^{i\omega t}$, apart from delta functions at $t = 0$ that integrate to constants. Then the solution, without the exponentially decaying terms, is

$$T_m(t) = \frac{a + (1-a)b\cos\zeta}{\tau_1}\omega\left(c_2 e^{-i\omega t} + c_3 e^{i\omega t}\right) + \frac{(1-a)b\sin\zeta}{\tau_1}\left(-c_2 i\omega e^{-i\omega t} + c_3 i\omega e^{i\omega t}\right) \quad (17)$$

This can be transformed to sine and cosine factors using the relationships $e^{i\omega t} = \cos\omega t + i\sin\omega t$ and $e^{-i\omega t} = \cos\omega t - i\sin\omega t$:
T_m

$$T_m(t) = \left(\frac{a + (1-a)b\cos\zeta}{1}\right)\left\{\left(\frac{(i-\omega\tau)}{2(1+\omega^2\tau^2)}\right)(\cos\omega t - i\sin\omega t) + \left(\frac{(-i-\omega\tau)}{2(1+\omega^2\tau^2)}\right)(\cos\omega t + i\sin\omega t)\right\}$$
$$+\left(\frac{(1-a)b\sin\zeta}{1}\right) i\left\{\left(\frac{-(i-\omega\tau)}{2(1+\omega^2\tau^2)}\right)(\cos\omega t - i\sin\omega t) + \left(\frac{(-i-\omega\tau)}{2(1+\omega^2\tau^2)}\right)(\cos\omega t + i\sin\omega t)\right\}$$
$$= C_1\cos\omega t + C_2\sin\omega t$$

where

$$C_1 = (a + (1-a)b\cos\zeta)\left(\frac{-\omega\tau}{1+\omega^2\tau^2}\right) + (1-a)b\sin\zeta\left(\frac{1}{1+\omega^2\tau^2}\right)$$
$$C_2 = (a + (1-a)b\cos\zeta)\left(\frac{1}{1+\omega^2\tau^2}\right) + (1-a)b\sin\zeta\left(\frac{\omega\tau}{1+\omega^2\tau^2}\right)$$

In this equation, C_1, C_2, b and ζ are all functions of frequency. It is useful to convert this solution into the form $T_m(t) = c(\omega)\sin(\omega t + \phi(\omega))$, so that the functions $c(\omega)$ and $\phi(\omega)$ represent the amplitude and phase of the response to an input of $\sin\omega t$. This then will characterize the transfer function. The result is that $c(f)\sin(2\pi ft+\phi(f)) = c(f)\left(\sin 2\pi ft\cos\phi(f) + \cos 2\pi\right.$ so $C_1 = c\sin\phi$ and $C_2 = c\cos\phi$. Thus, the amplitude and phase of the transfer function defining the response to sine waves of various frequencies are given by

$$c = \sqrt{C_1^2 + C_2^2} \quad (18)$$
$$\phi = \arctan(C_1/C_2) \quad (19)$$

The unheated Rosemount 102A2EL sensor can be represented by the parameters called "set 1" (a =0.73, τ_1 =0.03, τ_2 =0.45), as discussed in Section 3. Figure 1 shows the resulting frequency-dependent transfer function for sine-wave input.

As a check for the preceding derivation, an alternate solution was obtained as follows. Again isolate the factors multiplying $\sin\omega t$ and $\cos\omega t$:

$$c\omega\tau_1[\cos\omega t\cos\phi - \sin\omega t\sin\phi]$$

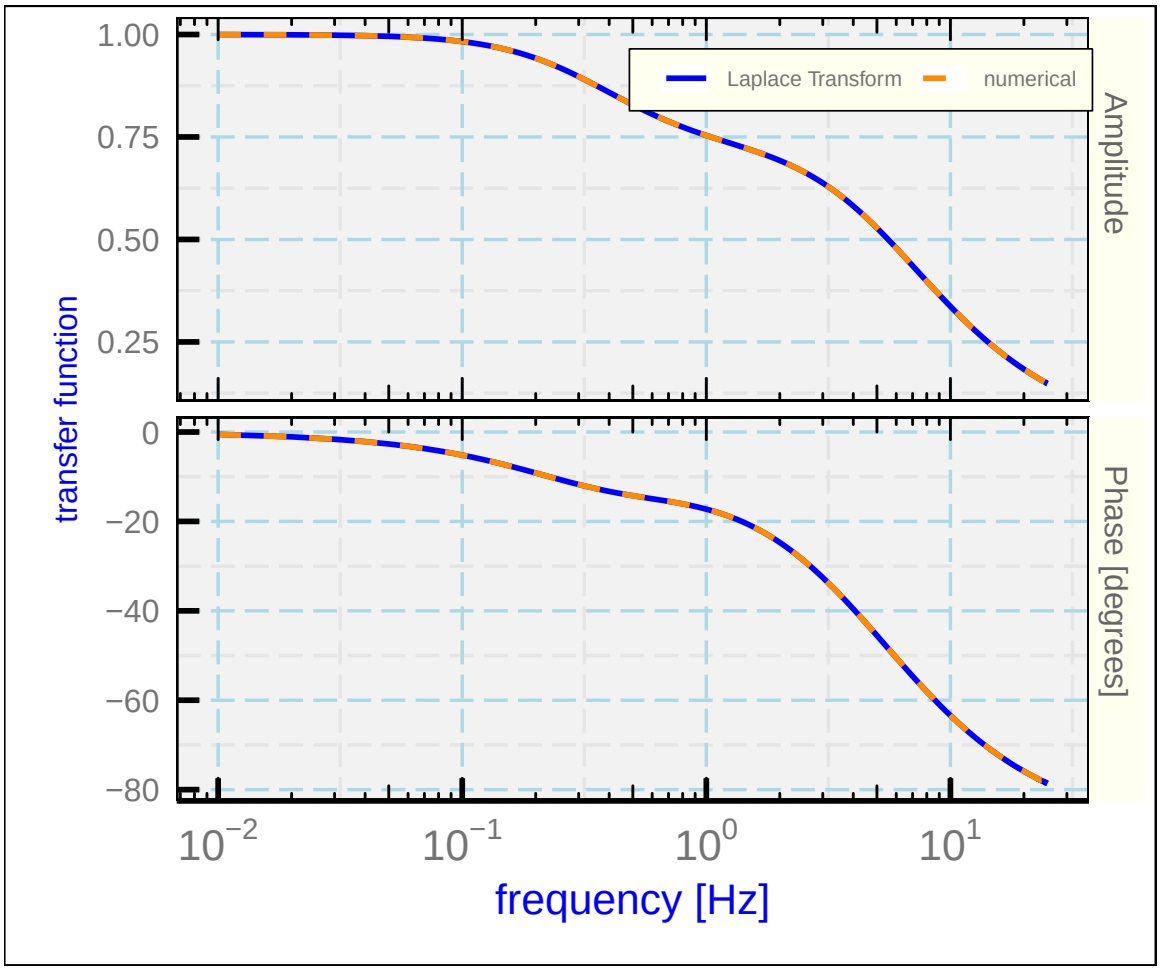

Figure 1: The amplitude and phase for the response of the Rosemount 102E4AL temperature sensor to a signal of the form $\sin(2\pi ft)$ where f is the frequency, for the Laplace-transform solution and for a numerical solution of simultaneous equations.

$$= a\sin\omega t + (1-a)b(\cos\zeta\sin\omega t + \sin\zeta\cos\omega t) - c[\cos\phi\sin\omega t + \sin\phi\cos\omega t]$$

Gathering terms:

$$[c\omega\tau_1\sin\phi + a + (1-a)b\cos\zeta - c\cos\phi]\sin\omega t$$

$$= [c\omega\tau_1\cos\phi - (1-a)b\sin\zeta + c\sin\phi]\cos\omega t$$

The simultaneous equations to be solved are then obtained by setting the factors in square brackets to zero:

$$[c\omega\tau_1\sin\phi + a + (1-a)b\cos\zeta - c\cos\phi] = 0 \tag{20}$$

$$[c\omega\tau_1\cos\phi - (1-a)b\sin\zeta + c\sin\phi] = 0 \tag{21}$$

Because it is assumed that a is known and because b and ζ are known from the solution above, the two unknowns for which (20) and (21) must be solved are c and ϕ, giving the amplitude and phase of the response to the input. They must be solved independently at each frequency because all the parameters $\{b, \zeta, c, \phi\}$ are functions of frequency. R code for this solution using the R package "nleqslv" is included in the "Rnw"-format file used to generate this memo. The results of the two methods of solution are the same to nearly the numerical precision of the software.

