RSKtools for Matlab processing RBR data

Table of Contents

RSKtools v3.4.0; RBR Ltd. Ottawa ON, Canada; support@rbr-global.com; 2020-02-14

Introduction

RSKtools includes a series of functions to post-process RBR logger data. Below we show how to implement some common processing steps to obtain the highest quality data possible.

All post-processing functions are customisable with name-value pair input arguments. Documentation for each function can be accessed using the Matlab commands `doc` and `help`. For example, to open the help page for `RSKsmooth`, type: `doc RSKsmooth` at the Matlab command prompt.

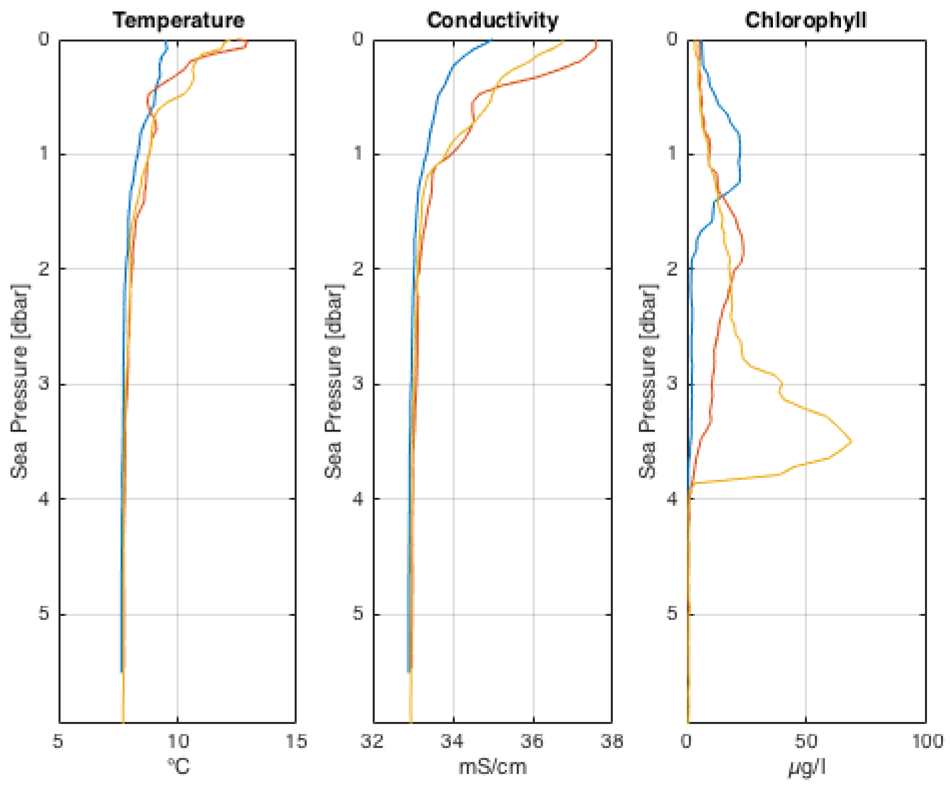
Review RSKtools Getting Started for an introduction on how to load RBR data into Matlab from RSK files, make plots, and access the data.

Example processing sequence

```
% First, while in the RSKtools/QuickStart directory, open a connection
% to the RSK logger database file:
rsk = RSKopen('../sample.rsk');

% read the upcast from profiles 1 - 20
rsk = RSKreadprofiles(rsk, 'profile', 1:20, 'direction', 'up');

% plot a few profiles of temperature, conductivity, and chlorophyll
RSKplotprofiles(rsk,'profile',[1 10 20],...
    'channel',{'temperature','conductivity','chlorophyll'});
```



Derive sea pressure from total pressure

We suggest deriving sea pressure first, especially when an atmospheric pressure other than the nominal value of 10.1325 dbar is desired. Functions that require sea pressure, such as `RSKderivedepth`, will always use the nominal value if a custom value is not specified at this stage. In this example we'll take atmospheric pressure to be 10 dbar.

```
rsk = RSKderiveseapressure(rsk,'patm',10);

% Keep a copy of the raw data to compare with the processed data.
raw = rsk;
raw = RSKderivesalinity(raw);
```

Correct for A2D zero-order hold

The analog-to-digital (A2D) converter on RBR instruments must recalibrate periodically. In the time it takes for the calibration to finish, one or more samples are missed. The onboard firmware fills the missed scan with the same data measured during the previous scan, a technique called a zero-order hold.

