RSI Technical Note 039

A Guide to Data Processing

(ODAS Matlab Library v4.4)

Rolf Lueck
Emma Murowinski
Justine McMillan

2020-07-27

Rockland Scientific International Inc.
520 Dupplin Rd
Victoria, BC, CANADA, V8Z 1C1
www.rocklandscientific.com

Copyright Notice

info@rocklandscientific.com
tel: +1 250 370 1688

Version History

Date	Description
2016-03-15	Initial version for ODAS Matlab Library v4.0.
2017-07-20	Updates for ODAS Matlab Library v4.2.
2018-03-30	Updates for ODAS Matlab Library v4.3.
2019-10-08	Updates for ODAS Matlab Library v4.4.
2020-03-12	Clarification in despiking section.
2020-06-18	Table 2: Fixed units of P, updated sh1 and sh2 descriptions Figure 2: Added note about assumed speed for shear signals. Added note about varying `constant_temp` to modify viscosity calculation. Changed RSI to Rockland.
2020-07-27	Table 1: Updated to include all supported vehicle types.

Contents

1 Introduction

This document will guide you through the post-cruise data processing of your Rockland Scientific data files using the ODAS Matlab Library. The functions can be used with any Rockland binary data file (`P`-file) acquired with either a real time or internally recording instrument using a `setup.cfg` configuration file. All functions described here are based on Matlab and are included in our ODAS Matlab Library v4.4. *The technical details of all functions are described in our ODAS Matlab Library v4.4 Technical User Guide.*

This technical note describes data processing in several sections. A discussion of the configuration string and depth-time history of your data file is presented in Section 2. The conversion from raw data into physical units using `odas_p2mat.m` is then described in Section 3 and data visualization using `quick_look.m` is described in Section 4. The subsequent sections outline the interpretation of results for instrument kinematics (Section 6), CT, Turbidity, and Chlorophyll sensors (Section 7), microstructure data from shear, thermistor, and micro-conductivity probes (Section 8), and their associated spectra (Section 9). In Section 10, the profile of the rate of dissipation of turbulent kinetic energy (TKE) is shown, and a summary of data structure output by `quick_look.m` is provided. Finally, some additional data processing techniques are explained in Section 11, including manipulation of hotel-files, *in situ* thermistor calibration, data despiking, and inspecting spectra.

NOTE: As of July 2019, a stand-alone software package called "Zissou Essentials" is available. This package does not require Matlab and can be used for basic data visualization and inspection. The more advanced techniques described in this technical note can only be performed, presently, using the ODAS Matlab Library and therefore Zissou Essentials is considered to be a complimentary tool to the Matlab library.

2 Before You Start

2.1 Handling the Configuration File String

The configuration file serves three purposes:

1. It contains all settings required to configure the Rockland data acquisition software for data collection. During data acquisition, a copy of the configuration file, in the form of a configuration string, is embedded in the P-file.
2. It stores information required to convert the recorded raw data into physical units.
3. It holds user provided information which is available when the data file is processed.

Configuration files are standard ASCII text files that consist of a basic structure composed of `parameters`, `sections`, and `comments`. The `parameters` store information and consist of two parts delimited by an equal sign: a `name` and a `value`.

```
name=value
```

Each line of the configuration file can only have one parameter. Groups of `parameters` are called `sections` and are identified by the `name` string enclosed by square brackets (`[]`). The declaration of a new `section` automatically ends the previous section. Parameters declared before the first section are automatically assigned to the mandatory `[root]` section. Additional mandatory sections are `[instrument_info]` and `[matrix]`. The `[instrument_info]` section has one required parameter (`vehicle`, Section 3.1), but it may also be used to provide additional information, such as the instrument model, serial number, etc. When converting the raw data into physical units, the method of estimating the speed of profiling is vehicle dependent (Table 1). The `[matrix]` section defines the channel address matrix, which specifies the order in which data channels will be sampled.

Many `sections` pertain to specific channels. Each `[channel]` section may contain two different kind of parameters: instrument-dependent parameters and sensor-dependent parameters. **Instrument-dependent parameters** depend on the internal electronics of the instrument and are only changed if a board is changed. **Sensor-dependent parameters** depend on the individual sensor that is installed for data acquisition. Each sensor has unique calibration coefficients. The configuration string must contain accurate sensor-specific and instrument-specific calibration coefficients to convert raw binary data into physical units. Please see the ODAS Matlab Library Manual v4.4 (Section 3) for an extensive discussion on the anatomy and application of the configuration file.

You may need to edit the configuration string post-cruise due to recalibration of the sensors, or if you changed probes at sea without updating the configuration file. The configuration string can be extracted from the data file using the matlab function

```
>> extract_setupstr('fname', 'setup_new.cfg')
```

producing a configuration file named `setup_new.cfg` that can be edited in a text editor. Once you have made the necessary changes to the configuration file, it can be patched back into the data file using the Matlab function

```
>> patch_setupstr('fname','setup_new.cfg')
```

The original configuration string used for data acquisition is automatically appended as comments, i.e., it can never be altered and, hence, it is never lost.

2.2 Depth-time History

Before you begin converting your data into physical units, you may wish to see an overview of the depth-time history of the data. The depth-time history of the P-file provides a birds-eye view of the file, including the length of the file, the number of profiles, and the depth of profiles. You may wish to break the data file into smaller files that contain specific portions of data. Please note that our processing software, `quick_look.m` (Section 4), is capable of identifying and individually processing profiles within your data file.

There are two functions that can be used to display the depth-time history of the data: (1) `show_P.m`, and (2) `show_ch.m`. These are described in the following subsections.

2.2.1 Using `show_P.m`

The function `show_P.m` calculates the record index and the record-average pressure from a P-file. The pressure channel must exist in the address matrix of the P-file and there must be a `[channel]` section containing the coefficients for converting raw pressure data into physical units. The returned vectors are suitable for plotting the depth-time history of the data file:

```
>> [P records] = show_P('fname');
>> plot(records, P);
```

or simply

```
>> plot(show_P('fname.p')); grid on; set(gca,'ydir','rev');
```

where `fname` is the name of the file to be processed, `P` is a vector of record-averaged pressure in physical units, and `record` is the record numbers corresponding to `P` (Figure 1). Note that in almost all cases a record is 1 second long.

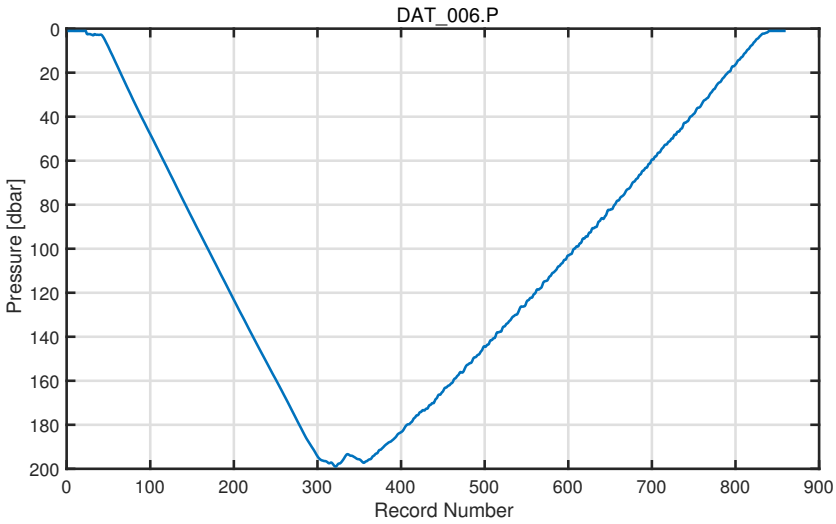

Figure 1: The pressure record of a data file plotted against record number.

2.2.2 Using `show_ch.m`

The function `show_ch.m` is more complex than `show_P.m`, in that it can extract any channel from a P-file, and hence is not limited to the pressure record. Both the channel data and associated time vector are returned and a time series figure is generated, allowing for a quick preview of channel data within a data file.

The typical syntax for `show_ch.m` is:

```
>> [ch, t] = show_ch('fname', 'channel')
```

where `fname` is the name of the file to be processed and `channel` is the name of one or more channels. If multiple channels are desired, the names need to be placed in a cell array. Note: If the function is called with only the `fname`, then all channels will be plotted. The output variables `ch` and `t`, are optional, and correspond to the extracted channel data and corresponding time vector, respectively. By default, the data are in physical units (Table 2).

The function can also be called with three or four inputs:

```
>> [ch, t] = show_ch('fname', 'channel',range,convert)
```

where `range` is a two element vector specifying the start and end time of a selected range and `convert` is a logical value (i.e. either `true` or `false`) that determines if data are deconvolved and converted into physical units.

As an example, the same depth-time history shown in Figure 1 can be generated using the command:

```
>> show_ch('DAT_006', 'P')
```

Additionally, both the pressure and the shear probe data can be plotted using:

```
>> show_ch('DAT_006',{ 'P','sh1','sh2'})
>> set(gca,'ydir','rev')
```

which is shown in Figure 2. Because the scales of the channels are often different by several orders of magnitude (i.e. the shear signals have a much smaller amplitude than the pressure signal in Figure 2), using `show_ch` to display all channels at once is not recommended for detailed data inspection[1].

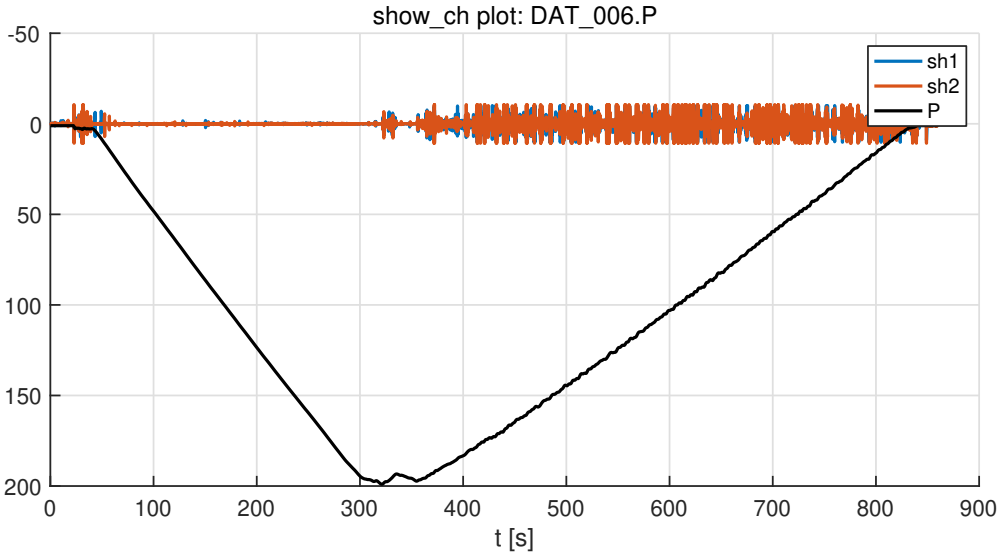

Figure 2: Time series of the pressure and shear signals generated using `show_ch`. The data are in physical units, i.e. `P` has units of dbar and `sh1` and `sh2` have units of s^{-1}. Note: A constant speed of $1\,m\,s^{-1}$ has been assumed for the conversion of the shear signals. In reality, $sh = sh_{plot}/speed^2$.

[1] Zissou Essentials is a good tool to use to assess data quality on all channels. Alternatively, Matlab can be used to generate user-specific plots after the `P`-file has been converted to a `mat`-file (Section 3).

3 Conversion to Physical Units

The function `odas_p2mat.m` uses configuration information embedded in the Rockland data file to convert the binary data into data vectors in physical units for both raw signals (Table 2) and derived signals (Table 3). The data vectors created by `odas_p2mat.m` are saved in a `mat`-file with the same name as the `P`-file. There are two options for calling `odas_p2mat.m`:

1) You can call `odas_p2mat.m` directly. Then, the generated `mat`-file can either be analyzed independently, or processed using `quick_look.m`.

2) You can call `quick_look.m`, which will automatically call `odas_p2mat.m`.

NOTE: Converting raw data into physical units uses the embedded configuration file string and the calibration parameters contained therein, so these values must be correct.

Upon being called, `odas_p2mat.m` first checks if a `mat`-file already exists with the same name and configuration information as found in the `p`-file (Figure 3). If a `mat`-file already exists with matching configuration information, then that `mat`-file is passed to `quick_look.m`. On the other hand, if a `mat` file does not exist, `odas_p2mat.m` will be called to convert the raw data into physical units using the configuration information embedded in the `P`-file. And finally, if a `mat`-file exists but with different configuration information, `odas_p2mat.m` will prompt the user with

```
>> Delete MAT file and continue?  [y,n].
```

If the user answers 'no' (i.e. `n`), then the existing `mat`-file will be used without incorporating the new configuration file. If the user answers 'yes' (i.e. `y`), then the existing `mat`-file will be deleted and the new configuration information will be used for conversion to physical units.

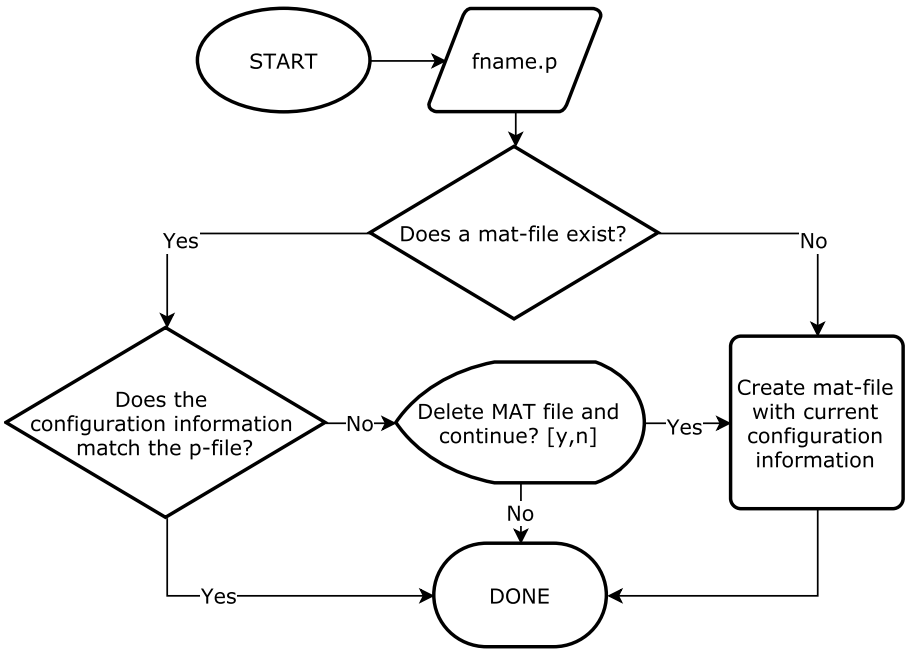

Figure 3: The logic used by `odas_p2mat.m` to determine if a `mat`-file must be generated.

The syntax for odas_p2mat.m is:

```
>> odas_p2mat( 'fname', convert_info)
```

Argument	Description
fname	Name of the file to be processed. The file extension is optional.
convert_info	Optional structure with fields that define the processing parameters (see below).

If the convert_info structure is not present when the function is called, default parameters will be used. You may explicitly create a variable with the default parameters by calling odas_p2mat.m without any input parameters, i.e.,

```
>> convert_info = odas_p2mat

convert_info =

    MF_extra_points: 0
               MF_k: 4
            MF_k_mag: 1.7000
              MF_len: 256
           MF_st_dev: 'st_dev'
        MF_threshold: []
                 aoa: []
      constant_speed: []
       constant_temp: []
        gradC_method: 'high_pass'
        gradT_method: 'high_pass'
          hotel_file: []
        speed_cutout: 0.0500
           speed_tau: []
         time_offset: 0
             vehicle: ''
```

You can then change the parameters (which are the fields within the returned structure convert_info) to values suited to your particular processing requirements.

3.1 Vehicle Specification

An instrument is identified by the vehicle parameter:

vehicle String identifying the vehicle that carries the Rockland instrument. Typically empty ('') in the convert_info parameters.

First, odas_p2mat.m searches the input arguments to identify the vehicle. If the input parameter is empty, the function will search the configuration string. The vehicle should be identified in the [instrument_info] section of the configuration-file. If the vehicle remains unspecified, it will assume that vehicle=vmp. You must specify this parameter in the input arguments only if it is not in the configuration-string (and the data is not from a downward profiling VMP), or if you want to explicitly override the value in the configuration string. This is bad practice but useful for testing purposes. Ultimately, you should correct the configuration string using the functions extract_setupstr.m and patch_setupstr.m.

The recognized `vehicle` types (Table 1) are listed in the `default_vehicle_attributes.ini` file included in the ODAS Matlab Library (Appendix A).

3.2 Profiling Speed

The parameters that are used to determine the speed of profiling are:

`constant_speed` Speed [$\mathrm{m\,s^{-1}}$] used to generate gradients and convert shear probe data into physical units. Must be empty or positive. Leave empty if a constant speed is not desired. Default = [ ].

`hotel_file` Name (string) of the `mat`-file containing speed information from a source outside of the raw Rockland data file. Default = [ ].

`speed_tau` Time-scale [s] of smoothing filter applied to speed data. Default is vehicle dependent. See the `default_vehicle_attributes.ini` file (Appendix A).

`speed_cutout` Minimum profiling speed [$\mathrm{m\,s^{-1}}$] used to convert data into physical units. Slower speeds are set to this value. Default = 0.05.