One final check was to simulate sine waves of various frequencies and then use a numerical solution of TN Eq. (4) to calculate the expected measurement. The results were consistent with Fig. 1; for example, the 1 Hz simulation agreed in amplitude to within 1% and in phase within 1° even for a simple Euler-method integration.

The following is verification of the solution by substituting into the differential equations. For (3),

$$b\omega\tau_2\cos(\omega t + \zeta) \stackrel{?}{=} \sin(\omega t) - b\sin(\omega t + \zeta)$$

$$b\omega\tau_2(\cos\omega t\cos\zeta - \sin\omega t\sin\zeta) \stackrel{?}{=} \sin(\omega t) - b(\sin\omega t\cos\zeta + \sin\zeta\cos\omega t)$$

$$\cos\omega t(b\omega\tau_2\cos\zeta + b\sin\zeta) \stackrel{?}{=} \sin(\omega t)(b\omega\tau_2\sin\zeta + 1 - b\cos\zeta)$$

The asserted solution gives zero for the coefficient multiplying $\cos\omega t$ on the left side. On the right side, the coefficient multiplying $\sin\omega t$ is $1 - b\cos\zeta(\omega^2\tau_2^2 + 1)$ or, using $\cos\zeta = 1/\sqrt{1+\tan^2\zeta} = 1/\sqrt{1+\omega^2\tau_2^2}$, $1 - b\sqrt{1+\omega^2\tau_2^2} = 0$ so the equality is satisfied.

For (4), the approach is similar:

$$-C_1\omega\tau_1\sin\omega t + C_2\omega\tau_1\cos\omega t$$

$$\stackrel{?}{=} \{a\sin\omega t + (1-a)b(\sin\omega t\cos\zeta + \cos\omega t\sin\zeta)\} - C_1\cos\omega t - C_2\sin\omega t$$

Again gathering coefficients multiplying $\sin\omega t$ and $\cos\omega t$ separately:

$$\cos\omega t\,(C_2\omega\tau_1 + C_1 - (1-a)b\sin\zeta) \stackrel{?}{=} \sin\omega t\,(a + (1-a)b\cos\zeta + C_1\omega\tau_1 - C_2)$$

Substituting the solutions for C_1, C_2, b and ζ gives, for the left side coefficient, after dividing by $b\cos\zeta$,

$$\left((\frac{a\sqrt{1+\omega^2\tau_2^2}}{\cos\zeta} + (1-a))\left(\frac{1}{1+\omega^2\tau_1^2}\right) - (1-a)\omega\tau_2\left(\frac{\omega\tau_1}{1+\omega^2\tau_1^2}\right)\right)\omega\tau_1$$
$$+\left((\frac{a\sqrt{1+\omega^2\tau_2^2}}{\cos\zeta} + (1-a))\left(\frac{-\omega\tau_1}{1+\omega^2\tau_1^2}\right) - (1-a)\omega\tau_2\left(\frac{1}{1+\omega^2\tau_1^2}\right)\right)$$
$$+(1-a)\omega\tau_2$$

which is zero. The right-side coefficient becomes, after similar division,

$$\frac{a\sqrt{1+\omega^2\tau_2^2}}{\cos\zeta} + (1-a) + \left((\frac{a\sqrt{(1+\omega^2\tau_2^2}}{\cos\zeta} + (1-a))\left(\frac{-\omega\tau_1}{1+\omega^2\tau_1^2}\right) - (1-a)\omega\tau_2\left(\frac{1}{1+\omega^2\tau_1^2}\right)\right)\omega\tau_1$$
$$-\left((\frac{a\sqrt{(1+\omega^2\tau_2^2}}{\cos\zeta} + (1-a))\left(\frac{1}{1+\omega^2\tau_1^2}\right) - (1-a)\omega\tau_2\left(\frac{\omega\tau_1}{1+\omega^2\tau_1^2}\right)\right)$$
$$= \frac{a\sqrt{1+\omega^2\tau_2^2}}{\cos\zeta} + (1-a)\left(1 - \frac{1+\omega^2\tau_1^2}{1+\omega^2\tau_1^2}\right) - \frac{a\sqrt{(1+\omega^2\tau_2^2}}{\cos\zeta}\left(\frac{1+\omega^2\tau_1^2}{1+\omega^2\tau_1^2}\right)$$

which is also zero. These results then verify that the differential equation is satisfied for the case where the true signal is a continuous sine wave.

5.2 Re: Section 2.2 (the transfer function):

The solution specifies the amplitude and phase of the response to a unit-amplitude sinusoidal input, and so specifies the magnitude and phase of the transfer function. This section uses assumed values for the response parameters (a, τ_1 and τ_2) as examples because those parameters have not been determined yet, but the values are those that result from subsequent analysis. The function "LTphase(f, P)" was constructed to represent the solution so that it could be used throughout the TN. Its arguments are the frequency "f", which can be a vector, and the values of the parameters, provided as "P" which should be a list with named components "a", "tau1" and "tau2." The function returns a list with two component vectors representing the amplitude and phase at the specified frequencies.

The function "LTphase()" accepts a vector argument for the frequency and returns vector arguments for the gain (or amplitude ratio) and phase. It is useful for the "LTphase()"

function to be able to accept a frequency of zero because conventional FFT results in R have the first coefficient equal to the constant for zero frequency. The function appropriately returns gain = 1 and phase = 0 for zero frequency.

The two-component faceted plot of the transfer function (TN Fig. 1) is generated by "Ranadu::ggplotWAC()" a plot function based on the "ggplot2" package (Wickham [2009]) that combines the plots of amplitude (or gain) and phase of the transfer function into one plot. See "help(ggplotWAC)" in R for information on the use of this routine to generate plots with multiple panels like this, and the "ggplot2" book in the reference list for complete information on using that package. All the plots of transfer functions in the TN rely on those routines and follow the same model used here.

At the end of TN Sect. 2.2 there is a brief discussion of how the transfer function can be used to correct the measurements to compensate for the time response of the sensor. This is included here as a brief reference because the correction procedure is used in some subsequent sections before it is developed and documented in more detail in TN Sect. 5.3.