The function identifies zero-hold points by looking for where consecutive differences for each channel are equal to zero, and replaces these samples with a NaN or an interpolated value. See RSKtools on-line user manual for further information.

```
rsk = RSKcorrecthold(rsk,'action','interp');
```

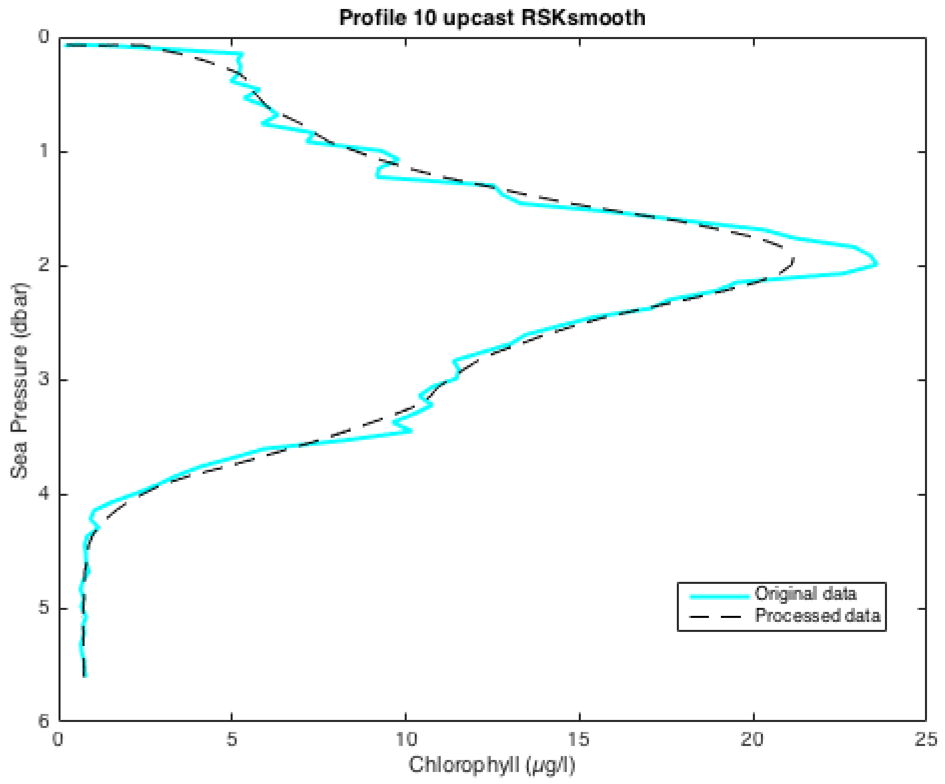
Low-pass filtering

Low-pass filtering is commonly used to reduce noise and to match sensor time constants. Applying a low pass filter to temperature and conductivity smooths high frequency variability and compensates for differences in sensor time constants. Users may also wish to smooth other channels to reduce noise (e.g., optical channels such as chlorophyll).

Most RBR instruments designed for profiling (branded as |*fast*) are equipped with thermistors that have a time constant of 100 msec, which is slower than the conductivity cell. When the time constants are different, salinity will contain spikes at strong gradients. The solution is to "slow down" the conductivity sensor to match the thermistor. In this example data set, the logger sampled at 6 Hz (found using `readsamplingperiod(rsk)`), so a 5 sample running average provides more than sufficient smoothing, and also reduces noise.

Note: All functions that alter the data have an optional input called `'visualize'` that, when activated, plots data before and after applying the function. Users specify which profile(s) to visualize.

```
rsk = RSKsmooth(rsk,'channel',
{'temperature','conductivity'}, 'windowLength', 5);
rsk = RSKsmooth(rsk,'channel','chlorophyll', 'windowLength',
 9, 'visualize', 10);
```



Alignment of conductivity and temperature

Conductivity, temperature, and pressure need to be aligned in time to account for the fact these sensors are not always co-located on the logger. The implication is that, under dynamic conditions (e.g., profiling), the sensors are measuring a slightly different parcel of water at any instant.