The purpose of the `speed_cutout` parameter is to avoid division by a small number. The shear and all other gradients are generated by dividing the raw data by the speed of profiling. The estimated speed is very small when an instrument is not actually profiling, such as a VMP while it is suspended near the surface before being released to fall freely. Shear probe and gradient data are meaningless during such occasions, and could be extremely large if the speed is not set to a cut-out value.

There are **three different methods to compute the profiling speed**. The order of precedence is as follows:

1) Specifying a `constant_speed` as an input parameter. A `constant_speed` is useful if you have unreliable speed data, for example, if your pressure sensor is broken or if you are profiling through large turbulent up- and down-drafts. The parameter is also useful if you have a vehicle for which it is not possible to calculate the speed using the raw Rockland data file, and you do not yet have a hotel-file.

2) Specifying a `hotel_file` that contains a structure named `speed` (See Section 11.1). This method is useful if the speed cannot be computed from the data in the Rockland datafile, or if you prefer an alternative method of estimating the speed than the default method (Table 1).

3) Using the speed algorithm implemented by Rockland that is determined by the `vehicle` parameter (Table 1). If the speed algorithm requires data vectors not available in the current data file, the required data vectors must be provided within a hotel-file.

Table 1: A table of the different vehicle types and vehicle-specific parameters. Different instruments and instrument configurations rely on different algorithms to calculate the speed of the instrument. The instrument speed is required to convert data into physical units and to calculate the dissipation rate. If your instrument does not record data that can be converted into speed, then you must specify a constant speed or a hotel-file that contains the velocity data. The parameter `speed_tau` is the time scale [s] of the smoothing filter applied to the speed data.

Vehicle type	Description	Profile dir.	`speed_tau` [s]	Default speed algorithm
`vmp`	A vertical profiler	Down	1.5	$speed = \|W\| = \partial P/\partial t$, when P is increasing.
`rvmp`	An uprising vertical profiler	Up	1.5	$speed = \|W\| = \partial P/\partial t$, when P is decreasing.
`argo_float`	A rising Argo-type vertical profiler with no pressure sensor	Up	60	Speed data is input from a hotel-file (Section 11.1).
`sea_glider`	A Seaglider with MicroPods and Datalogger	Glide	5.0	Speed data is input from a hotel-file (Section 11.1) because the sea glider configuration does not record pressure.
`slocum_glider`	A Slocum glider with MicroRider	Glide	3.0	$speed = \|W\|/\sin\theta_{glide}$, where $\theta_{glide} = \|\theta_y\| + \alpha$, $\|\theta_y\|$ is the pitch and α is the angle of attack.
`sea_explorer`	A Sea Explorer glider with Rockland instrument	Glide	3.0	$speed = \|W\|/\sin\theta_{glide}$, where $\theta_{glide} = \|\theta_y\| + \alpha$, $\|\theta_y\|$ is the pitch and α is the angle of attack.
`auv`	An AUV with integration of Rockland instrument	Horizontal	10.0	Speed data is input from a hotel-file (Section 11.1).
`auv_emc`	An AUV with integration of Rockland instrument and an EM current meter	Horizontal	10.0	Speed data obtained from the EM current meter.
`nemo`	Nemo float with MicroRider	Horizontal	60.0	Speed data is input from a hotel-file (Section 11.1) or Vector current meter.
`micro_squid`	A MicroSquid eddy correlation system	Horizontal	1.5	Speed data is input from a Vector current meter.
`stand`	A Rockland instrument mounted on a stand	Horizontal	1.5	Speed data is assumed to be a constant at $0.5\,\mathrm{m\,s^{-1}}$. This can be modified by changing the `constant_speed` input parameter (Section 3.2) or by using a hotel-file (Section 11.1).
`xmp`	Expendable microstructure profiler	Down	1.5	$speed = \|W\| = \partial P/\partial t$, when P is increasing.

3.3 Median Filter

A median filter is used to remove flyers, and other erroneous samples, from ADV (Acoustic Doppler Velocimeter) velocity data. Most vertical profilers do not have an ADV and thus you can ignore this subsection. See the function `median_filter.m` for a more detailed description of these parameters. The median filter can be disabled by setting either `MF_threshold` or `MF_k` to `NaN` or `inf`.

`MF_len` Length of data segment [samples] used to calculate the median and standard deviation. Default = 256.

`MF_threshold` Data points are defined as bad when they differ by more than this value from the median. Default = [ ].

`MF_st_dev` An empty string, or `'std_dev'`. If `'std_dev'`, the threshold is `MF_k*std(segment)`, where each segment is of length `MF_len`.

`MF_k` The scale factor for the `MF_st_dev` option (i.e. if `MF_st_dev` is not empty). Default = 4.

`MF_k_mag` Similar to `MF_k` but applied to the magnetometer signal. Default = 1.7.

`MF_extra_points` Number of points, adjacent to bad points, to also replace with interpolated values. This parameter is used when the velocity data are over sampled. Default = 0.

3.4 Scalar Gradients

There are two possible methods for computing the gradients of the conductivity and temperature microstructure signals. The two options are `'high_pass'` and `'first_difference'`. The high pass method applies a high pass filter to the pre-emphasized temperature signal to make it a time-derivative for all frequencies. The first difference method uses the high-resolution temperature to estimate the gradient by way of the first-difference operator. See Rockland Technical Note 005 for more information.

`gradC_method` Specifies the method to use to compute the gradient of the microstructure conductivity signal. Default = `'high_pass'`.

`gradT_method` Specifies the method to use to compute the gradient of the microstructure temperature signal. Default = `'high_pass'`.

3.5 Other parameters

Other controllable parameters of `odas_p2mat.m` are:

`aoa` Angle-of-attack [degrees] for a glider. It is the difference between the glide-path angle and the pitch angle. Default = 3.

`time_offset` Time [s] added to the instrument time, which is provided in the header of every record in the `P`-file. This facilitates time synchronization with other sources of data (such as a hotel-file) and a correction of an erroneously configured instrument clock.

`constant_temp` Temperature [°C], or temperature variable, used for calculating the kinematic viscosity of water. By default, this parameter is empty and `'T2_fast'` will be used. If only `'T1_fast'` is available, then it will be used instead. A constant temperature can be used if `constant_temp` is assigned a scalar value. If temperature is unavailable, and this parameter is empty, a nominal temperature of 10 °C is used. Default = [ ].

Table 2: Partial list of channel `id`s and their corresponding sensors and signals. For each channel, the sampling frequency, f_s, is listed as 'fast' or 'slow' and the actual frequency is calculated in the configuration file. Note: Not all instruments sample on all channels and channel assignments may be different depending on instrument configuration. For example, channels 80, 81 and 82 are GPIO board channels that can be used for a wide range of optional sensors.

ID	Name	Type	f_s	Units	Description
0	Gnd	gnd	slow	counts	Ground on first μASTP board
1	Ax	piezo	fast	counts	Piezo-vibration sensor x-axis
2	Ay	piezo	fast	counts	Piezo-vibration sensor y-axis
4	T1	therm	slow	counts	Temperature without pre-emphasis
5	T1_dT1	therm	fast	counts	Temperature with pre-emphasis
6	T2	therm	slow	counts	Temperature without pre-emphasis
7	T2_dT2	therm	fast	counts	Temperature with pre-emphasis
8	sh1	shear	fast	s^{-1}	Along-path gradient of u_1 (i.e. ∇u_1)
9	sh2	shear	fast	s^{-1}	Along-path gradient of u_2 (i.e. ∇u_2)
10	P	poly	slow	counts	Pressure without pre-emphasis
11	P_dP	poly	slow	counts	Pressure with pre-emphasis
12	PV	poly	slow	V	Voltage across pressure transducer
15	EMC_Cur	voltage	fast	V	EMC drive current reference signal
16,17	SBT1	sbt	slow	°C	Temperature from Sea-Bird SBE3F
18,19	SBC1	sbc	slow	mS cm^{-1}	Conductivity from Sea-Bird SBE3F
32	V_Bat	voltage	slow	V	Voltage of the main power supply
40	Incl_Y	inclxy	slow	°	θ_y, inclinometer; pitch
41	Incl_X	inclxy	slow	°	θ_x, inclinometer; roll
42	Incl_T	inclt	slow	°C	Temperature of inclinometer
48, 49	JAC_C	jac_c	slow	mS cm^{-1}	Conductivity from JAC CT
50	JAC_T	jac_t	slow	°C	Temperature from JAC CT
52	Chlorophyll	poly	fast	ppb	Chlorophyll from JAC Fluorometer
53	Turbidity	poly	fast	FTU	Turbidity from JAC Turbudity
64	C1	ucond	slow	counts	Conductivity without pre-emphasis
65	C1_dC1	ucond	fast	counts	Conductivity with pre-emphasis
66	C2	ucond	slow	counts	Conductivity without pre-emphasis
67	C2_dC2	ucond	fast	counts	Conductivity with pre-emphasis
80	Mz	magn	slow	µT	Vertical component of magnetic field
81	My	magn	slow	µT	Horizontal component of magnetic field
82	Mx	magn	slow	µT	Horizontal component of magnetic field
144	U_EM	aem1g_d	slow	$m\,s^{-1}$	Current speed from JAC EM digital output
255	ch255	–	slow	–	7FF0 hexi-decimal, 32 752 Decimal

Table 3: Derived signals and parameters in the mat-files. These are in addition to the ones listed in Table 2, and are instrument dependent.

Name	f_s	Units	Description
cfgobj	–	–	Parsed configuration file. (Structure)
C1_fast	fast	mS cm^{-1}	High-resolution conductivity.
C1_slow	slow	mS cm^{-1}	High-resolution conductivity.
date	–	–	Completion date of first record (String)
Day	–	–	Day of completion of first record.
filetime	–	–	Start time of the file in Matlab time format.
fs_fast	–	s^{-1}	Sampling rate of fast channels.
fs_slow	–	s^{-1}	Sampling rate of slow channels.
fullPath	–	–	Path where data is stored.
gradC1	fast	mS m^{-1}	Conductivity gradient, $\partial C_1/\partial z$.
gradC2	fast	mS m^{-1}	Conductivity gradient, $\partial C_2/\partial z$.
gradT1	fast	°C m^{-1}	Temperature gradient, $\partial T_1/\partial z$.
gradT2	fast	°C m^{-1}	Temperature gradient, $\partial T_2/\partial z$.
header	–	–	The header of every 1-second record. (Matrix)
header_version	–	–	Version of the header.
Hour	–	–	Hour of completion of first record.
input_parameters	–	–	Parameters passed to `odas_p2mat.m`. Structure.
Milli	–	–	Millisecond of completion of the first record.
Minute	–	–	Minute of completion of the first record.
Month	–	–	Month of completion of the first record.
odas_version	–	–	Version number of the ODAS Matlab library used to generate mat-file
P_fast	fast	dbar	High-resolution pressure.
P_slow	slow	dbar	High-resolution pressure.
params	–	–	Parameters used by `odas_p2mat.m`. (Structure)
Second	–	–	Second of completion of the first record.
setupfilestr	–	–	Information from the setup file. (String)
speed_fast	fast	m s^{-1}	The speed of the instrument (as per Table 1).
speed_slow	slow	m s^{-1}	The speed of the instrument (as per Table 1).
t_fast	fast	s	Fast sampling time since the start of the file.
t_fast_YD	fast	day	Fast sampling time in year-day.
t_slow	slow	s	Slow sampling time since the start of the file.
t_slow_YD	slow	day	Slow sampling time in year-day.
temperature_fast	fast	°C	Temperature to calculate kinematic viscosity.
time	–	–	Completion time of first record. (String)
T1_fast	fast	°C	High-resolution temperature.
T1_slow	slow	°C	High-resolution temperature.
T2_fast	fast	°C	High-resolution temperature.
T2_slow	slow	°C	High-resolution temperature.
vehicle_info	–	–	Vehicle-specific info used by `odas_p2mat.m`. (Structure)
W_fast	fast	m s^{-1}	The rate of change of pressure, W.
W_slow	slow	m s^{-1}	The rate of change of pressure, W.
Year	–	–	Year of completion of first record.

4 Data Visualization & Dissipation Calculations

The Matlab function, `quick_look.m`, is a Rockland turnkey function that provides data visualization and processing for the contents of a Rockland data file. The function works with all vertical profilers and for a MicroRider mounted on a Slocum glider, a Sea-glider and a Sea Explorer. As of version 4.4 of the ODAS Matlab library, it also works for data collected with horizontal profilers, such as an AUV, a moored float or a fixed instrument on a stand.

The `quick_look.m` function serves three main purposes. It (1) graphically displays the contents of a Rockland data file, (2) computes spectra for a selected depth or time range, and (3) returns a profile of the rate of dissipation of turbulent kinetic energy. The function performs these tasks in three stages (Figure 4).

The first step, if necessary, is for data to be converted into physical units using `odas_p2mat.m`[2] (Section 3). The data from a requested profile[3] are then plotted in several figures. By default, the kinematic properties of the profile (Section 6), the CT data (Section 7), and the microstructure data (Section 8) are all displayed. The spectra are also plotted against frequency and wavenumber for a pre-defined pressure or time range (Section 9) and the rate of dissipation of TKE for the entire profile is generated (Section 10). In the final stage, the results are returned to the user in a data structure (Section 10).

NOTE: For vertical profilers, the data are plotted versus pressure (on the vertical axis), whereas for horizontal profilers, the time vector is displayed on the vertical axis. All other features of `quick_look.m` remain the same, unless otherwise specified.

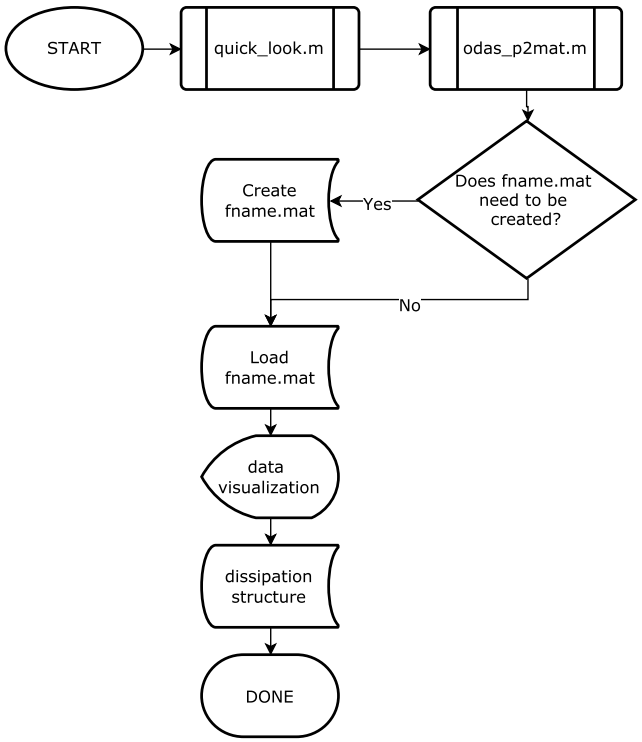

Figure 4: The stages used by `quick_look.m` when processing a data file.

[2] `odas_p2mat.m` can be called directly or implicitly by `quick_look.m`.
[3] For horizontally mounted instuments, the entire data file is processed.

To call `quick_look.m`, run the command:

```
>> diss = quick_look( 'fname', P_start, P_end, ql_info)
```

where the inputs and outputs are:

Argument	Description
`diss`	Output structure containing processed data (Section 10.1).
`fname`	Name of the file to be processed. The file extension is optional.
`P_start`	Start pressure or time (if horizontal profiler) of the data section used for spectral estimates (Section 9). This input can be empty (`[ ]`) if the spectral plots are not desired.
`P_end`	End pressure or time (if horizontal profiler) of the data section used for spectral estimates (Section 9). This input can be empty (`[ ]`) if the spectral plots are not desired.
`ql_info`	Input structure that defines variable processing parameters (see below).

The input structure, `ql_info`, contains parameters that apply to `quick_look.m` and parameters that apply to `odas_p2mat.m`. Each input parameter is unique and applies to either `quick_look.m` or `odas_p2mat.m`. The default input structure can be obtained by calling `quick_look.m` without input arguments, i.e.,

```
>> ql_info = quick_look;
```

The resulting output stucture is:

```
>> ql_info =
                HP_cut: 0.4000
                LP_cut: 30
                  YD_0: 0
              despike_A: [8 0.5000 0.0400]
              despike_C: [10 1 0.0400]
             despike_sh: [8 0.5000 0.0400]
            diss_length: 8
                   f_AA: 98
               f_limit: Inf
             fft_length: 2
              fit_2_isr: 1.5000e-05
              fit_order: 3
                goodman: 1
           make_figures: 1
                op_area: 'open_ocean'
                overlap: 4
           plot_battery: 0
        plot_dissipation: 1
         plot_kinematics: 1
           plot_rawaccel: 1
            plot_sensors: 1
            plot_spectra: 1
       plot_spectrograms: 0
           profile_min_P: 1
```

```
        profile_min_W: 0.2000
profile_min_duration: 20
         profile_num: 1
     MF_extra_points: 0
                MF_k: 4
            MF_k_mag: 1.7000
              MF_len: 256
           MF_st_dev: 'st_dev'
        MF_threshold: []
                 aoa: []
      constant_speed: []
       constant_temp: []
        gradC_method: 'high_pass'
        gradT_method: 'high_pass'
          hotel_file: []
        speed_cutout: 0.0500
           speed_tau: []
         time_offset: 0
             vehicle: ''
```

Any of the default fields can be changed to values that are appropriate for your particular application.

NOTE: The default values for each field are usually appropriate for a downward profiling VMP, and might need to be changed for a particular instrument or application.