6 Comments on TN Sect. 3

6.1 TN Sect. 3.1 (Response to dynamic heating)

6.1.1 The unheated Rosemount 102E4AL sensor

TN Sect. 3 determines the response parameters for various sensors by observing the response to fluctuations in dynamic heating. The first equation (TN Eq. (9)) is so complicated because the simple version ($\alpha_r V^2/(2C_p)$) uses the measured true airspeed (V) and the standard calculation of that involves the ambient temperature. If there are errors in the measurement of ambient temperature, they enter the estimate of Q. For that reason, the expression is revised to depend only on the absolute recovery temperature (T_r) and other factors that can be determined without reference to the ambient temperature. This makes it possible to use the measured recovery temperature as a first estimate, determine the response parameters, and then use those response parameters to correct the measured recovery temperature to obtain an improved estimate of the improved recovery temperature. Iteration of that process rapidly led to convergence at the set of response parameters found for the unheated Rosemount 102E4AL sensor.

The danger in the approach taken is that, if fluctuations in dynamic heating (via fluctuations in airspeed) are not the dominant source of variation in the measured recovery temperature (T_m) then the measured phase and amplitude ratio for the cross-spectrum of T_m and Q will be contaminated by other causes of fluctuation in the measured recovery temperature. Indeed, if such fluctuations in T_m are large compared to fluctuations in Q then the amplitude ratio determined from the variance spectra will not reflect the transfer function and cannot be used to determine it. Therefore regions and frequency intervals were sought where it appeared

that the fluctuations in T_m arose primarily from fluctuations in Q. This should be the case where the intensity of turbulence is high and would produce much larger fluctuations in T_m than the existing fluctuations in ambient temperature. This is the goal of the discussion leading to TN Fig. 4.

The selection of data and combination into a unified data set is described in the text of the TN and won't be repeated here. Some trial-and-error investigation of the number of bins to use in the construction of TN Figs. 5 and 6 led to the choice of 200 bins, which determines the number of plotted points and error bars in the plot. Although it leads to reduced clarity in the plots, a large number of bins was settled upon to avoid significant distortion of the mean when calculated for a region where the variable changes rapidly, as for example approaching 10 Hz in TN Fig. 5. Too large an interval, with the average defined at the mid-point of the interval, could distort the means toward higher values compared to the true distribution and hence would bias the determination of response variables. These plots have a more convincing appearance when a smaller number of bins is used, and become vary noisy if a much larger number of bins is used, so 200 bins was the compromise used in the TN for the unheated Rosemount sensor. Because the data sets used for other sensors were much smaller, smaller numbers of bins were used for those sensors.

A modified version of the Ranadu function "CohPhase()", named "CohP()" in the TN code, was used to combine the measurements of phase and amplitude for the flight segments used in TN Sect. 3.1.1. That function was modified to return a special data.frame containing the frequency limits of the bins and, for each frequency in the values returned by R function "spec.pgram()", the assignment of that frequency to one of the 200 logarithmically spaced bins and the phase and amplitude of the two spectra at that frequency. This function "CohP()" was called for each of the six ten-minute flight segments, with the measured recovery temperature and the expected forcing by dynamic heating (i.e., T_m and Q) and then the results from all calls were averaged in each of the frequency bins. The result was an average value (with standard deviation) for the phase and amplitude ratio, as needed to determine the transfer function. The observed phase at each bin-average frequency then was plotted in TN Fig. 5 to show the results for the phase, where in this case the measured value of the recovery temperature was "RTRR".[2] Two-standard-deviation values for the standard deviation of the mean are plotted because one-standard-deviation limits appeared hard to distinguish in this plot. The error-bar plot was generated using the ggplot2 function "geom_errorbar()" with the limits specified using the bin-average mean values and standard deviations.

The plotted "theoretical response" was generated by "LTphase()" using the standard values of the response parameters as found later by fitting to these values and the corresponding values for the amplitude ratio. The plotted frequencies are limited to those above 10^{-2}Hz because, below that frequency, bins had too few independent measurements to provide good statistics, although the available measurements below 0.01 Hz were consistent with zero

[2]At the time of the VOCALS project the recovery temperatures had names like TTRR instead of RTRR. Those names were changed subsequently to reflect that these are better described as recovery temperatures rather than total temperatures. In this analysis, the variable TTRR has been renamed RTRR to reflect the current usage. There is no variable RTRR in the referenced data files.

phase shift as would be expected from the theoretical response. The frequency interval was also limited to 12 Hz because the highest-frequency point in the plot otherwise departed significantly from the trend by being much higher (about $-43°$) and so appearing to be an outlier that should be excluded from the fit. The highest frequency included in the plot also shows some possible deviation and maybe should also be excluded.

TN Fig. 6 was generated in similar fashion, using the ratio of amplitudes (the square roots of the ratio of variance spectra) in place of the phase. For the fit the calculation of the chi-square included the deviations from the experimental phases and the experimental amplitude ratios with equal weights, except that the experimental amplitude ratios were only included for the frequency range from 0.1 to 3 Hz. This range was selected because outside that range the values departed significantly and systematically from the theoretical prediction. As explained in the TN, the R function "optim()", part of the standard "stats" package, was used for this fit. It the case of the unheated Rosemount sensor, the fit used the default Nelder-Mead minimization method. without imposed constraints on the values of the three parameters. This function returned a calculated Hessian and that was used to determine estimates of the uncertainty in the parameters. The function reported that it converged to a solution, and at the point of convergence the chi-square was reasonable at 262 for 224 degrees of freedom. In the calculation of the reported uncertainty limits the reported Hessian was divided by 4 to account for the use of two-standard-deviation standard deviations and then the square roots of the diagonal elements of the inverted Hessian matrix were reported. (This ignores significant correlations among the error estimates that perhaps should be considered and reported.)

The TN then reports on an iteration in which the measured recovery temperature was first corrected using the parameters as determine in the fit and then the fit was repeated. This process was performed outside the .Rnw code but the result did not change the fit parameters even after only one iteration so the iteration procedure was not included in the processing code.

6.1.2 The heated Rosemount sensor

A study of the heated Rosemount sensor was included in the .lyx file but excluded from the TN for these reasons:

1. The sensor is seldom used and has largely been replaced by the heated HARCO sensor.
2. The replacement HARCO is designed to have the same characteristics as the original.
3. Only a limited data set was available and it led to problematic parameters because the fit favored a value of the ratio of heat transfer to the air vs. that to the sensor (parameter a) of less than zero. As reported below, the fit procedure was changed to restrict the value of a to be positive but the fit then reported a best value of $a = 0$.

