RSKtools for Matlab processing RBR data

Table of Contents

Introduction 1
RSKtools help 1
Getting set up 1
Remove atmospheric pressure from measured total pressure 2
Correct for A2D zero-order hold 2
Low-pass filtering 3
Alignment of conductivity to temperature and pressure 4
Remove loops 4
Derive practical salinity 5
Compute an extra variable and add it to the RSK structure 5
Bin average all channels by pressure 6
Plot the bin averaged profiles 6
2D plot 7
Add station metadata 8
Output rsk structure as CSV files 9
Display a summary of all the processing steps 9
Other Resources 9
About this document 9

RSKtools v2.3.0; RBR Ltd. Ottawa ON, Canada; support@rbr-global.com; 2018-05-09

Introduction

Version 2.0.0 of RSKtools included new functions to post-process RBR logger data. Below we show how to implement some common processing steps to obtain the highest quality data possible.

RSKtools help

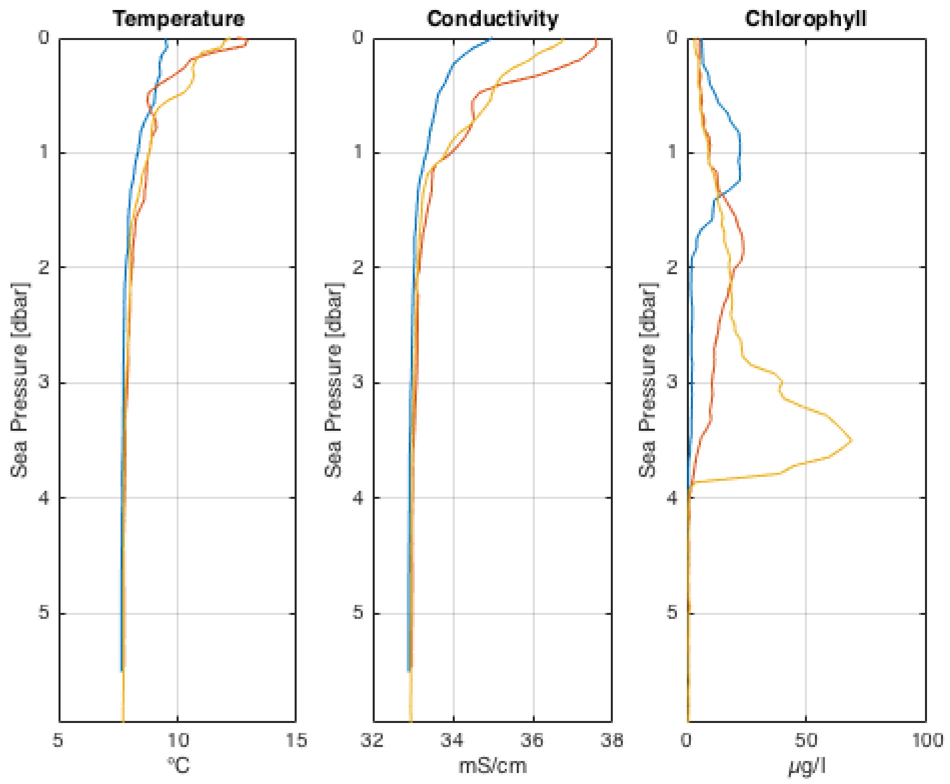
All post-processing functions are customisable with name-value pair input arguments. Documentation for each function can be accessed using the Matlab commands `doc` and `help`. For example, to open the help page for `RSKsmooth`, type: `doc RSKsmooth` at the Matlab command prompt.

Getting set up

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

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

% read the upcast from profiles 1 - 20
```

```
rsk = RSKreadprofiles(rsk, 'profile', 1:20, 'direction', 'up');

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



Remove atmospheric pressure from measured total pressure

We suggest deriving sea pressure first, especially when an atmospheric pressure other than the nominal value of 10.1325 dbar is used, because deriving salinity and depth requires sea pressure.

```
rsk = RSKderiveseapressure(rsk);

% Hang on to the raw data for plotting later.
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 simple 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,holdpts] = RSKcorrecthold(rsk,'channel',...
    {'temperature','conductivity','chlorophyll'},'action','interp');
```

