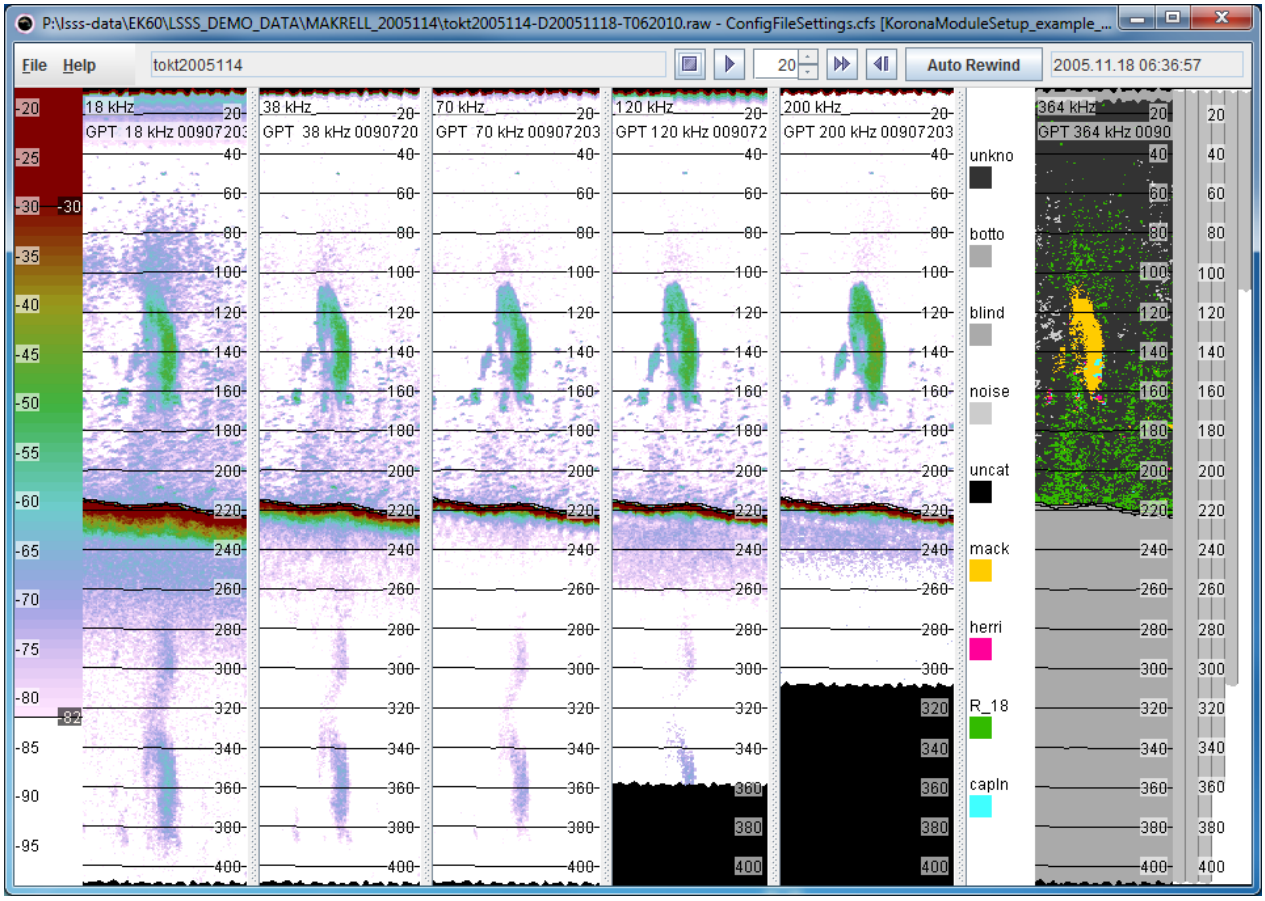


Korona

User Guide

Version 2.2.0-alpha

August 31, 2017

Contents

- KORONA (LSSS preprocessor)
  - Getting started
  - Working with Korona
    - Korona playbox
    - Korona relay
    - Korona batch
    - Config file settings
      - Module configuration
      - Horizontal transducer offsets
      - Vertical transducer offsets
      - Transducer ranges
      - Categorization library
      - Plankton
      - Broadband notch filters
      - Broadband splitter bands
      - Towfish
    - Calibration
    - Broadband calibration
    - Categorization
    - Examples
      - 1 - Data improvement
      - 2 - Data reduction
      - 3 - School detection
      - 4 - Echo trace detection
      - 5 - Categorisation
      - 6 - School categorisation
      - 7 - Zooplankton size estimation
      - 8 - Visualisation of intermediate results
      - 9 - Colour scales
      - 10 - Bottom detection
  - Modules
    - Broadband
      - Broadband notch filter
      - Broadband splitter
    - Categorization
      - Categorization
      - Isolation
      - School categorization
    - Data reduction
      - Angle deletion
      - Channel data removal
      - Channel removal
      - Combination remover
      - Complex to real
      - Data reduction
      - Downsampling
      - Empty ping removal
      - Ping thinning
      - Remove single channel
    - Filters
      - Dilate
      - Edge detection
      - Erode low values
      - Erode
      - Fill missing data
      - Filter 3×3
      - Median
      - Ping collapsing
      - Remove bottom
      - Threshold

        - I/O
            - Writer
        - Misc
            - Cds viewer
            - Rescale
            - Time interval
        - Noise
            - Bubble spike filter
            - Noise quantification
            - Noise remover
            - Noise visualization
            - Spike filter
            - Spot noise
        - Offset
            - Horizontal offset correction
            - Vertical offset correction
        - Plankton
            - Plankton inversion
                - Algorithms
                - Models
        - School (region)
            - School detection
            - School filtering
            - Trace border
        - Smooth (convolution)
            - Smoother
        - Towfish
            - Towfish
        - Tracking
            - Tracking
        - Bottom detection
        - Combination
        - Comment
        - Expression
        - Plugin
        - Temporary computations begin
        - Temporary computations end
    - Datagrams
        - Bot0Datagram
        - Cac0Datagram
        - Cad0Datagram
        - Cas0Datagram
        - Cds0Datagram
        - Con0Datagram
        - Dep0Datagram
        - Idx0Datagram
        - Nme0Datagram
        - Nqp0Datagram
        - Pic0Datagram
        - Pid0Datagram
        - Raw0Datagram
        - RegionBorderDatagram
        - RegionInfoDatagram
        - RegionTableOfContentsDatagram
        - Tbr0Datagram
        - Tnf0Datagram
        - Ttc0Datagram
        - UnknownDatagram
    - Reference
        - Literature references
        - EK80
            - Broadband data
            - Narrowband data

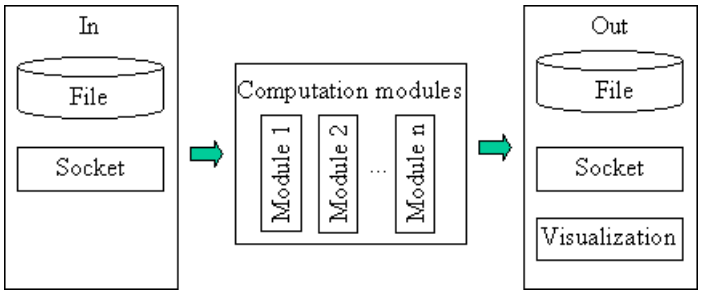
    - Split beam angles
    - Transducer impedance
- Miscellaneous
    - NMEA

Korona

Korona is the LSSS preprocessor, and is used for processing and visualizing data from an acoustic source. The acoustic sources may be e.g. EK60, EK500, ME70, MS70, SX90 or similar equipment.

See the getting started section for basic introduction to using the system.

Datagrams go through a sequence of processing modules as illustrated in the figure below.



Data processing in Korona.

The sequence of processing modules can be loaded from file, or configured using the module editor.

Getting started with Korona

The processing done by Korona can either be done as batch processing, using Korona relay, or interactively using Korona playbox.

Viewing a data file

1. Start Korona relay and specify the location of data files in "Source directory". A list of available data files in that directory will then appear below.
2. Select one of the files in the list by clicking on it.
3. Click the "Korona playbox" button to start Korona playbox.

Computation modules

Computation modules are specified by the "Module configuration" file in the Config file settings.

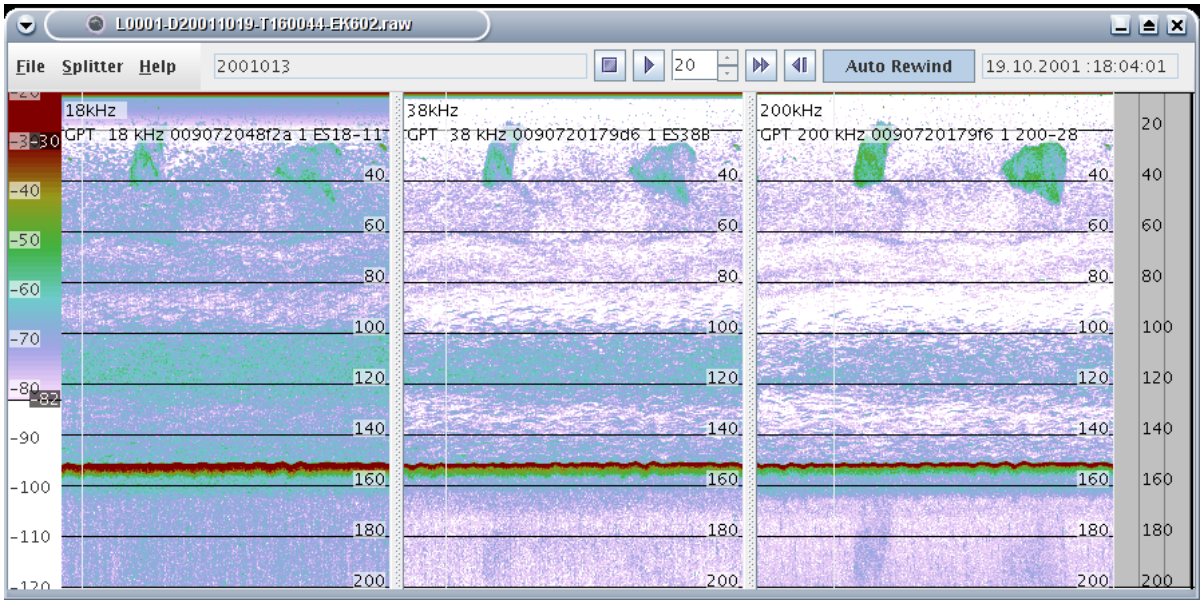
1. Click the "Edit" button for the "Config file settings" file in Korona relay.
2. Specify a "Module configuration" file by one of
    i. Click the "Browse" button to select an existing file.
    ii. Click the "Edit" button to specify modules using the module editor.

Working with Korona

This chapter describes some of the typical tasks when working with Korona.

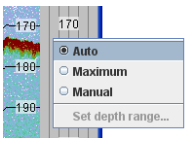
Korona playbox

Korona playbox allows for interactively exploring the processing. It can be launched from Korona relay.



Depth range menu

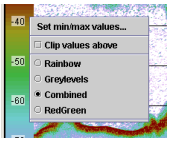
The panel on the far right shows the depth range of the data. Right-click here to access a menu with options for setting the depth range:



- *Auto:* Visible depth range is set automatically to a reasonable range based on the last 100 pings.
- *Maximum:* Visible depth range is set to the maximum (join) range during the last 100 pings.
- *Manual:* Visible depth range is user defined. Drag the two markers to change the visible range or open a dialog by selecting *Set depth range...:*

Colormap menu

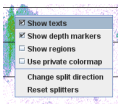
The panel on the far left shows the mapping from S_V values to colors. Drag the two markers to change maximum and minimum values for the colormap. Right-click to access a menu with these options:



- *Set min/max values...:* Specify maximum and minimum values for the colormap in a separate dialog.
- *Clip values above:* Toggle on and off using white instead of the colormap's extreme color for values above the maximum value.
- *Rainbow*| *Graylevels* | *Combined* | *RedGreen:* Choose which colormap to use.

Echogram menu

Right-click in an echogram panel to access a menu with the following options:

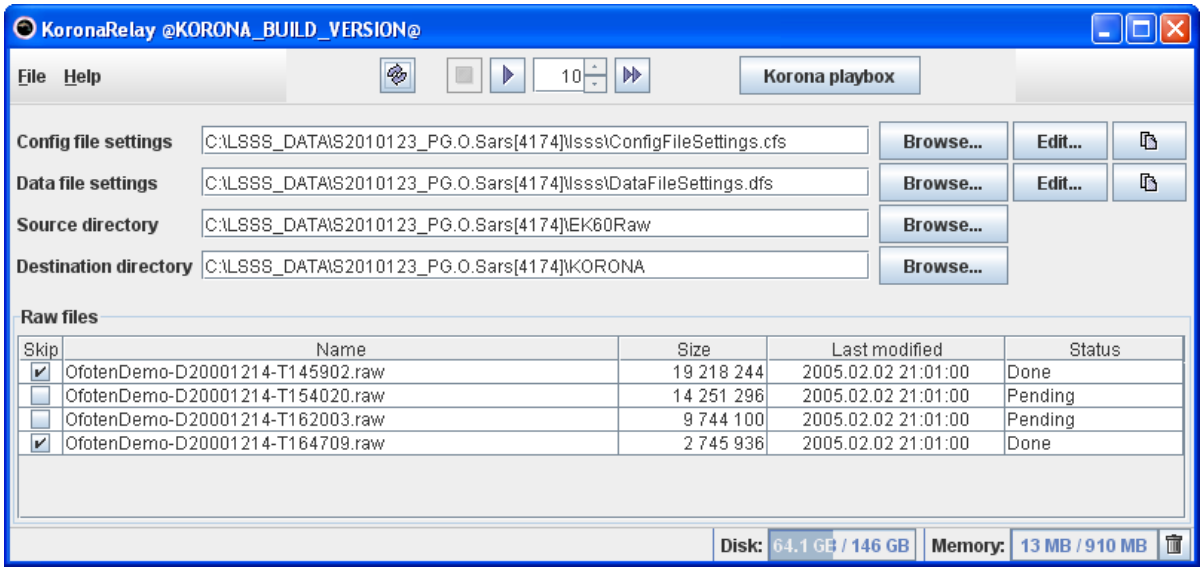


- *Showtexts:* Toggle on and off text labels in upper left corner.
- *Showdepth markers:* Toggle on and off depth labels and drawing of horizontal lines at constant depth.
- *Showregion:* Toggle on and off plotting of regions detected by School detection module.

- *Use private colormap:* Toggle on and off the use of a separate colormap panel for this echogram panel. This separate colormap panel has options to show categories.
- *Change split direction:* When a data file has more than one channel it is possible to view the channels either side by side, or stacked on top of each other. The size of each echogram panel may be adjusted by dragging the dividers with the mouse.
- *Reset split:* Sets equal distance between the splitters.

Korona relay

Korona relay is intended for batch processing. Launch Korona playbox for interactive exploration.



- *Config file settings:* A file containing the paths to config files used by the processing.
- *Source directory:* Location of files to be processed.
- *Destination directory:* Location of the processed files.

status.xml

As Korona relay processes files it writes the processing status to the file *status.xml* in the destination directory. This is done for several reasons:

- To speed up the processing, it is possible to use multiple instances of Korona relay on different computers. The *status.xml* file coordinates the collaboration by the different Korona relay instances.
- The processed files are written with an extra *.new*file name suffix instead of overwriting existing files. This is done to not interfere with other programs using the existing files. The *status.xml* file is used for coordinating when the existing files are replaced by the *.new*files.

If the *status.xml* file becomes corrupt or contains wrong information, it might be necessary to re-create it. This can be done in the file menu of Korona relay.

Korona batch

Korona batch is intended for scripted batch processing.

Command line options:

`--cfs`	The config file settings used by the processing.
`--source`	Directory with files to be processed.
`--destination`	Directory where the processed files will be written.
`--help`	Display help information.

Example:

```
KoronaBatch.bat --cfs=D:\ConfigFileSettings.cfs --source=D:\OriginalRaw --
destination=D:\ProcessedRaw
```

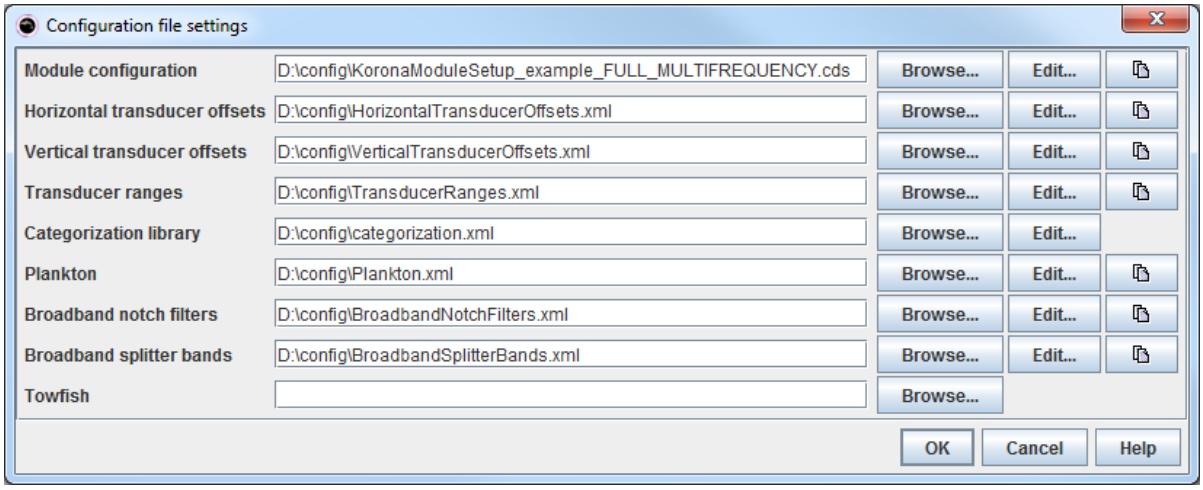
Notes:

- If file paths contain spaces then quotes are necessary, e.g. `"--source=D:\Raw files"`.

Config file settings

The config file settings is a set of config files defining a processing setup. The module configuration specifies the setup of modules, while the other files contain additional configuration referenced by certain modules.

The name of a configuration settings file normally ends with the suffix .cfs.



The files listed are:

- Module configuration
- Horizontal transducer offsets
- Vertical transducer offsets
- Transducer range
- Categorization library
- Plankton
- Broadband notch filters
- Broadband splitter bands
- Towfish configuration

Clicking the Edit... button will for each file bring up an editor with the contents of the config file.

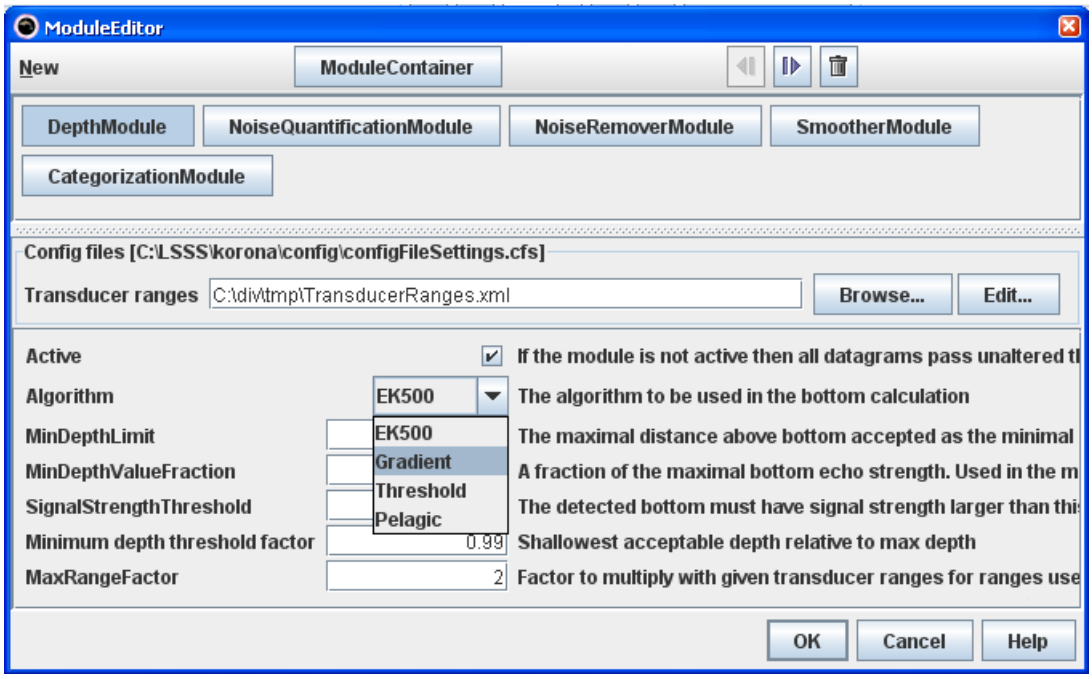
Module configuration

The configuration of a set of modules.

The name of a module configuration file normally ends with the suffix .cds.

Module Editor

The module editor is a tool for configuring the chain of computation modules, and for editing parameters of individual modules. The figure below shows the module editor with the chain of modules in the upper panel, and the parameters for the selected module in the lower panel. The order of the modules in the upper panel is left to right, top to bottom.



Editing the `Algorithm` parameter of `Bottom detection module`.