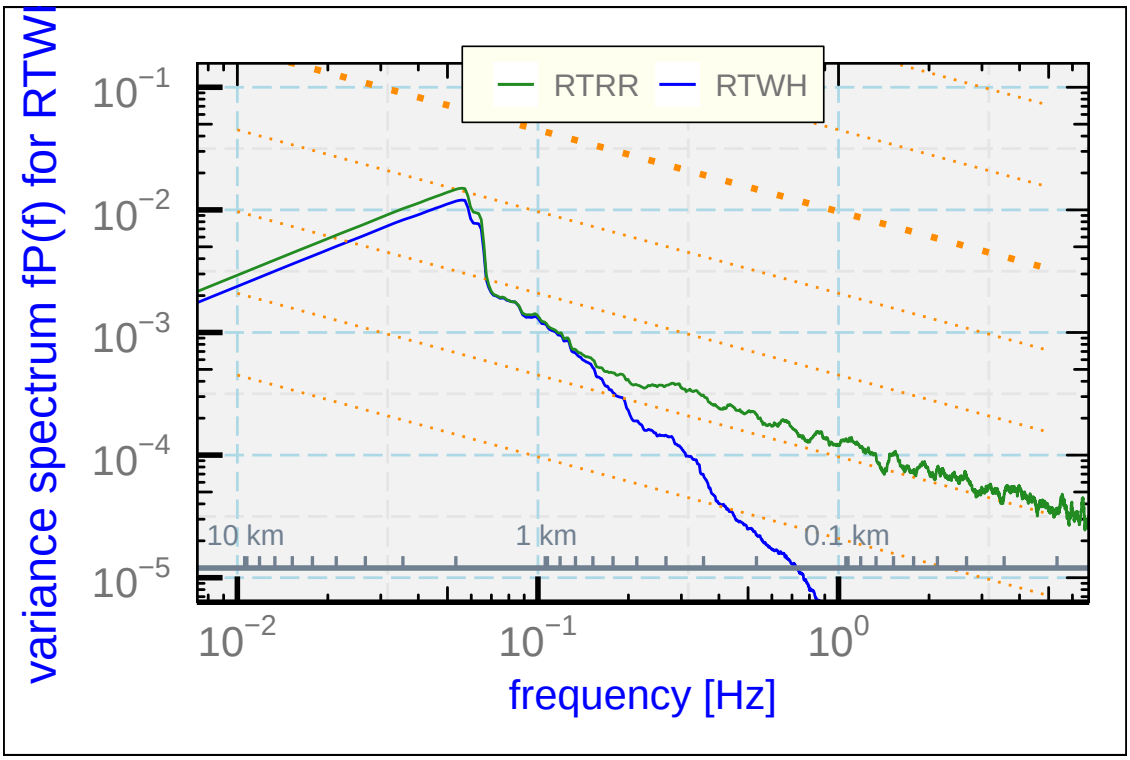

Figure 2: Variance spectrum for the recovery temperature from the heated Rosemount sensor (RTWH) and for the unheated Rosemount 102E4AL sensor (RTRR), for a low-level flight segment in the marine boundary layer.

In the .lyx file the inclusion of this material is controlled by the "Extra" branch which is not included in the TN but could be added by activating this branch. A version with this and other extra material included can be generated by the file "SensibleHeatFluxTechNoteExtra.Rnw", which is generated by LyX with the "Extra" branch activated.

Here is the text that would have been included in the TN but was excluded:

(TN Sect. 3, after 3.1.1) The heated Rosemount sensor

During the same flight used to study the unheated sensor, a heated Rosemount sensor was also present. Data were assembled as for the unheated sensor and the preceding analysis was repeated, except that two of the flight segments were omitted because they led to erratic indications of the phase. The heated sensor had almost no response to any fluctuations above 1 Hz, as shown in Fig. 1.

The measured phase between the measured recovery temperature and the calculated dynamic heating is shown in Fig. 2. Above 1 Hz, the lack of response led to erratic estimates of the phase and the coherence between the measured recovery temperature and dynamic heating was consistent with zero, so those measurements of the phase were omitted from the fit.

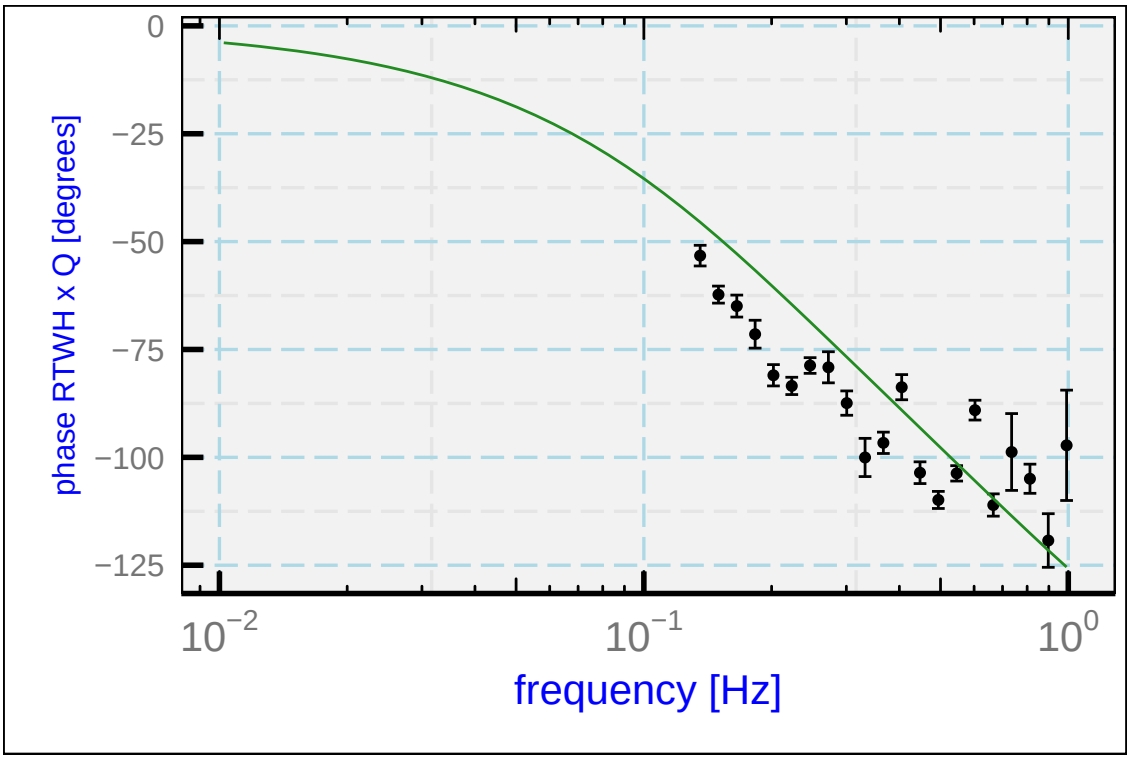

Figure 3: Phase lag of recovery temperature vs.\\ dynamic heating, for the measurements (error bars) from a Rosemount heated sensor, and the theoretical response for the best-fit parameters to the measurements with frequency below 1 Hz (green line).