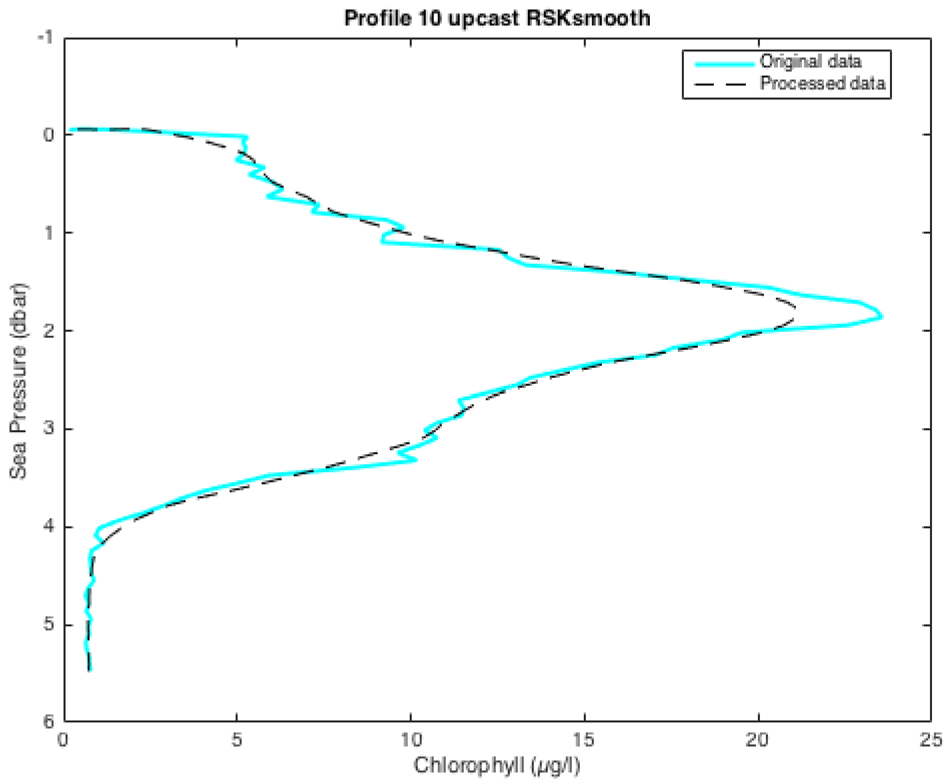
Low-pass filtering

Applying a low pass filter to temperature and conductivity smooths high frequency variability and compensates for differences in sensor time constants (the thermistor has a slower response to changes than the conductivity cell). Users may also wish to smooth other channels to reduce noise (e.g., optical channels such as chlorophyll). RSKtools includes a function called `RSKsmooth` for this purpose.

Most RBR profiling instruments ("|fast") are equipped with thermistors that have a time constant of 100 ms, so the conductivity should be smoothed to match that value. In this example, the logger sampled at 6 Hz (found using `RSKsamplingperiod(rsk)`), so a 5 sample running average should provide sufficient smoothing.

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



Alignment of conductivity to temperature and pressure

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. In other words, at any instant, the sensors are measuring a slightly different parcel of water. Furthermore, sensors with long time constants introduce a time lag to the data. For example, dissolved oxygen sensors often have a long time constant.

When temperature and conductivity are misaligned, the salinity will contain spikes at sharp interfaces and may even be biased. Properly aligning the sensors, together with matching the time response, will minimize spiking and bias in salinity.

In the case of RBR instruments, the pressure sensor is effectively an instantaneous measurement, and does not need to be lagged. However, conductivity and temperature must be aligned with respect to each other. In this example, we align conductivity to temperature by shifting conductivity in time.