A module may be

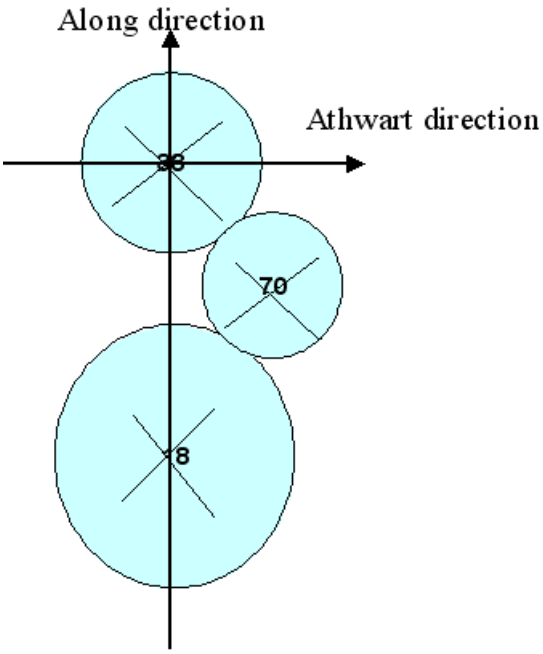
- *created* by selecting module type from the 'New' menu.
- *moved one step forward* in the processing chain by clicking .
- *moved one step backward* in the processing chain by clicking .
- *deleted* by clicking .
- *configured* by editing its parameters in the lower panel.
- *copied* by typing Ctrl-C.
- *cut* by typing Ctrl-X.
- *pasted* by typing Ctrl-V.
- *dragged* with the mouse.

Horizontal transducer offsets

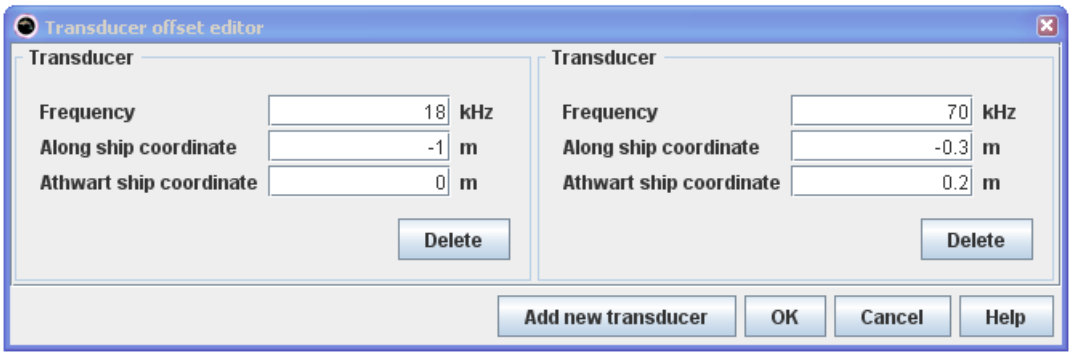
The horizontal positioning of transducers relative to the 38 kHz transducer.

Used by the Horizontal offset correction module.

The picture below shows a transducer configuration seen from above, and the coordinate system used. The positive z (depth) axis is inwards in the figure. Here the 18 kHz transducer has a negative alongship coordinate compared to the 38 kHz transducer, so a negative number should be inserted into the alongship coordinate below for 18 kHz (if 38 kHz is selected as origin). The transducer selected as origin should have zero offsets. The 70 kHz has a positive athwartship coordinate and a negative alongship coordinate compared to 38 kHz, so a positive number should be inserted into to athwartship coordinate for 70 kHz, and a negative number for alongship coordinate.



The transducer offsets are given with respect to one of the transducers, usually the 38 kHz transducer, and are specified in the window shown below. All values are in meters.



To add offset corrections to a new transducer, press "Add new transducer". Press delete to remove the corrections for a transducer.

Pressing OK will save the offsets.

Vertical transducer offsets

Vertical offset correction of transducers.

Used by the Vertical offset correction module.

The vertical offset is divided into two components. The depth coordinate component is offset due to the physical placement. A positive value means that the transducer is placed deeper than the reference depth. Note that this position can also be set in the Simrad logging software, so the offset here is considered an additional offset. The other component, PulseDelayDepthCorrection, is the offset due to pulse delay. A positive value for PulseDelayDepthCorrection is equivalent to an offset due to deeper placement of the transducer compared to the reference depth.

Transducer ranges

This is a configuration file specifying the blind zone and range of each transducer.

Used by several modules, including the Bottom detection module, the Data reduction module and the Categorization module.

If data to be processed contains a channel with a frequency not specified in this file, then the blind zone and the range will be interpolated. However, no extrapolation will be done.

- *Blind zone*: Interpolated to the maximum of the blind zones of the next lower and higher frequencies.
- *Range*: Interpolated linearly using the ranges of the next lower and higher frequencies.

Categorization library

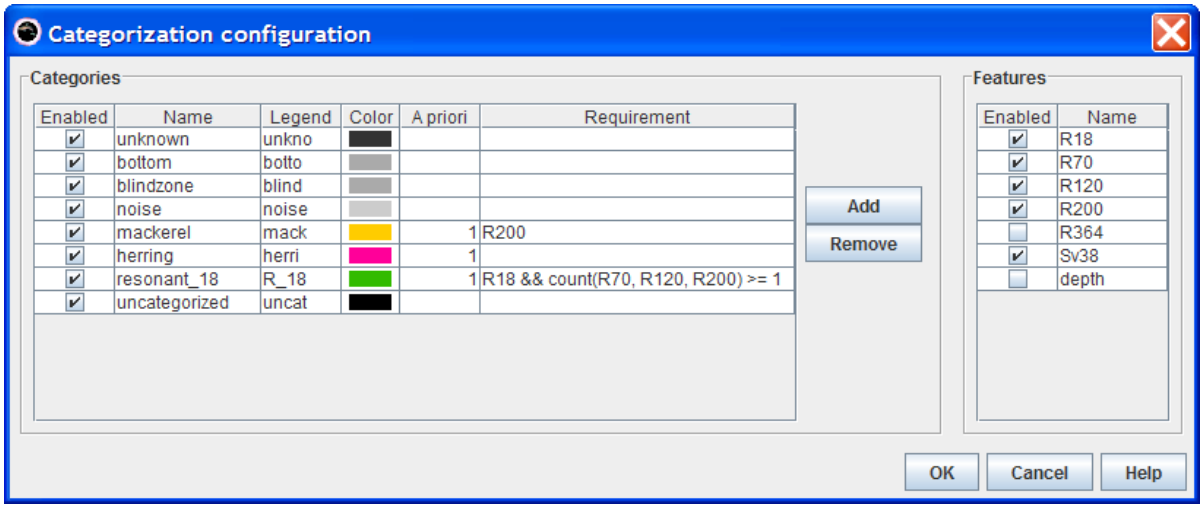
The categorization config file.

Used by the Categorization module.

Categorization library visualizer and editor

The edit menu

Categorization configuration



Here it is possible to define new categories with the **Add** button, and permanently remove categories with the **Remove** button. For each category a number of parameters can be set:

- **Enabled**: The checkboxes in the first column specifies whether the given category is enabled. Disabled categories become invisible to the user and will not be used in categorization.
- **Name**: In the second column the name for the categories can be edited. The names for the permanent categories such as noise, bottom etc. can not be edited.
- **Legend**: The third column is a shorter legend which is used in the colorbar when visualising results from categorization. The name should preferably be 5 characters or shorter.
- **Color**: By double-clicking on the color in the fourth column, the color for the category can be edited.
- **A priori**: In the fifth column the a priory probabilities for the different categories can be given.
- **Min Sv 38**: In the sixth column the a minimum Sv(38 kHz) for the different categories can be given.
- **Max Sv 38**: In the seventh column the a maximum Sv(38 kHz) for the different categories can be given.
- **Requirement**: The eight column specifies a requirement for using that category during categorization. The requirement is a Boolean expression that must evaluate to true for the category to be used.
  - If no expression is specified then the category is always used.
  - R18 evaluates to true if 18 kHz is available and false otherwise. Similarly for R70, R120, etc.
  - Boolean operators such as && and || can be used.
  - The function count(Boolean...) takes any number of Boolean parameters and returns the number of parameters with value true.

  Examples:
  - Example 1: If `R200` is specified for mackerel then 200 kHz must be available for anything to categorized as mackerel. If the transducer range for 200 kHz is set to 200 m in Vertical transducer offsets then nothing will be categorized as mackerel below 200 m.
  - Example 2: `R18 && count(R70, R120, R200) >= 1` specifies that 18 kHz must be available together with at least one of 70, 120 and 200 kHz.

The features known by the system can also be enabled/disabled though the panel to the right.

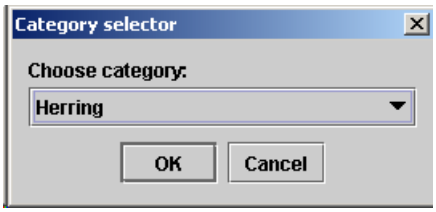
Categorization parameters



Parameters used by the categorization system can be set here.

- `OutlierFraction:` The points that lie in the percentile given by 1 minus this number are considered outliers.
- `ThinnedScatterSize:` The maximum number of points in the point clouds used for visualization. The points used are drawn randomly from all existing points.

Add to category



The user can add the extracted points to a category that can be selected from a drop down list. The category that the system deems to be the best automatically appears at the top of the list, then the second best and so on. For adding points to categories, see also Misclassification matrix.

Mark

Through this you can mark points that meet some criteria. Possible criteria are:

- Classified as: marks points that will be classified as the chosen category.
- Not classified as: marks points that will NOT be classified as the chosen category.
- Outlier: marks points that are outliers relative the chosen category.
- Inlier: marks points that are inliers relative the chosen category.

Unmark

Through this you can unmark points that meet some criteria. Possible criteria are:

- Classified as: unmarks points that will be classified as the chosen category.
- Not classified as: unmarks points that will NOT be classified as the chosen category.
- Outlier: unmarks points that are outliers relative the chosen category.
- Inlier: unmarks points that are inliers relative the chosen category.

Clear marking

Clears all current markings. Shortcut key: `Escape`.

Invert marking

Inverts all current markings.

Delete marked points

Deletes the marked points. Shortcut key: `Delete`.

Delete unmarked points

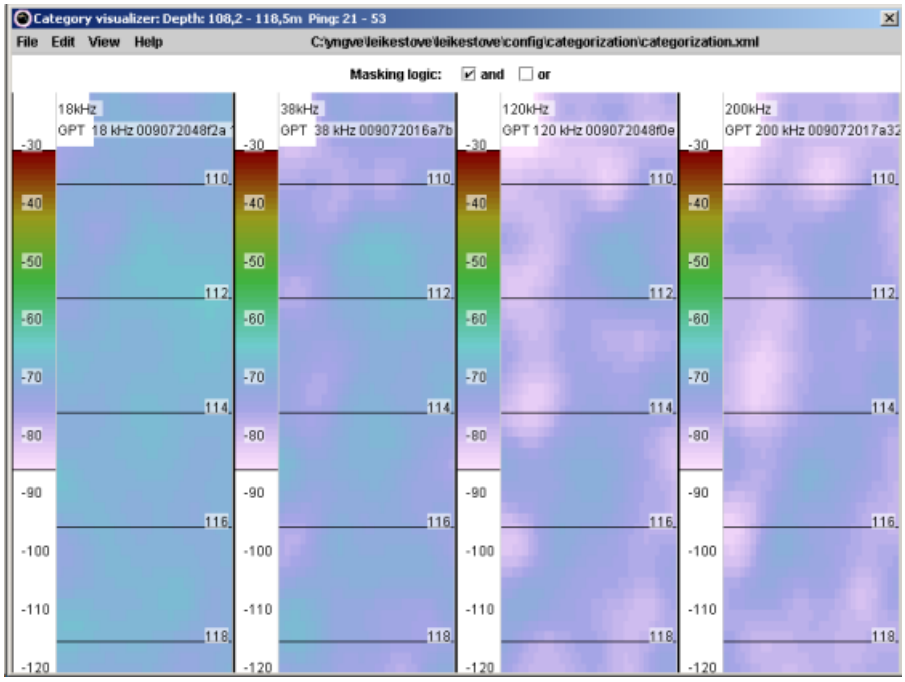
Deletes the unmarked points. Shortcut key: `Alt-Delete`.

Undo all modifications

Reads the data from file again so that all modifications before a Save are undone.

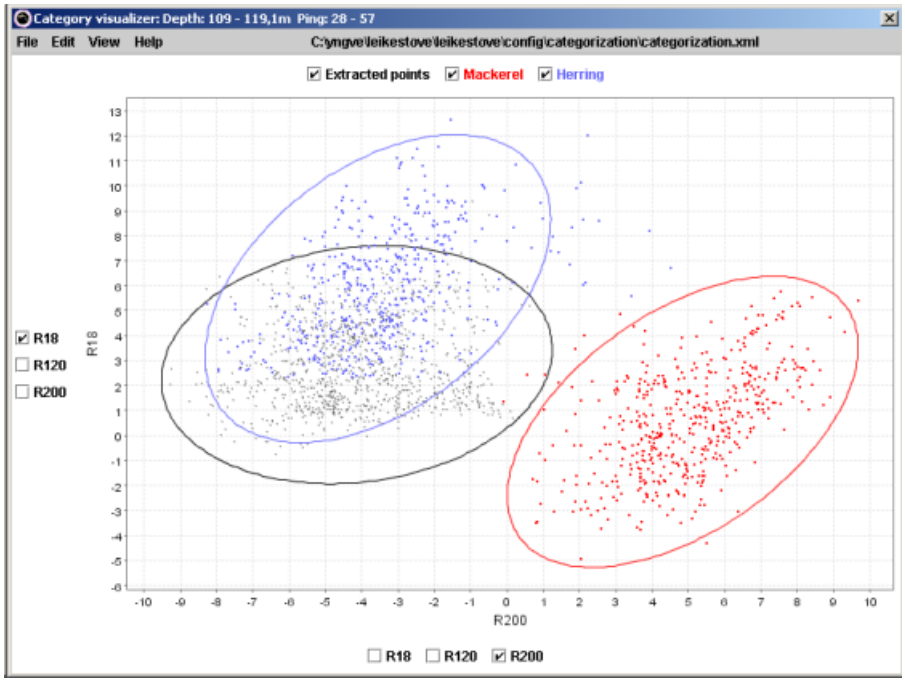
The view menu

Echogram



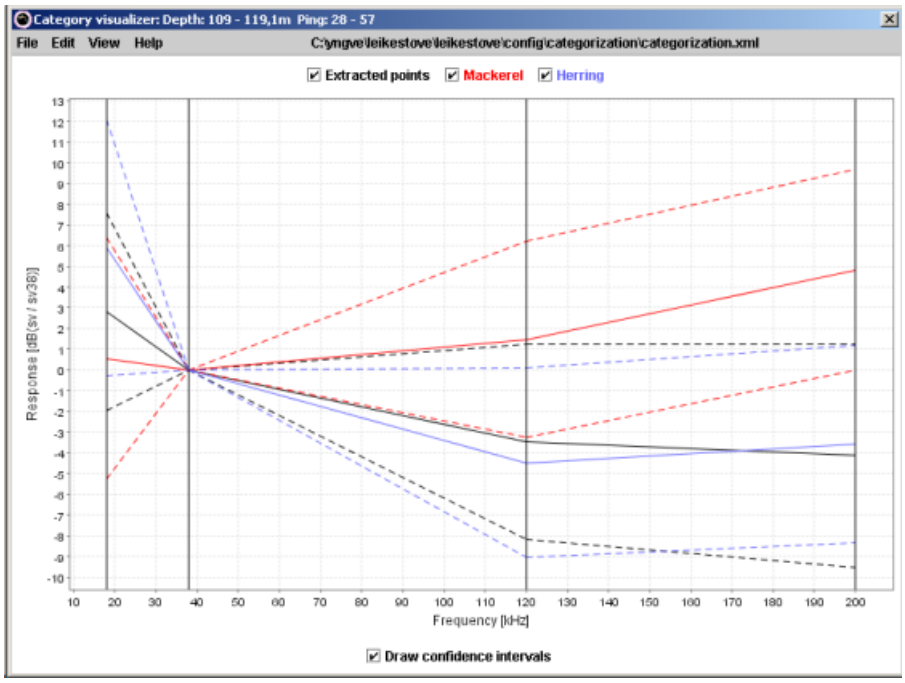
Shows the extracted rectangle with a separate colorbar for each frequency. The points will be masked/unmasked by dragging the colorbars. Points can also be masked by drawing rectangles directly in the echogram views. Holding `Ctrl` enables marking of several rectangles, and holding `Alt` unmarks the points inside the drawn rectangle.

Scatter



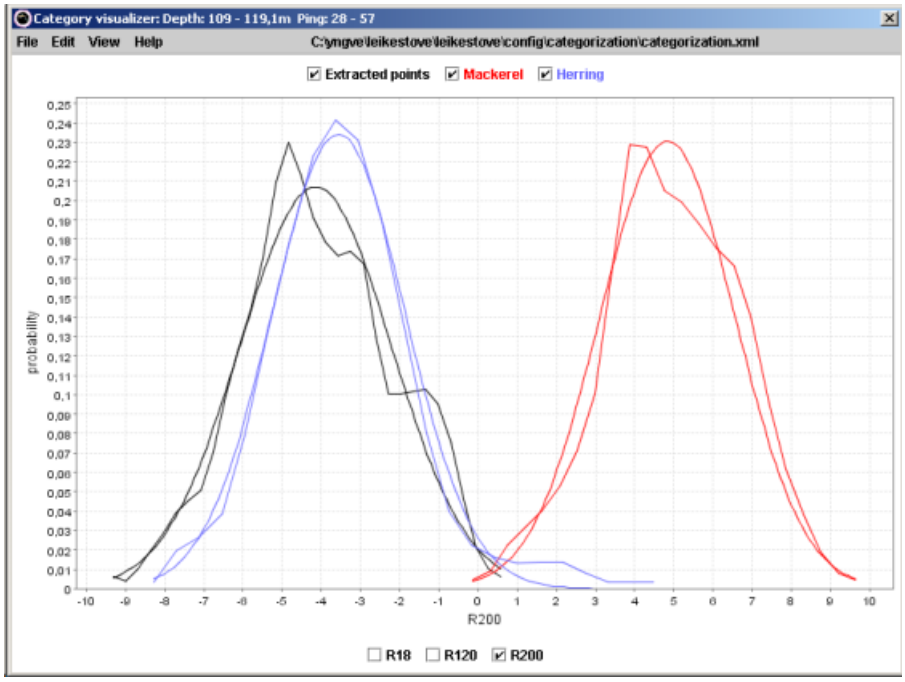
Shows a scatter plot of how the points of the categories are distributed in the feature space. A pair of feature axes can be selected by clicking the checkboxes to the left and below the plot. See also category toggles.

Frequency response



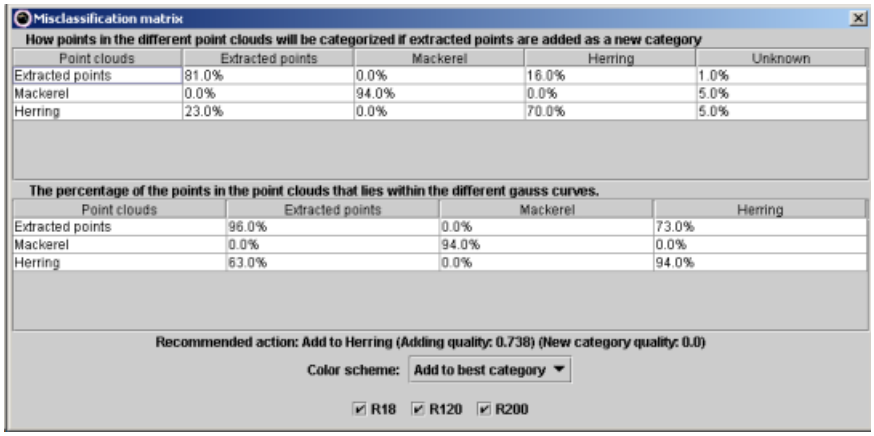
Shows the relative frequency response curves for the categories. See also category toggles.

Probabilities



Shows the probability density function for the categories for a selected feature. See also category toggles.

Misclassification matrix



The first table shows how point cloud `i`, where `i` is the row, would be categorized if the extracted points were added as a NEW category.

The second table shows the overlap between the different point clouds, i.e. the percentage of the points in category `i` that lies within the `1 - outlierFraction` contour of category `j`,where `i` is the row and `j` is the column.

This view also gives a recommendation of what can be done with the extracted points.

The user can specify different coloring schemes, depending on whether the user wishes to add the points to an existing category or create a new category. The system will then highlight the problematic values with either yellow or red color depending on the severity of the problem (yellow: somewhat severe, red: more severe).

The user can also turn on/off the enabled features (see Configuration to enable features) by clicking the toggles at the bottom of the view. The results will then be recalculated.

Rescale axes

Rescales the plot axes.

Common buttons for many views

- **Category toggles:** Positioned at the top of many views. By left clicking these toggles, the user can turn on/off the visualization of the given category. Right clicking the toggles brings up another view where it is possible to turn on/off the use of earlier saved point clouds. By right clicking the earlier point clouds, it is possible to delete them permanently.

Plankton configuration

The plankton inversion configuration file.

Used by the Plankton inversion module.

Broadband notch filters

Used by the Broadband notch filter module.

Example of xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<notchFilters>
   <filter RejectionFrequencyKHz="200" BandwidthKHz="1"/>
   <filter RejectionFrequencyKHz="170" BandwidthKHz="1" StopTime="2014-11-02T09:45:00Z"/>
   <filter RejectionFrequencyKHz="75" BandwidthKHz="2" StartTime="2014-11-02T010:00:00Z" Stop
   <filter RejectionFrequencyKHz="220" BandwidthKHz="1" StartTime="2014-11-02T11:00:00Z"/>
</notchFilters>
```

Broadband splitter bands

Used by the Broadband splitter module.

Example of xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<splitterBands>
   <band frequencyNominal="200" frequencyStart="180" frequencyStop="230"/>
   <band frequencyNominal="170" frequencyStart="150" frequencyStop="180"/>
   <band frequencyNominal="70" frequencyStart="60" frequencyStop="75"/>
</splitterBands>
```

Towfish configuration

Points to a directory with towfish data files.