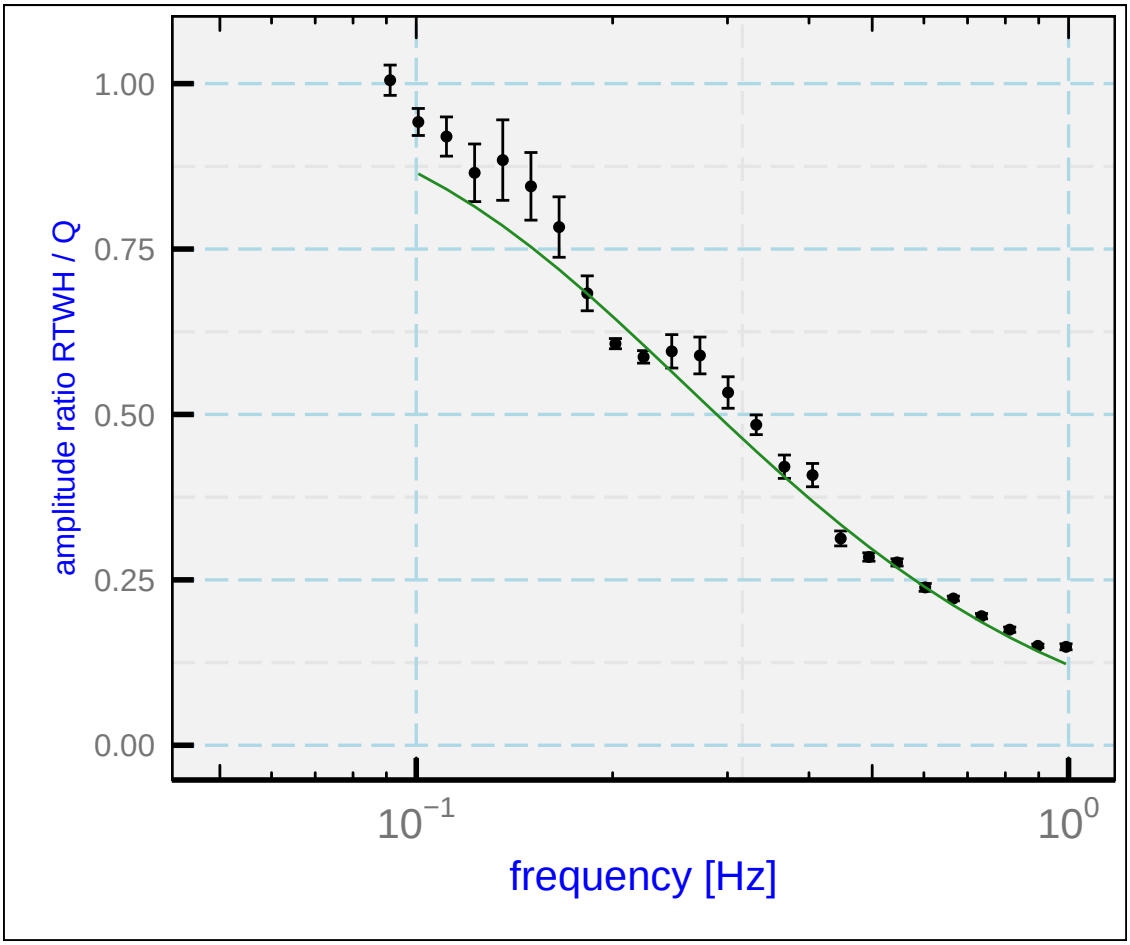

Figure 4: The ratio of the spectral amplitude for the measurement of recovery temperature ($T_m(t)$) from the heated Rosemount sensor to that for dynamic heating (Q), shown as the plotted data points. The green line is the prediction from the transfer function determined from the best-fit values matching the phase lag and amplitude ratio between these variables.

Similarly, the amplitude for frequencies above 1 Hz was not used in the fit, and measurements of amplitude for frequency below 0.13 Hz were similarly omitted because for these frequencies there appeared to be significant fluctuations in recovery temperature caused by real changes in air temperature and not primarily changes in dynamic heating. The best-fit parameters so obtained were a =0, τ_1 =0.16 s and τ_2 =0.9 s; these values are the basis for the theoretical lines in Figs. 2 and 3.[3]

The phase shift leads to these conclusions:

1. A two-time-constant formula does not provide a very good representation of the low-

[3]The fit was constrained to keep $a \geq 0$. Still smaller values of the χ^2 were obtained for negative a, but this would be inconsistent with the assumed model of heat transfer to the wire.

frequency portion of the observations that is between 0.01 and 0.04 Hz. This may indicate that still another time constant is involved in the response of this sensor and that the two-equation representation provided by (**??**) and (**??**) is incomplete.

2. The observed phase shift falls below $-90°$ above about 0.3 Hz. This is impossible for a single-time-constant sensor but is possible, as the theoretical line shows, for a two-time-constant representation of the response, so it is conclusive that at least two time constants are involved. This also indicates that wrong-sign contributions to sensible-heat flux would be measured by this sensor above about 0.3 Hz.

3. The low value of a, 0, indicates that the primary heat transfer from the wire is not to the air but to the support. For a equal to zero, the differential equations TN Eq. (3) and (4) do not separately constrain τ_1 and τ_2. The fit results show complete negative correlation for the errors in these parameters, so they cannot be determined independently. The result, however, is not equivalent to a single time constant, as demonstrated by the measured phase shifts below $-90°$. The solution to the differential equations with $a = 0$ becomes

$$C_1 = b\cos\zeta\left(\frac{-\omega\tau_1}{1+\omega^2\tau_1^2}\right) + b\sin\zeta\left(\frac{1}{1+\omega^2\tau_1^2}\right)$$

$$C_2 = b\cos\zeta\left(\frac{1}{1+\omega^2\tau_1^2}\right) + b\sin\zeta\left(\frac{\omega\tau_1}{1+\omega^2\tau_1^2}\right)$$

$$c = \sqrt{C_1^2 + C_2^2} = 2b^2\frac{1+\omega^2\tau_1^2}{(1+\omega^2\tau_1^2)^2} = 2\left(\frac{1}{1+\omega^2\tau_2^2}\right)\left(\frac{1}{1+\omega^2\tau_1^2}\right)$$

$$\tan\phi = (C_1/C_2) = \frac{-\omega\tau_1\cos\zeta + \sin\zeta}{\cos\zeta + \omega\tau_1\sin\zeta} = \frac{\tan\zeta - \omega\tau_1}{1+\omega\tau_1\tan\zeta} = \frac{-\omega(\tau_2+\tau_1)}{1-\omega^2\tau_1\tau_2}$$

These equations show that, for any solution, the values of τ_1 and τ_2 can be exchanged and the solution will remain the same. The particular solution found was influenced by the initial parameters for the search function, but reversed value for τ_1 and τ_2 could be obtained for other initial conditions. Nevertheless, it appears that the best interpretation is that the support responds more slowly, with τ_2 =0.9, and the sensing wire responds faster with time constant τ_1 =0.16.

The transfer function for this sensor is shown in Fig. 4. It does not appear to be feasible to measure sensible-heat flux with this sensor, even with the correction procedures to be developed in Section 5, because above about 1 Hz there does not appear to be enough signal to amplify by using the transfer function. Also, the phase shift is less than $-90°$ above this frequency.

(this is the end of the section re the heated Rosemount sensor, excluded from the TN.)