Sensors with long time constants introduce a time lag to the data. Dissolved oxygen sensors often have a long time constant, and this can be fixed to some degree by shifting the data in time.

When temperature and conductivity are misaligned, salinity will contain spikes at sharp interfaces and a bias in continuously stratified environments. Properly aligning the sensors, together with matching the time response, will minimize salinity spiking and bias.

In the case of RBR instruments, pressure is effectively an instantaneous measurement, and does not need to be lagged with respect to time. The most important consideration is the relative timing of conductivity and temperature. Therefore, in this example, we align conductivity to temperature by shifting conductivity in time.

The classic approach to determine the optimal lag is to compute and plot salinity for a range of lags, and choose the lag (often by eye) with the smallest salinity spikes at sharp temperature interfaces. As an alternative approach, RSKtools includes a function called `RSKcalculateCTlag` that estimates the optimal lag between conductivity and temperature by minimizing salinity spiking. We currently suggest using both approaches to check for consistency. See the `RSKcalculateCTlag` help page for more information. Typical lag values are 0, +1, or +2 samples for a 6 Hz instrument (about +0.2 sec).

```
lag = RSKcalculateCTlag(rsk);
rsk = RSKalignchannel(rsk,'channel','conductivity','lag',lag);
```

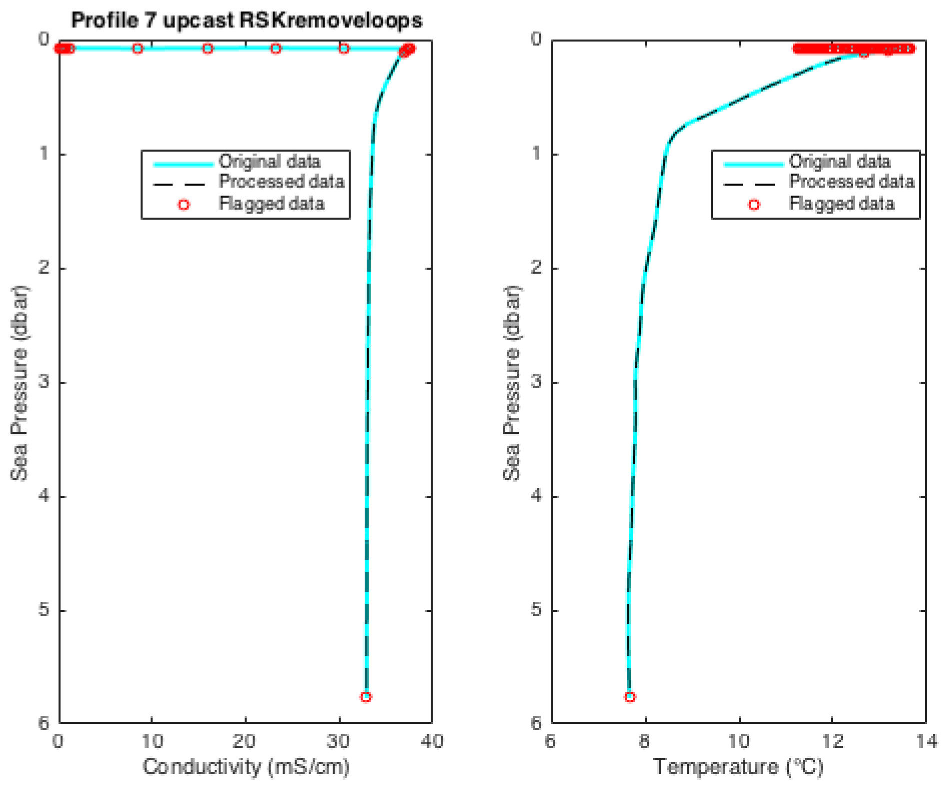
Remove loops

Profiling in rough seas can cause the CTD profiling rate to vary, and even change sign (i.e., the CTD momentarily changes direction). When this happens, the CTD effectively samples its own wake, degrading the quality of the profile in regions of strong gradients. The measurements taken when the instrument is profiling too slowly or during a pressure reversal should not be used for further analysis. We recommend using `RSKremoveloops` to flag and treat the data when the instrument 1) falls below a threshold speed and 2) when the pressure reverses (the CTD "loops"). Before using `RSKremoveloops`, use `RSKderivedepth` to calculate depth from sea pressure, and then use `RSKderivevelocity` to calculate profiling rate.