Used by the Towfish module.

Calibration

Calibration of data is done if a file named "calibration.xml" exists in the same directory as the raw data files.

Simple example

```
<?xml version="1.0" encoding="UTF-8"?>
<calibration>
   <default g="26.5" SA="64:-0.49, 128:-0.71, 256:-0.58, 512:-0.77,  1024:-0.66, 2048:-0.42, 
   <case channel="1" kHz="18" g="22.4" SA="512:-0.59, 1024:-0.65, 2048:-0.45, 4096:-0.4,  819
   <case channel="2" kHz="38" g="26.5" SA="256:-0.58, 512:-0.77,  1024:-0.66, 2048:-0.42, 409
   <case channel="3" kHz="70" g="27.0" SA="128:-0.88, 256:-0.82,  512:-0.53,  1024:-0.39, 204
   <case channel="4" kHz="120" g="27.0" SA="64:-0.49, 128:-0.71,  256:-0.66,  512:-0.41,  102
   <case channel="5" kHz="200" g="27.0" SA="64:-0.66, 128:-0.76,  256:-0.38,  512:-0.32,  102
   <case channel="6" kHz="333" g="27.0" SA="64:-0.50, 128:-0.70,  256:-0.35,  512:-0.30,  102
</calibration>
```

If the default element is present, then it will apply to all channels without a separate case element. If a separate case element is specified for a channel then none of the information from the default element is used.

At least one of the attributes 'id' and 'channel' must be specified. All other attributes are optional.

If an actual transducer configuration in a data file for a specified 'id' or 'channel' does not match all of the values specified for 'id', 'channel' and 'kHz' then a warning is given and the calibration is not applied.

Attribute	Description
id	The channel id
channel	The channel number
kHz	The frequency in kHz
g	New value for gain
SA	New values of the sa correction table given as a comma separated list mapping from pulse length (in microseconds) to sa correction
abs	Absorption coefficient
sound	Sound velocity
equivalentBeamAngle	Equivalent beam angle
beamWidthAlong	Beam width alongship
beamWidthAthwart	Beam width athwartship
angleOffsetAlong	Angle offset alongship
angleOffsetAthwart	Angle offset athwartship

Example with multiple time intervals

```
<?xml version="1.0" encoding="UTF-8"?>
<calibration>
   <calibration begin="2010-01-01T12:00:00Z" end="2010-02-01T12:00:00Z">
      <case channel="1" kHz="18" g="22.4" SA="512:-0.59, 1024:-0.65, 2048:-0.45, 4096:-0.4,  
      <case channel="2" kHz="38" g="26.5" SA="256:-0.58, 512:-0.77,  1024:-0.66, 2048:-0.42, 
   </calibration>
   <calibration begin="2010-03-01T12:00:00Z" end="2010-04-01T12:00:00Z">
      <case channel="1" kHz="18" g="22.4" SA="512:-0.59, 1024:-0.65, 2048:-0.45, 4096:-0.4,  
      <case channel="2" kHz="38" g="26.5" SA="256:-0.58, 512:-0.77,  1024:-0.66, 2048:-0.42, 
   </calibration>
</calibration>
```

The calibration element can have multiple nested calibration elements specifying begin and end times. The times are specified in UTC using the format `YYYY-MM-DDThh:mm:ssZ` in accordance with ISO 8601.

If the begin time is omitted it is taken to mean since the beginning of time.
If the end time is omitted it is taken to mean to the end of time.

NB! Note that the calibration element matching the time of the CON0 datagram will be used for the entire data file. This means that even if the time of a ping matches a different calibration element, then the calibration element matching the time of CON0 datagram will be used.

NB! Note that if the time matches the interval of a nested calibration element, then *only* that element is used. Specifically, top level default and case elements and not merged with those of a nested element. In the example below, only channel 2 will be modified between 2010-01-01T12:00:00Z and 2010-02-01T12:00:00Z, as the top level default and case elements do not apply.

```
<?xml version="1.0" encoding="UTF-8"?>
<calibration>
   <default g="26.5" SA="64:-0.49, 128:-0.71, 256:-0.58, 512:-0.77,  1024:-0.66, 2048:-0.42, 
   <case channel="1" kHz="18" g="22.4" SA="512:-0.59, 1024:-0.65, 2048:-0.45, 4096:-0.4,  819
   <calibration begin="2010-01-01T12:00:00Z" end="2010-02-01T12:00:00Z">
      <case channel="2" kHz="38" g="26.5" SA="256:-0.58, 512:-0.77,  1024:-0.66, 2048:-0.42, 
   </calibration>
</calibration>
```

In fact, the above example is equivalent to

```
<?xml version="1.0" encoding="UTF-8"?>
<calibration>
   <calibration>
      <default g="26.5" SA="64:-0.49, 128:-0.71, 256:-0.58, 512:-0.77,  1024:-0.66, 2048:-0.4
      <case channel="1" kHz="18" g="22.4" SA="512:-0.59, 1024:-0.65, 2048:-0.45, 4096:-0.4,  
   </calibration>
   <calibration begin="2010-01-01T12:00:00Z" end="2010-02-01T12:00:00Z">
      <case channel="2" kHz="38" g="26.5" SA="256:-0.58, 512:-0.77,  1024:-0.66, 2048:-0.42, 
   </calibration>
</calibration>
```

Broadband calibration

LSSS allows for calibration of a number of parameters used in different broadband calculations by adding appropriate xml elements to a file "calibration.xml" to be placed in the same directory as the raw data files.

Values given in the file are interpolated linearly between frequencies. Outside the frequency range covered in the calibration, constant extrapolation applies. If the calibration file does not exist, or does not contain calibration information for all parameters, the values are estimated using formulae based on nominal parameter values.

Parameter	XML element name	Affects	XML example

Gain	g	TS, Sv	`<g>`
			`  <case hz="180000" g="22.04"/>`
			`  <case hz="200000" g="22.5"/>`
			`  <case hz="220000" g="22.9"/>`
			`</g>`

Equivalent beam angle	eba	Sv	`<eba>`
			`  <case hz="180000" eba="-20.08"/>`
			`  <case hz="200000" eba="-21.00"/>`
			`  <case hz="220000" eba="-21.83"/>`
			`</eba>`

Beam width, alongship	albw	TS	`<albw>`
			`  <case hz="180000" albw="7.78"/>`
			`  <case hz="200000" albw="7.00"/>`
			`  <case hz="220000" albw="6.36"/>`
			`</albw>`

Beam width, athwartship	atbw	TS	`<atbw>`
			`  <case hz="180000" atbw="7.78"/>`
			`  <case hz="200000" atbw="7.00"/>`
			`  <case hz="220000" atbw="6.36"/>`
			`</atbw>`

Angle offset, alongship	alao	TS	`<alao>`
			`  <case hz="180000" alao="0.55"/>`
			`  <case hz="200000" alao="0.70"/>`
			`  <case hz="220000" alao="0.81"/>`
			`</alao>`

Angle offset, athwartship	atao	TS	`<atao>`
			`  <case hz="180000" atao="1.01"/>`
			`  <case hz="200000" atao="1.10"/>`
			`  <case hz="220000" atao="1.22"/>`
			`</atao>`

Complete example

```
<?xml version="1.0" encoding="UTF-8"?>
<calibration>
   <case channel="2">
      <broadband>
         <g>
            <case hz="180000" g="22.04"/>
            <case hz="200000" g="22.5"/>
            <case hz="220000" g="22.9"/>
         </g>
         <eba>
            <case hz="180000" eba="-20.08"/>
            <case hz="200000" eba="-21.00"/>
            <case hz="220000" eba="-21.83"/>
         </eba>
         <albw>
            <case hz="180000" albw="7.78"/>
            <case hz="200000" albw="7.00"/>
            <case hz="220000" albw="6.36"/>
         </albw>
         <atbw>
            <case hz="180000" atbw="7.78"/>
            <case hz="200000" atbw="7.00"/>
            <case hz="220000" atbw="6.36"/>
         </atbw>
         <alao>
            <case hz="180000" alao="0.55"/>
            <case hz="200000" alao="0.70"/>
            <case hz="220000" alao="0.81"/>
         </alao>
         <atao>
            <case hz="180000" atao="1.01"/>
            <case hz="200000" atao="1.10"/>
            <case hz="220000" atao="1.22"/>
         </atao>
      </broadband>
   </case>
</calibration>
```

Correction formulae

The here given formulae are used to estimate the value of some frequency dependent parameters of broadband data, if no broadband calibration is specified. The values of the parameter at a given frequency f are estimated from the parameter value at the nominal frequency (denoted by a the subscript "nom").

Gain

$g(f) = g_{nom} + 10\ log_{10}(f / f_{nom})$

Equivalent beam angle

$\psi(f) = \&psi_{nom} + 20\ log_{10}(f_{nom} / f)$

Beam width (either axis)

$bw_{axis} = 2\ arcsin\left(sin(bw_{axis,nom} / 2) \cdot (f_{nom} / f) \right)$

Angle offset (either axis)

$ao_{axis}(f) = ao_{axis,nom}$

Categorization

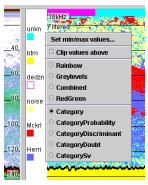
Categorization is done by CategorizationModule, which is typically used in the following setup:

Bottom detection module ➡ Noise quantification module ➡ Noise remover module ➡ Smoother module ➡ Categorization module

Typical module configuration when doing categorization.

The categorization library used is specified in the config file settings.

Viewing categorization results



Make sure categorization datagrams, Cac0Datagram and Cad0Datagram, are available and start playback. Right-click in one of the echogram panels and select 'Use private colormap'. Right-click in the separate colormap panel to access a colormap menu with the following extensions:

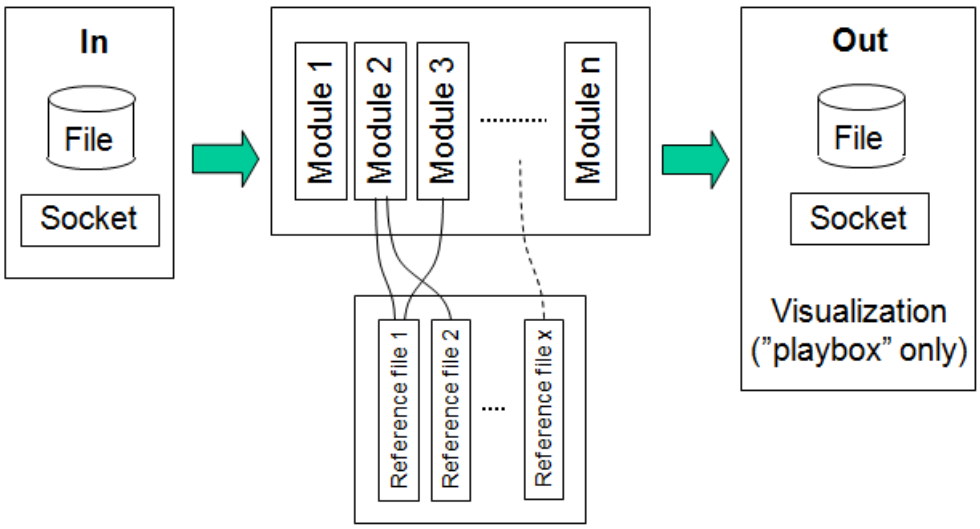
- *Category:* Each pixel is colored according to its category.
- *CategoryProbability:* Each pixel is colored according to its category with light intensity proportional to category conditional probability.
- *CategoryDiscriminant:* Each pixel is colored according to its category with light intensity proportional to discriminant value.
- *CategoryDoubt:* Each pixel is colored according to its category with light intensity proportional to the difference in discriminant value of the two best categories.
- *CategorySv:* Each pixel is colored according to its category with light intensity proportional to signal strength, S_V.

The following actions are possible:

- Left-click on a category's square in the colormap panel to toggle on and off plotting of that category. When a category is turned off, the next best category is plotted instead.
- For category colorings that modify the light intensity (the four last choices): Drag the two markers in the colormap panel.

Examples using Korona

This section describes possible use cases for preprocessing in Korona. The data flows through the KORONA modules in a sequence described in the module setup, and some of the processing modules use the content of a reference file as illustrated in the figure below. Several modules may use the same reference file. *Note that the sequence of the processing modules may affect the result.* For example, noise removal (NoiseRemovalModule) need to be done after noise quantification (NoiseQuantificationModule). Further, the best data used to quantify noise is picked somewhat "below" the detected bottom (really: some time after bottom echo), so the Bottom detection module should be used before the NoiseQuantificationModule.



The processing modules can be grouped into 6 classes:

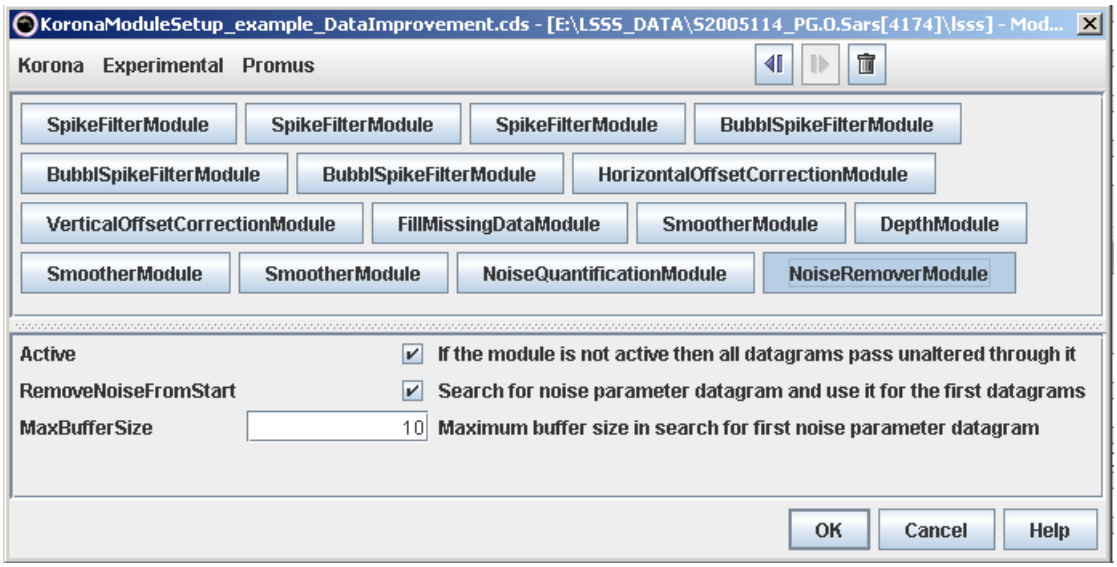
1. Data improvement modules. These manipulate data, but the data-format and file-size remain unchanged.
2. Data reduction modules. These remove or compress the data, but do not add information in form of new data-telegrams.
3. Echo-trace detection modules. These detect the extent of e.g. a school or a single-fish track, and calculate parameters that characterize the echo-trace and pass this information as new data-telegrams in the data in addition to those defined by the data-file format *.raw.
4. Identification of acoustic scattering classes by means of an acoustic feature library or by scattering models. The information is passed as new data-telegrams in the data.
5. Generation and removal of data-channels trough mathematical operators.
6. Modules for visualizing intermediate results, e.g. echogram between any two processing modules.

Examples

Example 1 - Data improvement.
Example 2 - Data reduction.
Example 3 - School detection.
Example 4 - Echo-trace detection.
Example 5 - Categorisation.
Example 6 - School categorisation.
Example 7 - Zooplankton size estimation.
Example 8 - Visualisation of intermediate results.
Example 9 - Colour scales.
Example 10 - Bottom detection.

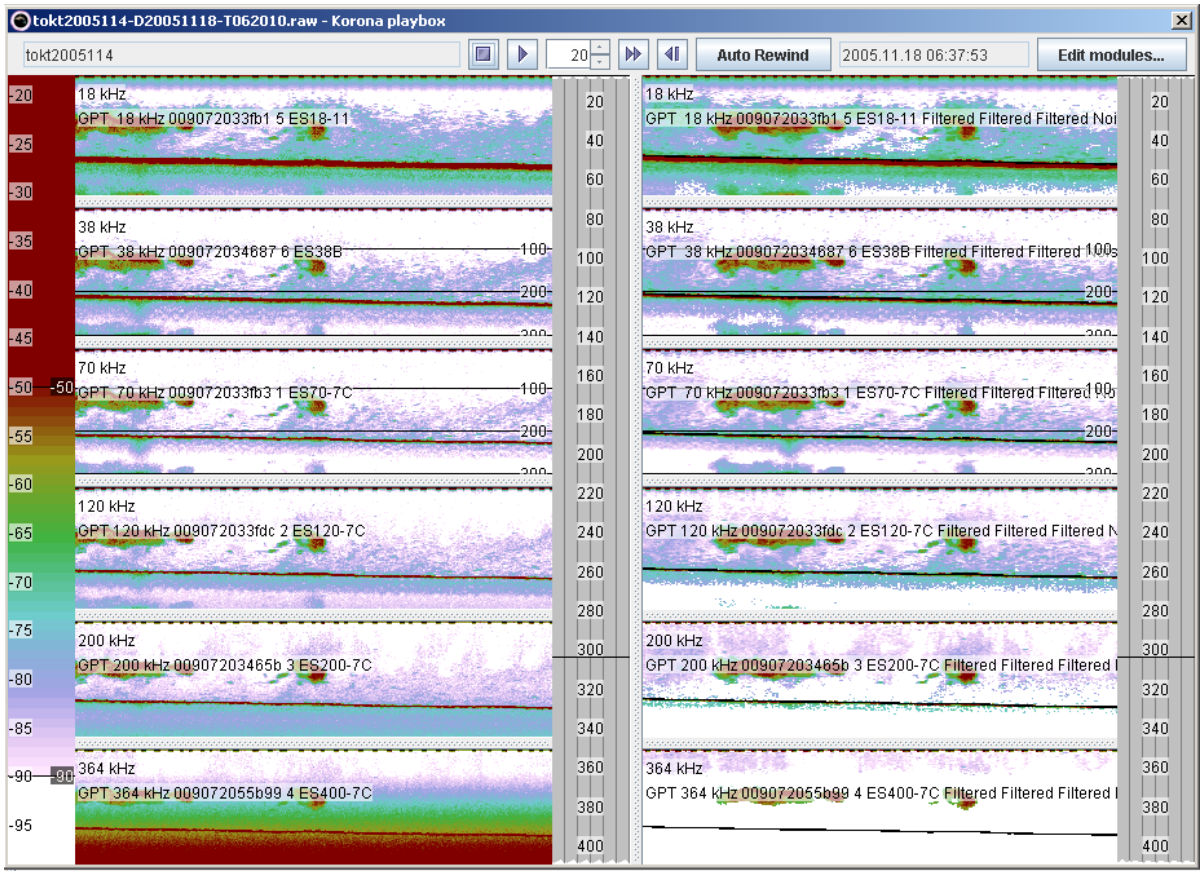
Example 1 - Data improvement through noise removal and smoothing

The module setup below shows one way to improve data. The data flows trough each of the modules starting with the one in the upper left corner and ending with the one in the lower right.



- There are three spike filters used to remove very strong values with a vertical extent, and replace those values with the median value of the surrounding measurements. The spike filters are set up to work at different depths: 10-100m, 100-250m, 250-2000m, i.e. for each depth there is only one spike-filter used. The reason why there are three different is the shape of the beam, so that spikes at shallow depths appear to be stronger.
- The three bubble-spike filters work in a similar way as the spike-filters, except that vertical regions with significantly lower values than the surroundings are replaced by the median of the surrounding values. The filters are called bubble-spike-filters because such regions often appear when there are bubbles under the ship.
- The HorizontalOffsetCorrectionModule adjusts the data at different acoustic frequencies for horizontal positioning of the transducers. The transducer position is read from a reference-file specified by HorizontalOffsetCorrectionModule
- The VerticalOffsetCorrectionModule corrects the data at different frequencies for the frequency-, and filter-dependent pulse-transmission delay. The transmission delays (i.e. the frequency-dependent vertical correction of the measurements) are read from a reference-file specified by VerticalOffsetCorrectionModule
- The FillMissingDataModule duplicate data at one frequency if data at a given time does not exist for that frequency, but do exist for other frequencies.
- The first smoother module smooths only in horizontal direction, so that the bottom-detection done by the Bottom detection module is better if the backscatter from the bottom varies much. The two next SmootherModule smooths data above and below the detected bottom. These smoothings are done separately to avoid losing fish that is close to the bottom. The smoothing is done below bottom because those data are used to quantify noise in the NoiseQuantificationModule, and those results are used by the NoiseQuantificationModule to remove noise.

Results are shown below: left panel is before data improvement for all 6 frequencies, right panel is after.