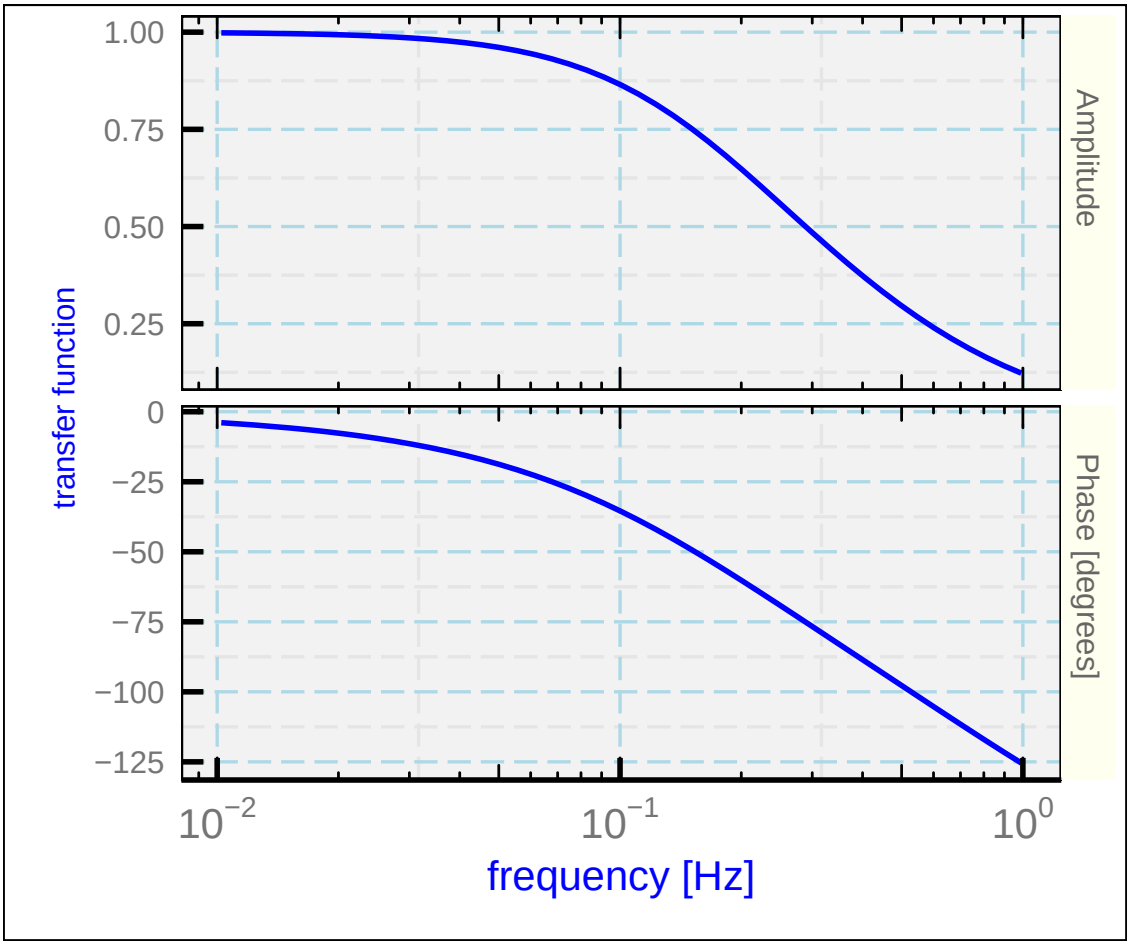

Figure 5: The transfer function for the heated Rosemount sensor on the C-130 research aircraft, based on the best-fit parameters that match the observed phase shift relative to dynamic heating.

6.1.3 Comments on TN Sect. 3.1.2, the heated HARCO sensor

The first try was based on WECAN data, but the results did not seem to give a response dominated by dynamic-heating forcing. That code still resides in the .Rnw file, but it was superceded by a selected flight segment from the SOCRATES experiment, and that is the source of the data in the TN. Procedures followed the pattern explained above for the unheated Rosemount sensor, and a transfer function for the HARCO was plotted as Fig. 10 using the same procedures.

6.2 Comments on TN Sect. 3.2, response to a step change

The VOCALS project provided a good opportunity to search for discontinuities topping the marine boundary layer because most flights included numerous climbs and descents through that structure. A large number of these were examined in a search for sharp transitions, but almost all showed evidence of mixing at the top and were not sharp as would be desired to test response to a step change. The case selected was the best case, and indeed the only good case, from about 50 examined. Some other projects were also considered, but good cases where the unheated Rosemount was in use were not found. It appears that this is a difficult way to test the response characteristics of the sensor, although other situations (terrestrial boundary layer, frontal surface, etc.) might provide better opportunities.

6.3 TN Sect. 3.3, application to a speed run

The initial approach was to shift the measurements in time to minimize the standard deviation about the fit, but the problem with this is that the Mach number varies significantly during the speed run so the appropriate time shift, according to TN Eq. (13), should also vary significantly. Therefore an alternate approach was used in which the measured recovery temperature was corrected for the sensor response. That eliminated most but not all evidence of hysteresis, as shown in TN Fig. 19. Still better fits (in terms of the residual standard deviation) were obtained if the time constants were increased slightly, both by 25%.

6.4 TN Sect. 3.4 (suppressed)

The LyX file and the "Extra" .Rnw file contain some additional material here that has been omitted from the TN because its significance was not clear and no good explanation was found for the shape of the corrected variance spectrum. It may be useful to return to this someday.

7 TN Sect. 4: Revised treatment of dynamic heating

The calculation of Q' via integration of the differential equations initially used the R function "runge.kutta()" from the "rmutil" package of Swihart and Lindsey [2019]. This implements the standard fourth-order Runge-Kutta integration and was used here with a 25 Hz time series for Q. Because the first time constant for the unheated Rosemount 102E4AL sensor is so small, the second required integration became unstable and inaccurate when used with 1 Hz measurements, as is discussed further in Appendix C. No problems were encountered in the applications in TN Sect. 4, but out of caution and because it actually executed faster the new "rk4.integrate()" function (cf. Sect. 4.5) was used for the integrations in this section.

The function "reviseDH()" was constructed that returns a new ambient-temperature variable with the dynamic-heating term adjusted as described in this section of the TN. That may eventually be modified to be more general for addition to the "Ranadu" package, but as used in the present .Rnw routine it is used in a special way: It must be called with an argument that is a data.frame containing at least TASX and ATX (airspeed and ambient temperature), a set of response parameters in a list, and a recovery facter that may be a vector matching the rows in the data.frame. It returns a revised value of ATX with the dynamic-heating term filtered as specified by the response parameters. The function should only be used on 25-Hz measurements; a revised function is developed in TN Appendix B.

Here are some details regarding the filter discussed in Sect. 4.2. The filter was specified in the code chunk labeled "designFilter" as follows, following suggestions from Press et al. [1992]:

1. Use the response parameters for the sensor (e.g., for the unheated Rosemount 102E4AL sensor) and a large set of frequencies spanning the interval from -12.5 to 12.5 Hz, e.g., with resolution between assumed frequencies of $(1/600)$ Hz, in the solution specified by TN Eqs. 5–8 to specify the transfer function. This solution is stored in a vector with frequencies in the order 0–12.5 Hz, then $(-12.5 + (1/600)$ to $-(1/600)$ Hz as needed for the Fourier transform.

2. The inverse Fourier transform then is the impulse function at 15,000 delays, many of them representing negative delays. The values in the central part of this array were mostly very small.

3. To obtain a manageable number of moving-average coefficients, all values in the array representing the impulse function were set to zero for indices k with values $M + 2 \leq k \leq N - M$ where N is the length of the calculated impulse function and $M = 100$ to leave 201 non-zero coefficients. This gave coefficients spanning about 8 s at 25 Hz, or a time long compared to the expected impulse response of the sensor.

4. The upper-100 coefficients represent negative delays in the impulse response because of the cyclic nature of the Fourier transform, so the coefficients were re-arranged into a sequence with the last-100 coefficients first and the initial-101 coefficients moved to

the end of the array. These coefficients were then moving average coefficients that implement a filter matching the transfer function, except for some omitted terms outside the 200-coefficient range that are assumed negligible.

5. The resulting set of moving-average coefficients can then be applied to the measured dynamic-heating term Q to produce a filtered version.

6. The filtered result then needs to be shifted in time by 4 s to correct for the offset in the filter.

Some auxiliary studies not documented in the TN evaluated the significance of the change in dynamic heating in terms of possible contributions to the correlation between updraft and measured temperature. Those cursory studies did not find a significant effect on measurements of sensible-heat flux because, while the error arising from using the standard processing scheme is weakly correlated with updraft in some cases, the magnitude is minor in comparison to other contributions to the flux. The comments supporting use of this revised correction anyway have been retained in the TN because these checks were not exhaustive and it could be the case that the correction will be more significant in other cases.