In the example data set, good data is collected on the *upcast*. `RSKremoveloops`, when applied to this data, removes data when the instrument profiled slowly near the surface.

```
rsk = RSKderivedepth(rsk);
rsk = RSKderivevelocity(rsk);

% Apply the algorithm
rsk = RSKremoveloops(rsk,'threshold',0.3,'visualize',7);
```



Derive Practical Salinity

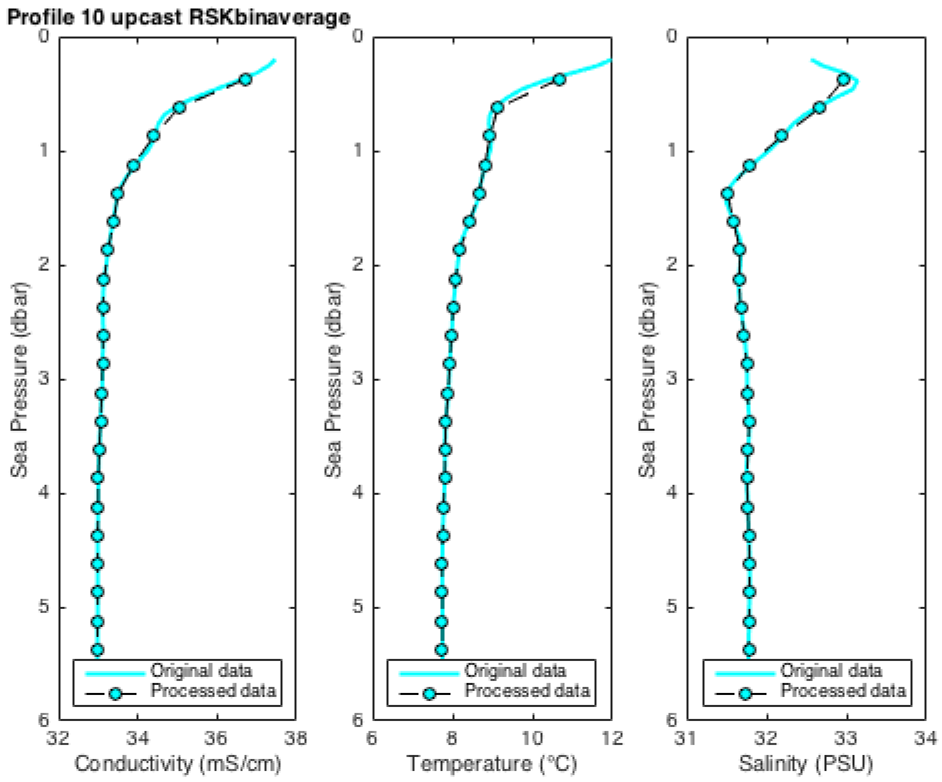
RSKtools includes a convenience function to derive Practical Salinity using the TEOS-10 GSW function `gsw_SP_from_C`. The TEOS-10 GSW Matlab toolbox is freely available from http://www.teos-10.org/software.htm. The result is stored in the data table along with other measured and derived channels.

```
rsk = RSKderivesalinity(rsk);
```

Bin average all channels by sea pressure

Bin averaging reduces sensor noise and ensures that each profile is referenced to a common grid. The latter is often an advantage for plotting data as "heatmaps." `RSKbinaverage` allows users to bin channels according to any reference, but the most common choices are time, depth, and sea pressure. It also can handle grids with a variable bin size. In the following, the data are averaged into 0.25 dbar bins.