4.1 Parameters that specify a profile

A single data file may contain multiple profiles. For example, a vertical profiler that was raised and lowered multiple times while recording data into a single file. The `quick_look.m` function detects profiles using five specifications. These are:

`profile_num` Index to the requested profile from the set of detected profiles. Default = 1.

`profile_min_P` The minimum pressure [dbar] of a profile. Default = 1.

`profile_min_W` The minimum vertical speed [dbar/s] of profiling. Default = 0.2.

`profile_min_duration` The minimum duration [s] in which the minimum pressure and speed must be satisfied. Default = 20.

The fifth specification is the `direction`, which is implicitly determined by the `vehicle` parameter. See Table 1 and the file `default_vehicle_attributes.ini` (Appendix A) for the direction of each recognized `vehicle`.

4.2 Parameters that control the calculation of dissipation rate

The rate of dissipation of turbulent kinetic energy, ϵ, is estimated using the function `get_diss_odas.m` and follows the method described in Rockland Technical Note 028. The controlling parameters are:

`diss_length` The time span [s] over which to make each estimate of the rate of dissipation. Default = 8.

`overlap` The overlap [s] of each dissipation estimate. Default = 4.

`fft_length` The length [s] of the fft-segments that will be ensemble-averaged into a spectrum. Default = 2.

`fit_2_isr` The rate of dissipation [$\mathrm{W\,kg^{-1}}$] above which the function will switch from the method of spectral-integration to the method of fitting to the inertial subrange, to estimate the rate of dissipation. Default = 1.5e-5.

`fit_order` The order of the polynomial to fit to the shear spectrum, in log-log space, to estimate the wavenumber at which the spectrum has a minimum. This is one of several constraints on the upper limit of spectral integration. Default = 3.

`f_limit` The upper frequency limit [Hz] to be used for estimating the rate of dissipation. Default = inf (no unconditional limit).

`f_AA` The cut-off frequency [Hz] of the anti-aliasing filter in your instrument. This value is instrument dependent but is almost always 98, unless you have an instrument that has been customized to sample at rates faster than 512 per second. Default = 98.

4.3 Parameters that control the processing of microstructure data

The parameters that control the processing of the microstructure signals pertain to the high-pass filtering of the shear-probe signals, and the despiking of the shear-probe, acceleration and micro-conductivity signals.

`HP_cut` The cut-off frequency [Hz] of the high-pass filter applied to the shear-probe signals. Default = 0.4.

`despike_sh` The triplet of parameters for the despiking function applied to the shear-probe signals. The first value is the threshold. The second value is the cut-off frequency [Hz] of the low-pass smoothing filter. The third value is the duration [s] of data to remove around a spike. See the function `despike.m` (Section 11.3) for more information on the parameters. You can suppress the despike function by specifying an infinite threshold, e.g. [inf 0.5 0.07]. Default = [8 0.5 0.04].

`despike_A` The triplet of parameters for the despiking function applied to the vibration sensor signals. Default = [8 0.5 0.04].

NOTE: For data collected with a glider, it may be necessary to supress despiking. The intermittent vibrations from battery movement and fin actuators create short duration vibrations that are easily confused with spikes, but such data is needed for coherent-noise removal.

`despike_C` The triplet of parameters for the despiking function applied to the micro-conductivity signals. Default = [10 1.0 0.04].

4.4 Parameters that control data visualization

The parameters that control the data visualization include flags that either suppress or display the figures. By default, profiles of the instrument kinematics and oceanographic data (CT and microstructure) will be displayed. The spectra (over a specified pressure range) and the dissipation rates (for the entire profile) will also be shown. On the other hand, plots of the battery voltage and spectrograms are suppressed by default, and will only be shown if the appropriate flags are set to `true`.

`make_figures` A flag that determines if any figures are generated. Setting this to false will speed up the data processing. Default = `true`.

`plot_battery` A flag to plot a profile of the battery voltage. Default = `false`.

`plot_dissipation` A flag to plot a profile of the dissipation rate. Default = `true`.

`plot_kinematics` A flag to generate figures of the kinematic behaviour of the vehicle. This includes the inclination, acceleration and speed of the vehicle. Default = `true`.

`plot_rawaccel` A flag to plot a profile of both the raw and de-spiked vibration sensor data. Default = `true`.

`plot_sensors` A flag to generate profiles of the data collected by the oceanographic sensors. This includes both the microstructure data and the ancillary data (e.g. JAC-CT). The figures that get generated will depend on the instrument configuration. Default = `true`.

`plot_spectra` A flag to generate spectra (both frequency and wavenumber) for the range of data specified by the input arguments. Default = `true`.

`plot_spectrograms` A flag to generate spectrograms of the accelerometer and shear data. Default = `false`.

4.5 Parameters for other purposes

`LP_cut` The cut-off frequency [Hz] of the low-pass filter applied to the microstructure profile signals, for graphical display only. It does not affect the estimation of the rate of dissipation. Default = 30.

`YD_0` The year-day subtracted from the time axis of figures. Note: This parameter is currently not used. Default = 0.

`op_area` The operational area of your instrument. Recognized values are 'open_ocean' and 'tidal_ch'. It controls the scale on certain figures. Default = 'open_ocean'.

The remaining parameters in `ql_info` are passed directly to `odas_p2mat.m` (Section 3) to convert the raw data into physical units.

5 Pressure Record and Profile Selection

The first figure generated by `quick_look.m` displays the pressure record for the entire data file and highlights the profile being analyzed. This figure is useful for ensuring that the desired profile has been selected.

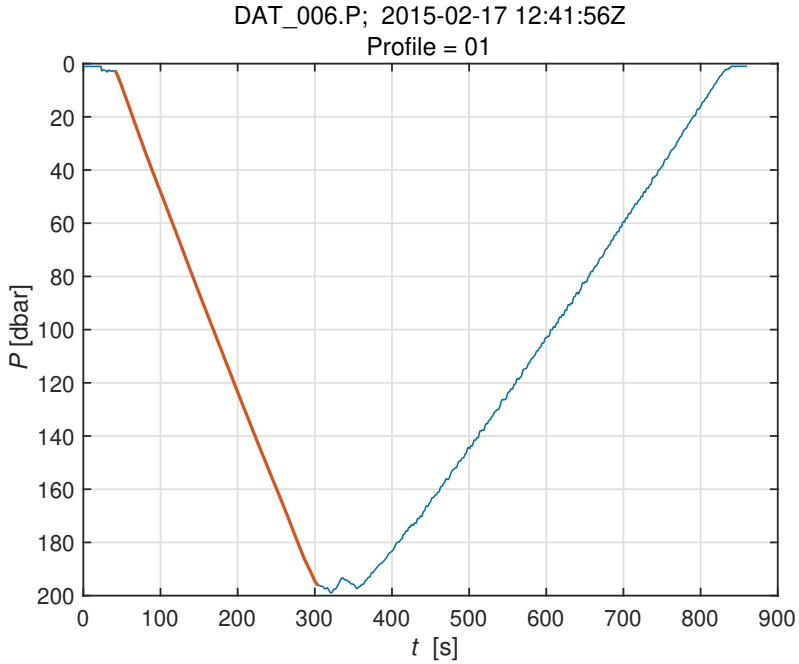

Figure 5: The pressure record for the entire data file (blue) and the specific segment being analyzed (red).

6 Kinematics

For each instrument, `quick_look.m` displays the kinematic properties of a single profile[4] by generating two figures. The first figure shows the fall rate and the inclination of the instrument (Figures 6 and 7), whereas the second shows the vibrational data (Figures 8 and 9). **A poor deployment produces poor data, so it is recommended to check the quality of your profiles while at sea, paying special attention to the kinematics displayed in these figures.**

For instruments that profile vertically (e.g. `vehicle = vmp` or `vehicle = rvmp`), the first kinematics figure contains two subpanels (Figure 6). The left panel shows the 'roll' angle, θ_x, as measured by the inclinometer. A free-falling instrument, such as a VMP, should profile through the water column without rolling more than $\pm 5°$ (Figure 6, left panel). In quiescent waters, the typical roll angle is less than $\pm 0.5°$. A large roll angle (for a VMP) could indicate that the tether line is not slack and the VMP is coupled with the motion of the deployment platform. The right panel shows the rate of change of pressure of the profiler. The rate of change of pressure is calculated by low-pass filtering the gradient of the slow pressure vector of the profile of interest. It is expected that, upon deployment, the instrument quickly accelerates to its operating speed of $0.6\,\mathrm{dbar\,s^{-1}}$ to $0.8\,\mathrm{dbar\,s^{-1}}$ (Figure 6, right panel).

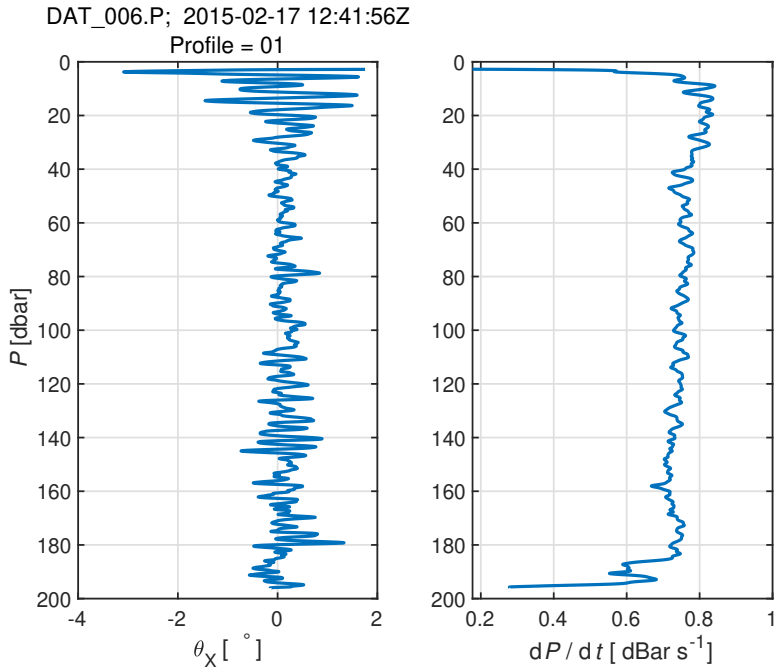

Figure 6: The roll (left) and the rate of change of pressure (right) of a vertically profiling instrument, such as a VMP.

For a MicroRider mounted on a Slocum Glider (i.e. `vehicle = slocum_glider`) the first kinematics figure contains three panels (Figure 7). In addition to the roll of the instrument, the left panel includes the instrument pitch, θ_Y. Although some rolling[5] and pitching[6] is anticipated, the inclination of the MicroRider should remain steady during the profile. The middle panel shows the rate of change of pressure (described above for a `vmp`) and the right panel

[4] See Section 4 for an explanation of how to choose the profile
[5] Roll, θ_x, is rotation around the x-axis.
[6] Pitch, θ_y, is rotation around the y-axis.

shows the estimated speed of the MicroRider. The speed is calculated using the algorithm specified in Table 1 (blue line), and – if present – an eletro-magnetic current meter (red line).

NOTE: Currently, the speed estimated from the angle of attack and the rate of change of pressure is used for data processing; the data from the electro-magnetic (EM) current meter is only shown for comparison. To explicitly use the EM sensor data, the signals need to be assigned to a `speed` vector in a hotel-file (Section 11.1).

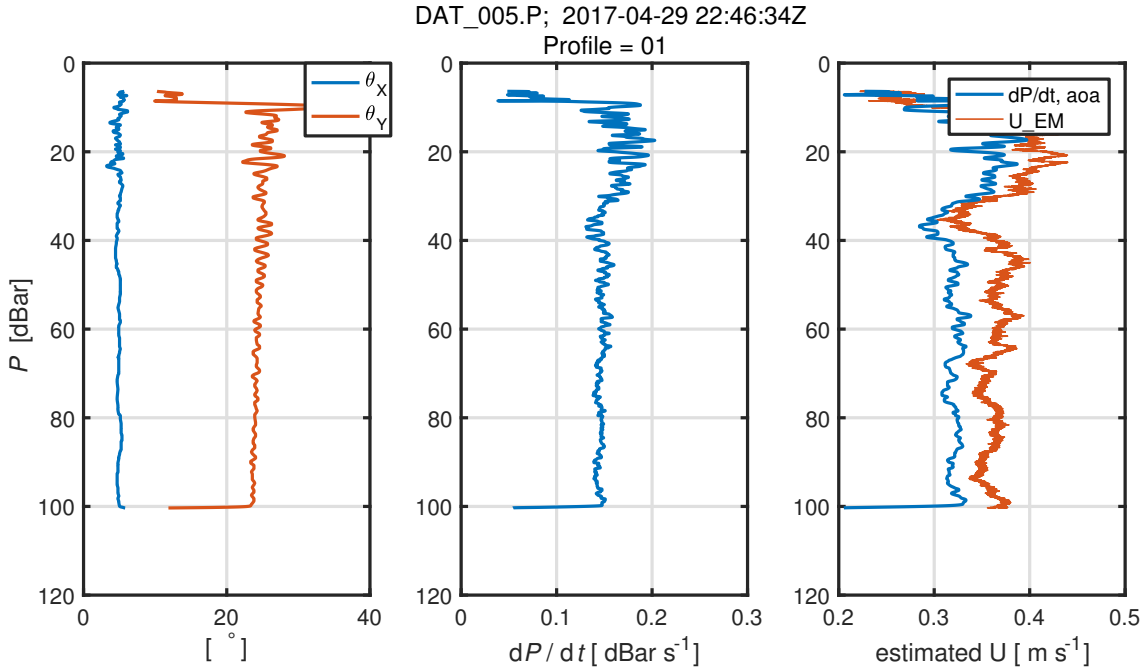

Figure 7: The kinematics of a MicroRider mounted on a Slocum glider. The subpanels show the pitch and roll (left), the rate of change of pressure (center), and the estimated speed of the glider (right).

The second kinematics figure (Figures 8 or 9, depending on the input parameters) shows the instrument vibrations as sensed by two 1-axis piezoelectric vibration sensors[7]. The vibration sensors measure vibrations in the band of 0.1 Hz to 100 Hz. By default, the vibration sensor data is shown in two subpanels (Figure 8) – the left and right panels are without and with the despiking algorithm applied, respectively. The title of the right subplot indicates the number of spikes that have been removed. The left subpanel can be suppressed by setting the `plot_rawaccel = false` (Figure 9).

NOTE: The vibration sensors are labelled `Ax` and `Ay`, but are not truly 'accelerometers'. They only sense relative vibrations and are not calibrated.

The vibration sensor data is used to remove vibrational contamination of the shear data using the "coherent noise" removal technique described by Goodman et al. (2006). The vibration data is not converted into physical units because the Goodman technique requires only a signal that is linear with respect to acceleration, therefore the data are left in terms of raw output, [counts], from the analog-to-digital converter.

[7] Older instruments may have a linear accelerometer installed. If so, the linear accelerometer data will be plotted in this figure. Check the channel descriptions in your `setup.cfg` file or contact Rockland Scientific if you are unsure of the type you have installed.

Like the examples shown in Figures 8 and 9, the instruments typically vibrate more strongly in the x-direction (blue trace)[8] than the y-direction (red trace). This is due to different stiffness of internal mechanical components in these two directions. The low-frequency vibrations (gold and purple traces in Figures 8 and 9) are similar and quite small.

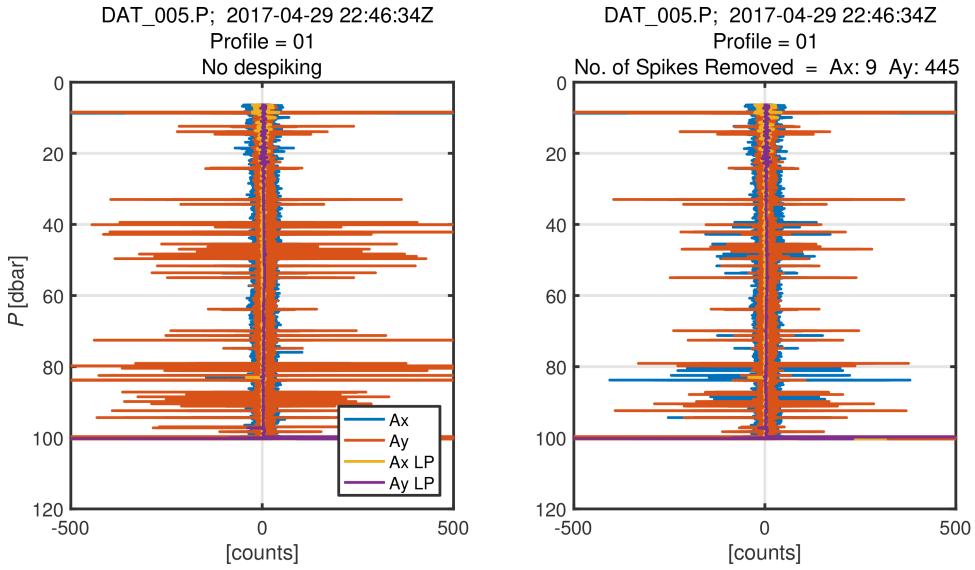

Figure 8: The vibrations in the x- (blue) and y-directions (red) directions. The x- and y-data are also low-passed at 1 Hz (yellow and purple, respectively). The raw data are shown in the left panel, and the despiked signals are shown on the right.