8 TN Sect. 5: Correcting the cospectrum

Following the introductory discussion of the cospectrum, there is a suppressed section that plots and discussed the quadrature spectrum. This section is included in the "Extra" branch so it is accessible to anyone interested, but the material and discussion did not seem be contribute to the main theme of this section in any significant way so it has been omitted from the standard branch. It is worth noting, however, that the quadrature spectrum showed more indication of significant values than might have been expected, and some aspects of that distribution were not understood or explained convincingly in the suppressed material. An alternative to the correction procedure used in this section would be to transform between the cospectrum and quadrature spectrum according to the transfer-function phase angle, and that was explored but was abandoned when it did not appear to produce results consistent with the approach taken here.

8.1 TN Sect. 5.1: Outline of the correction procedure

Two methods are discussed, but only one is used in the TN, the first of the two methods. This was coded into the "Ranadu" function "flux()" to provide an optional correction to the cospectrum. The procedure is to pass the time-response parameters to this function in a list ("Par") so that the corresponding transfer function can be generated and applied to correct the cospectrum. Only the time-response parameters from the list are used; other characteristics (b, zeta, frq) are generated in "flux()" from the time-response parameters and the frequencies generated there.

8.2 TN Sect. 5.2: Application to examples

The plots for these examples were all generated by "Ranadu::flux()", optionally with time-response parameters provided to use when calculating the corrected cospectrum. The corrected plots show the corrected cospectrum, but they also indicate (as a dashed brown line) the exceedance that was obtained before correction. For more information on "Ranadu::flux()" see R help for this function or the "flux.Rd" help file in the GitHub repository. Some additional aspects of this routine are described here because of their importance to this study:

1. The arguments include "scaleFactor", which is multiplied to the cospectrum to obtain a flux. For sensible heat, the scale factor is air density multiplied by specific heat at constant pressure ($\rho_a C_p$), as shown by TN eq. (1). This can be a vector of length equal to the number of rows in the data.frame supplied to "flux( )", and that is how it is supplied as used in the TN. It can also be a single value.

2. Units can be specified using the "bquote()" function. For the sensible-heat examples, the appropriate specification is "Units = bquote("W" ~ m ^ -2)". The units are also used in the title to the plot where the calculated flux is displayed.

3. The default number of smoothing bins (argument "smoothBins") is zero, and in that case no binning and smoothing will occur and the average-value dots will not appear on the plot. The option to plot an uncertainty ribbon ("plotRibbon") will have no effect because this depends on the binning process to determine the standard deviation that is plotted in the ribbon.

4. In addition to the plot, "flux( )" also returns a data.frame containing the cospectrum and quadrature spectrum along with the exceedance distributions (called "ogives" in that data.frame). This makes it possible to construct plots in other formats. This may be useful in cases where the logarithmic plot generated by "flux( )" is not appropriate.

5. As plotted, the cospectrum is weighted by frequency, as is appropriate for a logarithmic abscissa. Although the units are listed as supplied, the true units for a plot in this format are those units per logarithm in frequency. This weighting would preserve the relationship between area and the contribution to variance if the ordinate axis were linear, but in this case of a logarithmic ordinate axis one must take the logarithmic scale into account when visually relating area on the plot to the contribution to variance.

6. "flux( )" calls "Ranadu::CohPhase( )" to calculate the cospectrum. That routine in turn calls the "fft( )" function from the R "stats" package to calculate the Fourier transforms and then finds the cospectrum and quadrature spectrum from the product of those transforms. The scaling convention is that the integral of the variance spectrum over positive frequencies will equal the total variance.

7. The argument "Par" should only be supplied if a corrected cospectrum using those parameters should be calculated and plotted, *and at present it is specific to temperature sensors.* The "flux( )" routine calculates the solution to the differential equations using only the three response parameters (a, tau1, tau2) in "Par", which should be a list containing at least those values. The solution does not call "LTphase()" but recalculates the equation solution specified by TN Eqs. (7) and (8), and then it divides the calculated cospectrum by the gain (amplitude) of the transfer function to correct for the sensor time response. It also divides by the cosine of the phase of the transfer function.

Some of the numbers quoted don't arise directly from the code but are from special evaluations performed by replication of segments of the code. For example, the fraction of the flux missed was evaluated by running the same code without the correction procedure and noting how the measured flux changed, and in the SOCRATES case the estimate of flux from frequencies above 1 Hz was estimated similarly by special execution of the "SOCp1" code chunk with the wavelength limit changed to 125 m. For true reproducibility, these steps should have been incorporated into the .Rnw file; perhaps that can be fixed eventually.

8.3 TN Sect. 5.3: Correcting the temperature directly

The method described here is applied to 25-Hz time series and will fail if applied to 1-Hz measurements because, for the short first time constant of the unheated Rosemount 102E4AL sensor, the Runge-Kutta integration becomes unstable and generates large errors if the time step is as large as one second. Appendix B discusses an alternate integration method that can be applied to 1 Hz files.

There is a suppressed segment discussing the heated Rosemount sensor and the special considerations that arise because the best-fit value for the parameter a is zero. That segment can be retrieved by activating the "Extra" branch in the LyX file or using the appropriate .Rnw file. The problem addressed there might have some interest because the best-fit result for the heated Rosemount led to modified differential equations that needed a different method for their solution.

For Fourier transforms used here and elsewhere in the TN, the structure of the complex length-N vector produced by "fft()" is as follows:

1. The first entry is the constant zero-frequency value. For real-number input to fft(), this will be a real number (imaginary part zero).

2. The next N/2 values have positive frequencies in increments of "Rate / N" from a minimum frequency of "Rate / N" (e.g., for a 25-Hz file spanning 10 min with N = 15000, $\delta\nu = 1/600\,\mathrm{Hz}$) up to the Nyquist frequency of "Rate/2". There value at the Nyquist frequency will be real (imaginary part zero) for real input to fft(). That frequency only appears once; the value is the same for the frequencies +Rate/2 and −Rate/2 so it is not duplicated.

3. These are followed by the N/2-1 frequencies from "-Rate/2+$\delta\nu$" to $-\delta\nu$. For real-number input to the fft() function, these will be the complex conjugates of the corresponding values for positive frequencies.

To match this sequence of frequencies, transfer functions are generated at frequencies specified, for example, by statements like this:

```
f <- fft (DT$TTRR)
N <- length(f)
df <- Rate / N
frq <- c(seq(0, Rate / 2, by = df), seq(-Rate / 2 + df, -df, by = df))
```

The function generating the transfer function (here, LTphase( )) must be valid for negative and zero frequency.

9 TN Appendix A Comments

The generation of a variance spectrum with $-5/3$ slope (spectral variance vs frequency) started by using work from an earlier study, documented here. In the course of working on this TN, it became evident that there is a better way: generate a Gaussian noise spectrum (flat vs. frequency) and then modify the spectral density to have $-5/3$ slope before transforming back to the simulated measurement sequence. This was the approach used in the TN. The code is listed below because the code segment is fairly short and the approach might be useful elsewhere. This is much faster than the original method used in the TN and in the referenced document above. However, there appeared to be some advantages to the original method in terms of accuracy and consistency of results for different random sequences. Perhaps those advantages arose from generating originally at 50 Hz and then reducing the sample rate, or they may have arisen from generation as complete sine waves spanning the entire time period (which is unrealistic). In the code below, some truncation is included for frequency components smaller than 0.02 Hz, as a commented option. This seemed to improve the consistency of the results at the highest frequency and so was included in the latest simulation used in theTN.