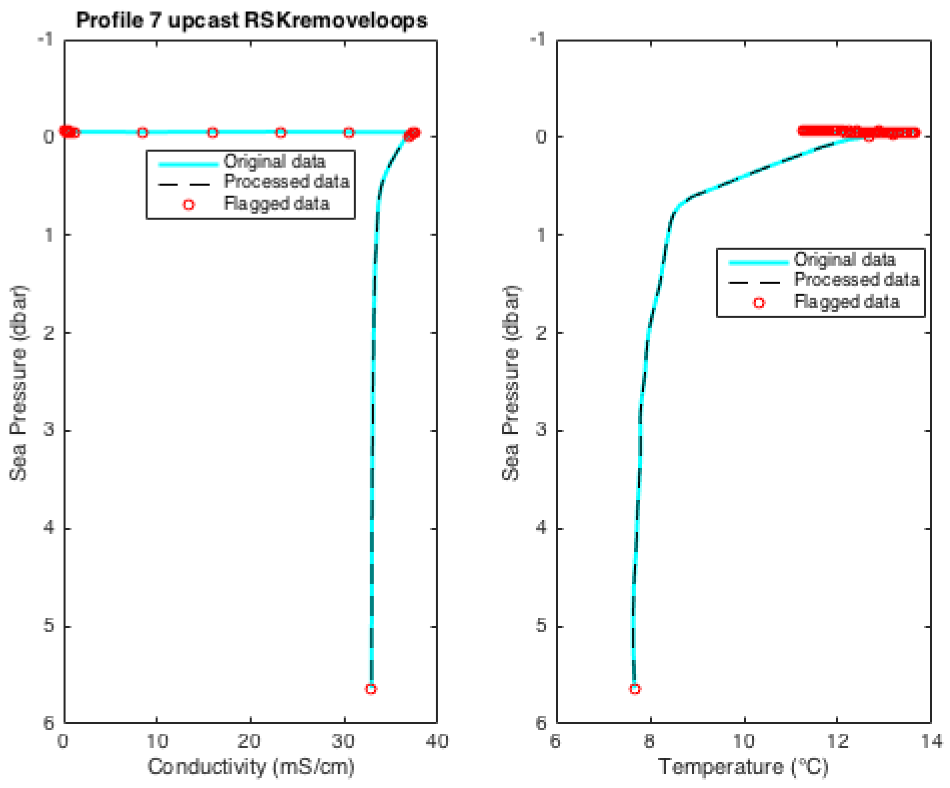
The classic approach to choose the optimal lag is to compute the salinity for a range of lags, plot each curve, and choose the curve (often by eye) with the smallest salinity spikes at sharp interfaces. As an alternative approach, RSKtools provides a function called `RSKcalculateCTlag` that estimates the optimal lag between conductivity and temperature by minimising salinity spiking. We currently suggest using both approaches to check for consistency. See the `RSKcalculateCTlag` help page for more information.

```
lag = RSKcalculateCTlag(rsk);
rsk = RSKalignchannel(rsk, 'Conductivity', lag);
```

Remove loops

Profiling during rough seas can cause the CTD profiling rate to vary, or even change sign (i.e., the CTD momentarily changes direction). When this happens, the CTD can effectively sample its own wake, potentially 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 this particular example, the `RSKremoveloops` algorithm removes the surface soak.

```
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);
```

Compute an extra variable and add it to the RSK structure

Users may wish to add additional data to the RSK structure. We illustrate how this is done by computing Absolute Salinity and adding it to the RSK structure