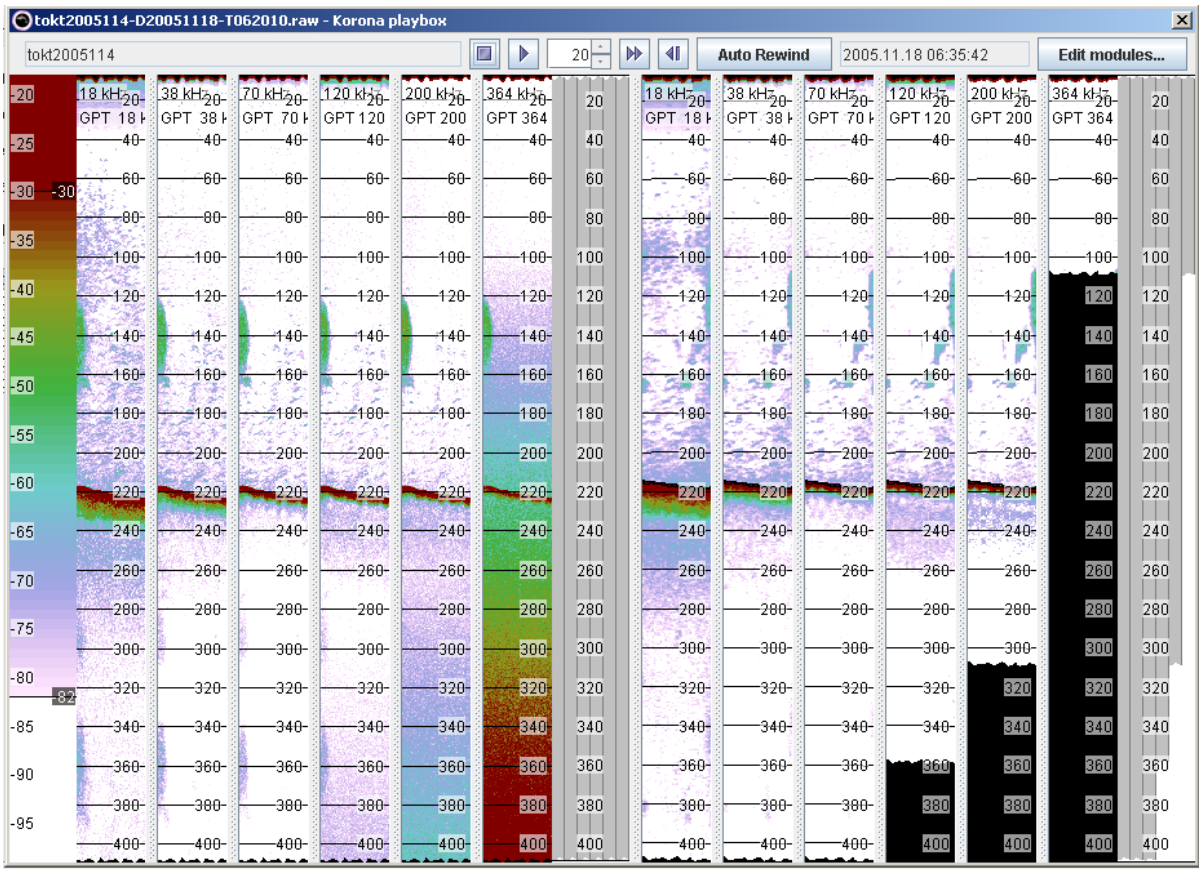

tokt2005114-D20051118-T062010.raw - Korona playbox
tokt2005114
20
Auto Rewind
2005.11.18 06:37:53
Edit modules...
18 kHz
GPT 18 kHz 009072033fb1 5 ES18-11
38 kHz
GPT 38 kHz 009072034687 6 ES38B
70 kHz
GPT 70 kHz 009072033fb3 1 ES70-7C
120 kHz
GPT 120 kHz 009072033fdc 2 ES120-7C
200 kHz
GPT 200 kHz 00907203465b 3 ES200-7C
364 kHz
GPT 364 kHz 009072055b99 4 ES400-7C
18 kHz
GPT 18 kHz 009072033fb1 5 ES18-11 Filtered Filtered Filtered Noi
38 kHz
GPT 38 kHz 009072034687 6 ES38B Filtered Filtered Filtered
70 kHz
GPT 70 kHz 009072033fb3 1 ES70-7C Filtered Filtered Filtered
120 kHz
GPT 120 kHz 009072033fdc 2 ES120-7C Filtered Filtered Filtered
200 kHz
GPT 200 kHz 00907203465b 3 ES200-7C Filtered Filtered Filtered
364 kHz
GPT 364 kHz 009072055b99 4 ES400-7C Filtered Filtered Filtered

Example 2 - Data reduction

The processing setup is similar to Example 1, except that the **DataReductionModule** is inserted between the NoiseQuantificationModule and the NoiseRemovalModule. Note that the final result would be the same if the DataReductionModule was placed after the NoiseRemovalModule, but placing if before makes the processing go faster since time is not wasted in removing noise from data that is later removed.

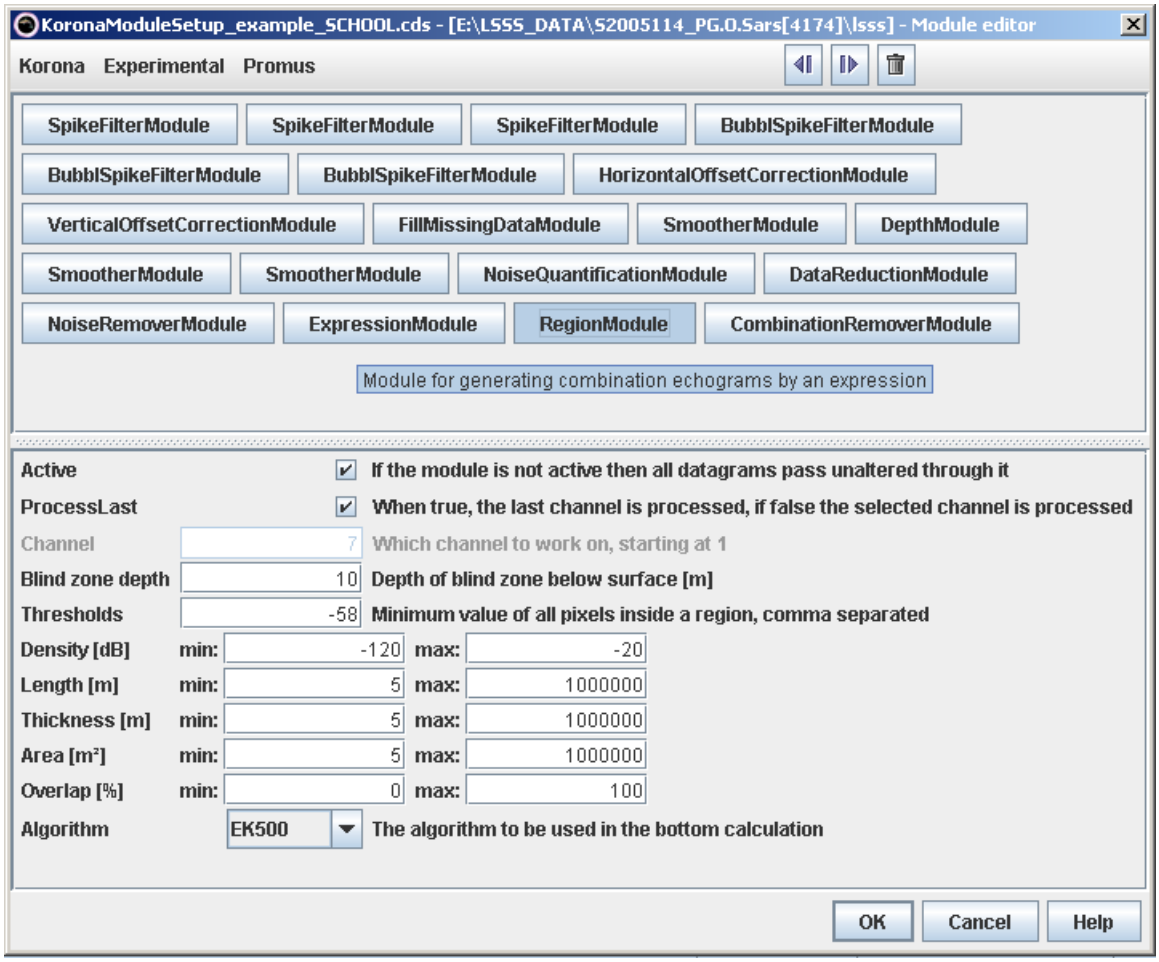


Results are shown below: left panel is original data for all 6 frequencies, right panel is after data improvement and data reduction. Black colour means that there is no data.

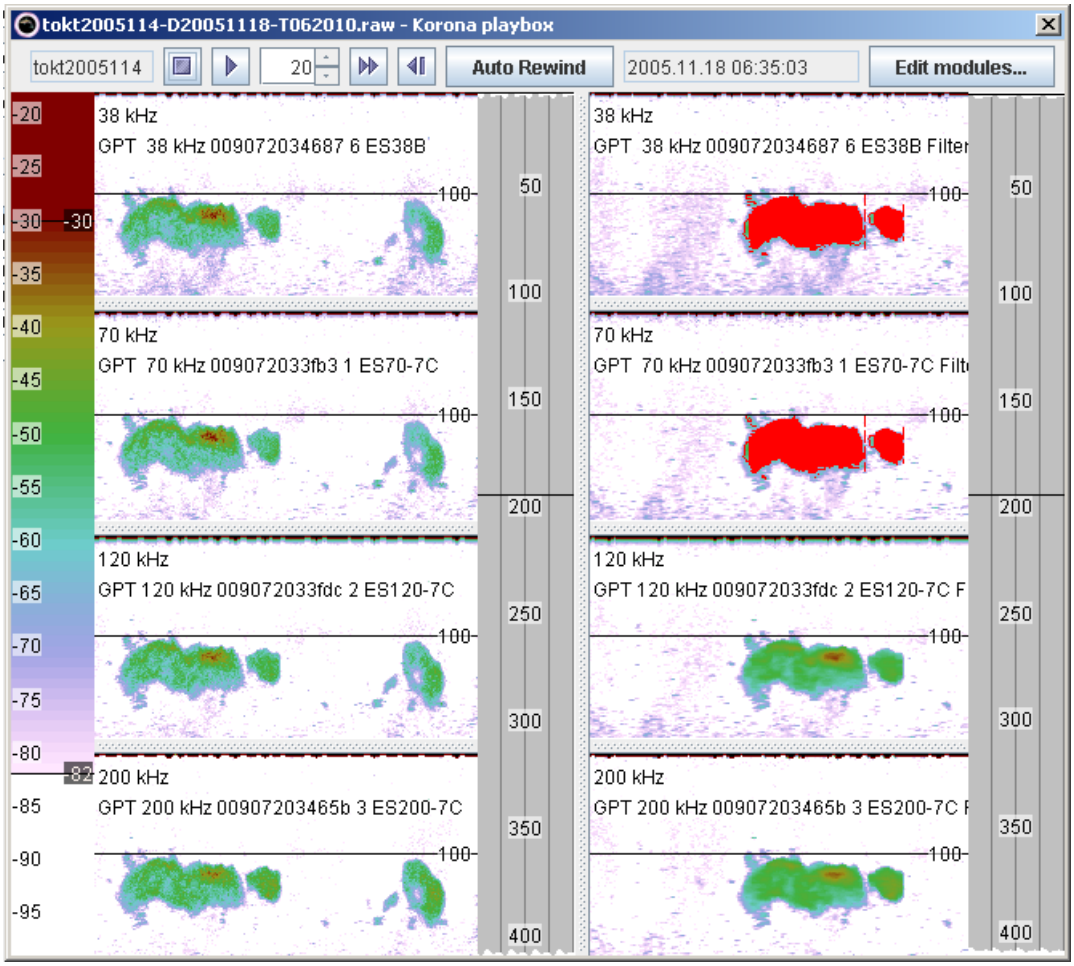


Example 3 - School detection

School detection is done by means of the School detection module, and is typically done on a combined-frequency channel after data improvement (see modules before School detection module, and Example 1) as shown in the figure below. Further, there is no need to keep the synthetic data-channel when the school is detected, so in the example that channel is removed. The School detection module add a new data-telegram. The results of the school-detection is visualised in KORONA by setting "Show regions", and in LSSS by checking the "Preprocessed regions" from the "Overlay menu". The "Overlay menu" is started e.g. by clicking the right-mouse-button in the pelagic echogram and select "Configure pelagic echogram"



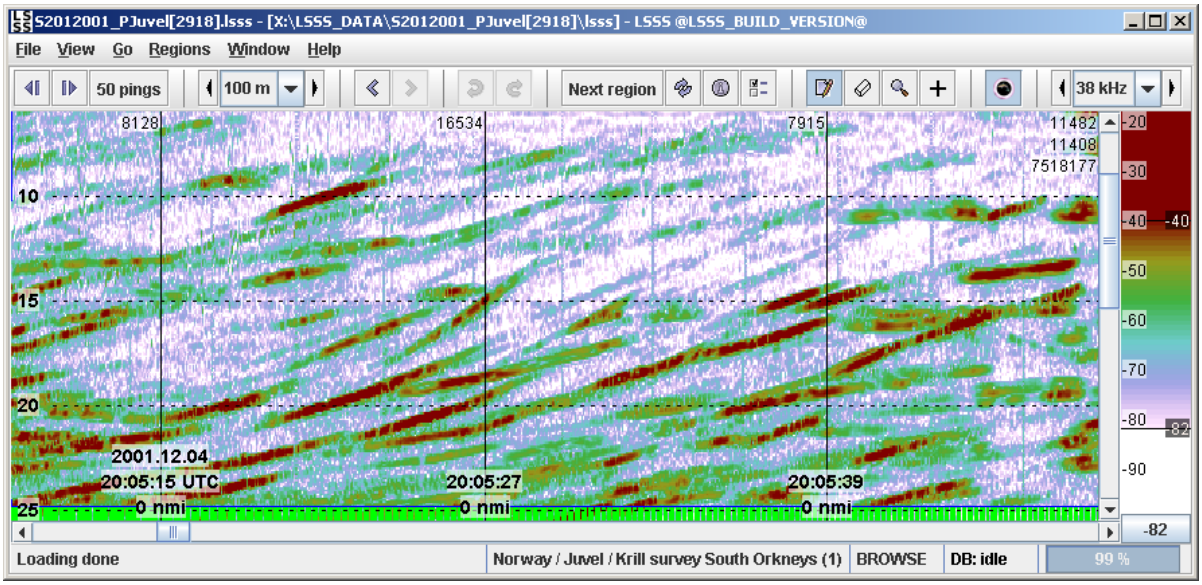
The figure below show detected schools. The left panel shows the 4 frequencies 38, 70, 120 and 200 kHz, and the right panel shows the same frequencies preprocessed, and with the detected schools shown in the 18 and 38 kHz echogram.


tokt2005114-D20051118-T062010.raw - Korona playbox
tokt2005114
20
Auto Rewind
2005.11.18 06:35:03
Edit modules...
38 kHz
GPT 38 kHz 009072034687 6 ES38B
38 kHz
GPT 38 kHz 009072034687 6 ES38B Filter
70 kHz
GPT 70 kHz 009072033fb3 1 ES70-7C
70 kHz
GPT 70 kHz 009072033fb3 1 ES70-7C Filt
120 kHz
GPT 120 kHz 009072033fdc 2 ES120-7C
120 kHz
GPT 120 kHz 009072033fdc 2 ES120-7C F
200 kHz
GPT 200 kHz 00907203465b 3 ES200-7C
200 kHz
GPT 200 kHz 00907203465b 3 ES200-7C F
-20
-25
-30
-35
-40
-45
-50
-55
-60
-65
-70
-75
-80
-85
-90
-95
50
100
150
200
250
300
350
400

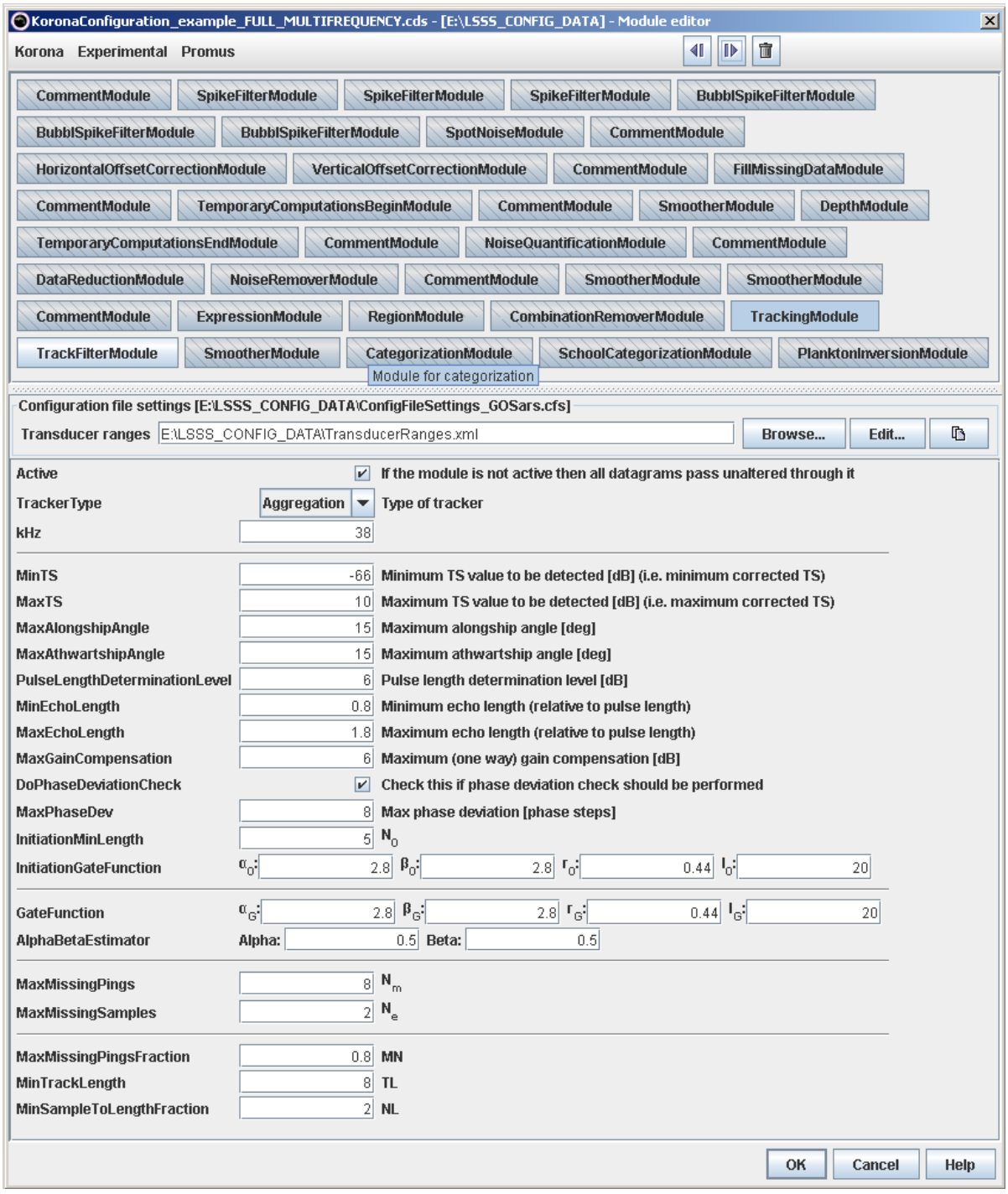
Example 4 - Echo trace detection

The tracks will be marked. The acoustic tracking module is based on the publication: *Nils Olav Handegard, Ruben Patel and Vidar Hjellvik (2005). "Tracking individual fish from a moving platform using a split-beam transducer", J. Acoust. Soc. Am., Vol 118(4), pp 2210-2223.*

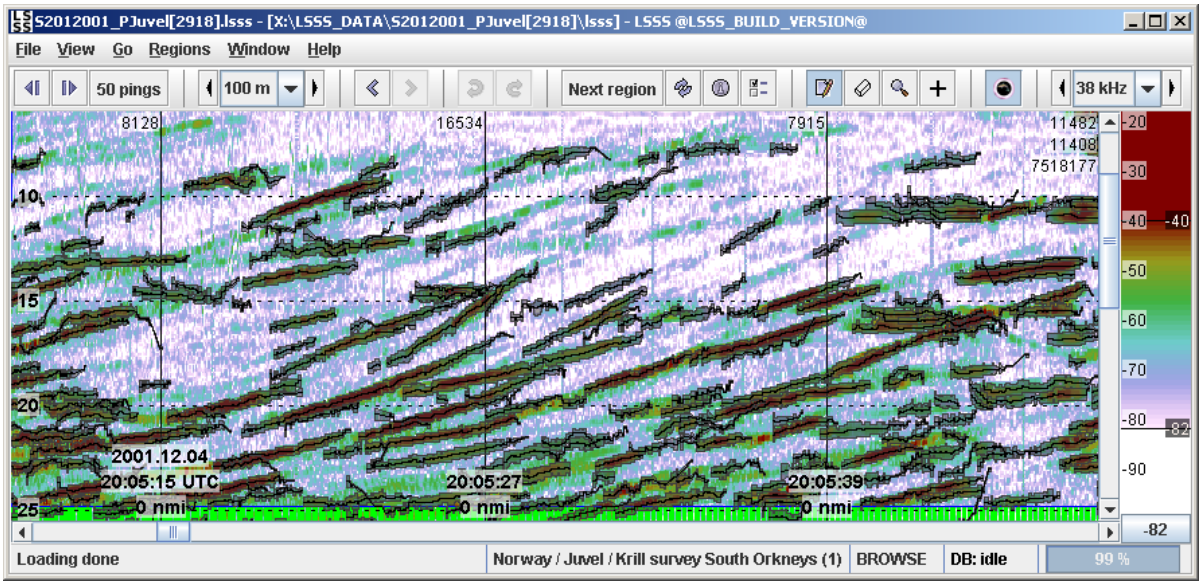
The echogram below shows the 38 kHz data to be tracked.



The figure below shows the module setup, e.g. the limitations put onto the tracking algorithm.