The TN also includes commented sections to use integrations instead of filtering to predict the response of the sensor to the recovery temperature and to the dynamic heating. Those have been suppressed in favor of filtering, which provides better performance.

```
Rate <- 25
duration <- 2^16 / Rate
epsilon <- 1.e-4  ## Specify the eddy dissipation rate, mks units
V <- 200   ## assumed flight speed, m/s
```

```
## Spectral variance amplitude, lateral component
C <- (2 / 3) * (2 * pi / V) ^ (-2 / 3) * epsilon ^ (2 / 3)
Time <- seq(0, duration * Rate - 1) / Rate
N <- length(Time)
A <- sqrt(C * Rate / 2)
u <- rnorm(N, 0, A) * sqrt(3/4)  ## sqrt(3/4) to get 3:4 ratio, spectra
v <- rnorm(N, 0, A)
w <- rnorm(N, 0, A)
D <- data.frame(Time = Time, TASX = V + u, u = u, v = v, w = w)
attr(D, 'Rate') <- Rate
f1 <- fft(D$u)
f2 <- fft(D$v)
f3 <- fft(D$w)
df <- Rate / N
frq <- c(seq(0, Rate/2, by = df), seq(-Rate/2+df, -df, by = df))
f1[2:N] <- f1[2:N] * abs(frq[2:N]) ^ (-5/6)
f2[2:N] <- f2[2:N] * abs(frq[2:N]) ^ (-5/6)
f3[2:N] <- f3[2:N] * abs(frq[2:N]) ^ (-5/6)
# f1[abs(frq) < 0.02] <- 1.e-5  ## Truncate low-frequency (improves high-f accuracy)
# f2[abs(frq) < 0.02] <- 1.e-5  ## (Suppressed here but used in the TN simulation)
# f3[abs(frq) < 0.02] <- 1.e-5
D$u <- Re(fft(f1, inverse = TRUE) / N)
D$v <- Re(fft(f2, inverse = TRUE) / N)
D$w <- Re(fft(f3, inverse = TRUE) / N)
## Modify time to get a POSIXct value as expected by Ranadu:
load('chunks/Time_units.Rdata')  ## Saved from a conventional file
attr(D$Time, 'units') <- Time_units$value
tref <- sub ('seconds since ', '', attr (D$Time, 'units'))
D$Time <- as.POSIXct(D$Time, tz = 'UTC', origin = tref)
D %>% select(Time, TASX, u, v, w) %>% VSpec(method = 'MEM', WACtheme = 1)
```

10 TN Appendix B: Processing Algorithms

This Appendix mostly provides context for the correction script "ReviseT.R" that accompanies this TN. Section B.2.2, however, is new and is not addressed in the main body of the report. It is isolated to the Appendix because it applies specifically to the GV and will not be applicable more generally, although the correction technique might have other uses.

This discussion focuses on "ReviseT.R". [To be added ...]

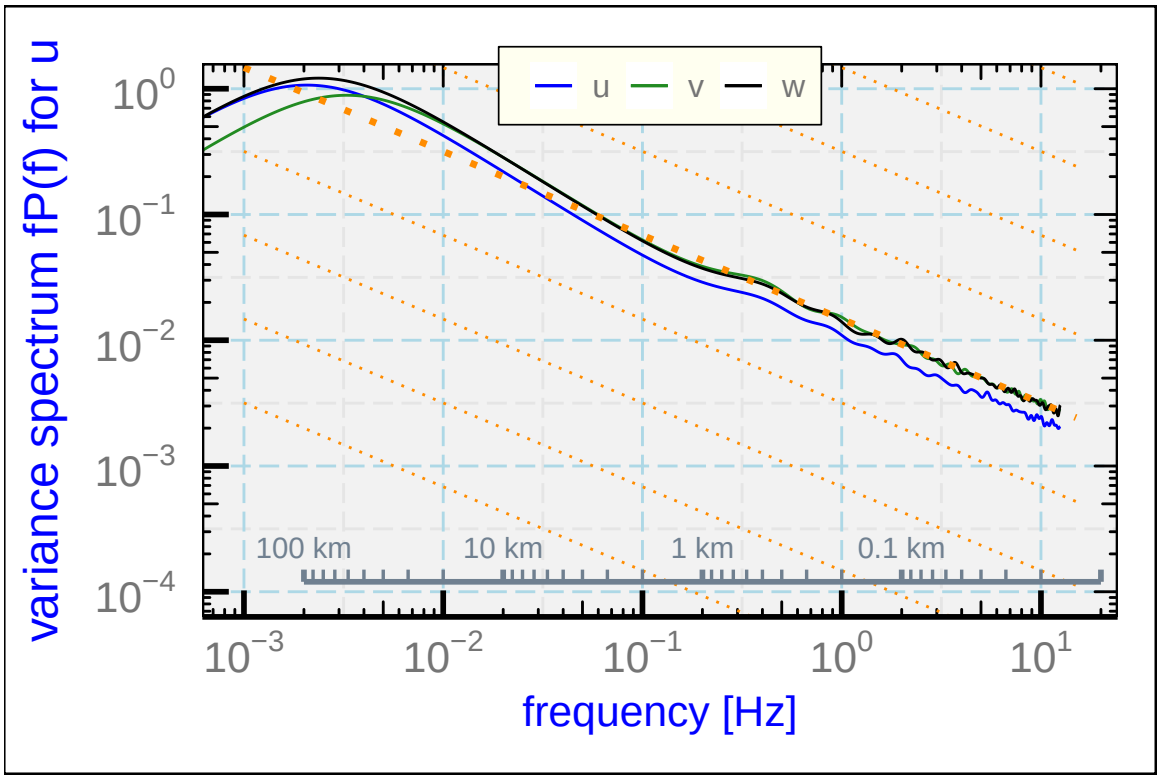

Figure 6: Variance spectra for simulated −5/3-slope variables (with 3:4 ratio between longitudinal and lateral components).

References

Jeff R Cash and Alan 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, R. B. Friesen, M. Hayman, J. B. Jensen, D. H. Lenschow, P. A. Romashkin, A. J. Schanot, S. M. Spuler, J. L. Stith, and C. Wolff. Characterization of uncertainty in measurements of wind from the NSF/NCAR Gulfstream V research aircraft. NCAR technical note NCAR/TN-528+STR, Earth Observing Laboratory, NCAR, Boulder, CO, USA, jul 2016. URL `http://n2t.net/ark:/85065/d7qr4zqr`.

E. Kreyszig. *Advanced Engineering Mathematics*. Wiley, 1962. ISBN 047133328X. URL `https://books.google.com/books?id=eAK7QgAACAAJ`.

D. H. Lenschow and P. Spyers-Duran. Measurement techniques: air motion sensing. Technical report, National Center for Atmospheric Research, 1989. URL `https://www.eol.ucar.edu/raf/Bulletins/bulletin23.html`.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C (2nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992. ISBN 0-521-43108-5.

R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL `http://www.R-project.org`.

RStudio. *RStudio: Integrated development environment for R (Version 0.98.879)*, 2009. URL `http://www.rstudio.org`.

Bruce Swihart and Jim Lindsey. *rmutil: Utilities for Nonlinear Regression and Repeated Measurements Models*, 2019. URL `https://CRAN.R-project.org/package=rmutil`. R package version 1.1.3.

H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL `http://had.co.nz/ggplot2/book`.

Y. Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2013. URL `http://yihui.name/knitr/`. ISBN 978-1482203530.

Y. Xie. *knitr: A general-purpose package for dynamic report generation in R*, 2014. URL `http://yihui.name/knitr/`. R package version 1.6.

– End of Memo –