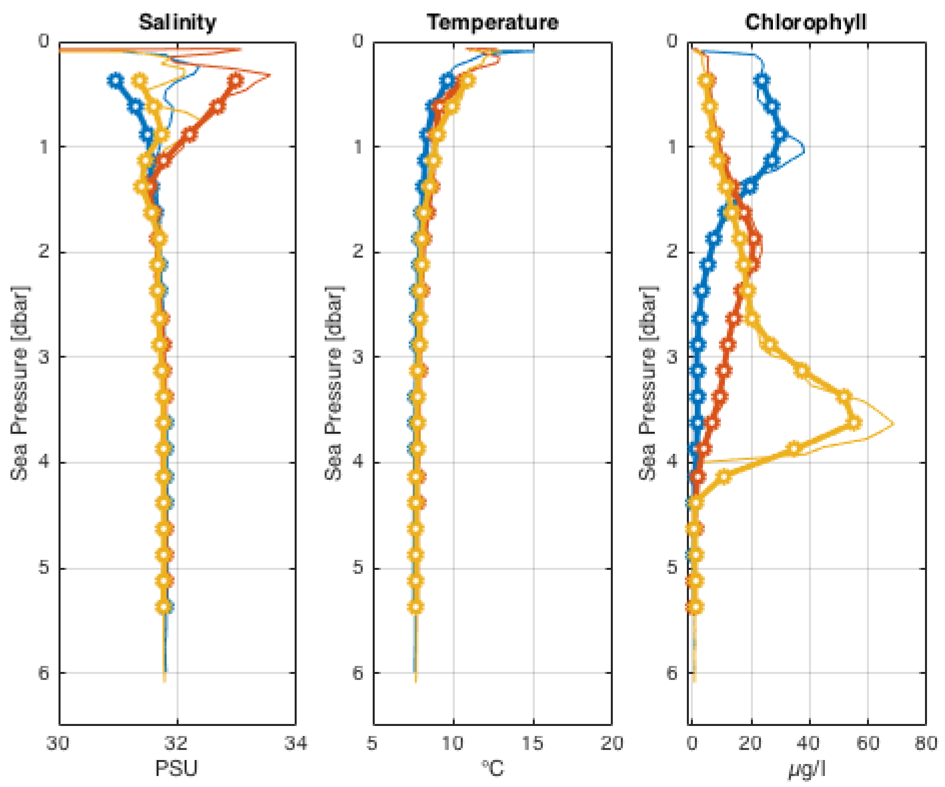
```
rsk = RSKbinaverage(rsk, 'binBy', 'Sea Pressure', 'binSize', 0.25, ...
                         'boundary', [0.5 5.5],'direction', 'up',...
                         'visualize', 10);
h = findobj(gcf,'type','line');
set(h(1:2:end),'marker','o','markerfacecolor','c')
```



Compare the raw and processed data

Use `RSKplotprofiles` to compare the binned data to the raw data for a few example profiles. Processed data are represented with thicker lines.

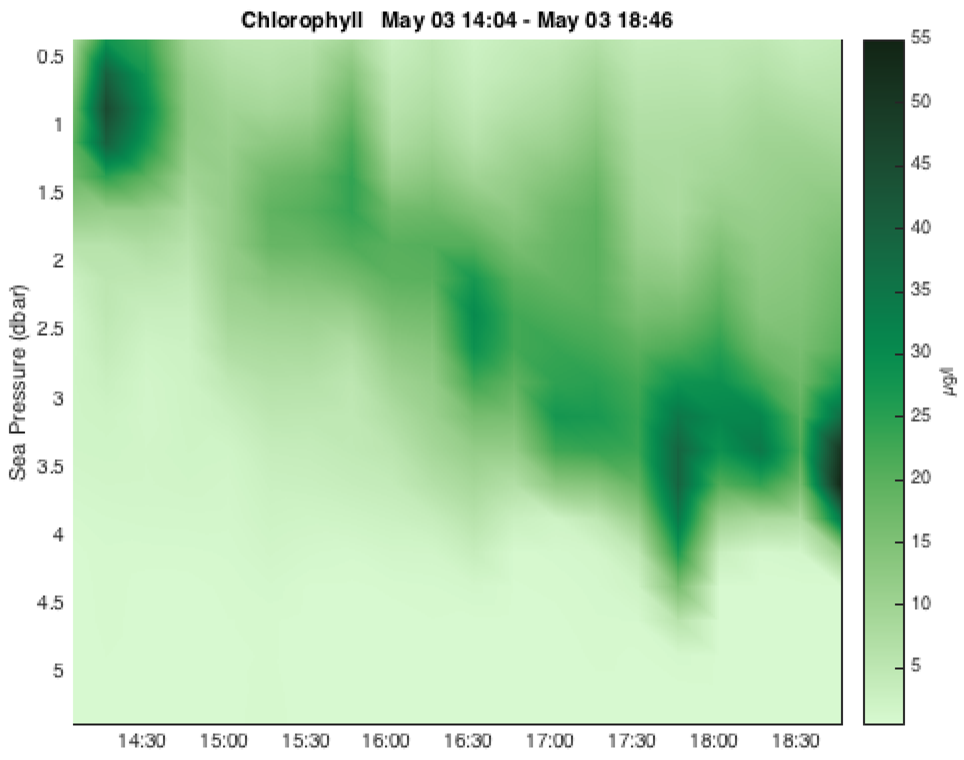
```
figure
channel = {'salinity','temperature','chlorophyll'};
profile  = [3 10 20];
h1 = RSKplotprofiles(raw,'profile',profile,'channel',channel);
h2 = RSKplotprofiles(rsk,'profile',profile,'channel',channel);
set(h2,'linewidth',3,'marker','o','markerfacecolor','w')
ax = findobj(gcf,'type','axes');
set(ax(1),'xlim',[-2 80])
set(ax(3),'xlim',[30 34])
set(ax,'ylim',[0 6.5])
```