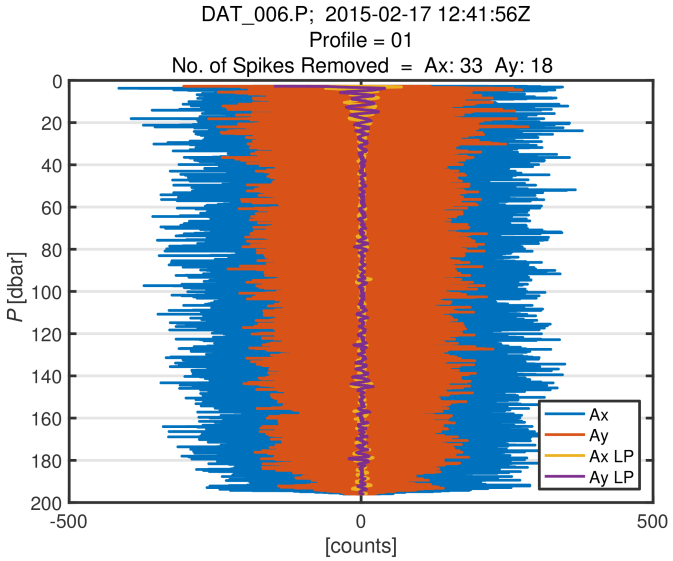

Figure 9: Vibrations in the x- (blue) and y-directions (red) after the despiking function has been applied. The number of spikes removed are indicated in the title. The x- and y-data are also low-passed at 1 Hz (yellow and purple, respectively).

[8] Please see your Instrument User Manual for the definitions of the x-, y-, and z-axis. Nominally, for a vertically profiling instrument, the z-direction is the axis of symmetry, positive in the upwards direction, the x-axis is the direction of the magnet (or pressure port), and the y-axis is other axis. The x- and y-directions are orthogonal and lie in the horizontal plane.

7 CT, Turbidity, and Chlorophyll Data

If your instrument is equipped with a conductivity-temperature (CT) sensor from SeaBird (SBE) or JFE Advantech (JAC), the `quick_look.m` function will produce a figure showing the profile of temperature, conductivity, salinity and potential density as functions of pressure (Figure 10). The potential density is computed relative to zero pressure using the measured temperature and computed salinity.

NOTE: For Seabird (SBE) sensors, the salinity profile makes no correction for the short-term mismatch of the C and T sensor, nor for the thermal inertia of the conductivity cell. For JFE Advantech (JAC) sensors the conductivity signal is lagged and low-pass filtered to try to match the apparent response times of the conductivity and temperature signals, prior to computing the salinity. The correction is speed dependent, but careful analysis of the resulting signal is required to ensure matching is accurate for your data set.

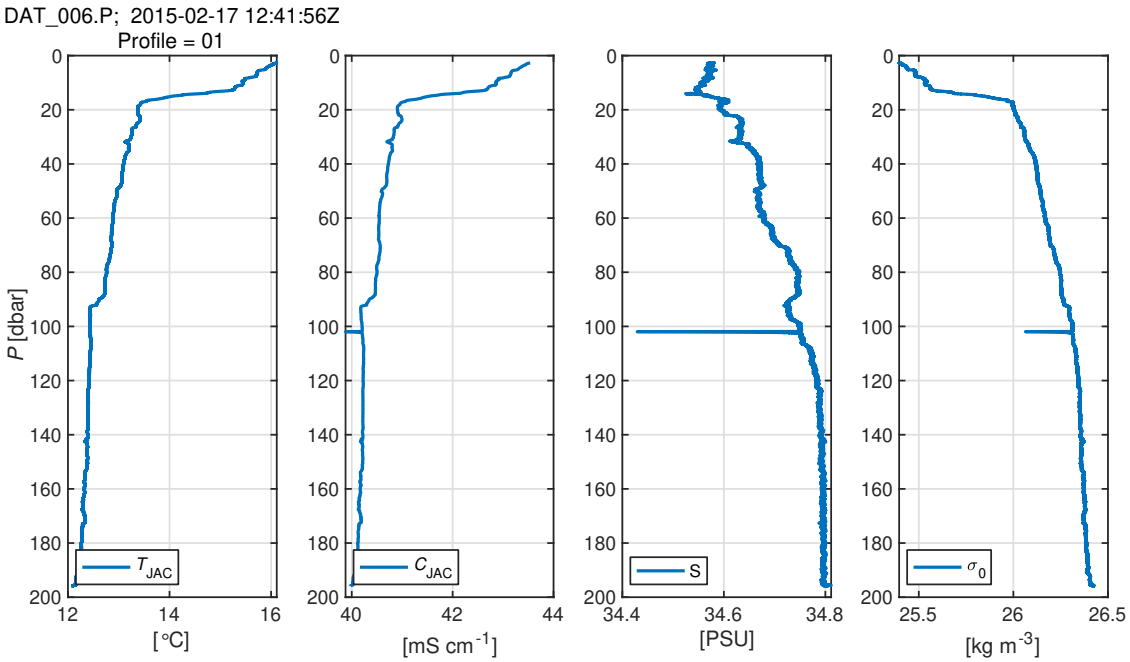

Figure 10: Data traces of the temperature (far left), conductivity (centre left), and salinity (centre right) as measured by the JFE Advantech sensor. The potential density anomaly (far right) is computed relative to zero pressure using the measured temperature and salinity. In this profile, plankton was caught in the conductivity sensor at 100 dbar, causing the spike in the conductivity and salinity traces.

Additionally, some instruments are equipped with a JFE Advantech Turbidity and Chlorophyll sensor used to take a fine vertical-resolution measure of the chlorophyll-a and particulate concentration (Figure 11). This sensor measures fluorescence expressed in units of parts per billion, i.e. ppb. The fluorescence of phytoplankton depends on the concentration of phytoplankton and its species composition. Some species fluoresce more than others even when their concentration, expressed in mass of organic carbon per unit of volume, is identical. The sensor is linear and, if the species composition is unchanged, the output is proportional to the concentration of phytoplankton. To interpret the fluorescence data in terms of phytoplankton concentration requires water samples from the depth range of a profile, and a laboratory

analysis that is beyond the scope of this document. The turbidity signal is calibrated by the manufacturer using standard methods and is displayed in units of Formazin Turbidity Unit, i.e. FTU.

NOTE: The chlorophyll and turbidity profile may appear unusual for people familiar with conventional (low spatial resolution) measurements. The unit has a vertical resolution of ~ 0.01 m and shows the micro-scale variability in the water column. The variability is easily mis-interpreted as noise.

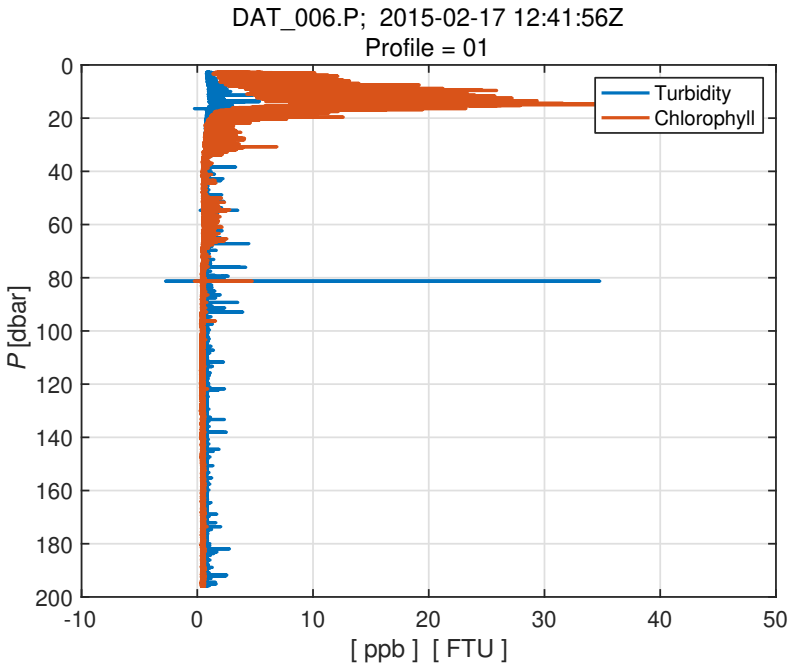

Figure 11: The vertical profile of backscatter in units of FTU (blue line) and chlorophyll in units of ppb (red line) measured with the JAC sensor.

8 MicroStructure Profiles

8.1 Thermistors

The temperature microstructure data reveal small-scale fluctuations in the vertical temperature profile (Figure 12, red and blue). The data are measured with a fast thermistor called an FP07. Under standard configuration, the fast thermistor is sampled at 512 Hz and measures temperature overturns and fluctuations down to the centimeter-scale. Temperature data from an optional conductivity-temperature (CT) sensor[9] are plotted on the same figure (yellow trace). CT sensors provide a reliable, stable temperature measurement that does not register the small-scale fluctuations. CT sensors sample at 1/8th the sampling rate of the thermistor. In our example, the thermistors have not been calibrated and thus differ from the CT data by 3 °C and 7 °C for T_1 and T_2, respectively. **The thermistor can be calibrated either at the Rockland factory or using *in situ* measurements from a CT (Section 11.2).**

NOTE: If the T2 channel is sampled (i.e. listed in the address matrix), its data will, by default, be used to estimate the kinematic viscosity, ν, which is used to calculate the dissipation rate, ϵ. If the thermistor installed in the T2 channel breaks, or a test probe is used as a placeholder, erroneous values of the temperature will yield erroneous values of ν. In these situations, the `constant_temp` field in the input structure should be used to specify a different variable for the temperature data (e.g. `T1_fast` or `JAC_T`). Example syntax when using `quick_look`:

```
>> quick_look( 'fname', P_start, P_end, 'constant_temp','T1_fast')
```

A constant temperature (in degrees Celsius) can also be specified as follows:

```
>> quick_look( 'fname', P_start, P_end, 'constant_temp',13)
```

Note: The `constant_temp` parameter can also be defined as a field in the `ql_info` structure (Section 4). It is used by the `odas_p2mat.m` function (see Section 3.5).

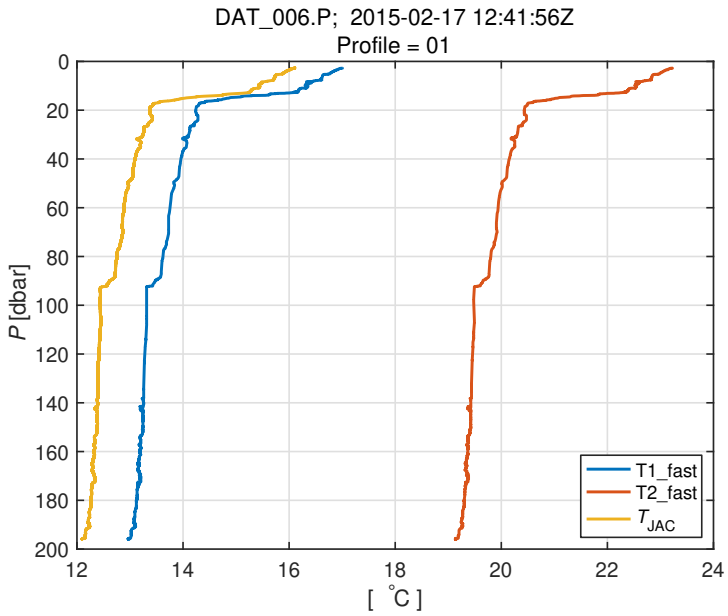

Figure 12: The vertical profiles of temperature from the two thermistors, T_1 (blue) and T_2 (red), and from the stable CT sensor (T_{JAC}, gold). The thermistors have not been calibrated.

[9] CTs are provided by JFE Advantech or SeaBird

8.2 Micro-Conductivity

The `quick_look.m` function will also show profiles of the data measured by the micro-conductivity and CT sensors (if available). The micro-conductivity data are measured with a fast micro-conductivity (a.k.a. microC) sensor, which is typically sampled at 512 Hz. Conductivity data are also acquired with an optional CT, which is sampled at 1/8th the sampling rate of the microC. The microC can be calibrated either at the Rockland factory or using *in situ* measurements from a CT sensor.

NOTE: The ODAS Matlab Library does NOT contain a function that will perform this calibration for you; however, you can choose a depth range and perform a linear regression. Use the slope to adjust the cell constant, `k`. Change the value in the configuration string and recalculate your data.

8.3 Shear and Scalar Gradients

The shear data from each shear probe are plotted twice in Figure 13 (left panel). The full bandwidth shear data are high pass filtered at a cutoff frequency of `HP_cut` (blue and red traces). These high pass filtered data are eventually used to estimate the dissipation rate (Section 10); however – for display purposes only – the shear data are also bandpassed between frequencies of `HP_cut` and `LP_cut` (gold and purple traces). By removing the high frequency signal, vertical variations in the shear signals are more visible.

The scalar microstructure data – i.e. temperature and conductivity (if available) – are plotted in two panels (Figure 13, middle and right panels). Both the measured profiles (right panel) and the along-path gradients (middle panel) help to qualitatively corroborate the shear data. In the absence of turbulence, the temperature tends to decrease with increasing depth and the scalar gradients will be small. On the other hand, in regions of high turbulence, the gradient of temperature is bipolar and rich in vertical detail. The background temperature in these turbulent regions is nearly uniform or inverted (i.e., increasing with depth) due to the presence of overturns.

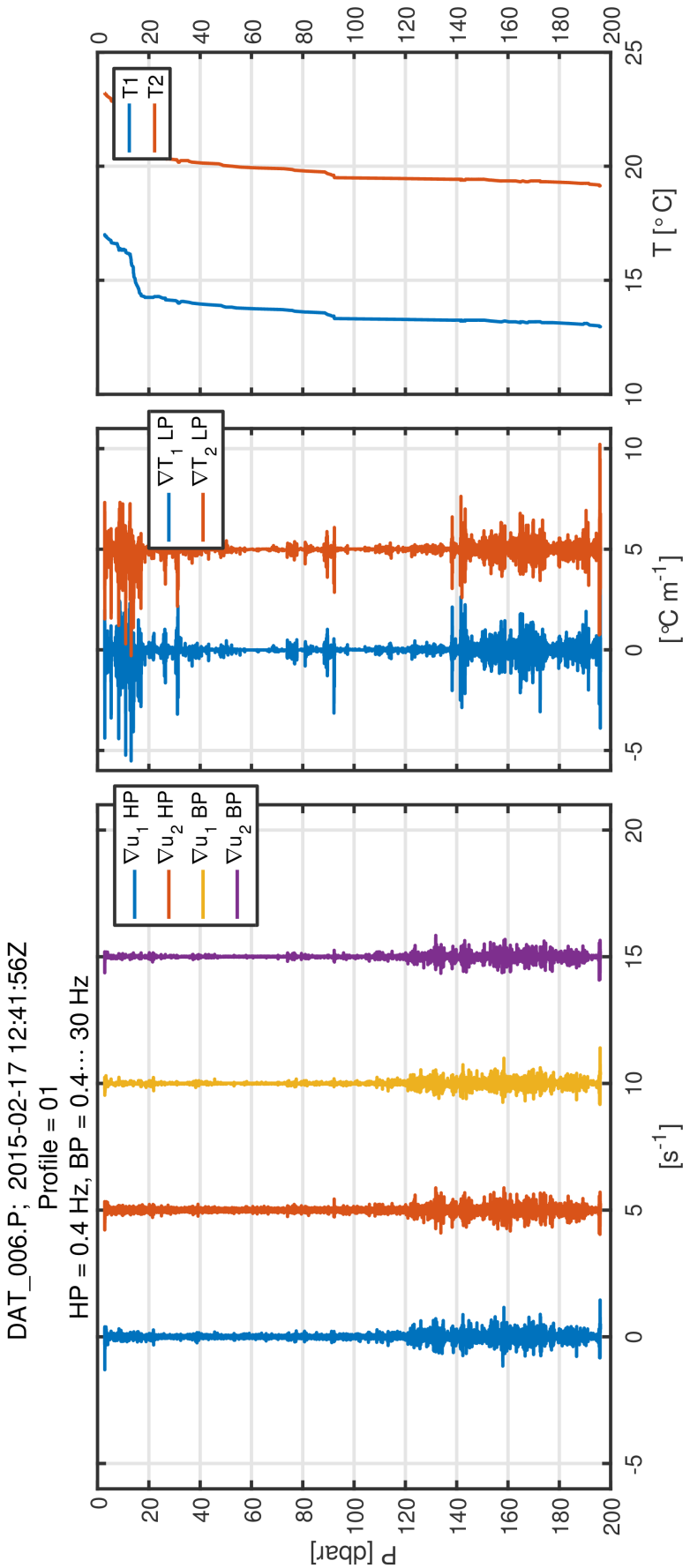

Figure 13: Left panel: The vertical profiles of data from shear probe 1 (sh1, $\partial u/\partial z$, blue and gold), and from shear probe 2 (sh2, $\partial v/\partial z$, red and purple). HP and BP indicate high- and band-pass filtering of shear probe data, respectively. Middle panel: The vertical gradients of temperature from probe 1 ($\partial T_1/\partial z$, green) and probe 2 ($\partial T_2/\partial z$, cyan). The temperature, T_1 and T_2 are based on nominal calibration coefficients in this example. All vertical gradients have a mean of nearly zero, and have been plotted with an offset of multiple units of 5.

9 Spectra

Spectra are calculated over a depth range specified in the call to `quick_look.m` (Section 4). The spectra are calculated for the vibration sensors (or accelerometers), the shear probes, and the gradients of temperature and micro-conductivity. Both the wavenumber and frequency spectra of the signals are plotted (Figure 14).

The frequency spectra (Figure 14, right panel) are computed from the measured inputs using the parameters specified in `ql_info`. The important features of the figure are:

- The vibration spectra (A_x and A_y) are not scaled to physical units. They have been reduced by a factor of 10^{-4} to be on the scale with the shear spectra. For the example shown in Figure 14, there is a moderate band of vibrations between 3 Hz to 20 Hz. Beyond 20 Hz, the vibration in the x-direction increases with increasing frequency and it peaks near 80 Hz (thick cyan line). The vibration in the y-direction does not increase until 60 Hz and reaches a lower peak (thick bright green line). Both spectra roll-off above 100 Hz due to the anti-aliasing filters in the instrument.