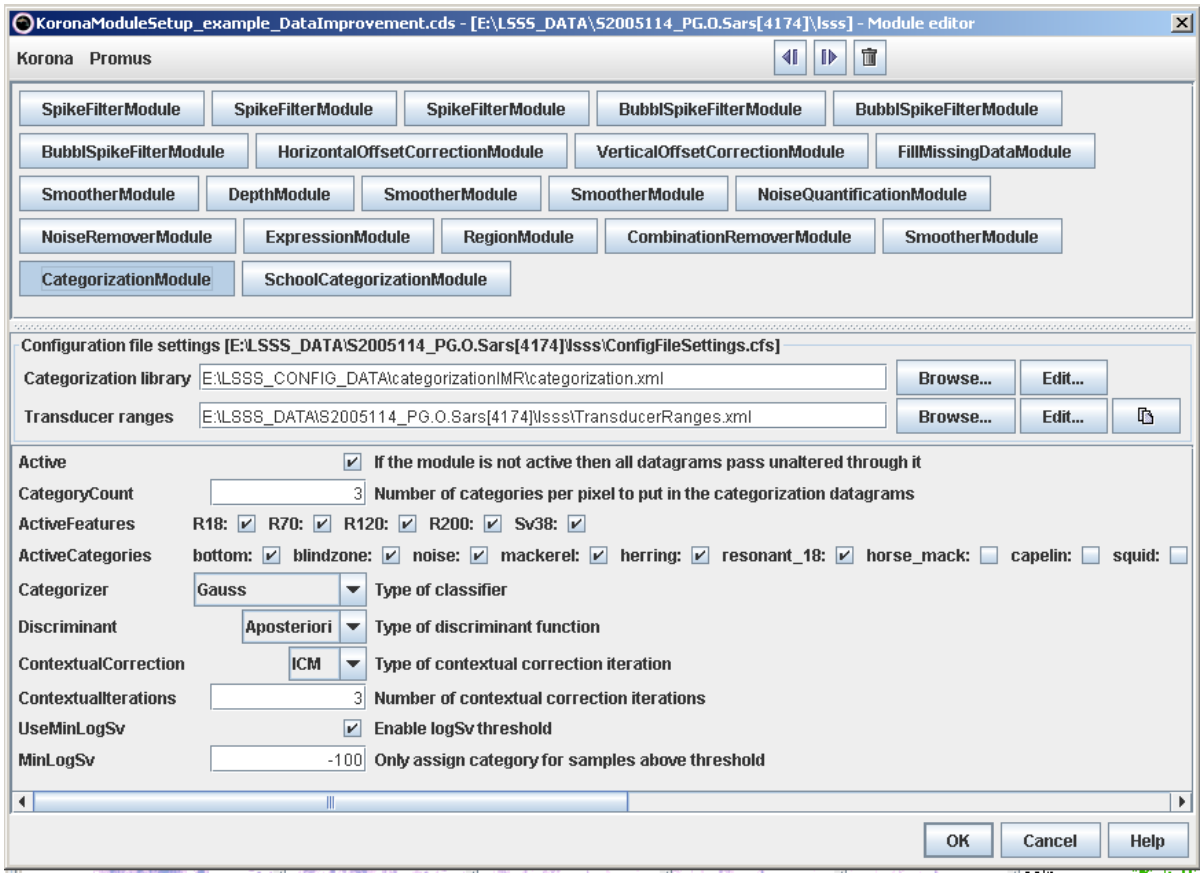


The echogram below shows the result of the tracking (marked in grey).

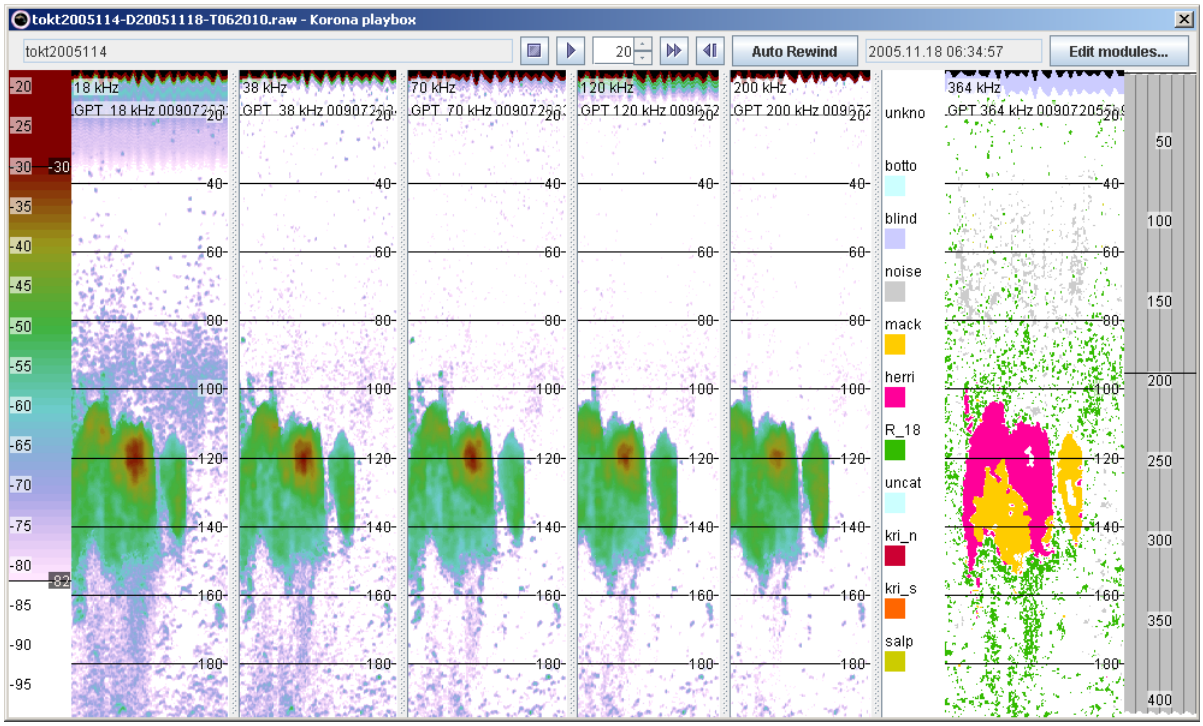

S2012001_PJuvel[2918].lsss - [X:\LSSS_DATA\S2012001_PJuvel[2918]\lsss] - LSSS @LSSS_BUILD_VERSION@
File View Go Regions Window Help
50 pings
100 m
Next region
38 kHz
8128
16534
7915
11482
11408
7518177
10
15
20
25
2001.12.04
20:05:15 UTC
20:05:27
20:05:39
0 nmi
0 nmi
0 nmi
-20
-30
-40
-40
-50
-60
-70
-80
-82
-90
-82
Loading done
Norway / Juvel / Krill survey South Orkneys (1)
BROWSE
DB: idle
99 %

Example 5 - Categorisation (pixel categorisation)

The acoustic categories may be a single species (e.g. Atlantic mackerel), a group of species (e.g. swimbladdered fish) and sometimes even a subset of a species (different year-classes of sandeel). The figure below shows an example of a module setup used to identify acoustic categories. The CategorisationModule is placed after the data improvement modules, and is showed in a somewhat darker grey colour than the other modules.



The figure below shows the result of the acoustic categorisation. The CategorisationModule adds a new datagram, and the results are visualised in KORONA by setting the colour-scale (Use private colourmap) "Categories".



References

- Korneliussen, R. J., and Ona, E. 2002. An operational system for processing and visualizing multi-frequency acoustic data. – ICES Journal of Marine Science, 159: 293–313.
- Korneliussen, R. J., and Ona, E. 2003. Synthetic echograms generated from the relative frequency response. – ICES Journal of Marine Science, 60: 636–640.
- Korneliussen, R. J., Heggelund, Y., Eliassen, I. K., and Johansen, G. O. 2009. Acoustic species identification of schooling fish. – ICES Journal of Marine Science, 66: 1111–1118.
- Pedersen, G., and Korneliussen, R. J. 2009. The relative frequency response derived from individually separated targets of northeast Arctic cod (Gadus morhua), saithe (Pollachius virens), and Norway pout (Trisopterus esmarkii). – ICES Journal of Marine Science, 66: 1149–1154.
- Korneliussen, R. J. 2010. The acoustic identification of Atlantic mackerel. – ICES Journal of Marine Science, 67: 1749–1758.

Example 6 - School categorisation

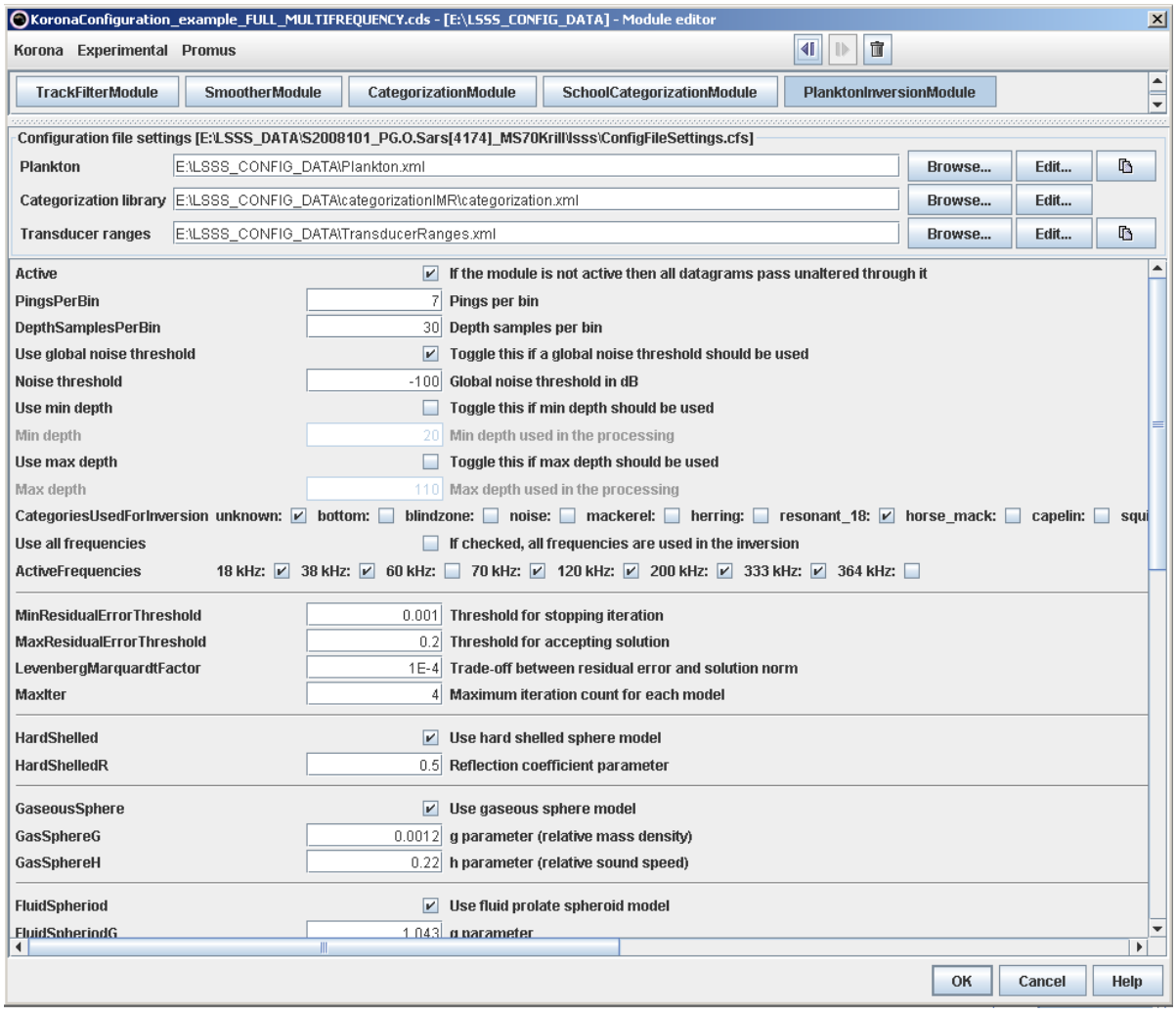
The identification of acoustic categories is improved if the variance is reduced, e.g. by assuming that a school contains one species only. The school categorisation module splits schools into segments, and compare those to investigate if the school is likely to contain one or several species. The **SchoolCategorisationModule** has to be used after the CategorisationModule as shown in the first figure of Example 5. The results are similar, except that whole schools receive the same acoustic category.

References

- Korneliussen, R. J., Heggelund, Y., Eliassen, I. K., and Johansen, G. O. 2009. Acoustic species identification of schooling fish. – ICES Journal of Marine Science, 66: 1111–1118.

Example 7 - Zooplankton size estimation

Acoustic scattering models are used to calculate size of different zooplankton groups. This is done by means of data improvement (optional) followed by acoustic categorisation (Example 5) and inversion by means of the PlanktonInversionModule. The figure below show which of the acoustic categories output from the CategorisationModule that is used in this example. The results of the inversion are sizes of the zooplankton specimens, and these sizes for the active region may be visualized in the Plankton window.



Example 8 - Visualisation of intermediate results

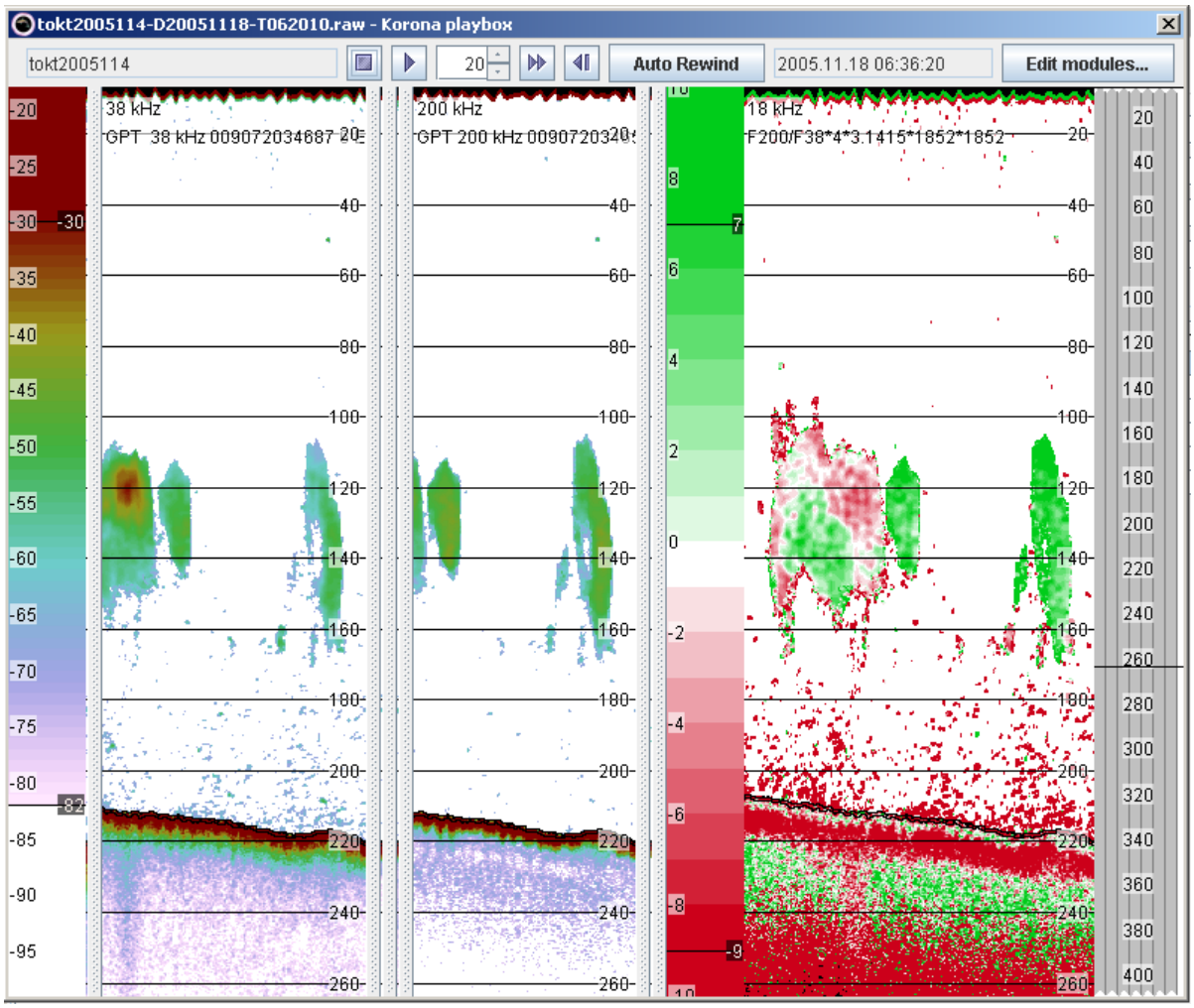
This section describes how intermediate results may be displayed in KORONA and KoronaRelay. This functionality is typically used during the creation of new KORONA module setups. One FishViewDisplayModule is inserted prior to the modules to be investigated, and another FishViewDisplayModule is placed after.



The FishViewDisplayModule is used also in Example 1, Example 2, Example 3.

Example 9 - Use of the red-green colour scale

This section describes how the red-green colour-scale may be used to compare backscatter at two frequencies. The module setup is expanded from Example 3 (school detection): A **ThresholdModule** is inserted at the end to remove all data at all frequencies weaker than -70 dB, and then an **ExpressionModule**. The ExpressionModule generates a new synthetic data-channel: F200/F38*(4*3.1415*1852*1852). F200 is the 200 kHz data-channel and F38 is the 38 kHz data-channel. The somewhat strange looking constant $4*PI*1852^2$ is because all colour-scales expect to read power-values and then convert those to s_v, so the factor is needed to make the resulting echogram symmetric around 0 dB. By applying the red-green colour-scale, and setting the limits between -10 (dB) and 10 (dB), the result below appears. The left panel is 38 kHz data, the middle is 200 kHz, and the right is the combined channel with the red-green colour-scale used. In the example below, herring (that has swimbladder) appear in red colour and Atlantic mackerel (no swimbladder) in green.

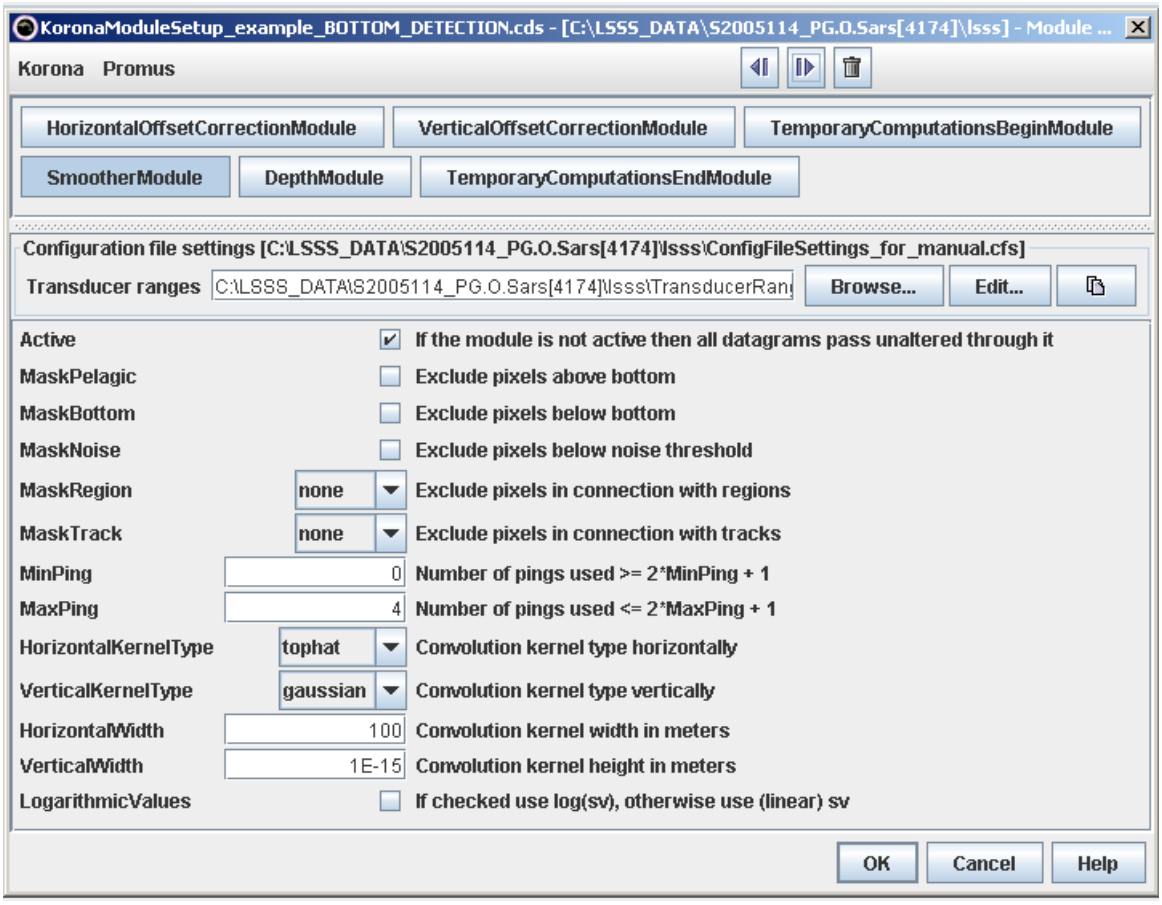


Example 10 - Bottom detection by means of temporary calculations

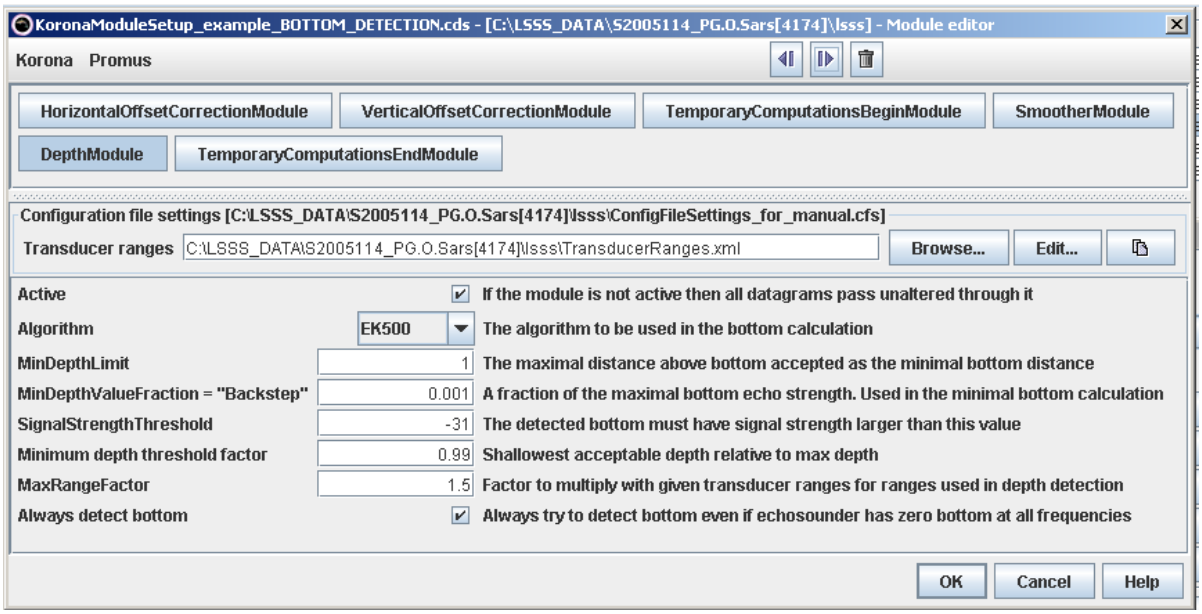
The common bottom detection in KORONA use the same algorithm as Simrad EK500 and Simrad EK60. However, the settings used by the algorithm may be different. How well a particular bottom detection setup works depends on the bottom, i.e. a setup that works well in Norwegian waters may not work as well in Icelandic waters. The use of temporary calculations may solve or at least reduce this problem. All calculations between temporary calculations are rejected, but new datagrams (e.g. bottom datagram) are kept. The following KORONA module setup usually solves the challenge of possibly using the improper settings to the bottom algorithm, and may work unless the bottom depth changes very rapidly:

