# 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 ..... 3
Remove loops ..... 3
Derive practical salinity ..... 4
Compute an extra variable and add it to the RSK structure ..... 4
Bin average all channels ..... 4
Plot the bin averaged profiles ..... 4
2D plot ..... 5
Display a summary of all the processing steps ..... 6
Other Resources ..... 6
About this document ..... 6



RSKtools v2.2.0; RBR Ltd. Ottawa ON, Canada; support@rbr-global.com; 2018-01-25


# Introduction

A suite of new functions are included in RSKtools v2.0.0 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. To process all data using the default parameters no name-value pair arguments are required. All the information above is available for each function using `help`, for example: `>> help RSKsmooth`.

# Getting set up

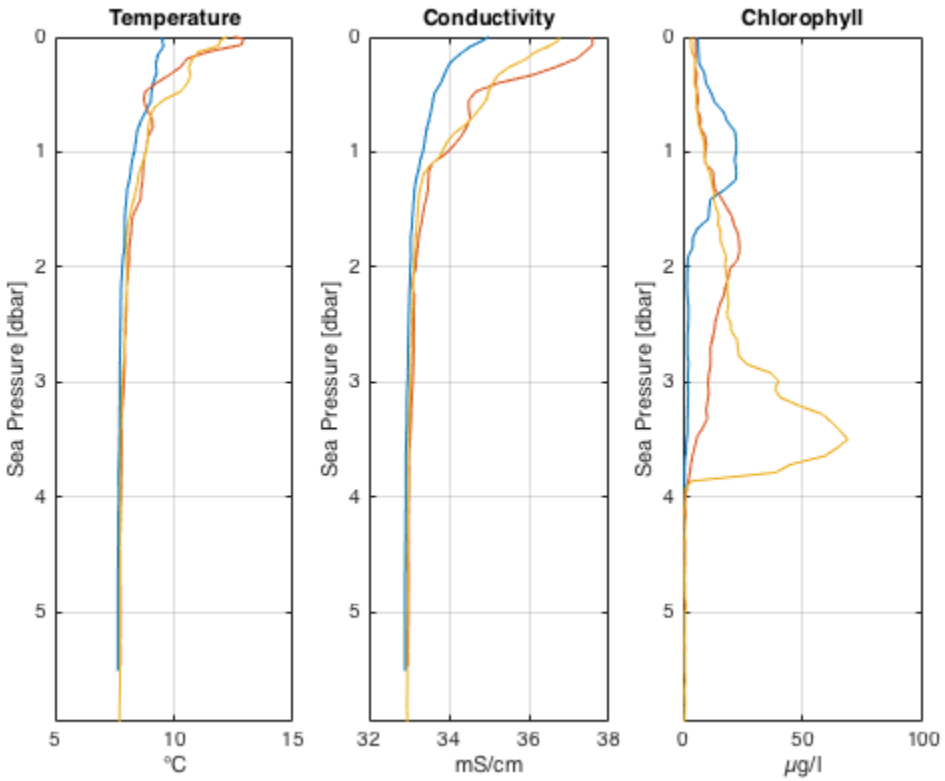
Review Standard for help.

```
% 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.

The standard RBR thermistor on profiling instruments has time constant of about 0.6 s, so the conductivity should be smoothed to match that value (RBR's fast thermistors have a 0.1 s response time). In this example, the logger samples at 6 Hz (`rsk.continuous.samplingPeriod`), so a 5 sample running average should provide sufficient smoothing.

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

# 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 physically co-located on the logger. In other words, at any instant, the sensors are measuring a slightly different parcel of water. 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.

The classic approach 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, RSKtools provides a function called `RSKcalculateCTlag` that estimates the optimal lag between conductivity and temperature by minimising salinity spiking. See `help RSKcalculateCTlag`.

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

# Remove loops

Profiling during rough seas can cause the CTD descent (or ascent) 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 find and NaN 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.

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

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

# 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

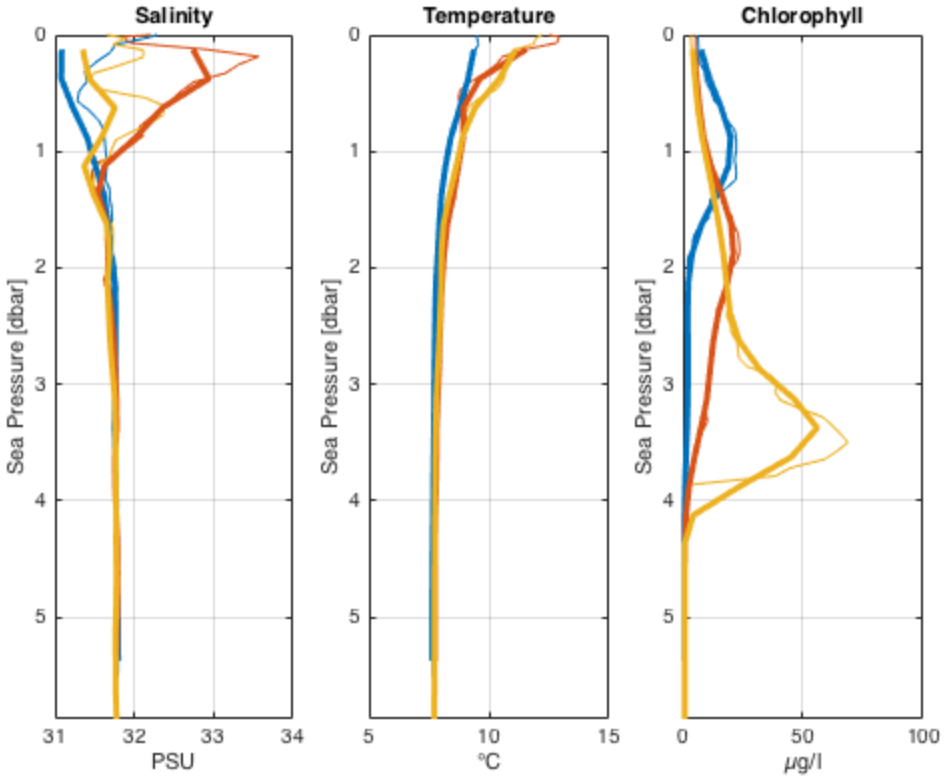
Average the data into 0.25 dbar bins using `RSKbinaverage`.