- The shear spectra (red and blue lines) rise in proportion to $f^{1/3}$, then fall off with increasing frequency up to about 60 Hz. Beyond this, there is a band of elevated shear in the same frequency range as the elevated vibrations. For yet higher frequencies, the shear spectra roll off monotonically to the noise level of $\sim 10^{-8}\,\mathrm{s}^{-2}\,\mathrm{Hz}^{-1}$. It should be noted that the frequency spectra of the shear data have NOT been corrected for the vibrational contamination, so peaks in the vibrational spectra will also appear in the shear spectra. On the other hand, the shear spectra have been corrected to account for the reduction in variance due to the spatial averaging imparted by the probe. This averaging occurs when the eddies are small compared to the width of the probe. The half-power wavenumber response is 48 cpm according to Macoun and Lueck (2004) and takes the form
$$H(k) = \frac{1}{1 + (k/48)^2}$$
where k is the wavenumber in cpm. The correction is applied up to the frequency corresponding to a wavenumber of 150 cpm (i.e. $f = 150U$, where U is the speed of the instrument).

- The black lines in the right panel of Figure 14 are the dimensional Nasmyth spectra for the indicated $\log_{10}\epsilon$ values. These curves are the canonical form of the shear spectrum and they provide a guide for the approximate dissipation rates.

- The temperature gradient frequency spectra (right panel, yellow and green lines) – when computed using the default `'high_pass'` method – have NO corrections applied to them. On the other hand, if the `'first_difference'` method is used, the spectra have been subjected to two frequency-dependent corrections. The first correction is applied because pre-emphasis of the thermistor occurs in the continuous domain while deconvolution occurs in the discrete domain (Mudge and Lueck, 1994). The second correction occurs because the gradient is calculated using the first difference method, which differs from a continuous-domain gradient. Note: NO correction for the frequency response of the thermistor (Vachon and Lueck, 1984) is applied to the frequency spectra when using either the `'high_pass'` or `'first_difference'` method. This differs from the wavenumber spectra (left panel) as explained below.

The wavenumber spectra (Figure 14, left panel) are created from the frequency spectra using Taylor's Frozen Field Hypothesis[10]. The important features of the plot are:

- The shear spectra (thick blue and red lines) have been cleaned to remove the vibrational contamination using the Goodman coherent noise removal algorithm. The uncorrected spectra are plotted by the thin lines for comparison. The vibrations above 80 Hz that are evident in the frequency spectra (right panel), show up at wavenumbers larger than 90 cpm. The shear signal contamination in this band is coherent with the vibration signals and is therefore removed to form the "clean" spectra of shear.
- The two black lines are the Nasmyth spectra for the estimated rates of dissipation, which are derived from an integration of the spectra from 0 cpm to the wavenumber indicated by the red and blue triangles. The upper limits of integration are given in the figure legend and equal 76 cpm and 69 cpm for sh1 and sh2, respectively. More than 90 % of the shear variance is located below these wavenumbers. The data follow the Nasmyth empirical spectrum closely for all wavenumbers up to 140 cpm and the two black curves are essentially indistinguishable, suggesting agreement between the probes.
- The temperature gradient spectra (yellow and green lines) *have been corrected minimally for the frequency response of the thermistor (Vachon and Lueck, 1984).* The correction is only approximate because the frequency response of the FP07 themistors is not well understood, and is likely sensor dependent. Due to this uncertainty, the temperature gradient spectra are not used to compute any additional parameters and are instead used as a qualitative measure to identify turbulent regions. Note: In the dissipation structure that is output from `quick_look.m` (Section 10.1.4), the wavenumber spectra are NOT corrected for the frequency response; i.e. the correction is only made for display purposes in Figure 14 (left) and is not saved to the computed values.

[10]Details on calculating the wavenumber spectra are described in Rockland Technical Note 028 - Calculating the Rate of Dissipation of Turbulent Kinetic Energy.

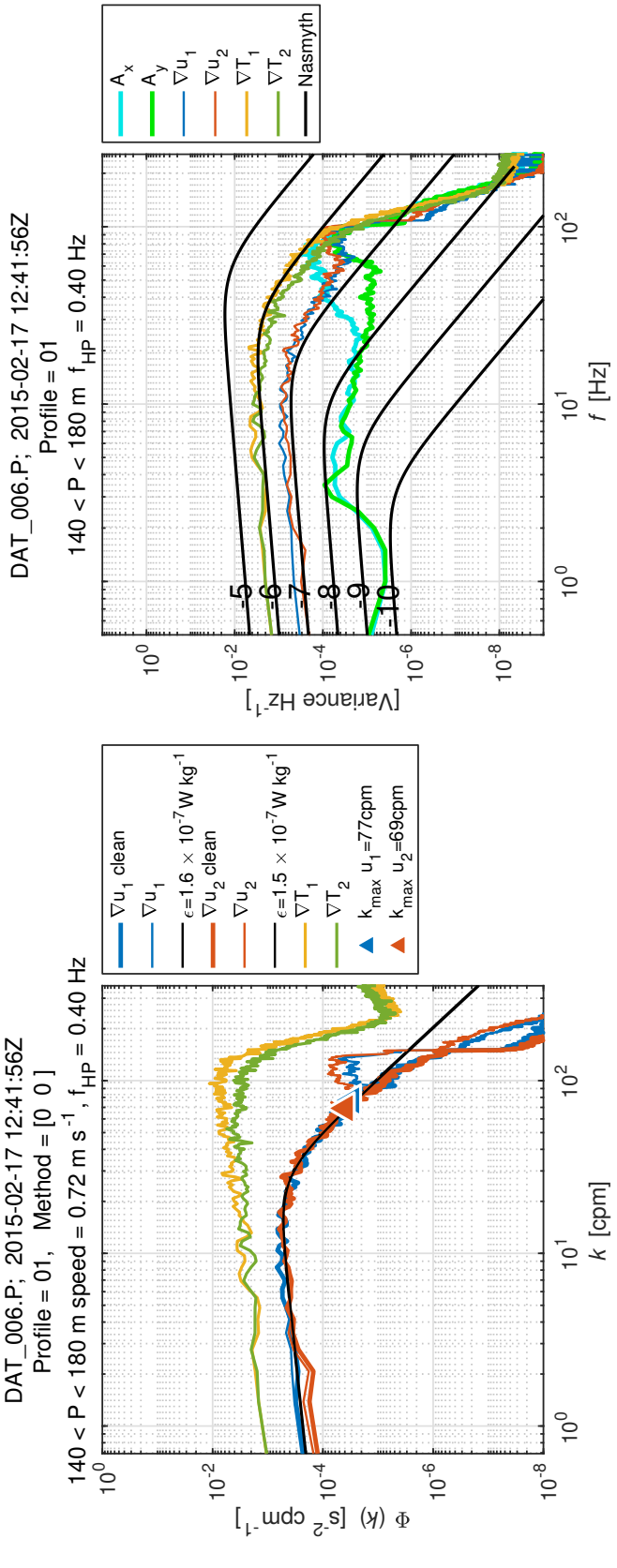

Figure 14: The wavenumber (left) and frequency (right) spectra from the depth range of 140 m to 180 m . Left panel: The thick (thin) red and blue lines are the shear spectra with (without) coherent noise correction. The triangles indicate the maximum wavenumber used for estimating the rate of dissipation, ϵ. The black lines are the Nasmyth spectra in dimensional form for the estimated rates of 1.6×10^{-7} and 1.5×10^{-7} W kg^{-1}. The temperature gradient spectra have been minimally corrected for the frequency response of the thermistors. Right panel: The frequency spectra of the vibration sensors (i.e. `Ax` and `Ay`) are shown by the thick lines. The shear and temperature gradient data are plotted in the same colours as the left panel, and the black lines show the dimensional Nasmyth spectra for the indicated $\log_{10}\epsilon$ values.

10 Rate of Dissipation of Turbulent Kinetic Energy

The rate of dissipation, ϵ, is calculated following Rockland Technical Note 028 titled "Calculating the Rate of Dissipation of Turbulent Kinetic Energy". The rate of dissipation is calculated for successive time intervals of `diss_length` (Section 4) with an overlap of `overlap` between adjacent estimates. Each estimate is composed of multiple fft calculations, each of length `fft_length`. The `quick_look.m` function displays a profile of the computed dissipation rates (Figure 15). **On average, the values for ϵ_1 and ϵ_2 should agree.**

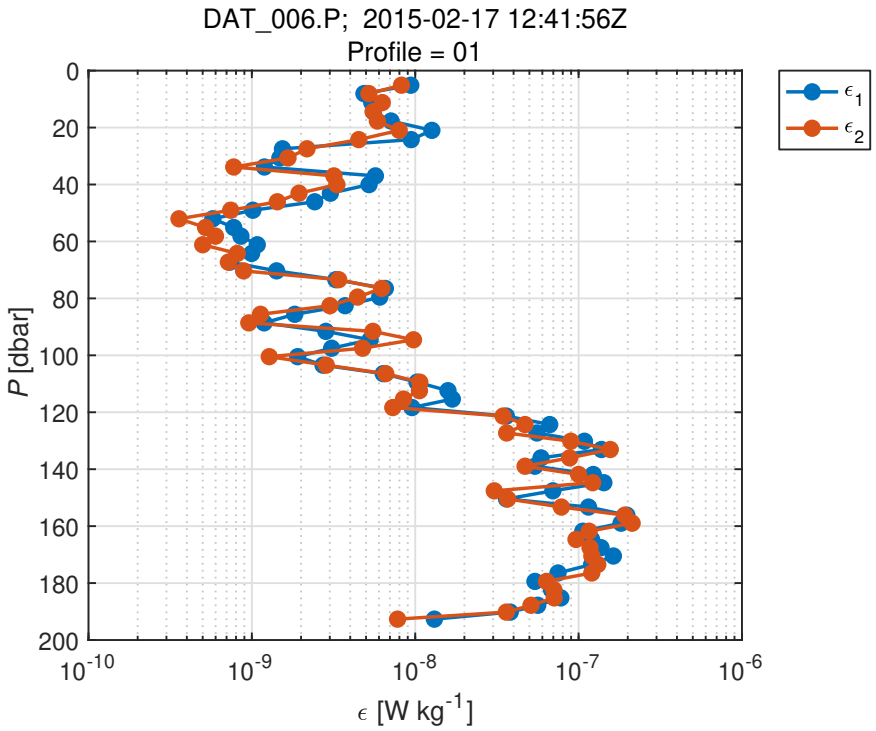

Figure 15: The vertical profiles of the rate of dissipation of TKE, ϵ, for shear probe 1 (blue) and 2 (red), estimated over time bins of 8 seconds, with 50% overlap.

10.1 Dissipation Structure

The output structure from `quick_look.m` depends slightly on the channels in your instrument. The typical fields are explained below. They are grouped below into various topics for ease of description. The topics are:

- fields associated with the rate of dissipation of TKE and shear spectra
- mean values at each dissipation estimate
- fields associated with despiking
- fields associated with the scalar signals
- additional parameters related to data processing

10.1.1 Fields associated with the rate of dissipation of TKE and shear spectra

The first group of output fields is associated with the calculation of the profile of the rate of dissipation. An example of the relevant section of the structure is:

```
                e: [2x64 double]
            K_max: [2x64 double]
           method: [2x64 double]
         dof_spec: 15.2000
            dof_e: [2x64 double]
              mad: [2x64 double]
               FM: [2×64 double]
      Nasmyth_spec: [513x2x64 double]
         sh_clean: [4-D double]
               sh: [4-D double]
               AA: [4-D double]
               UA: [4-D double]
                F: [513x64 double]
                K: [513x64 double]
          fs_fast: 512.0437
          fs_slow: 64.0055
             f_AA: 88.2000
          f_limit: Inf
        fit_order: 3
      diss_length: 4096
          overlap: 2048
       fft_length: 1024
```

The elements of the structure are:

`e` Rate of dissipation of turbulent kinetic energy [$\mathrm{W\,kg^{-1}}$]. Its size is `[N P]`, where `N` is the number of shear probes and `P` is the number of dissipation estimates.

`K_max` A matrix of the maximum wavenumber used for each dissipation estimate. Its size is `[N P]`, where `N` is the number of shear probes and `P` is the number of dissipation estimates.

`method` A matrix indicating the method used to make each dissipation estimate. Its size is `[N P]`, where `N` is the number of shear probes and `P` is the number of dissipation estimates. A value of 0 indicates the variance method and a value of 1 indicates the inertial subrange fitting method (See Technical Note 028 for a description of the two methods).

`dof_spec` The number of degrees of freedom of the spectral estimates based on Nuttall (1971), i.e. `dof_spec = 1.9*num_of_ffts`. This is a constant value for all dissipation estimates that is based on the `diss_length` and `fft_length`.

`dof_e` A matrix of the *approximate* degrees of freedom in each dissipation estimate. Its size is `[N P]`, where `N` is the number of shear probes and `P` is the number of estimates. The value is equal to the number of wavenumber points used to estimate the dissipation rate, divided by the Nyquist bandwidth, times the `diss_length` (in samples). *IMPORTANT: This is only an approximation and work to determine a more appropriate estimate is ongoing.*

`mad` A matrix of the mean absolute deviation (MAD) of the base-10 logarithm of the spectral values divided by the Nasmyth spectrum over the wavenumber range used to make each dissipation estimate. Its size is `[N P]`, where `N` is the number of shear probes and `P` is the number of dissipation estimates.

`FM` A 'figure of merit' given by `FM = mad*sqrt(dof_spec)`. This metric can be used for quality control of the dissipation estimates. Its size is `[N P]`, where `N` is the number of shear probes and `P` is the number of estimates.

`Nasmyth_spec` A three-dimensional matrix of the Nasmyth spectra [$s^{-2}\,cpm^{-1}$] for each shear probe and each dissipation estimate. Its size is `[M N P]`, where `M` is the number of wavenumber elements in the spectra, `N` is the number of shear probes and `P` is the number of dissipation estimates.

`sh_clean` A matrix of the wavenumber cross spectra of all shear probe signals [$s^{-2}\,cpm^{-1}$] at each dissipation estimate. This is a 4-dimensional matrix of size `[M N N P]`, where `M` is the number of wavenumber elements in each cross-spectrum, `N` is the number of shear probes and `P` is the number of dissipation estimates. The "diagonal" elements of the `N` by `N` submatrix are the auto-spectra.

`sh` Same as `sh_clean` but without cleaning by the Goodman coherent noise removal algorithm.

`AA` A matrix of the cross-spectra of acceleration signals [$counts^2\,Hz^{-1}$] for each dissipation estimate. This is a 4-dimensional matrix of size `[M N N P]`, where `M` is the number of wavenumber elements in each cross-spectrum, `N` is the number of acceleration signals and `P` is the number of dissipation estimates. The "diagonal" elements of the `N` by `N` submatrix are the auto-spectra.

`UA` A matrix of the cross-spectra of shear probe and acceleration signals [$counts\,s^{-1}\,Hz^{-1}$] for each dissipation estimate. Its size is `[M N_s N_a P]`, where `M` is the number of wavenumber elements in each cross-spectrum `N_s` is the number of shear probes, `N_a` is the number of acceleration signals and `P` is the number of dissipation estimates.

`F` The frequencies [Hz] of the spectral estimates. Its size is `[M P]`, where `M` is the number of spectral elements and `P` is the number of dissipation estimates. Currently every column is identical because the `fft_length` is uniform for all dissipation estimates.

`K` The wavenumbers [cpm, or cycles per metre] of the spectral estimates. Its size is `[M P]`, where `M` is the number of spectral elements and `P` is the number of dissipation estimates. The number of rows equals the number of wavenumber elements in the cross-spectra.

`fs_fast` The actual fast sampling rate [Hz] of the data.

`fs_slow` The actual slow sampling rate [Hz] of the data.

`f_AA` The frequency [Hz] that is 90% of the anti-aliasing frequency. *Note: Different from input value of* `f_AA`.

`f_limit` The maximum frequency [Hz] used when estimating the rate of dissipation. The value will be `Inf`, unless a value for `f_limit` was specified as an input parameter.

`fit_order` The order of the polynomial fit to the shear spectra, in log-log space, that was used to estimate the spectral-minimum wavenumber.

`diss_length` The length of the segment, in points, used to estimate each dissipation rate.

`overlap` The overlap of adjacent dissipation estimates, in points.

`fft_length` The length of the fft-segments, in points, that were ensemble-averaged into a spectrum.

10.1.2 Mean values at each dissipation estimate

The second group of output fields are related to mean values of the ancillary parameters at each dissipation estimate. An example of the relevant section of the structure is:

```
          speed: [64×1 double]
             nu: [64×1 double]
              P: [64×1 double]
              T: [64×1 double]
              t: [64×1 double]
            AOA: []
      Data_fast: [21×64 double]
      Data_slow: [33×64 double]
      fast_list: {1×21 cell}
      slow_list: {1×33 cell}
```

The elements of the structure are:

`speed` A column vector of the mean speed [$\mathrm{m\,s^{-1}}$] at each dissipation estimate.

`nu` A column vector of the kinematic viscosity [$\mathrm{m^2\,s^{-1}}$] at each dissipation estimate[11].

`P` A column vector of pressure [dbar] at each dissipation estimate.

`T` A column vector of temperature [°C] at each dissipation estimate.

`t` A column vector of time [s] at each dissipation estimate, as measured from the start of the data file.