Visualize data with a 2D plot

`RSKimages` generates a time/depth heat-map of a channel. The x-axis is time; the y-axis is a reference channel (default is sea pressure). All of the profiles must be evaluated on the same reference channel levels, which is accomplished with `RSKbinaverage`. The function supports customisable rendering to determine the length of gap shown on the plot. For more details, see [RSKtools on-line user manual]

```
figure
RSKimages(rsk,'channel','chlorophyll','direction','up');
```



Add station data

RSKaddstationdata appends station data to the profile data structure. The allowable fields are: latitude, longitude, station, cruise, vessel, depth, date, weather, crew, comment and description. The function is vectorized, which allows multiple station data inputs for multiple profiles. When there is only one station data input specified for multiple profiles, all profiles will be assigned with the same value.

Station data is written into both the CSV and ODV file headers when the data is exported with RSK2CSV and RSK2ODV.

```
% apply the same station data to each profile
rsk = RSKaddstationdata(rsk,'latitude',45,'longitude',-25,...
                         'station',{'SK1'},'vessel',{'R/V RBR'},...
                         'cruise',{'Skootamatta Lake 1'});

% or use vectorized inputs
rsk = RSKaddstationdata(rsk,'profile',4:6,'station',
{'S1','S2','S3'},...
                         'latitude',[45,44,46],...
                         'longitude',[-25,-24,-23],...
                         'comment','RSKtools demo');
```

To view the station data in the 4th profile:

```
disp(rsk.data(4))

           tstamp: [21x1 double]
           values: [21x11 double]
        direction: 'up'
    profilenumber: 4
     samplesinbin: [21x1 double]
         latitude: 45
        longitude: -25
          station: {'S1'}
           cruise: {'Skootamatta Lake 1'}
           vessel: {'R/V RBR'}
          comment: {'RSKtools demo'}
```

Export logger data to CSV files

`RSK2CSV` writes logger data and metadata to one or more CSV files. The CSV files contain a header with important logger metadata, station metadata (if it exists), and a row of variable names and units above each column of channel data. If the data has been parsed into profiles, then one file will be written for each profile and an extra column called 'cast_direction' will be included. The column will contain 'd' or 'u' to indicate whether the sample is part of the downcast or upcast, respectively. Users can select which channels and profiles are written, the output directory, and also specify comments to be placed after the metadata in the file header.

```
RSK2CSV(rsk,'channel',{'Conductivity','Pressure','Dissolved   O2'},...
'profile', 1:3 ,'comment','Hey Jude');
```

RSKtools also has export functions to write Ocean Data View files, MAT files, and Ruskin RSK files.

Display a summary of all the processing steps

Type `rsk.log{:,2}` at the command prompt.

Other Resources

We recommend reading:

- the RSKtools on-line user manual for detailed RSKtools function documentation.
- the RSKtools Getting Started for an introduction on how to load RBR data into Matlab from RSK files, make plots, and access the data.

About this document

This document was created using Matlab™ Markup Publishing. To publish it as an HTML page, run the command:

```
publish('PostProcessing.m');
```

See `help publish` for more document export options.

Published with MATLAB® R2015b