- Mark start of temporary calculations
- Do heavy horizontal smoothing
- Detect bottom
- Mark stop temporary calculations

The module setup of the smoother module and bottom detection module in the KORONA module setup is shown below:

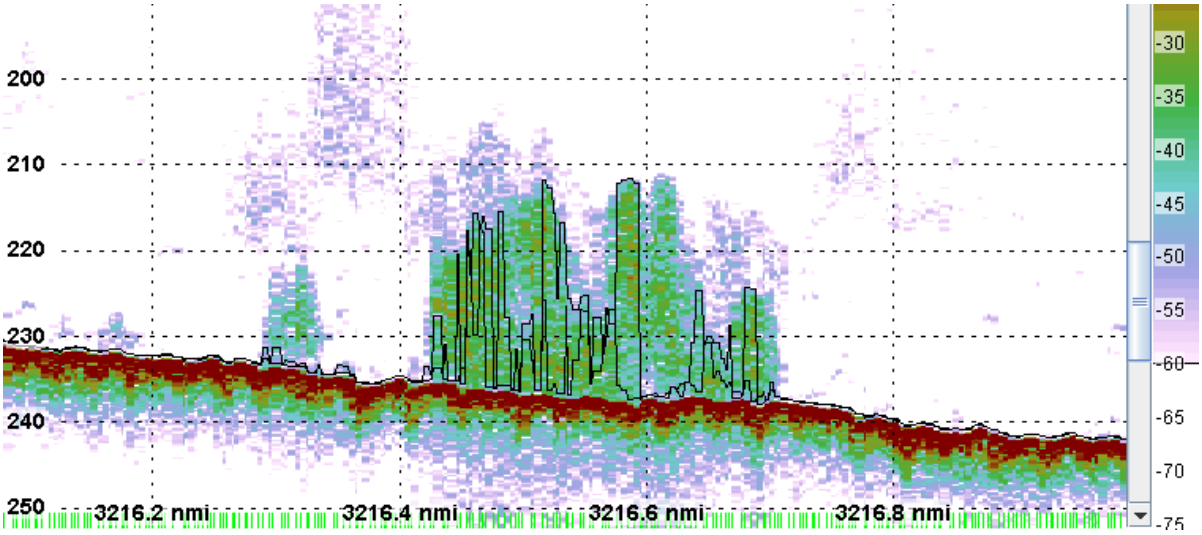


Settings in the smoother module

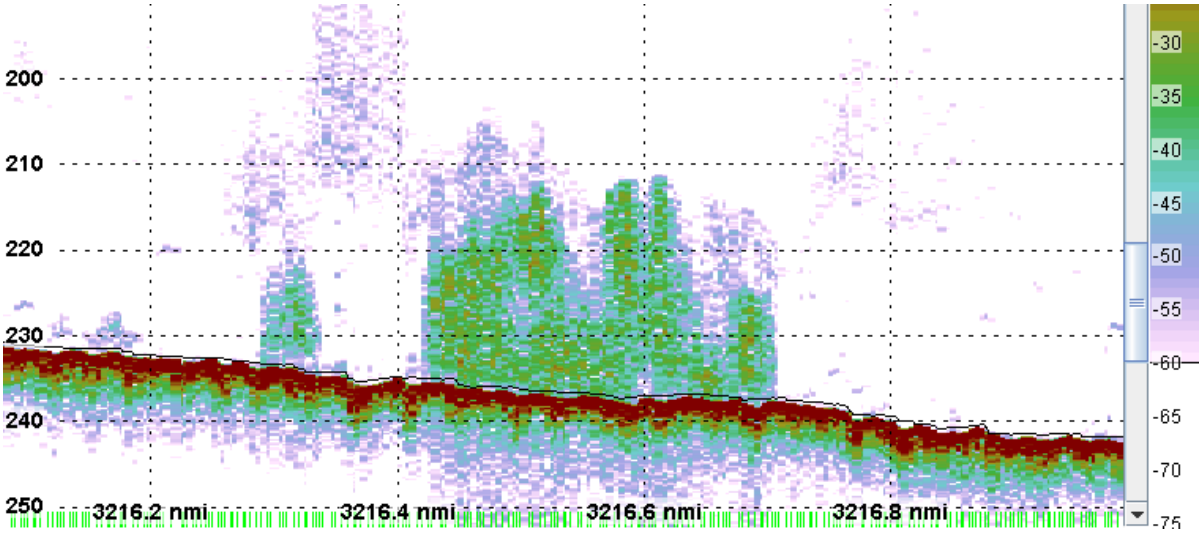


Settings in the bottom detection module

The figures below shows bottom detection from EK60 (38 kHz) and from KORONA using the module setup above.



Bottom detection from EK60 at 38 kHz



Bottom detection from KORONA

Modules

Angle deletion
Bottom detection
Broadband notch filter
Broadband splitter
Bubble spike filter
Categorization
Cds viewer
Channel data removal
Channel removal
Combination
Combination remover
Complex to real
Comment
Data reduction
Dilate
Downsampling
Edge detection
Empty ping removal
Erode low values
Erode
Expression
Fill missing data
Filter 3×3
Horizontal offset correction
Isolation
Median
Noise quantification
Noise remover
Noise visualization
Ping collapsing
Ping thinning
Plankton inversion
Plugin
Remove bottom
Remove single channel
Rescale
School categorization
School detection
School filtering
Smoother
Spike filter
Spot noise
Temporary computations begin
Temporary computations end
Threshold
Time interval selection
Towfish
Trace border
Tracking
Vertical offset correction
Writer

Broadband notch filter module

This module removes spike noise from the frequency spectrum of the broadband signal and write outs the filtered broadband channels.

The noise removal is achieved by applying one or several notch filters, i.e. a (very) narrow bandstop filter. See the Details section below for the exact mathematical definition and further background information.

In order to apply a notch filter in LSSS, its parameters need to be specified in the file.

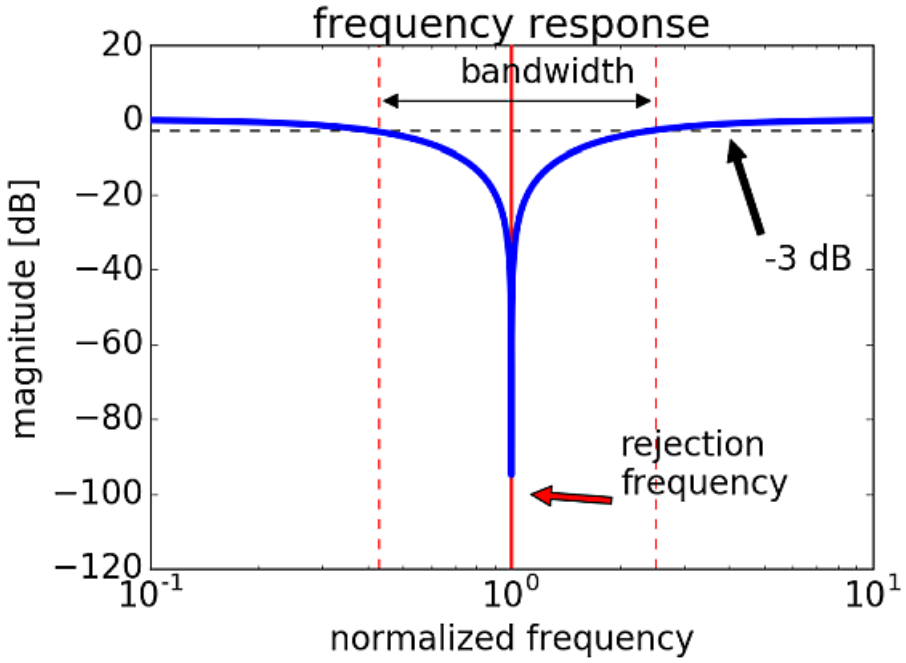
Parameters

The module requires a xml file containing the definition of the notch filters to apply to the data. The file can be edited directly from the configuration menu of the module. See here for an example of the xml file.

Each notch filter is defined by the following required parameters

- rejection frequency, and
- bandwidth.

The figure below showing the magnitude frequency response of a single notch filter illustrated the meaning of these two parameters.



Besides the filter parameters, it is also possible to specify the temporal validity of each of the notch filters by giving start and/or stop time. Time is to be specified in ISO 8601 format, i.e. [YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]Z.

It should be noted that the frequency resolution in the data is limited by the inverse of the pulse length. Thus, the filter bandwidth should be at least 2-3 times this limit resolution in order to capture the notch properly. E.g. for a pulse duration of 2 milliseconds, the filter width should be at least 1-1.5 kHz.

If already filtered data are processed by the module, only filters not already applied are considered. In other words, the set of filters applied is the union of previously applied and the currently specified filters, no filter is applied more than once.

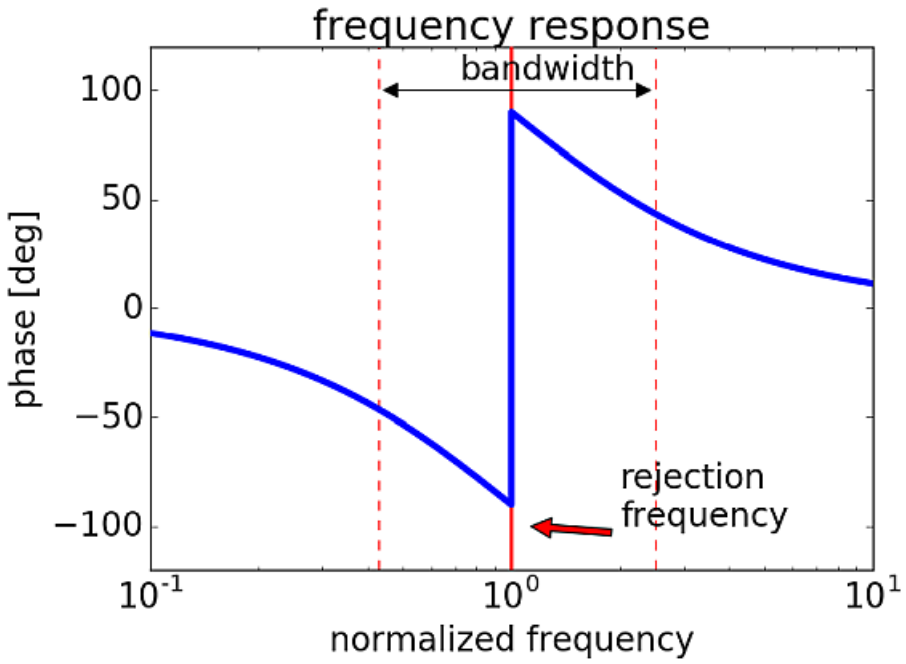
Details

The used notch filter is defined by the following transfer function

$$H(s) = \frac{s^2 + \omega^2}{s^2 + \frac{\omega}{q}s + \omega^2}$$

where ω is the rejection frequency and q the quality factor determining the bandwidth. The q-factor is found iteratively for a prescribed bandwidth.

The figure below shows the phase response of a notch filter with a rejection frequency of 1 Hz. For the magnitue response, see the figure above. The bandwidth of the filter is user chosen.



Broadband splitter module

This module splits the frequency band of each broadband data channel into narrower frequency bands and saves each of them as a single frequency channel.

The splitting of the original frequency band is performed by bandpass filtering the pulse compressed signal and computing sv at the centre frequency of this passband. See the Details section below for the exact mathematical definition and further background information.

It is worth noticing that the width of the sub-bands / number of single frequency channels affects the spatial resolution of the output. Narrower bands / more single frequency channels yield worse resolution.

Furthermore, the width of the bands is limited by the inverse of the pulse length. In order to yield meaningful results, the bandwidth of the generated frequency bands cannot be below this number. E.g. for a pulse length of 2 milliseconds, the minimal bandwidth is 500 Hz.

It is possible to perform downsampling during splitting. Angles may be included in the generated single frequency data. If downsampling is activated, also angles are downsampled.

Parameters

The module requires a xml file containing frequency bands to be converted to single frequency data, specified by their nominal frequency, their start frequency, and their stop frequency. The file can be edited directly from the configuration menu of the module. See here for an example of the xml file.

In addition, the following parameters are available:

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`AutoSplit`**: Split the original broadband channels into equally large sub-bands.
- **`SplitCount`**: The number of single frequency channels created from each broadband channel.
- **`StopBandDistance`**: Distance between passband and the stopbands, i.e. the totally attenuated frequencies, in kilohertz.
- **`MinBandwidth`**: Do not generate single frequency data from bands narrower then the specified bandwidth.
- **`Downsampling`**: Choose downsampling method.
- **`ComputeAngles`**: Whether or not angles should be computed.

Details

Some technical details about the processing yielding the single frequency channels.

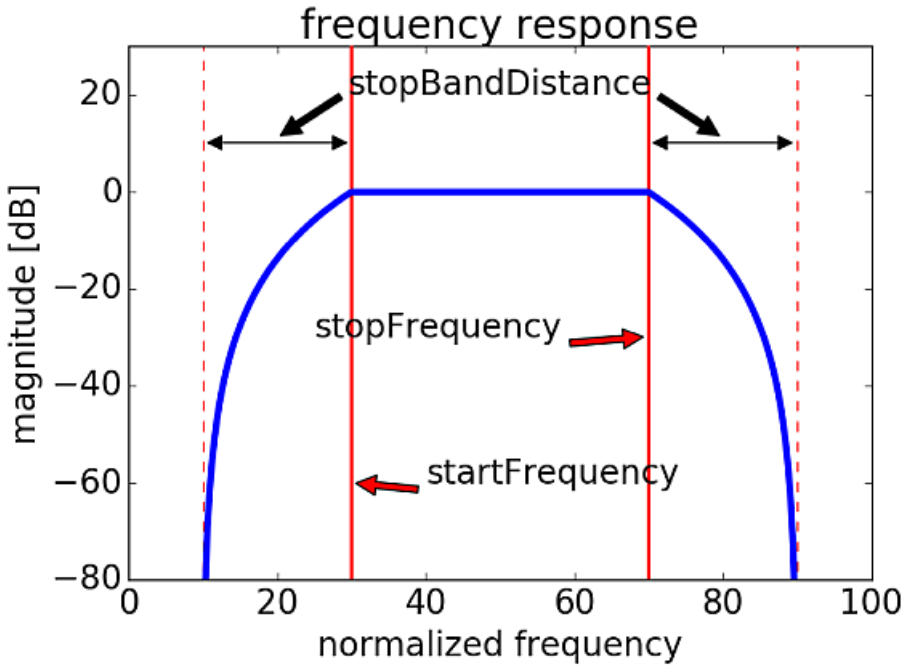
Bandpass filter

The used bandpass filter can be represented by the following transfer function

$$H(s) = \begin{cases} 0 & \text{for } \Im(s) \leq \text{startFrequency} - \text{stopBandDistance} \\ 1 - \frac{\text{startFrequency} - \Im(s)}{\text{stopBandDistance}} & \text{for startFrequency} - \text{stopBandDistance} < \Im(s) \leq \text{startFrequency} \\ 1 & \text{for startFrequency} < \Im(s) \leq \text{stopFrequency} \\ 1 - \frac{\Im(s) - \text{stopFrequency}}{\text{stopBandDistance}} & \text{for stopFrequency} < 0\Im(s) \leq \text{stopFrequency} + \text{stopBandDistance} \\ 0 & \text{for } \Im(s) \geq \text{startFrequency - stopBandDistance} \end{cases}$$

where I(s) denotes the imaginary part of a complex number.

The figure below shows the frequency response of a bandpass filter with a start frequency of 30 kHz, stop frequency of 70 kHz, and stopband distance of 20 kHz.



Other

- **Effective pulse length:** The effective pulse length τ is calculated from the restricted sent pulse.
- **Transducer gain:** The transducer gain is corrected to correspond to the centre frequency of each sub-band.

Categorization module

Module for categorization. See references for description of original methods.

Requirements

- Depth datagrams, Dep0Datagram, see Bottom detection module, if category *bottom* is active.
- Noise datagrams, Nqp0Datagram, see NoiseQuantificationModule, if category *noise* is active.
- Smoothing, see SmootherModule, is recommended.

New datagrams

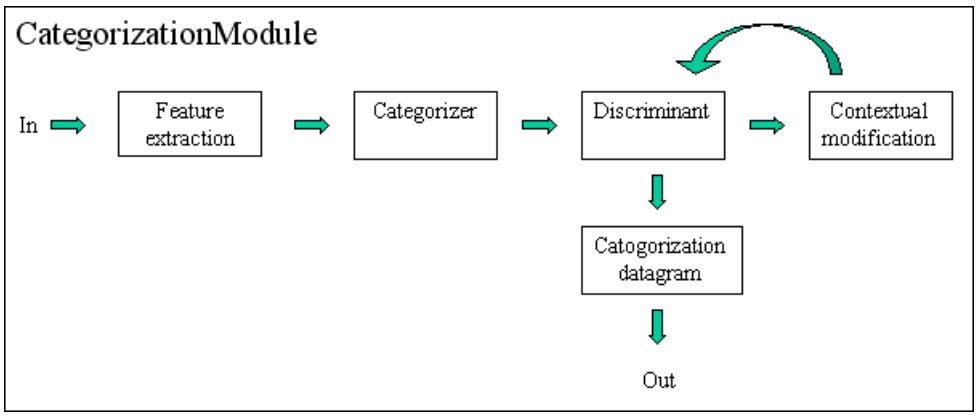
- One initial categorization configuration datagram, Cac0Datagram.
- Subsequently one categorization datagram, Cad0Datagram, for each ping.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`CategoryCount`**: Number of categories per pixel to put in the categorization datagrams.
  Allowed values: `CategoryCount` >= 1
- **`ActiveFeatures`**: Which features to use. Only enabled features can be set to active.
- **`ActiveCategories`**: Which categories to use. Only enabled categories can be set to active.
- **`BlindZoneDepth`**: Depth of surface dead zone.
  Allowed values: `BlindZoneDepth` >= 0.0
- **`Categorizer`**: Type of classifier.
  Allowed values: `{ Gauss, Nearest neighbor, Combined }`
- **`Discriminant`**: Type of discriminant function.
  Allowed values: `{ Aposteriori, Cost }`
- **`ContextualCorrection`**: Type of contextual correction iteration.
  Allowed values: `{ None, ICM }`
- **`ContextualIterations`**: Number of contextual correction iterations.
  Allowed values: `ContextualIterations` >= 0

Details

The data flow through the categorization module is depicted in the following figure:



References

- Korneliussen, R. J., and Ona, E. 2002. An operational system for processing and visualizing multi-

frequency acoustic data. – ICES Journal of Marine Science, 159: 293–313.
- Korneliussen, R. J., and Ona, E. 2003. Synthetic echograms generated from the relative frequency response. – ICES Journal of Marine Science, 60: 636–640.
- Korneliussen, R. J., Heggelund, Y., Eliassen, I. K., and Johansen, G. O. 2009. Acoustic species identification of schooling fish. – ICES Journal of Marine Science, 66: 1111–1118.
- Pedersen, G., and Korneliussen, R. J. 2009. The relative frequency response derived from individually separated targets of northeast Arctic cod (Gadus morhua), saithe (Pollachius virens), and Norway pout (Trisopterus esmarkii). – ICES Journal of Marine Science, 66: 1149–1154.
- Korneliussen, R. J. 2010. The acoustic identification of Atlantic mackerel. – ICES Journal of Marine Science, 67: 1749–1758.

Isolation module

Isolates one category by zeroing out all pixels of different categories.

Requirements

- Categorization datagrams, Cac0Datagram and Cad0Datagram, see CategorizationModule.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`Category`**: The category to isolate.

School categorization module

Module for performing categorization of whole schools.

Requirements

- Region border datagrams, RegionBorderDatagram, see School detection module.
- Region info datagrams, RegionInfoDatagram, see School detection module.
- Categorization module, see CategorizationModule, must be before this module in the module setup.