`AOA` A vector containing the maximum angle of attack [°] of the velocity field. The output will be empty unless the three velocity components (U, V, W) were recorded by an ADV (or other external instrument).

`Data_fast` A matrix where every row is a fast vector that has been averaged over the interval of each dissipation estimate. The number of columns equals the length of `P`. This matrix can be empty. The row names are listed in `fast_list`.

`Data_slow` Same as `Data_fast` except that it is for the slow vectors. The row names are listed in `slow_list`.

`fast_list` A cell array that gives the names of the signals that are sampled on the fast channels. The names correspond to the rows of data in the `Data_fast` matrix (see above). In the above example, there are 21 fast vectors. *Note: The Matlab functions `find.m` and `strcmpi.m` can be used to identify the row corresponding to a particular signal.*

`slow_list` The same as `fast_list` except that the names correspond to the slow channels and the rows of data in `Data_slow`. In the above example, there are 33 slow vectors.

[11] The value is estimated based on the temperature from the T2 sensor, if the channel was sampled. Otherwise, the T1 data is used, unless a `constant_temp` value is specified as an input parameter, which overrides any measured signal.

10.1.3 Fields associated with despiking

The next group of parameters is associated with the despiking of the accelerometer, shear probe, and micro-conductivity signals. The `despike.m` function is described in more detail in Section 11.3. An example of the relevant section of the `quick_look` output structure is:

```
      spikes_A:  {[4x1 double], []}
  pass_count_A:  [2x1 double]
    fraction_A:  [2x1 double]
     spikes_sh:  {[247x1 double], [269x1 double]}
 pass_count_sh:  [2x1 double]
   fraction_sh:  [2x1 double]
      spikes_C:  {}
  pass_count_C:  [0x1 double]
    fraction_C:  [0x1 double]
```

The elements of the structure are:

`spikes_A` A cell array containing the indices to the spikes located in the signals from the vibration sensors. The array has `N_a` elements, where `N_a` is the number of sensors. Typically, two sensors are used, and the first and second cell vectors correspond to `Ax` and `Ay` signals, respectively. In the example above, four spikes were detected in `Ax` and no spikes were detected in `Ay`.

`pass_count_A` A vector indicating the number of passes of the `despike.m` function to remove spikes in the vibration data (See Section 11.3). The vector has `N_a` elements, where `N_a` is the number of sensors. Typically, the first and second elements of the vector correspond to the `Ax` and `Ay` signals, respectively.

`fraction_A` The fraction of vibration data removed by the `despike.m` function[12]. The vector has `N_a` elements, where `N_a` is the number of sensors. Typically, the first and second elements of the vector correspond to `Ax` and `Ay` signals, respectively.

`spikes_sh` The same as `spikes_A`, except for the shear probes. The cell array has `N` elements, where `N` is the number of shear probes. In the above example, 247 spikes were detected in `sh1` and 269 spikes were detected in `sh2`.

`pass_count_sh` The same as `pass_count_A`, except for the shear probes. The vector has `N` elements, where `N` is the number of shear probes.

`fraction_sh` The same as `fraction_A`, except for the shear probes. The vector has `N` elements, where `N` is the number of shear probes.

`spikes_C` The same as `spikes_A`, except for the micro-conductivity sensor(s). The cell array has `N_c` elements, where `N_c` is the number of micro-conductivity sensors. In the above example, no spikes were detected in the micro-conductivity data.

`pass_count_C` The same as `pass_count_A`, except for the micro-conductivity sensor(s). The vector has `N_c` elements, where `N_c` is the number of shear probes.

`fraction_C` The same as `fraction_A`, except for the micro-conductivity sensor(s). The vector has `N_c` elements, where `N_c` is the number of micro-conductivity sensors.

[12]This parameter, in addition to `fraction_sh` and `fraction_C`, can be used for quality control of your data. The removal of a high percentage of data could be a sign of electronic noise or a broken probe.

10.1.4 Fields associated with the scalar signals

The next group of output parameters is associated with the scalar signals. An example of the relevant section of the structure is:

```
          scalar_spectra:  [1x1 struct]
    scalar_vector_list:  {'gradT1'  'gradT2'}
           scalar_info:  [1x1 struct]
```

The elements of the structure are:

`scalar_spectra` A structure with the wavenumber spectra of the scalar gradients and relevant ancillary information. The spectra of the scalar gradients [variance cpm^{-1}] are contained within the subfield `scalar_spec` (i.e. `scalar_spectra.scalar_spec`), which has a size of `[M N_s P]`, where `M` is the number of wavenumber elements in each spectrum, `N_s` is the number of scalar signals and `P` is the number of spectra estimates (i.e. depth bins). The remaining fields are similar to those for the shear spectra which are listed in Section 10.1.1. Refer to the `get_scalar_spectra_odas.m` function (or the Manual for the ODAS Matlab Library) for a detailed description of the fields.

NOTE: The wavenumber spectra of the temperature gradients have NOT been corrected for the frequency response of the thermistors. This correction is left to the scientist because the correction is controversial, and there is no agreement in the research community on what the correction should be. The thermistors are also "hand made" and the response will vary among probes. Refer to Sommer et al. (2013) for more information on the frequency response of the FP07s.

`scalar_vector_list` A cell array with the names of the scalar vectors contained within the field `scalar_spectra.scalar_spec`. The length of the cell array will be `N_s`, where `N_s` is the number of scalar signals.

`scalar_info` A structure containing the parameters used to compute the scalar spectra. The fields are described in the `get_scalar_spectra_odas.m` function and the ODAS Matlab Library manual.

10.1.5 Additional parameters related to the data processing

The remaining fields in the output structure contain additional information related to the data processing. An example of the remaining fields are:

```
                  piezo: 1
profiles_in_this_file: 3
         speed_source: 'Rate of change of pressure'
               params: [1×1 struct]
         vehicle_info: [1×1 struct]
              ql_info: [1×1 struct]
           ql_info_in: [1×1 struct]
```

The elements of the structure are:

`piezo` A logical parameter that identifies the type of accelerometer in the instrument. If the value of `piezo` is 1, piezo-accelerometers were used, whereas if the value is 0, linear accelerometers were used.

`profiles_in_this_file` The number of profiles detected in this file.

`speed_source` A string specifying the source of the estimate of speed of profiling. Typical outputs are `'Rate of change of pressure'` for vertical profilers and `'Rate of change of pressure and glide angle'` for gliders. Other common outputs are `'Hotel file'` or `'Forced to be a constant speed by input parameter'`.

`params` A structure containing the parameters used by `odas_p2mat.m` to create the mat-file from the P-file.

`vehicle_info` A structure with vehicle specific info that was used by `odas_p2mat.m`. The fields of the structure identify the profiling direction, the speed algorithm and the time scale, τ, values described in Table 1.

`ql_info` The structure that was used to process the data (Refer to Section 4 for a description of the fields).

`ql_info_in` The structure that was passed to `quick_look` to process the data. If default parameters were used, the `ql_info_in` structure will be empty.

11 Further Data Processing Techniques

11.1 hotel-files

If your Rockland instrument is mounted on a vehicle[13] that provides mission files, you may wish to integrate the data provided in the mission file into your data processing. In some cases the data recorded by the vehicle are required to process the P-file. There are currently four scripts and functions[14] for extracting information out of mission files and placing it into a hotel-file.

The hotel-file is ingested by `odas_p2mat.m` (Section 3) and interpolated on to the time vector `t_slow` in your P-file. The resulting data vectors are saved to the same `mat`-file as the data vectors produced from the P-files.

NOTE: The most common use of a hotel-file is to import speed data from a vehicle when the Rockland instrument is not able to measure the speed itself (such as an AUV equipped with a MicroRider, or a Seaglider equipped with MicroPods). An accurate speed estimate is required to convert shear probe data into physical units and to compute gradients of temperature and conductivity. Other data of interest measured by the controller in a vehicle include CT data, pressure, pitch, roll and heading.

11.1.1 hotel-file content

For a hotel-file to be usable by `odas_p2mat.m`, it must be a Matlab mat-file that contains structures.

NOTE: Each structure must have two fields, one named `data` and the other named `time`.

For example, if you have a file containing GPS data, then you may want to use this GPS-file to create a structure called `gps_long`, and place the longitudinal data into `gps_long.data` and the time of each sample of longitude into `gps_long.time`. Save this structure and any others that you may have created into a file, say `my_gps_data.mat`. This is your hotel-file and `odas_p2mat.m` will extract the information in `my_gps_data.mat`, place it into variables, and interpolate the data onto the time base of `t_slow`. The names of the variables will be the names of the structures appended with `_slow`. For example, the longitudinal data will be in the variable `gps_long_slow` and can be plotted against the variable `t_slow`, because they have identical lengths.

There are a few **restrictions**:

1. The data in the field `time` must increase monotonically.
2. The data must also span the time range of the data in your Rockland data file.
3. It is best to make the field `time` in units of Matlab time. However, `odas_p2mat.m` will also recognize 'unix' time based on January 1, 1970 or a time vector that starts at 0 (zero) and is in units of seconds.

Note: The field `time` **does not have to be uniformly spaced** – there can be gaps with no data. If you have multiple Rockland data files that need to be processed, feel free to make the

[13] Vehicles upon which our instruments are frequently mounted include Kongsberg Seaglider, Teledyne Webb Slocum Glider, Hydroid REMUS AUV, and Nemo Floats equipped with a Nortek Vector Current Meter

[14] `hotelfile_Nortek_vector.m`, `hotelfile_Remus_mat.m`, `hotelfile_seaglider_netcdf.m`, and `hotelfile_slocum_netcdf.m`.

range of the field `time` to be as large as necessary to encompass all of your Rockland data. The `odas_p2mat.m` function will only extract the portion that overlaps with the time base in your Rockland data file.

The feature that allows the time to be in seconds starting from zero is useful for situations where you have an algorithm that uses the pitch and rate of change-of-depth to calculate the speed of a glider based on a model such as that of Merckelbach et al. (2010). As opposed to the Rockland algorithm, which assumes a constant angle of attack, a glider model computes the angle-of-attack as a function of time and, hence, provides a more accurate estimate of speed. You would, for example, do an initial conversion to physical units using a `constant_speed = 1`, use the pitch ($-$`Incl_Y`), rate-of-change of pressure (`W_slow`), and time (`t_slow`) data (which was derived from the conversion to physical units) in the model of glider speed, and produce a speed vector. The derived speed estimates are placed into `speed.data` and the time of the estimates is placed in `speed.time`. This structure is saved into a hotel-file and can then be used to convert the data into physical units using the speed estimated from the glider model.

How you create the structures in the hotel-file depends on your source of data. One usually writes a script that generates the desired hotel-file.

NOTE: It is wise to give the structures unique names that do not conflict with any of the names of the variables created by `odas_p2mat.m`. Otherwise, a variable in your Rockland data file may be over-written by the contents in your hotel-file.

Finally, take some care to check how the time base of your source matches against the time base of your Rockland data file. Instruments with independent clocks will have an offset error (because the time was not set properly) and a drift with respect to time (because the clocks will inevitably tick at different rates). The only way to be really sure that the time bases are aligned is to plot – on the same graph – a signal that is common to both instruments. For example, the pitch or roll signals are recorded by both the MicroRider and the glider controller. Any signal that has significant and unpredictable fluctuations is useful for time alignment. If the fluctuations do not align, then the time bases have not been aligned. For example, you may need to add or subtract a fixed value from `glider_pitch.time` in order to make the fluctuations in the signal recorded by the vehicle controller align with the inclinometer signal recorded by the MicroRider. You may even wish to stretch (or compress) the values in `time` to account for the differing clock rates.

11.1.2 Seaglider Example

MicroPods can be mounted on a Seaglider. MicroPods do not have a means of measuring pressure or speed, and require pressure and speed to be ingested from the hotel-file. The data contained within the hotel-files are extracted from NetCDF mission files generated by Kongsberg-supplied software.

The script to generate the hotel-file is called `hotelfile_seaglider_netcdf.m`. The parameters at the top of the script may need to be modified to represent your deployment data. In particular, you must ensure that the parameters `file_num_range`, `file_base_name`, and `output_file_name` are correct and consistent with your NetCDF files. You may also need to edit the variable `var_data`, which a two-column list of the names of the mission-file variables that you wish to extract and the names that they will be given in the hotel-file. The names in the second column are also the names that they will have in your final Matlab data file that is created by `odas_p2mat.m`.

To run the script, navigate to the directory containing the NetCDF files and execute the following command:

```
>> hotelfile_seaglider_netcdf
```

The resulting hotel-file can then be used with `odas_p2mat.m`. In this example, a Rockland raw binary file ending in `16` is used.

```
>> odas_p2mat('*16', 'vehicle', 'sea_glider', ...
      'hotel_file','my_hotel_file.mat' );
```

If there is a time offset between the Rockland data and the data supplied within the mission file, a time offset can be added to the Rockland instrument time. In the following example, a `time_offset` of -5 indicates that 5 seconds will be removed from the time logged in the header of the `P`-file.

```
>> odas_p2mat('*16', 'vehicle', 'sea_glider', ...
      'hotel_file','my_hotel_file.mat',...
      'time_offset', -5 );
```

Once the hotel file is generated, `quick_look.m` (Section 4) can be used for further data processing.

11.2 *In Situ* Thermistor Calibration

The FP07 thermistor is often sold uncalibrated, but can be calibrated post-deployment using *in situ* data. (This was apparent in the example data shown in Figure 12, where the thermistor profiles were offset from the CT profiles of temperature.) The *in situ* calibration can be performed using a reference temperature recorded by your instrument by either a SeaBird-CT or JAC-CT. Alternatively, a hotel-file can be created from a vehicle, such as a Slocum Glider, that is equipped with its own CT (see Section 11.1).

Once you have converted your data into physical units and, if necessary, incorporated data from a hotel-file, you can call `cal_FP07_in_situ.m` to calibrate your FP07 thermistor. The syntax is as follows:

```
[T_0, beta, Lag] = cal_FP07_in_situ('fname', 'T_ref','T','SN',temp_info)
```

where the input and output arguments are:

Argument	Description
`fname`	Name of the file that contains the thermistor data to be calibrated and a reference temperature.
`T_ref`	Name of reference temperature variable.
`T`	Name of the FP07 thermistor to be calibrated.
`SN`	The serial number of your thermistor.
`temp_info`	Input structure that defines variable processing parameters (see below).
`T_0`	Value of parameter T_0 used in the Steinhart-Hart (SH) equation.
`beta`	Beta coefficients, in ascending order, of the fit to the SH equation.
`Lag`	Delay in seconds between the thermistor and the reference thermometer.

The variables within the structure `cal_info` control the behaviour of `cal_FP07_in_situ`. The default input structure is attained by running the `cal_FP07_in_situ.m` without any arguments, i.e.,

```
>> temp_info = cal_FP07_in_situ

temp_info =

         make_figures: 1
                order: 1
           plot_range: 1
         plot_regress: 1
          plot_result: 1
          plot_scaled: 1
           plot_xcorr: 1
        profile_min_P: 1
        profile_min_W: 0.2000
 profile_min_duration: 20
          profile_num: 1
         vehicle_info: []
```

The fields of `temp_info` are categorized and described in the following subsections.

11.2.1 Fields that control the temperature calibration

There is only one parameter used to control the calibration procedure:

`order` The fit order of the regression to the Steinhart-Hart equation. Value can be 1, 2, or 3. For small temperature ranges, order = 1 is recommended. Default = 2.

NOTE: If the temperature range is less than 8 °C and an order 2 fit is requested, a warning message will recommend using a first-order calibration.

11.2.2 Fields that control profile selection

The fields that control the profile selection are the same as those used by `quick_look`. More specifically:

`profile_min_P` The minimum pressure of a profile, [dbar]. Default = 1.

`profile_min_duration` The minimum time [s] for a profile to be deemed a profile. Default = 20.

`profile_min_W` The minimum vertical speed [dbar/s] for the detection of a profile. Default = 0.2.

`profile_num` An integer identifying the profile number to be analyzed. Default = 1.

`vehicle_info` A structure found in all `mat`-files that is used to determine the direction of profiling. If empty, the information will be loaded from the datafile. If a change is desired, the profiling direction needs to be or 'glide' for gliders. Default = [ ].

11.2.3 Parameters that control data visualization

The parameters that control the data visualization include flags that either suppress or display the figures. By default, figures that display the time range of the calibration, the regression to the Steinhart-Hart equation and the result of the calibration are shown. On the other hand, plots of the analysis steps (i.e. detrending and cross-correlation) will only be shown if the appropriate flags are set to `true`.

`make_figures` A logical parameter that determines if figures are generated. `make_figures = false` suppresses the generation of figures to speed up the data processing. Default = `true`.

`plot_range` A logical parameter that determines if a figure of the selected range is generated. Default =`true`.

`plot_regress` A logical parameter that determines if the regression data and fit are plotted. Default = `true`.

`plot_result` A logical parameter that determines if figures of the calibration results (i.e. comparison and difference) are shown. Default = `true`.

`plot_scaled` A logical parameter that determines if a figure of the detrended signals and lag are plotted. Default = `false`.

`plot_xcorr` A logical parameter that determines if a figure of the cross correlation is plotted. Default = `false`.

11.2.4 Output Parameters of FP07 Calibration function

The `beta` and `T_0` values that are output from `cal_FP07_in_situ.m` are the calibration coefficients. To finalize the calibration, the determined coefficients must be input into the configuration string (Section 2.1) of the data file and the data re-converted into physical units using either `odas_p2mat.m` or `quick_look.m`.

11.2.5 Example

As an example, we will perform an *in situ* calibration of the temperature data previously presented in Section 8 (Figure 12).