```
rsk = RSKbinaverage(rsk, 'binBy', 'Sea Pressure', 'binSize',
 0.25, 'direction', 'up');
```

# Plot the bin averaged profiles

Compare the binned data to the raw data for a few example profiles, processed data are represented with thicker lines.

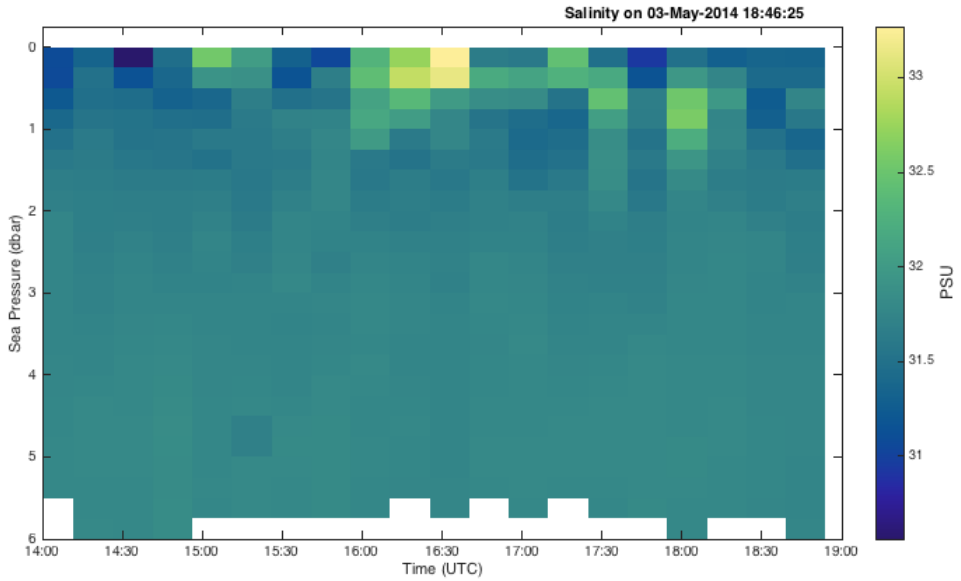
```
clf
h1 = RSKplotprofiles(raw,'profile',[1 10 20],'channel',
{'salinity','temperature','chlorophyll'});
h2 = RSKplotprofiles(rsk,'profile',[1 10 20],'channel',
{'salinity','temperature','chlorophyll'});
set(h2,{'linewidth'},{3})
```




# 2D plot

`RSKplot2D` generates a plot of the profiles over time. The x-axis is time; the y-axis is a reference channel (default is sea pressure). All data elements must have identical reference channel samples. `RSKbinaverage` has achieved this here.

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




# 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*