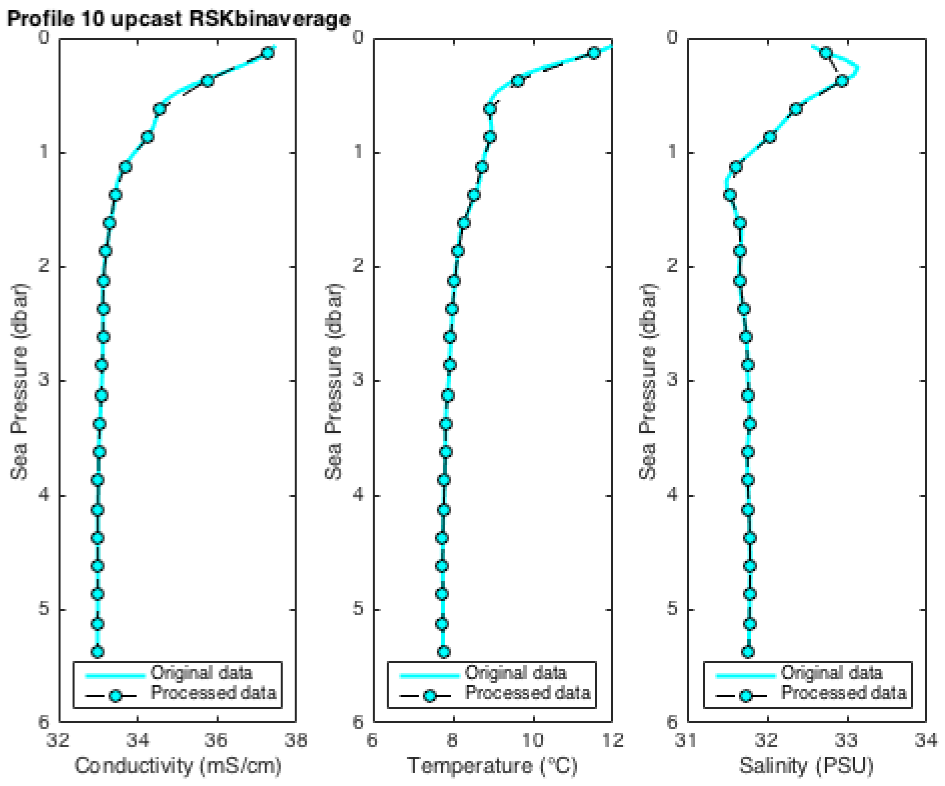
```
p = getchannelindex(rsk,'sea pressure');
sp= getchannelindex(rsk,'salinity');

ncast = length(rsk.data);
sa = repmat(struct('values',[]),1,ncast);
for k=1:ncast,
  sa(k).values = gsw_SA_from_SP(rsk.data(k).values(:,sp),...
                                rsk.data(k).values(:,p),-150,49);
end
```

```
rsk = RSKaddchannel(rsk,sa,'Absolute Salinity','g/kg');
```

Bin average all channels by pressure

Average the data into 0.25 dbar bins using `RSKbinaverage`.

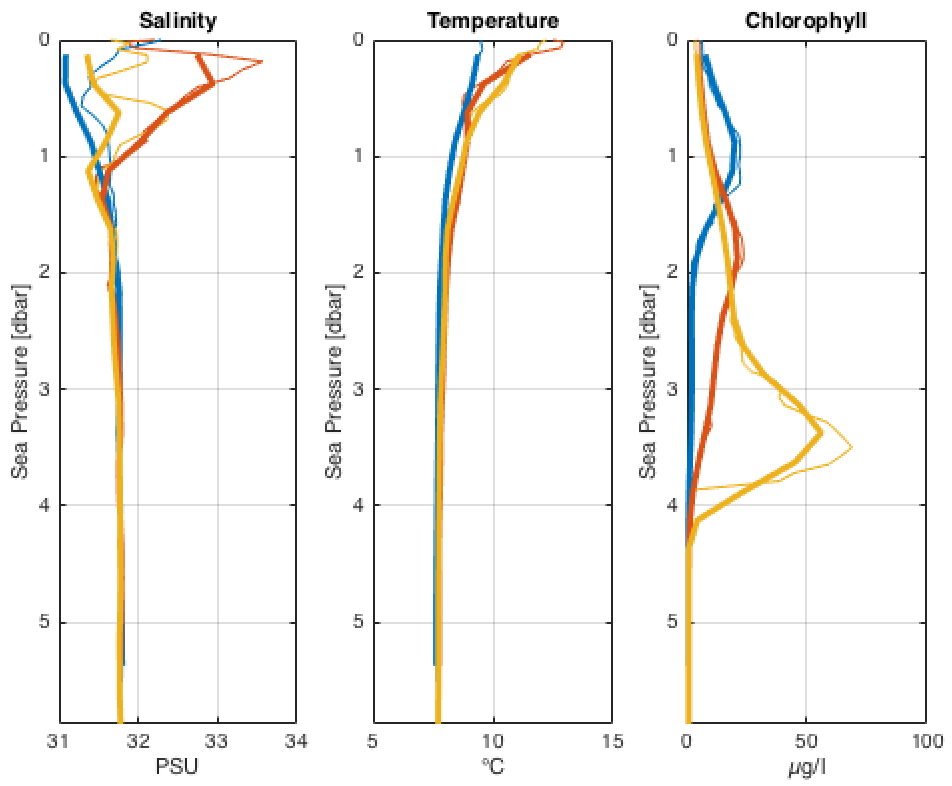
```
rsk = RSKbinaverage(rsk, 'binBy', 'Sea Pressure', 'binSize',
 0.25, 'direction', 'up', 'visualize', 10);
h = findobj(gcf,'type','line');
set(h(1:2:end),'marker','o','markerfacecolor','c')
```