First, create the temperature input structure from the default values:

```
>> temp_info = cal_FP07_in_situ;
```

In our application, the default `temp_info` values are appropriate, so the *in situ* calibration script can be run next by:

```
>> [T_0, beta, lag] = cal_FP07_in_situ('DAT_006','JAC_T','T1',...
      '1089',temp_info)
```

By default, three figures are produced to illustrate the calibration. First, the selected section of the data file is highlighted (Figure 16). In the second figure, the natural logarithm of the resistance ratio against the inverse of the absolute temperature is plotted to show the quality of the regression (Figure 17). In the third figure, the reference temperature signal, the newly calibrated thermistor signal, and their difference are plotted (Figure 18).

NOTE: Importantly, the calibration coefficients are displayed in the title of Figure 17.

Two additional figures can be generated to show the results of intermediate steps of the calibration procedure. The first of these optional figures is generated by setting `plot_scaled=true` (Figure 19). The figure shows the timeseries of the detrended reference temperature and the scaled thermistor data that will be used for the calibration. This shows a more direct comparison than the bottom panel of Figure 16. The second optional figure is generated by setting `plot_xcorr=true` (Figure 20). This figure shows the cross-correlation of the thermistor and reference temperature signal. The maximum correlation and corresponding lag that is used to offset the thermistor data are identified. The lag is caused by the physical separation of the thermistor and reference thermometer, as well as the different frequency response of the thermistor and reference thermometer.

NOTE: The `T_0` and `beta` calibration coefficients produced by the *in situ* calibration can now be added to the configuration file for `DAT_006` (Section 2.1) and the `P`-file reconverted into physical units (Section 3).

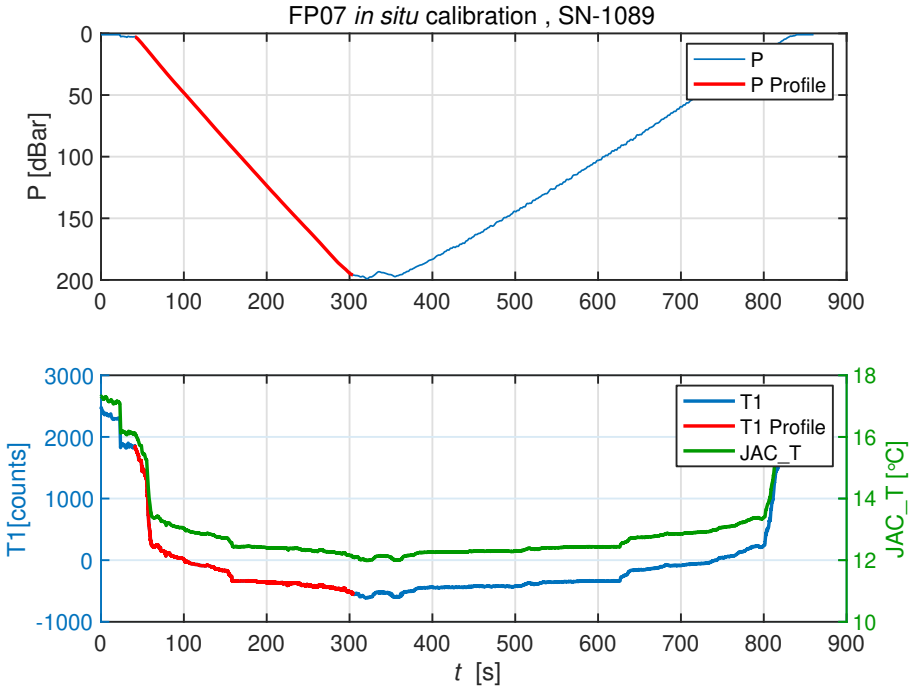

Figure 16: A profile is selected using the parameters described in the input structure `cal_info`. (Top panel) The pressure signal for the entire datafile is shown in blue and the segment corresponding to the selected profile is shown in red. (Bottom panel) The temperature signal in raw counts for the entire datafile is shown in blue and the segment corresponding to the selected profile is shown in red. The data from the reference thermometer in degrees Celcius is shown in green on the right axis.

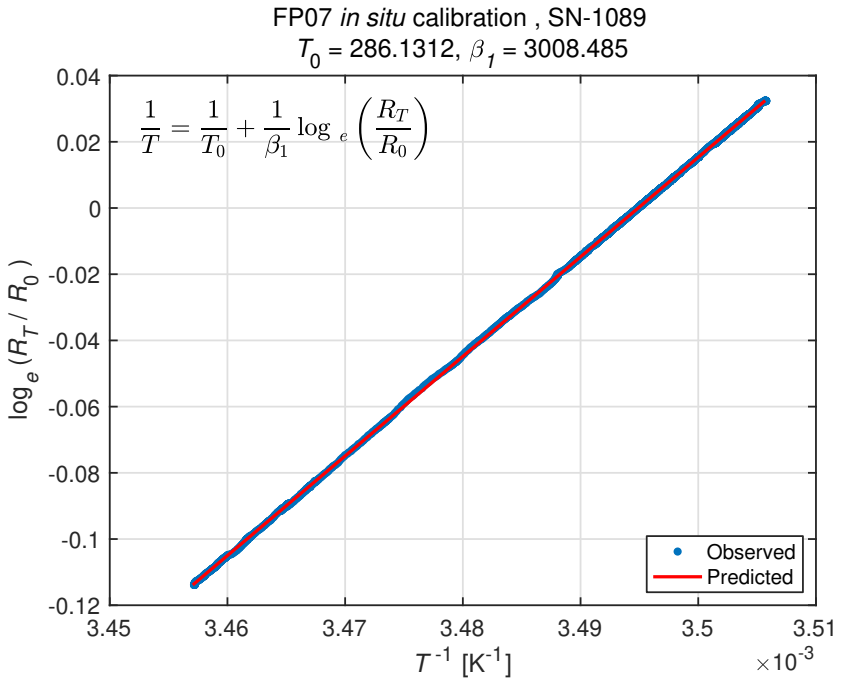

Figure 17: The natural logarithm of the resistance ratio plotted against the inverse of the absolute temperature to show the quality of the regression.

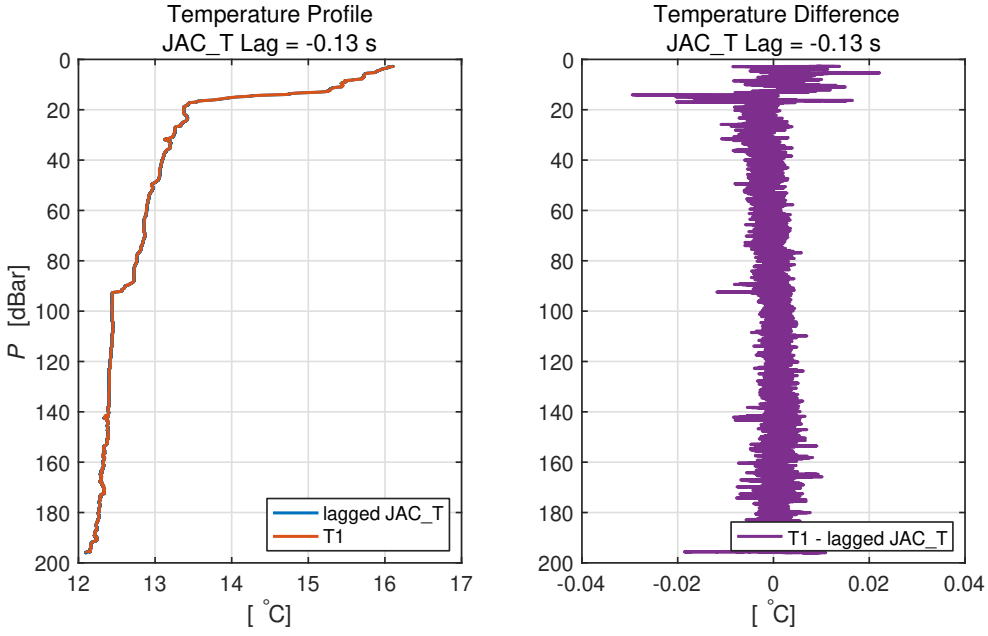

Figure 18: The lagged reference temperature (blue), the newly calibrated thermistor (red), and their difference (violet).

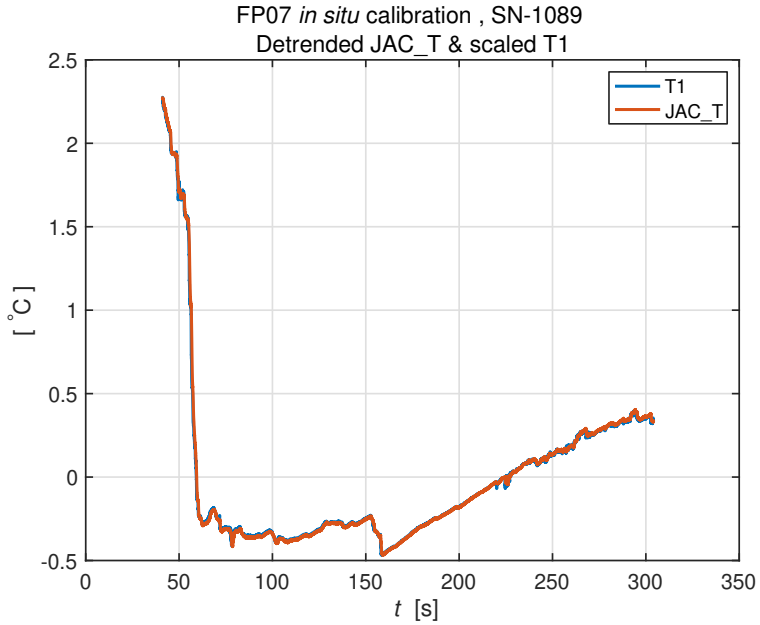

Figure 19: The thermistor data and CT data are de-trended. This is an optional figure that can be generated by `cal_FP07_in_situ.m`.

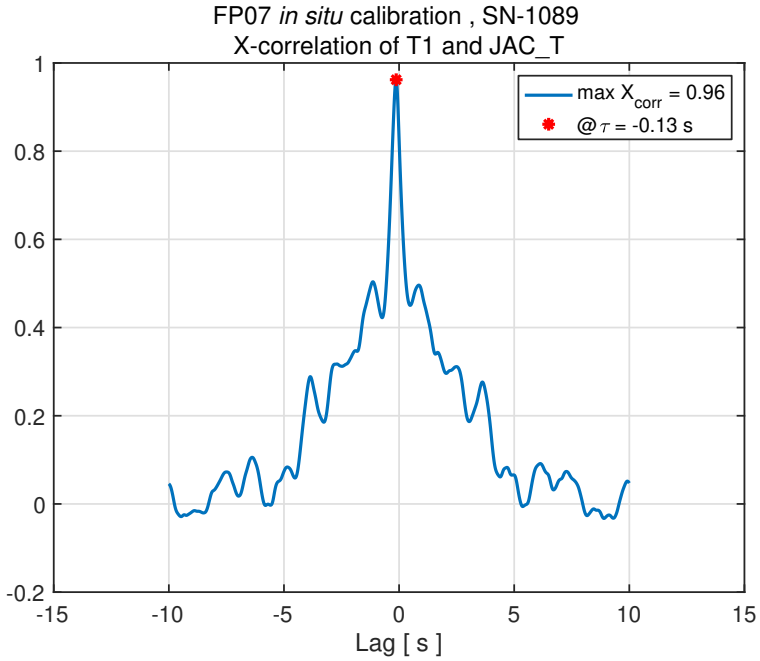

Figure 20: The cross-correlation of thermistor and CT data is calculated and the lag of maximum cross-correlation is identified. This is an optional figure that can be generated by `cal_FP07_in_situ.m`.

11.3 Despiking Data

Spikes in data can be caused by environmental debris in the water column (phytoplankton, jelly fish, etc.). Regions of high biological productivity are more likely to result in lots of spikes in your data set. Such spikes – for example near 10 dbar in ∇u_1 in Figure 21 (left) – can change the spectra and skew estimates of dissipation (Figure 21, right). The `despike.m` function is designed to detect anomalous spikes in data and replace the spikes and adjoining data points with a local average of the signal. Appropriately despiked data produces cleaner spectra and more accurate dissipation estimates (Figure 22).

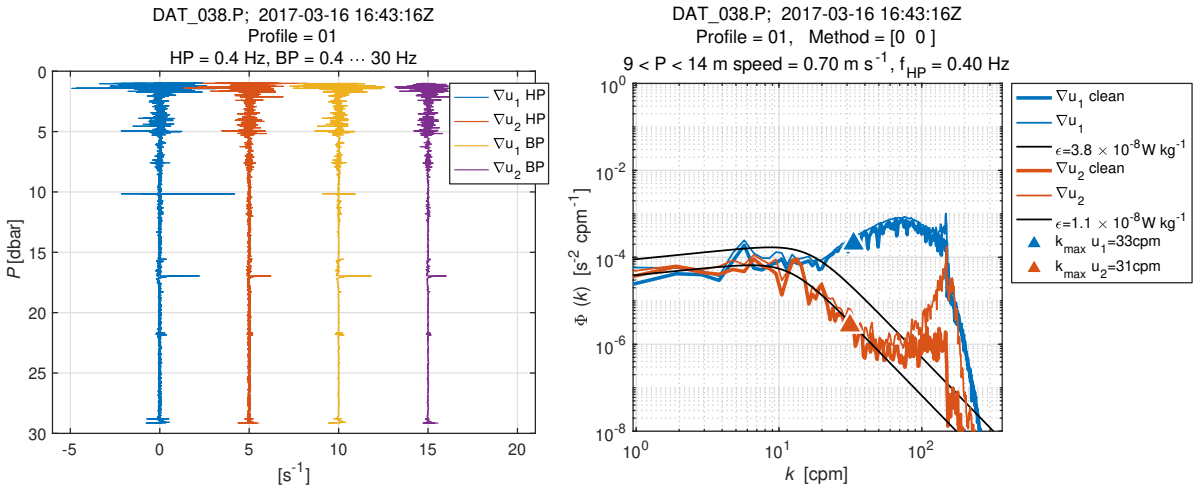

Figure 21: (Left) Data profiles with no despiking applied to the shear signals, which renders all spikes visible. Note the large spike near 10 dbar in ∇u_1 (i.e. `sh1`). (Right) Shear spectra of the signals with no despiking applied calculated between 9 and 14 dbar. The spectra for `sh1` deviates significantly from the Nasmyth spectrum.

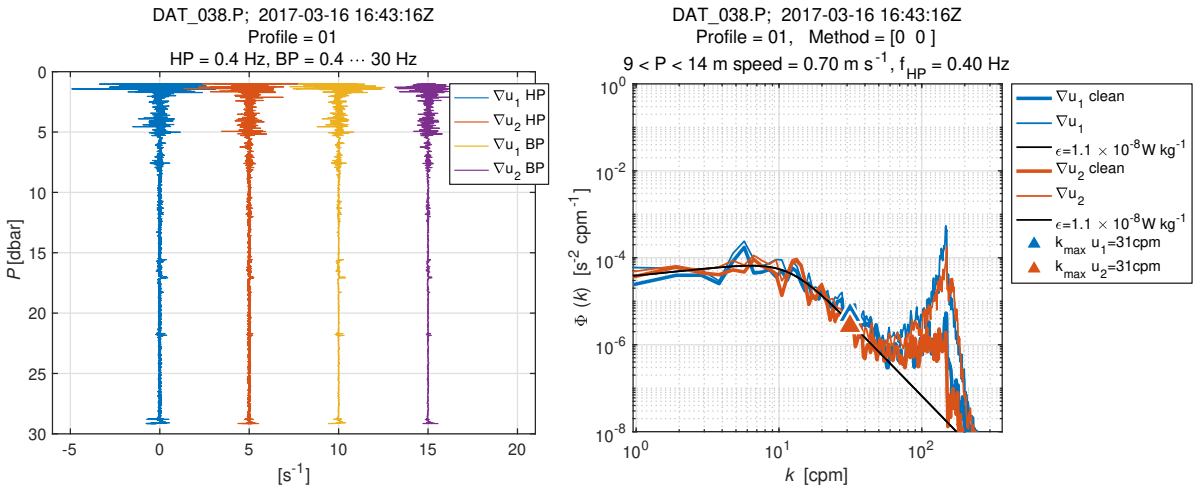

Figure 22: (Left) Data profiles with despiking applied to the same shear signals as in Figure 21. The spike near 10 dbar in Figure 21 has been eliminated. (Right) Shear spectra calculated between 9 and 14 dbar. The spectrum of `sh1` is no longer anomalous with respect to the Nasmyth spectrum and the `sh2` spectrum.

Despiking is applied by default in `quick_look.m` to the shear, vibration and micro-conductivity signals. The `despike.m` function, which is called by `quick_look.m`, performs the following steps:

1. The input signal is high-pass filtered and rectified (Figure 23, top panel, blue line).
2. The local pseudo-standard deviation is calculated by smoothing the rectified signal with a low-pass zero-phase filter (Figure 23, top panel, red line).
3. Spikes are identified in the time-domain by comparing the instantaneous rectified signal against the smoothed pseudo-standard deviation calculated in Step (2). If the ratio of the instantaneous signal to the smoothed signal is above a threshold, a spike is identified (Figure 23, top panel, yellow stars).
4. The spike and adjoining points of the input signal are replaced by an average calculated using the good points on either side of the spike. The resulting signal is referred to as the 'clean' signal (Figure 23, bottom panel, red line).

Steps (1) - (4) are repeated, using the 'clean' signal produced in Step (4) as the input for the subsequent iteration of Step (1). This process is repeated until either no further spikes are identified or a maximum of ten iterations has been performed.