New datagrams

- A school categorization datagram, Cas0Datagram, for each detected region which has not been classified as multi-species. If the module has been unable to determine whether the region is single- or multi-species, the region is classified as uncategorized.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Horizontal cell size:`** Cell size in number of pings.
- **`Vertical cell size:`** Cell size in number of samples.
- **`Upper sample offset:`** Number of samples excluded at top of region.
- **`Lower sample offset:`** Number of samples excluded at bottom of region.
- **`Horizontal offset:`** Number of pings excluded at the left and right edges of region.
- **`Secondary category fraction:`** Fraction of cells in region that can be allowed to be different from dominant category.
- **`School categorization distribution:`** Distribution to use when categorizing whole schools. Allowed values: `{ CELL, SCHOOL }`
- **`Fill factor for cells:`** The fill factor of cells must be above this value for cells to be used in the analysis of schools.
- **`Minimum fraction categorized cells:`** The fraction of categorized cells has to be above this factor for the school to be categorized.

References

- Korneliussen, R. J., Heggelund, Y., Eliassen, I. K., and Johansen, G. O. 2009. Acoustic species identification of schooling fish. – ICES Journal of Marine Science, 66: 1111–1118.

Angle deletion module

This module converts split beam data to single beam data.

Real data

For real data containing power and angles, e.g. Raw0Datagrams, the angles are simply discarded.

Complex data

For complex data containing multiple sectors with complex numbers, e.g. broadband data or narrowband data , then the sectors are reduced to a single sector such that values of $S_V(f)$ and uncorrected TS(f) are preserved.

If y_i is the complex sample value of sector $i \in [1, n]$, then the replacement single beam value is computed as

$$y = (1 / \sqrt{n}) \sum y_i$$

Channel data removal module

Removes data on specified channels. The channels themselves are not removed.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`Channels`**: Comma separated list of channel numbers.
- **`Channels from end`**: Comma separated list of channel numbers from end. (1 = last channel, 2 = second last channel, etc.)
- **`Frequencies`**: Comma separated list of kHz.
- **`Keep specified`**: If selected then keep data on only the specified channels.

Channel removal module

Removes specified channels.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Channels:`** Comma separated list of channel numbers.
- **`Channels from end:`** Comma separated list of channel numbers from end. (1 = last channel, 2 = second last channel, etc.)
- **`Frequencies:`** Comma separated list of kHz.
- **`Keep specified:`** If selected then keep only the specified channels.

Combination remover module

This module removes the last channel. It can be used to remove the channel created by the CombinationModule. For instance if creating certain datagrams requires changing the pingdata and one wants the original channels to stay unaltered one can make a copy of the channel one wants to alter, manipulate the copy, add datagrams into the stream based on the copy and then remove the copy using this module. That is how the School detection module works.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.

Complex to real module

This module converts complex broadband or narrowband data to sv values and angles.

For broadband data the center frequency is used for sv and angles. The nominal frequency is left as is.

Conversion of broadband data to multiple single frequency datagrams per channel is possible by using the BroadbandSplitterModule.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`Compute angles`**: If selected then angles are computed.

Data reduction module

Removes data above and below selected ranges.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`BlindZone`**: Remove data above the blind zone configured in Transducer ranges file.
- **`MinRange`**: Remove data above this range.
- **`MinDepth`**: Remove data above this depth.
- **`TransducerRange`**: Remove data below the range configured in Transducer ranges file.
- **`MaxRange`**: Remove data below this range.
- **`MaxDepth`**: Remove data below this depth.

Downsampling module

This is a module for reducing vertical resolution by downsampling. Sv is downsampled by averaging. Angles are downsampled by using the angles of the center sample in the downsample interval.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Downsampling:`** The method to find the downsampling factor.
- **`DownsamplingFactor:`** If `Downsampling = FACTOR` then this is the downsampling factor.
- **`DownsamplingSampleSize:`** If `Downsampling = SAMPLE_SIZE` then the downsampling factor calculated to give approximately this sample size.

Empty ping removal module

Removes pings with no sample datagrams.

Parameters

- `Active`: If the module is not active then all datagrams pass unaltered through it.

Ping thinning module

Removes pings.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`Initial skip`**: The number of pings to skip initially.
- **`Periodical skip`**: The number of pings to skip between each included ping.

Remove single channel module

Removes a specified channel.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`ChannelNumber`**: Channel number to remove. Refers to echosounder channel numbering.

Dilate module

A dilation filter. Each pixel gets the value of the max value of its neighbors. Size of neighborhood vertically is defined by setting `VertResolution` , if 1 it is a 3*3 neighborhood with the current pixel in the middle of it. In general a 3* (2*`VertResolution`+1) neighborhood. The module buffers up one ping thereby delaying the stream by one ping. Informally the filter has the effect of growing all objects by adding pixels around the object edges.

Parameters

- **`Active`:** If the module is not active then all datagrams pass unaltered through it.
- **`LogarithmicValues`:** Use log(sv) or sv.
- **`ShowDetails`:** Shows more detailed settings (OnlyLast,AutomaticDepthRange,StartDepth,EndDepth).
- **`OnlyLast`:** When true, only last channel is processed, if false all channels are processed.
- **`AutomaticDepthRange`:** If true and Bottom detection module is used then the range goes to the detected seabed, if true and Bottom detection module is not used the range goes to max depth of the ping.
- **`StartDepth`:** Start depth in meter for filter.
- **`EndDepth`:** End depth in meter for filter.
- **`VertResolution`:** Vertical resolution of filter(integer).

Edge detection module

Detects edges by using sobel horizontal and vertical filter. Filter can be prolonged vertically by setting mVertRes. Value if gradient is 0 is -120, and increases linearly with strength of edge. Value is multiplied with scaleFactor.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`LogarithmicValues`**: Use log(sv) or sv.
- **`ShowDetails`**: Shows more detailed settings (OnlyLast,AutomaticDepthRange,StartDepth,EndDepth).
- **`OnlyLast`**: When true, only last channel is processed, if false all channels are processed.
- **`AutomaticDepthRange`**: If true and Bottom detection module is used then the range goes to the detected seabed, if true and Bottom detection module is not used the range goes to max depth of the ping.
- **`StartDepth`**: Start depth in meter for filter.
- **`EndDepth`**: End depth in meter for filter.
- **`ScaleFactor`**: Scale of filter.
- **`VertResolution`**: Vertical resolution of filter(integer).

Erode low values module

This is a module for eroding low values. All samples close to another sample with a value lower or equal than a given threshold are set to a predefined value. A sample is close to another if they are on the same or neighbouring pings and their vertical distance is less than a configured distance in meters.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`LogarithmicValues`**: Use log(sv) or sv.
- **`ShowDetails`**: Shows more detailed settings (OnlyLast,AutomaticDepthRange,StartDepth,EndDepth).
- **`OnlyLast`**: When true, only last channel is processed, if false all channels are processed.
- **`AutomaticDepthRange`**: If true and Bottom detection module is used then the range goes to the detected seabed, if true and Bottom detection module is not used the range goes to max depth of the ping.
- **`StartDepth`**: Start depth in meter for filter.
- **`EndDepth`**: End depth in meter for filter.
- **`Threshold`**: The threshold for determining low values.
- **`Replacement value`**: The value to set to samples that are close to samples with value below the threshold.
- **`Vertical extent`**: The vertical distance that low values are eroded.

Erode module

An erode filter. Each pixel gets the value of the min value of its neighbors. Size of neighborhood vertically is defined by setting `VertResolution`, if set to 1 it is a 3*3 neighborhood with the current pixel in the middle of it. In general it is a 3*(2*`VertResolution`+1) neighborhood. Informally the filter has the effect of shrinking all objects by removing all the pixels around the edges.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`LogarithmicValues`**: Use log(sv) or sv.
- **`ShowDetails`**: Shows more detailed settings (OnlyLast,AutomaticDepthRange,StartDepth,EndDepth).
- **`OnlyLast`**: When true, only last channel is processed, if false all channels are processed.
- **`AutomaticDepthRange`**: If true and Bottom detection module is used then the range goes to the detected seabed, if true and Bottom detection module is not used the range goes to max depth of the ping.
- **`StartDepth`**: Start depth in meter for filter.
- **`EndDepth`**: End depth in meter for filter.
- **`VertResolution`**: Vertical resolution of filter(integer).

Fill missing data module

Creates missing pings. Pings with missing raw0Datagrams will after passing through this module have same values on the missing raw0datagrams as the previous ping had. This module is used in front of other modules that don't handle missing pings in the middle of a stream.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.

Filter 3×3 module

Class implementing a 3*3 filter. Either choose a filter from the dropdownmenu or specify one manually by setting all 3*3 values.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`LogarithmicValues`**: Use log(sv) or sv.
- **`ShowDetails`**: Shows more detailed settings (OnlyLast,AutomaticDepthRange,StartDepth,EndDepth).
- **`OnlyLast`**: When true, only last channel is processed, if false all channels are processed.
- **`AutomaticDepthRange`**: If true and Bottom detection module is used then the range goes to the detected seabed, if true and Bottom detection module is not used the range goes to max depth of the ping.
- **`StartDepth`**: Start depth in meter for filter.
- **`EndDepth`**: End depth in meter for filter.
- **`FilterType`**: Specifies what filter to use
  Allowed values: `{ User defined, /*smooth1*/{{1,2,1},{2,1,2},{1,2,1}}, /*smooth2*/{{1,1,1},{1,1,1},{1,1,1}},`
  `/*lowpass*/{{1,2,1},{2,4,2},{1,2,1}}, /*sharpen*/{{-1,-1,-1},{-1,9,-1},{-1,-1,-1}},`
  `/*Laplacian edge (Deltasquared)*/{{1,-2,1},{-2,4,-2},{1,-2,1}},`
  `... }`
- **`00`**: matrix value.
- **`01`**: matrix value.
- **`02`**: matrix value.
- **`10`**: matrix value.
- **`11`**: matrix value.
- **`12`**: matrix value.
- **`20`**: matrix value.
- **`21`**: matrix value.
- **`22`**: matrix value.

Median module

Sets pixelvalue to median of its 8 neighbors and itself. Has the effect of removing spikes and of smoothing the data whilst preserving edges.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`LogarithmicValues`**: Use log(sv) or sv.
- **`ShowDetails`**: Shows more detailed settings (OnlyLast,AutomaticDepthRange,StartDepth,EndDepth).
- **`OnlyLast`**: When true, only last channel is processed, if false all channels are processed.
- **`AutomaticDepthRange`**: If true and Bottom detection module is used then the range goes to the detected seabed, if true and Bottom detection module is not used the range goes to max depth of the ping.
- **`StartDepth`**: Start depth in meter for filter.
- **`EndDepth`**: End depth in meter for filter.

Ping collapsing module

Collapses sequential pinging.

Parameters

- `Active`: If the module is not active then all datagrams pass unaltered through it.

Remove bottom module

Module for removing data around the detected bottom.

Data in three different intervals can be removed:

- Above upper bottom limit
- Between upper and lower bottom limits
- Below lower bottom limit

If `AutomaticDepthRange` is deselected then `StartDepth` and `EndDepth` can be specified manually, and these will be used as upper and lower bottom limits.

Requirements

Depth datagrams, Dep0Datagram, must be computed, see Bottom detection module.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`LogarithmicValues:`** Use log(sv) or sv.
- **`ShowDetails:`** Shows more detailed settings (OnlyLast,AutomaticDepthRange,StartDepth,EndDepth).
- **`OnlyLast:`** When true, only last channel is processed, if false all channels are processed.
- **`AutomaticDepthRange:`** If true and Bottom detection module is used then the range goes to the detected seabed, if true and Bottom detection module is not used the range goes to max depth of the ping.
- **`StartDepth:`** Start depth in meter for filter.
- **`EndDepth:`** End depth in meter for filter.
- **`Use neighbours:`** Use minimum depth from current ping and neighbours.
- **`Above:`** Upper bottom limit is this many samples above bottom.
- **`Below:`** Lower bottom limit is this many samples below bottom.
- **`Remove above bottom:`** Remove values above upper bottom limit.
- **`Remove bottom:`** Remove values between upper and lower bottom limits.
- **`Remove below bottom:`** Remove values below lower bottom limit.
- **`Fill value:`** Value to set pixels to.

Threshold module

Masks out values in specific range. This module sets all values >= Threshold to value AboveThreshold or does not change them if AboveSame is true. Similarly it sets all values < Threshold to value BelowThreshold or does not change them if BelowSame is true

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`LogarithmicValues`**: Use log(sv) or sv.
- **`ShowDetails`**: Shows more detailed settings (OnlyLast,AutomaticDepthRange,StartDepth,EndDepth).
- **`OnlyLast`**: When true, only last channel is processed, if false all channels are processed.
- **`AutomaticDepthRange`**: If true and Bottom detection module is used then the range goes to the detected seabed, if true and Bottom detection module is not used the range goes to max depth of the ping.
- **`StartDepth`**: Start depth in meter for filter.
- **`EndDepth`**: End depth in meter for filter.
- **`Threshold`**: Threshold value.
- **`Cut below`**: If true all samples with values < threshold will become 'Value below'. Otherwise, the sample values will remain the same.
- **`Cut above`**: If true all samples with values ≥ threshold will become 'Value above'. Otherwise, the sample values will remain the same.
- **`Value below`**: Value to set to samples with value < threshold. The value is given in log(sv) with unit dB.
- **`AboveVal`**: Value to set to samples with value ≥ threshold. The value is given in log(sv) with unit dB.

Writer module

Writes to a .cmr file for loading into EK60.

New datagrams

- One initial module configuration datagram, Cds0Datagram, is written to the file.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`FileName`**: Output file relative to the directory deduced from the input file.

Cds viewer module

For viewing the module configuration stored in a processed file. Each time a Cds0Datagram is encountered, a ModuleEditor window appears displaying its configuration. Edits in this ModuleEditor have no effects.

Requirements

- Module configuration datagram, Cds0Datagram.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.

Rescale module

Rescales some channels to fit into the logarithmic Sv range.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`RescaleChannels`**: Which channels to rescale.
  Allowed values: `{ all, half, last, none }`
- **`DesiredMinimum`**: Desired minimum.
- **`DesiredMaximum`**: Desired maximum.
- **`MinMaxFraction`**: Discard this fraction of lower and upper values when deciding the rescaling.
  Allowed values: 0.0 <= `MinMaxFraction` <= 1.0
- **`InitializationDatagramCount`**: Number of datagrams from each channel used to determine rescaling factors.
  Allowed values: `InitializationDatagramCount` >= 1
- **`InitializationMaxDatagramCount`**: Maximum number of datagrams to buffer up when finding rescaling factors.
  Allowed values: `InitializationMaxDatagramCount` >= 1

Time interval selection module

Module for processing only the datagrams in a specified time interval. The start and end dates are in the same format as is displayed in the upper right in the Korona main window. The empty string implies no restriction.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`StartDateString`**: The start date and time in UTC (`dd.MM.yyyy HH:mm:ss`). The empty string implies no restriction.
- **`EndDateString`**: The end date and time in UTC (`dd.MM.yyyy HH:mm:ss`). The empty string implies no restriction.

Bubble spike filter module

A module for removal of narrow vertical spikes in the echogram, typically caused by bubble clouds that blocks the sound below the ship.

This filter works in a similar way as the Spike filter module, except that is it is looking for vertical structures that are significant weaker than the surroundings (in contrast to the Spike filter module that looks for vertical structures that are stronger).

A sample is considered a spike if two criteria hold:

1. The current search column is sufficiently weaker than the neighbouring search columns, i.e.,

   Neighbouring search column median < Current search column median + **`VerticalDelta`**

   is true for at most one neighbouring search column.
2. The current sample is sufficiently weaker than the search window, i.e.,

   Current sample value < search window median - **`TotalDelta`**

Parameters

See Spike filter module for a listing of parameters.

Noise quantification module

Produces noise quantification datagrams. The datagrams contain:

1. The average noise level.
2. The noise upper limit.
3. The quality of the noise estimate as number in the range [0, 100]. The quality depends on where in the water column data representative of noise is found.

The overall procedure of this module is:

1. Select parts of the data representative of noise.
2. Accumulate that noise data in a histogram.
3. Use the histogram to quantify the noise level.

Selecting noise values

The noise values input to the histogram are found ping by ping:

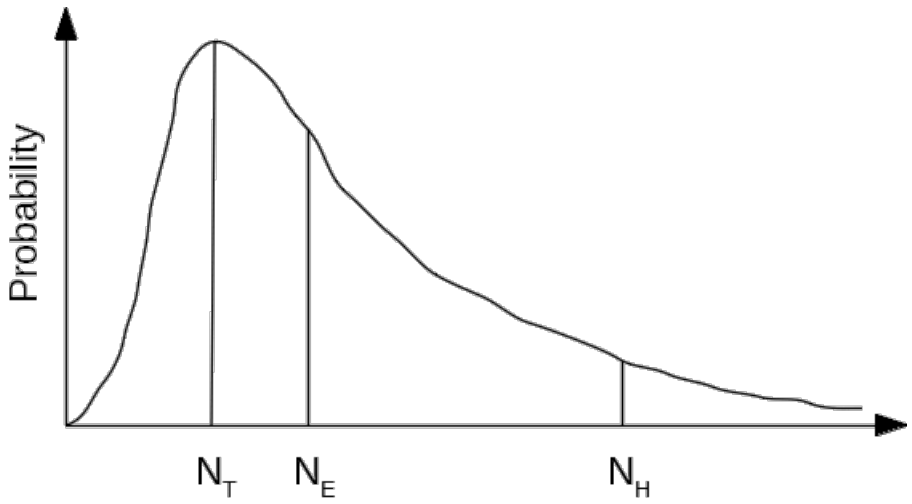
1. Remove very low values
2. Smooth the data vertically
3. Select noise data
4. Filter on quality
5. Filter on threshold

The selection of noise data in a ping is done in one of these ways:

1. Between first and second bottom echo. Quality = 100.
2. Below a frequency dependent deep depth limit D_1. Quality = 100.
3. Between a frequency dependent depth limit D_2 and the bottom. Quality in the range [50, 100].
4. Below D_2. Quality = 25.
5. All data. Quality = 0.

Estimating noise parameters

The selected noise data that passes the filtering on quality and threshold is accumulated in a histogram, which is used for estimating the noise parameters:



First the value N_T at peak probability is found. Then N_H is found at 2 to 4 standard deviations above N_T, depending on noise being close to normal distributed or not. Finally, N_E is calculated as the average of noise values below N_H.

The quality of the noise parameters is set to the average quality of the data in the histogram. If probability(N_H) is not low enough compared to probability(N_E), then the quality is reduced by a factor between 0.6 and 1.

Fallback noise

Throughout processing, the file noise.xml in the processing destination directory updated. This file is divided in 1 hour intervals and each interval contains the noise parameters with the highest quality and the lowest value.

If not enough noise data samples are found, then the noise parameters from the closest available interval in noise.xml is used. In this case, the quality in the generated datagrams is in the range [10, 20], depending on the quality in noise.xml.

Requirements

- Depth datagrams, Dep0Datagram, see Bottom detection module.

New datagrams

- Noise quantification datagrams, Nqp0Datagram.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`Smooth`**: Type of smoothing.
  Allowed values: `{ none, running averager, averager }`
- **`SmoothInterval`**: Number of samples to smooth vertically before making histogram.
  Allowed values: `SmoothInterval` >= 0
- **`Mask`**: Type of masking.
  Allowed values: `{ none, dynamic }`
- **`MinimumQuality`**: Minimum quality of histogram data. The quality for each ping is set based on where the data for the histogram is picked, and then the mean quality for all data used is calculated:
  Quality = 100. Data between first and second bottom echo are used.
  Quality = 50. Data below a certain (frequency-dependent) range from the transducer, but above the first bottom echo are used.
  Quality = 0. All data are used.
  Allowed values: 0.0 <= `MinimumQuality` <= 100.0
- **`ThresholdMasking`**: Masks values above a threshold and reflections below bottom.
- **`Threshold`**: The threshold to use for threshold masking.
- **`Histogram`**: Type of histogram.
  Allowed values: `{ geometric, linear, constant }`
- **`UseTimeStepBuffer`**: Try to center noise parameters.
- **`TimeStepBufferMaxSize`**: Maximum number of time steps in buffer.
- **`HistogramInitializationCellCount`**: Number of cells in the histograms at initialization.
  Allowed values: `HistogramInitializationCellCount` >= 1
- **`HistogramMaximumCellCount`**: Maximum number of cells in the histograms.
  Allowed values: `HistogramMaximumCellCount` >= 1
- **`HistogramInitializationSampleCount`**: Minimum count of noise samples at initialization of the histograms.
  Allowed values: `HistogramInitializationSampleCount` >= 1
- **`HistogramMinimumSampleCount`**: Minimum number of noise samples in the histograms.
  Allowed values: `HistogramMinimumSampleCount` >= 1
- **`HistogramSmooth`**: Whether or not to smooth the probability distribution.
- **`HistogramSmoothFactor`**: Diffusion time step = HistogramSmoothFactor * dx^2.
  Allowed values: `HistogramSmoothFactor` > 0.0

References

- Korneliussen, R. J. 2000. Measurement and removal of echo integration noise. ICES Journal of Marine Science, 57: 1204–1217.

Noise remover module

Removes noise based on parameters in noise quantification datagrams.

The noise power index N is modified for each sample:

$N = N - N_E$ if $N \geq N_H$
$N = 0$ if $N < N_H$

where N_E and N_H are the noise average and upper limit.

Requirements

- Noise quantification datagrams, Nqp0Datagram, see NoiseQuantificationModule.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`RemoveNoiseFromStart:`** Search for noise parameter datagram and use it for the first datagrams.
- **`MaxBufferSize:`** Maximum buffer size in search for first noise parameter datagram.
  Allowed values: `MaxBufferSize` >= 0

Noise visualization module

Visualizes noise parameters in a separate window.