Plot the bin averaged profiles

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

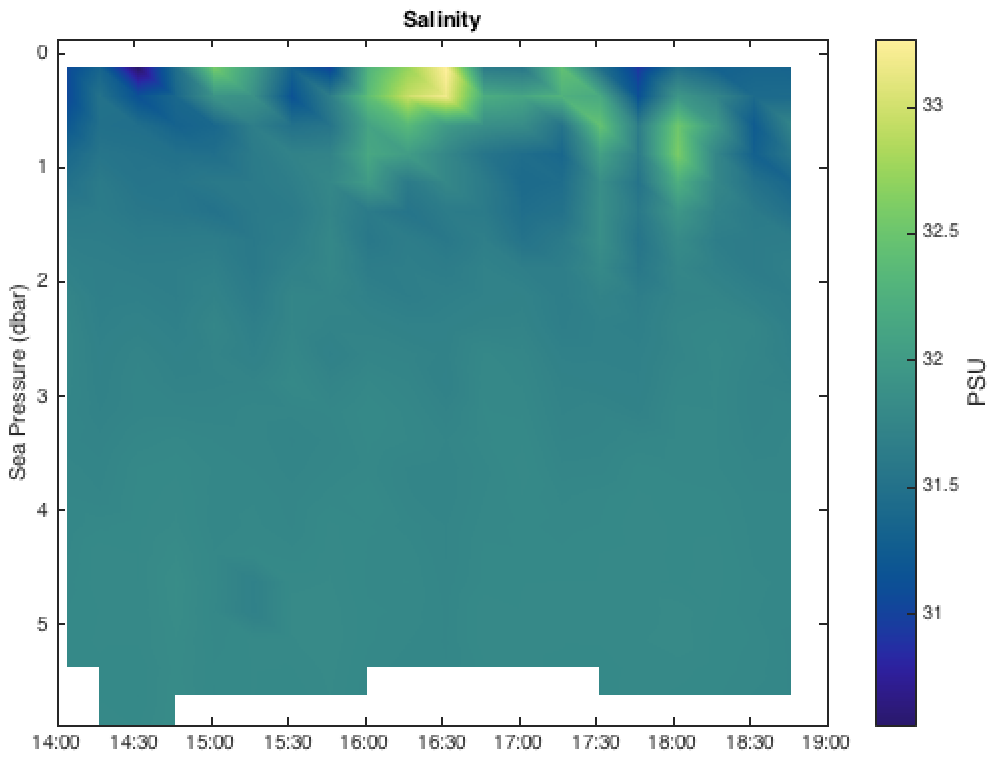
```
clf
h1 = RSKplotprofiles(raw,'profile',[1 10 20],'channel',
{'salinity','temperature','chlorophyll'});
h2 = RSKplotprofiles(rsk,'profile',[1 10 20],'channel',
{'salinity','temperature','chlorophyll'});
set(h2,{'linewidth'},{3})
```



2D plot

`RSKplot2D` 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

```
clf
RSKplot2D(rsk, 'Salinity','direction','up');
```



Add station metadata

`RSKaddmetadata` appends station metadata 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 metadata inputs for multiple profiles. When there is only one metadata input specified for multiple profiles, all profiles will be assigned with the same value.

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

```
rsk = RSKaddmetadata(rsk,'latitude',45,'longitude',-25,...
                         'station',{'SK1'},'vessel',{'R/V RBR'},...
                         'cruise',{'Skootamatta Lake 1'});

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

To view the metadata in the 4th profile:

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

```
           tstamp: [25x1 double]
           values: [25x12 double]
```

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

Output rsk structure as CSV files

`RSK2CSV` outputs the RSK structure format into one or more CSV file. The CSV file contains important logger metadata 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. Besides, 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 customize which channel, profile for outputs, output directory and comments attached to the end of the header.

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

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