The `despike.m` function can be called directly using the command:

```
>> despike(dv, thresh, smooth, Fs, N, '-debug')
```

where:

- `dv` is the signal to be despiked (e.g. `sh1`),
- `thresh` is the threshold value for identifying spikes (`quick_look.m` uses a default of 8),
- `smooth` is the cut-off frequency [Hz] of the first order Butterworth filter that is used to smooth the rectified input signal (`quick_look.m` uses a default of 0.5 Hz),
- `Fs` is the sampling rate (our instruments usually sample at 512 Hz), and
- `N` dictates how many data points are removed for each detected spike (`quick_look.m` uses a default of `N = 0.04*Fs`).

NOTE: The total number of points removed for each spike is 1.5`N`. `N/2` data points are removed before the spike and `N` points are removed after the spike. The spike and adjoining points are replaced by an average calculated using the 'good' points on either side of the spike, which are defined over a region of size `0.25*Fs /smooth`.

The `'-debug'` option produces a figure with two subplots to visualize the despiking processing (Figure 23). The despiking routine will pause at the end of each iteration and move forward to the next pass by pressing any key when prompted to do so. The plots will update to include the despiked signal from the previous iteration being used as the smoothed, rectified signal for the current iteration. The input signal in the lower plot will remain the original input signal. **It is recommended that you use the '`-debug`' option to test different `thresh`, `smooth`, and `N` parameters. Once you are satisfied with the despiking, these parameters can be input into `quick_look.m` to perform the despiking for you (Section 4.3).**

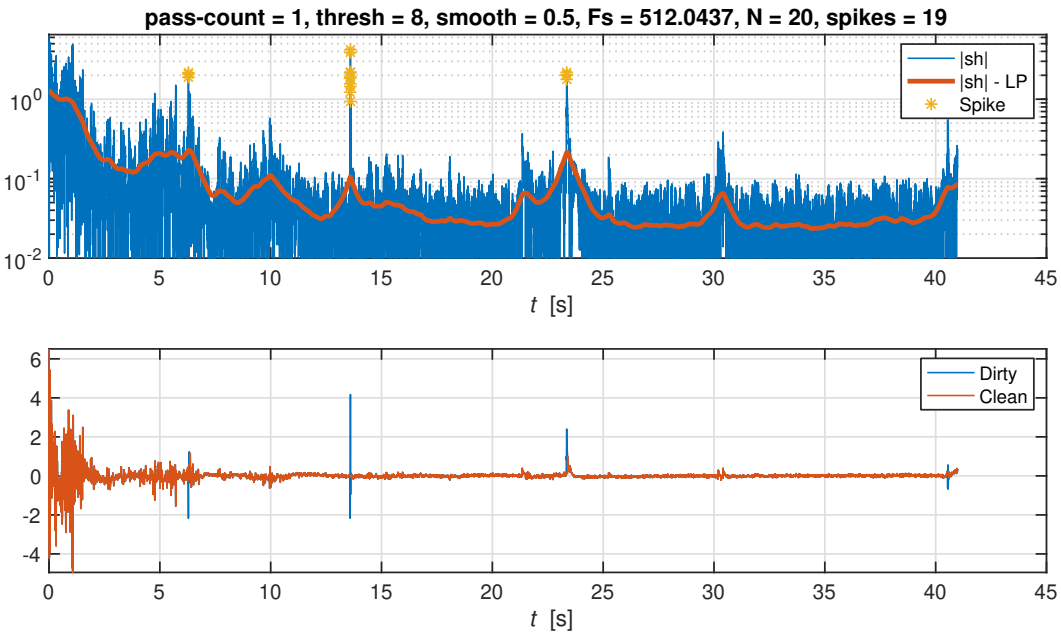

Figure 23: The figure generated by the `'-debug'` option of the `despike.m` function. The rectified signal (blue) and its smoothed version (red) are plotted in the upper panel with spikes identified (yellow stars). The original signal (blue) and cleaned signal (red) are plotted in the lower panel. The segment of data is the same as that used for Figures 21 and 22, where the spike at 10 dbar appears at 13.6 s in the time-domain.

11.4 Inspecting Spectra

After calculating the dissipation of turbulent kinetic energy using `quick_look.m`, it may be useful to examine the spectra that were used to determine each individual dissipation estimate. The function to perform this task is called `show_spec.m`[15]. The structure, `diss`, produced by `quick_look.m` contains a set of spectra, one set for every dissipation estimate within the structure. The `show_spec.m` function plots the spectra for shear, Nasmyth, scalar-gradients, and acceleration are plotted against frequency and wave number. The function is called using:

```
>> show_spec(diss, 'title string')
```

which will generate a figure window that displays the spectra from one segment of the `diss` structure used to estimate the rate of dissipation, ϵ (Figure 24). The segment is identified by its index number in the title, along with the segment-average pressure, speed, and temperature. You can transverse between sets of spectra using a slide bar, or the arrow keys.

Spectra of interest can be exported using the "Export" button found at the top of the screen. This button opens a dialog box that simplifies the saving of a plot into a file. Details on how to use `show_spec.m` function can be found in the ODAS Matlab Library Manual v4.4.

[15] Matlab version 2014b or later is required for `show_spec.m`

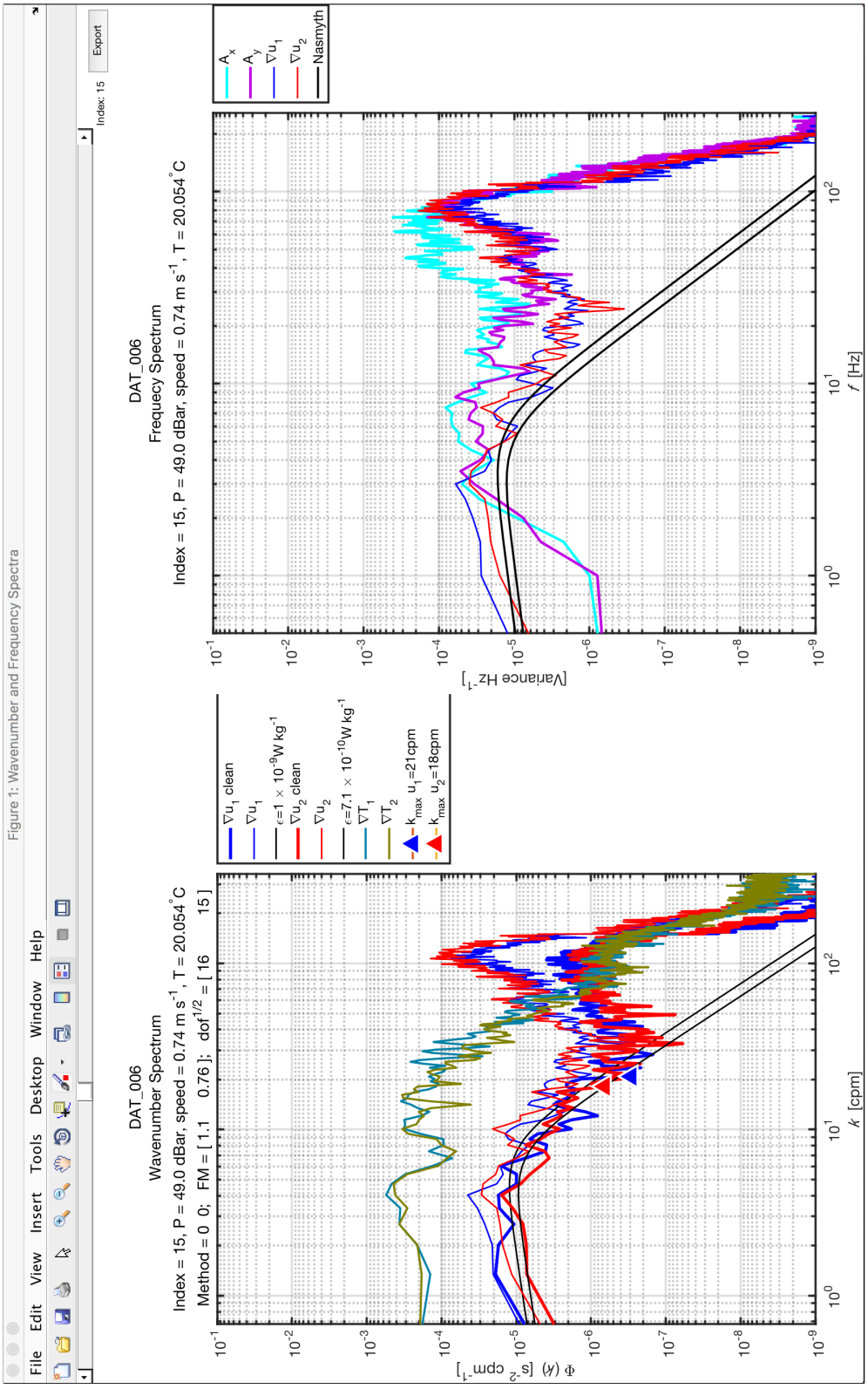

Figure 24: The figure generated by `show_spec.m`. The wavenumber spectra (left) and the frequency spectra (right) are plotted for individual estimates of dissipation. You can navigate between segments by either using the arrow keys, changing the scroll bar at the top of the window, or by entering a specific index into the text field location at the top right corner of the window.

A default_vehicle_attributes

```
; The speed algorithm must match one of the speed algorithms implemented by
; odas_p2mat.  Available algorithms include:
;   pressure - speed determined by pressure (ie, fall rate)
;   hotel    - speed data must be injected using an hotel file
;   glide    - fall rate and angle determine the fall rate
;   vector   - Three velocity vectors (U,V,W) are use to generate speed
;   emc      - Use of AEM1-G electromagnetic velocity sensor - type aem1g_a
;              for analog output or aem1g_d for RS232 output.
;   constant - A constant speed is applied as determined by the constant_speed
;              parameter value.

[vmp]
; Vertical Microstructure Profiler
profile_dir     = down
speed_algorithm = pressure
tau             = 1.5
aoa             = 0

[rvmp]
; Rising (or reverse) Vertical Microstructure Profiler
profile_dir     = up
speed_algorithm = pressure
tau             = 1.5
aoa             = 0

[argo_float]
; Rising Argo type of float vertical profiler
profile_dir     = up
speed_algorithm = hotel
tau             = 60
aoa             = 0

[sea_glider]
profile_dir     = glide
speed_algorithm = hotel
tau             = 5.0
aoa             = 0

[slocum_glider]
profile_dir     = glide
speed_algorithm = glide
tau             = 3.0
aoa             = 3
```

```
[sea_explorer]
profile_dir     = glide
speed_algorithm = glide
tau             = 3.0
aoa             = 3

[auv]
profile_dir     = horizontal
speed_algorithm = hotel
tau             = 10.0
aoa             = 0

[auv_emc]
profile_dir     = horizontal
speed_algorithm = emc
tau             = 10.0
aoa             = 0

[nemo]
profile_dir     = horizontal
speed_algorithm = vector
tau             = 60
aoa             = 0

[micro_squid]
profile_dir     = horizontal
speed_algorithm = vector
tau             = 1.5
aoa             = 0

[stand]
profile_dir     = horizontal
speed_algorithm = constant
constant_speed  = 0.5
tau             = 1.5
aoa             = 0

[xmp]
; Expendable Microstructure Profiler
profile_dir     = down
speed_algorithm = pressure
tau             = 1.5
aoa             = 0
```

B hotelfile_seaglider_netcdf

```
%% hotelfile_seaglider_netcdf
% Generate a hotel-file from the NetCDF files generated by a Seaglider

% <latex>\index{Scripts!hotelfile\_seaglider\_netcdf}</latex>
%
%%% Syntax
%   User Script:  hotelfile_seaglider_netcdf
%
% * [file_num_range] Scan input NetCDF mission-files within the specified
%      range.
% * [file_base_name] Base name of the input NetCDF mission-files.
% * []
% * [output_file_name] Hotel-file created with the specified name. The
%      default name is derrived from the input-file names.
%
%%% Description
% Generate a hotel-file from the NetCDF mission-files of a Sea-glider that
% were generated by Kongsberg-supplied software.
%
% The resulting hotel-file contains the subset of data from the NetCDF
% mission-files that are useful for converting Rockland binary data files into
% physical units and for other purposes. All of the vectors that are
% extracted from the mission-files are interpolated to the Rockland
% $\texttt{t\_fast}$ and $\texttt{t\_slow}$ time vectors. The mission data
% vectors are renamed using the $\texttt{\_fast}$ and $\texttt{\_slow}$
% postfixes. A direct comparison of mission-file data and Rockland recorded data
% is then possible.
%
%%% Examples
% The hotel-file can be generated by navigating to the directory containing
% the NetCDF files.  This script is then executed.
%
%    >> hotelfile_seaglider_netcdf
%
% The resulting hotel-file can then be used with $\texttt{odas\_p2mat}$. In
% this example, a Rockland raw binary data file ending in $\texttt{16}$ is used.
%
%    >> odas_p2mat('*16', 'vehicle', 'sea_glider', ...
%                  'hotel_file', 'my_hotel_file_name.mat');
%
% If there is a time offset between the Rockland data and the data supplied
% within in the mission-file, a time offset can be added to the Rockland
% instrument time.
%
%    >> odas_p2mat('*16', 'vehicle', 'sea_glider', ...
%                  'hotel_file', 'my_hotel_file_name.mat', ...
%                  'time_offset', -5);
```

```
%
% This is a script that you must edit. The parameters
% $\texttt{file\_num\_range}$, $\texttt{file\_base\_name}$ and
% $\texttt{output\_file\_name}$ are near the top of the function. The names
% of the desired data in the mission-file, and the names that they will
% attain in your Rockland data mat-file are tabulated just below the parameter
% names.

%%%%%%%% Credit

% Version History:
%
% 2015-02-03 RGL, Original version
% 2015-04-22 RGL, Original version
% 2015-04-23 RGL, added ability to read CTD data.
% 2015-04-20 WID, Slightly modified + documented.
% 2015-10-26 WID, Complete rewrite - support for final hotel-file format.
% 2015-10-28 RGL, Document changes.
% 2015-11-18 RGL, Update documentation.

file_num_range   = 1:28;
file_base_name   = 'p613';
output_file_name = sprintf('%s_%04d_%04d', file_base_name,    ...
                                           file_num_range(1), ...
                                           file_num_range(end));

var_data = {
%   Sea-glider variable name  ODAS_P2MAT required name
    'time'                    'time'
    'speed'                   'speed'
    'speed_qc'                'speed_qc'
    'speed_gsm'               'speed_gsm'
    'horz_speed'              'H_speed'
    'vert_speed'              'V_speed'
    'depth'                   'P'
    'glide_angle'             'glide_angle'
    'eng_pitchAng'            'pitch'
    'eng_rollAng'             'roll'
};

% Construct NetCDF input data file names
inputfiles = {};
for file_num = file_num_range
    inputfiles{end+1} = sprintf('%s%04d.nc', file_base_name, file_num);
end

for sensor = var_data'
    data = [];
    for file = inputfiles
```

```
        try
            data2 = ncread(file{1}, sensor{1});
            data = [data; data2];
        catch; end
    end
    if strcmpi(sensor{2}, 'time')
        time = data;
    elseif ~isempty(data) && ~isempty(time)
        result.(sensor{2}).data = data;
        result.(sensor{2}).time = time;
    end
end

if ~exist('result', 'var')
    error('hotel-file data not found.');
end

% Modify pressure / depth values so they are in units of dBar
%for f = {'P', 'depth', 'P_CTD'}
%    if isfield(result,f{1})
%        result.(f{1}).data = result.(f{1}).data * 10;
%    end
%end

% Fix for speed_qc. Value provided as characters - convert into doubles.
if isfield(result,'speed_qc')
    result.speed_qc.data = double(result.speed_qc.data) - double('0');
end

% Modify speed values so they are in units of m/s
for f = {'speed', 'speed_gsm', 'H_speed', 'V_speed'}
    if isfield(result,f{1})
        result.(f{1}).data = result.(f{1}).data / 100;
    end
end


% Output file declared at the begining of the file.
save(output_file_name, '-struct', 'result')
```

References

Goodman, L., E. R. Levine, and R. G. Lueck, 2006: On measuring the terms of the turbulent kinetic energy budget from an AUV. *Journal of Atmospheric and Oceanic Technology*, 977–990.

Macoun, P., and R. Lueck, 2004: Modelling the spatial response of the airfoil shear probe using different sized probes. *Journal of Atmospheric and Oceanic Technology*, **21**, 284–297.

Merckelbach, L., D. Smeed, and G. Griffiths, 2010: Vertical water velocity from underwater gliders. *Journal of Atmospheric and Oceanic Technology*, **21**, 547–5637.

Mudge, T. D., and R. G. Lueck, 1994: Digital signal processing to enhance oceanographic observations. *Journal of Atmospheric and Oceanic Technology*, **11**, 825–836.

Nuttall, A. H., 1971: Spectral estimation by means of overlapped Fast Fourier Transform processing of windowed data. NUSC Tech. Rep. No. 4169, 40 pp. [Available online at http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=AD0739315].

Sommer, T., J. R. Carpenter, M. Schmid, R. G. Lueck, and A. Wuest, 2013: Revisiting microstructure sensor responses with implications for double-diffusive fluxes. *J. Atmos. Oceanic Technol.*, **30 (8)**, 1907–1923, doi:10.1175/JTECH-D-12-00272.1.

Vachon, P., and R. G. Lueck, 1984: A small combined temperature-conductivity probe. *Proceedings of the 1984 STD Conference and Workshop*, San Diego, California, USA.

————————End of Document————————