Requirements

- Noise quantification datagrams, Nqp0Datagram, see NoiseQuantificationModule.
- Noise removing, see NoiseRemoverModule, should be done *after* noise visualization, since the median of equidistant samples is plotted.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`UpdatePeriod:`** Number of pings between each graph update.
  Allowed values: `UpdatePeriod` >= 1
- **`TimeSeriesLength:`** Number of pings in the graphs.
  Allowed values: `TimeSeriesLength` >= 1
- **`HistogramSpan:`** Upper limit of plotted histogram domain in multiples of N_H.
  Allowed values: `HistogramSpan` > 0.0
- **`MedianSampleCount:`** Number of equidistant samples used for finding median noise.
  Allowed values: `MedianSampleCount` >= 0

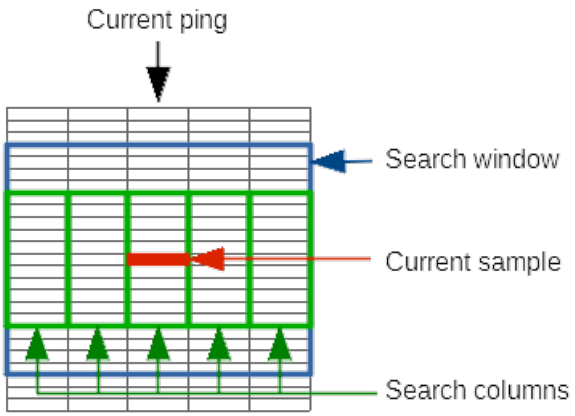
Spike filter module

A module for removal of narrow vertical spikes in the echogram, typically caused by instrumentation on the ship.

The module works on a per sample basis, and for each sample computes:

- The median of a *search window* of dimension 5 pings × (2 · `WindowMedianSearchHeight` + 1) samples.
- The medians of each of 5 *search columns* of height 2 · `VerticalMedianSearchHeight` + 1 samples.

The search window and search columns are centered around the current sample as illustrated in the following figure:



If the heights are specified as duration or distance, then the corresponding number of samples is calculated for each ping.

A sample is considered a spike if two criteria hold:

1. The current search column is sufficiently stronger than the neighbouring search columns, i.e.,

   Neighbouring search column median > Current search column median - `VerticalDelta`

   is true for at most one neighbouring search column.
2. The current sample is sufficiently stronger than the search window, i.e.,

   Current sample value > search window median + `TotalDelta`

If the current sample is considered i spike then its value is replaced by search window median.

Parameters

- `Active`: If the module is not active then all datagrams pass unaltered through it.
- `TotalDelta`: Minimum difference of current sample to search window median to be spike candidate.
- `VerticalDelta`: Minimum difference of vertical median to (most) neighbouring pings to be spike candidate.
- `Debug`: Replace spikes by 0 instead of median.
- `VerticalUnit`: The unit for specifying the height of the *search window* and the *search column*.
- `VerticalMedianSearchHeight`: Half the height of the search column in number of samples.
- `WindowMedianSearchHeight`: Half the height of the search window in number of samples.
- `VerticalMedianSearchDuration`: Half the height of the search column in milliseconds.
- `WindowMedianSearchDuration`: Half the height of the search window in milliseconds.
- `VerticalMedianSearchDistance`: Half the height of the search column in meters.
- `WindowMedianSearchDistance`: Half the height of the search window in meters.

Spot noise module

A module for removal of spot-noise in the echogram. The module works on a per sample basis, and compares each sample to surrounding samples in a user defined *search window* of dimension 3 pings X 5 samples

The 93% percentile value (=testValue) is calculated by the filter for each sample. The value of the current sample is then compared against this testValue + **`Delta`**, and is replace by the median of the 3 x 5 search window.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`Delta`**: Require sample > (window 93% percentile + Delta) to consider sample as being spot-noise. Given in log(s_V).
- **`Debug`**: Replace spikes by 0 instead of median.

Horizontal offset correction module

Module for horizontal offset correction.

The offsets are configured in the Horizontal transducer offsets part of the Config file settings.

Only the value in the field AlongCorrection is used in this module.

Parameters

- `Active`: If the module is not active then all datagrams pass unaltered through it.

Vertical offset correction module

Module for vertical offset correction.

The offsets are configured in the Vertical transducer offsets part of the Config file settings.

Only the values specified in the fields DepthCorrection and PulseDelayDepthCorrection are used in this module.

Parameters

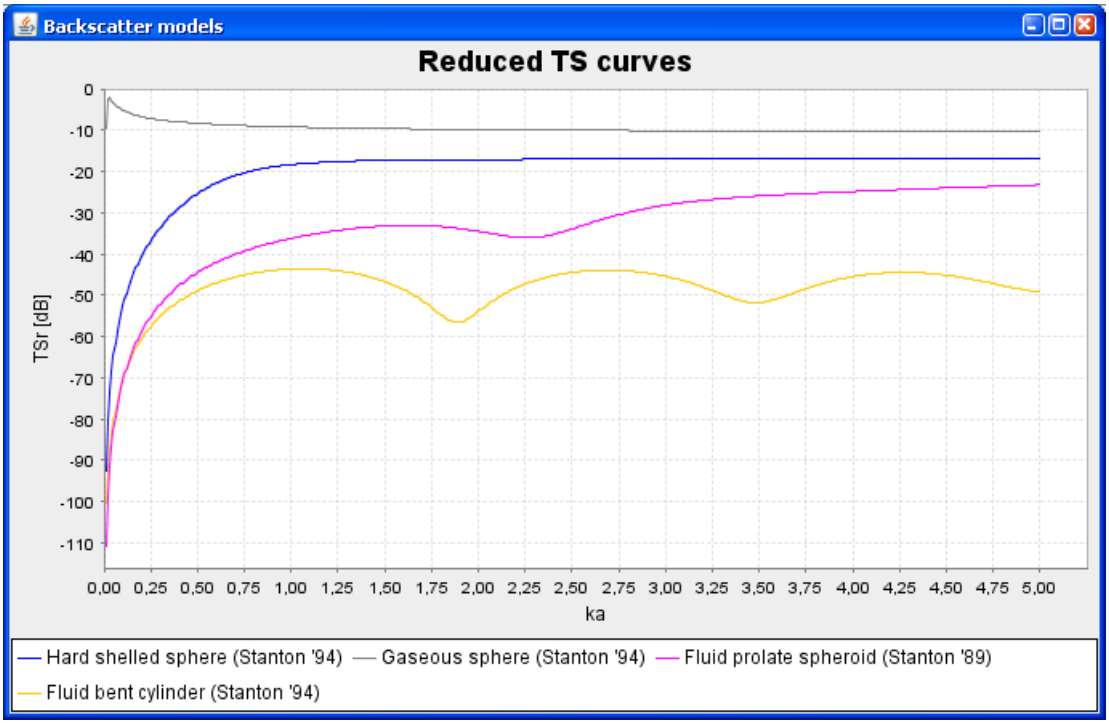
- **`Active`**: If the module is not active then all datagrams pass unaltered through it.

Plankton inversion module

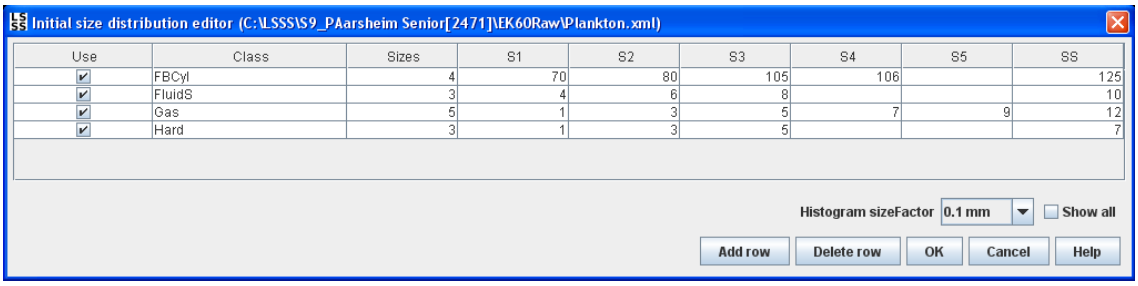
Produces plankton inversion datagrams. For more about the algorithm see plankton inversion algorithm. For more about the implemented plankton models, see plankton models.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`PingsPerBin`**: The number of pings per bin.
- **`DepthSamplesPerBin`**: The number of samples per bin.
- **`Use global noise threshold`**: Toggle this if the global noise threshold should be enabled. Using the global noise threshold means that the sample values for all enabled frequencies should be above the given threshold value for the sample to be used in inversion.
- **`Noise threshold`**: The global noise threshold. See above.
- **`CategoriesUsedForInversion`**: Here the user can toggle which categories (see CategorizationModule) should be used as input to plankton inversion.
- **`ActiveFrequencies`**: The user can toggle which frequencies should be used in the inversion. The effect on the domain of backscatter models can be seen in the backscatter curve plot.
- **`MinResidualErrorThreshold`**: The minimal residual error. The inversion iterations for a model stops if the residual is below this value.
  Allowed values: `MinResidualErrorThreshold` >= 0.0
- **`MaxResidualErrorThreshold`**: The maximal residual error. The plankton sample is set to no category if the residual for the best-fit model is higher than this number.
  Allowed values: `MaxResidualErrorThreshold` >= 0.0
- **`LevenbergMarquardtFactor`**: The Levenberg-Marquardt factor. Used in during the inversion to give preference to solution with the smallest norm. A higher value means that a small norm is more important than a small residual. Suggested value between 1e-6 and 1e-4.
  Allowed values: `LevenbergMarquardtFactor` >= 0.0
- **`MaxIter`**: The maximum number of iterations for each model inversion.
  Allowed values: `MaxIter` >= 1
- **`HardShelled`**: Toggle on if the hard shelled sphere model (Stanton 1994) is to be used in the inversion.
- **`HardShelledR`**: Reflection coefficient.
- **`GaseousSphere`**: Toggle on if the gaseous sphere model (Stanton 1994) is to be used in the inversion.
- **`GasSphereG`**: Relative density.
- **`GasSphereH`**: Relative sound speed.
- **`FluidSpheriod`**: Toggle on if the fluid prolate spheroid model (Stanton 1989) is to be used in the inversion.
- **`FluidSpheriodG`**: Relative density.
- **`FluidSpheriodH`**: Relative sound speed.
- **`FluidSpheriodBeta`**: Length to width ratio.
- **`FluidBent`**: Toggle on if the fluid bent cylinder model (Stanton 1994) is to be used in the inversion.
- **`FluidBentR`**: Reflection coefficient.
- **`FluidBentS`**: Standard deviation of length.
- **`FluidBentBeta`**: Length to width ratio.
- **`SDWBA`**: Toggle on if the SDWBA model (Conti 2006) is to be used in the inversion.
- **`SDWBA coefficients`**: Select which set of SDWBA coefficients should be used (defined in Conti 2006)
- **`BackscatterCurves`**: Opens a view of the backscatter curves with the current configuration of the models. The unused part of the backscatter curves will be drawn with a thinner line width than the parts that can be used given the set of frequencies and the given initial size distributions of the individuals belonging to the models.



- **`Editor:`** Opens an editor for the initial size distribution for the models:



This dialog lets the user specify an initial size distribution for the various models used in the plankton inversion.

Each row defines an initial size distribution histogram for a given class [FBCyl, FluidS, Gas, Hard]. The species column is used for entering a catch. By clicking "Show all", it is possible to specify a start/stop date/time/depth for each distribution, defining a window of validity in terms of time and depth.

The number of bins in the histogram is given in the "size" column. Entering a higher/lower number here will dynamically alter the number of columns in the table to match the number of bins in the histogram. The SX gives the lower edge of each histogram bin, while SS gives the upper edge of the last bin. NX gives the contents of each bin. If a divider is entered that is inconsistent with the previous/next divider, the user is asked if the surrounding inconsistent dividers should be automatically adjusted. The dividers will be adjusted with one unit with respect to the entered divider.

The initial size factor in the combo box gives the scaling factor for the histogram bin edges.

Additional initial size distributions can be specified by pressing "Add row". Distributions are deleted by selecting a row and pressing "Delete row".

Pressing OK saves the setting to the XML file indicated in the window header.

Plankton inversion algorithm

The purpose of the plankton inversion algorithm is to fit the gathered volume backscatter coefficients for multiple frequencies to analytical plankton backscatter models. The data is fitted to each model by the non-negative least squares algorithm (Lawson & Hanson). The model with the smallest residual is selected as the best fit model if the residual is smaller than a configurable maximal residual.

Description of algorithm

Let an analytical backscatter model be given by **σ(f, l)**, where **f** is a transducer frequency and **l** is a plankton size. The models used in this software and their parameters are described in Plankton inversion module

Let $[f_1, f_2, ..., f_M]$ be the frequencies used, let **c** = $[s_{v1}, s_{v2}, ..., s_{vM}]$ be the measured backscatter for a collection of samples, and let **N** = $[n_1, n_2, ..., n_N]$ be an abundance vector for an initial size distribution **b** = **l** = $[l_1, l_2, ..., l_N]$.

The system of equations to be solved can with respect to the abundance vector **N** be expressed in matrix form as
A N = c,
where the elements in **A**, $A_{ij} = \sigma(f_i, l_j)$, is the backscatter computed by the plankton model for frequency number i in **c** and length number j in **b**.

Rows are added to the system:
c = $[s_{v1}, s_{v2}, ..., s_{vM}, 0, ..., 0]$ (length M + N).
A is extended by a submatrix of size N×N, given by $\mathbf{I_N} \times \mathbf{||A||} \times \mathbf{LMfactor}$,

$$A = \overbrace{\left\{ \begin{array}{c} A_{ij} \\ I_N \, ||A|| \, \mathrm{LMfactor} \end{array} \right.}^{N} \Bigg\} M + N$$

where $\mathbf{I_N}$ is the identity matrix of size N×N, **||A||** is the norm of **A**, and **LMfactor** is the Levenberg-Marquardt factor (see Lawson & Hanson). The purpose of the Levenberg-Marquardt factor is to give a preference to the solution of smallest norm if several solutions are acceptable. It seems like a value between 1e-6 and 1e-4 gives solutions with reasonable norms without much change in residual.

The solution algorithm is described in (Lebourges 2005). In short, a non-negative least squares inversion is performed for each model with respect to the abundance vector N. After each inversion, the size vector is optimized and a new matrix A is computed based on the new size vector. New inversions are then performed until convergence is reached, or until a maximal number of iterations. After iterating over all the models, the model with the smallest residual is kept if the residual is smaller than a user configured maximal residual.

Computation of residual

Each inversion of the system **A N = c**, gives a best fit solution vector **N'** which is an approximation for **N**. The residual is computed as **|| A N' - c || / || c ||**, where the norm is the Euclidean norm.

Optimization of the size vector

The purpose of the optimization is to redistribute the size vector around the sizes where non-null abundance has been found. The number of sizes in the size vector is kept constant. If the number of sizes is not a multiple of non-null elements, the elements with the smallest size are divided first.

Plankton models

There are five plankton models implemented in the Plankton inversion module.

FluidBentCylinder

This models the backscattering of a fluid bent cylinder, taken from Stanton 1994.

The model has three configurable parameters:

- Reflection coefficient
- Relative standard deviation of length - std dev/L
- Length to width ratio

The bend of the cylinder is not a parameter, as the average echo is independent of the curvature when the distribution of orientations is sufficiently broad.

Bio volume

$V = L * \pi * r^2$
where L is the length of the cylinder and r is the radius.

FluidProlateSpheroid

This models the backscattering of a prolate spheroid filled with a fluid, taken from Stanton 1989.

The model has three configurable parameters:

- g - density body material/density surrounding fluid
- h - compressional speed of sound in body material/compressional speed of sound in surrounding fluid
- beta - Length to width ratio.

Bio volume

$V = \pi /6 * L * d * d$
where L and d the diameters of the sphere.

GaseousSphere

This models the backscattering of a gas filled sphere, taken from Stanton 1989.

The model has two parameters

- g - density body material/density surrounding fluid
- h - compressional speed of sound in body material/compressional speed of sound in surrounding fluid

BioVolume

$V = \pi /6 * L^3$
where L is the diameter of the sphere.

HardShelledSphere

This models the backscattering of a hard shelled sphere, taken from Stanton 1989.

The model has one parameter

- R - The reflection factor.

Bio volume

$V = \pi /6 * L^3$
where L is the diameter of the sphere.

SDWBA

This models the backscattering of a reference size krill, using a parametrization of the Stochastic Distorted Wave Born Approximation (SDWBA), taken from Conti 2006.

The model takes one parameter.

- SDWBA coeff set - The parametrization of the SDWBA is given in terms of the distribution of krill orientations N(theta, sigma), where theta is the mean krill orientation, and sigma is the standard deviation of the distribution of orientations. Coefficients for the distributions presented in the paper above are provided in this implementation.

Bio volume

$V = V_0 * (L/L_0)^3$
where V_0 is the volume of the reference krill, L is the length and L_0 is the length of the reference krill.

School detection module

For a specified channel this module detects regions with pixelstrength higher than a certain threshold and creates datagrams describing them.
See TraceBorderModule for more info.

New datagrams

- RegionInfoDatagram
- RegionBorderDatagram

RegionInfoDatagrams are created when the end of a region is reached and the region values are within the legal intervals specified in the module. RegionBorderDatagrams are created so that they describe the borders of all regions with values higher than the specified threshold independent of the regions values. However the RegionBorderDatagrams contain a boolean flag which is set to true if it would be the case that if the region ended at that datagram a RegionInfoDatagram would be created, and set to false otherwise. This is visualised by filling the regionBorderDatagrams that have the flag on, and only outlining the regionBorderDatagrams with the flag off. RegionInfoDatagrams are visualised by a horizontal line starting at the highest depth of the region to the lowest depth of the region.

This module encapsulates a module setup that looks like this:
Fill missing data module -> Bottom detection module-> Combination module -> Remove bottom module -> Median module*3 -> Erode module -> Dilate module -> TraceBorder module * #thresholds -> Combination remover module
The setup is loaded from cfgSchool.xml

The threshold value is set in the Thresholds parameter. If one wants to detect several thresholds then separate them with comma. The TraceBorderModule is instantiated as many times as there are thresholds, this is invisible for the user.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`ProcessLast:`** If selected, the last channel will be used for region detection. The Channel setting is in this case ignored.
- **`Channel:`** What channel module will work on.
- **`Blind zone depth:`** Depth of blind zone below surface [m].
- **`Thresholds:`** Minimum value of all pixels inside a region, comma separated [dB].
- **`Density:`** Density range of region; This range is compared with the average echostrength for the region (Integrated echostrength in whole region / Pixels in region).
- **`Length:`** Minimum length of region in meters must be in this range.
- **`Thickness:`** Minimum thickness of region in meters must be in this range.
- **`Area:`** Area of region in square meter.
- **`Overlap:`** Overlap of distribution with values outside region.

School filtering module

Filters away regionInfoDatagrams that fall outside of the ranges specified in the module so that they are not visualised with a horizontal line, the module also has options for printing the data in the regionInfoDatagram. The module does not actually remove the regionInfoDatagrams that are outside the ranges, it only sets a flag in the datagram to false. The visualiser does not show regionInfoDatagrams that have this flag set to false. Originally all regionInfoDatagrams created by the School detection module have their flags set to true.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Thresholds:`** What threshold datagram to filter.
- **`Density:`** Density range of region; This range is compared with the average echostrength for the region (Integrated echostrength in whole region / Pixels in region).
- **`Length:`** Minimum length of region in meters.
- **`Thickness:`** Minimum thickness of region in meters.
- **`Area:`** Area of region in square meter.
- **`Overlap:`** Overlap of distribution with values outside region.
- **`PrintInfo:`** Prints out info about the regionEnd datagram.
- **`PrintHistogram:`** Prints out histogram for region.

Trace border module

Detects regions with special properties and creates datagrams describing them.

New datagrams

- RegionInfoDatagram
- RegionBorderDatagram

Using this module alone won't work, it will disable itself automatically after writing a warning, use the School detection module instead.

A region is defined as a closed, connected area where all values are equal to or above a certain threshold, all values outside the region are below this threshold. A region is allowed to have holes in it. An inner boundary(border) of a region is the set of pixels within the region that have one or more neighbors outside the region.

The TraceBorderModule finds regions by tracing their inner boundaries with respect to a threshold value. For each region different values for the region are calculated (RegionInfoDatagram). 'Interesting' regions can be separated from 'uninteresting' regions by means of the region values. RegionInfoDatagrams are only created if the region values are within the range specified in the module.

Requirements

This module requires modules in front of it in this order where XX means any module:

Fill missing data module -> XX -> Bottom detection module-> Remove bottom module -> XX -> (maybe some smoothing, a quick solution is 3* Median module) -> Erode module -> Dilate module -> TraceBorderModule*REP

If REP > 1, the thresholds of the TraceBorderModules must be in increasing order.

Parameters

- **`Active:`** If the module is not active, all datagrams pass unaltered through it.
- **`LogarithmicValues:`** Use log(sv) or sv.
- **`ShowDetails:`** Shows more detailed settings (OnlyLast,AutomaticDepthRange,StartDepth,EndDepth).
- **`OnlyLast:`** When true, only last channel is processed, if false all channels are processed.
- **`AutomaticDepthRange:`** If true and Bottom detection module is used then the range goes to the detected seabed, if true and Bottom detection module is not used the range goes to max depth of the ping.
- **`StartDepth:`** Start depth in meter for filter.
- **`EndDepth:`** End depth in meter for filter.
- **`Channel:`** What channel module will work on.

'Satisfaction values':

- **`Density:`** Density range of region; This range is compared with the average echostrength for the region (Integrated echostrength in whole region / Pixels in region).
- **`Length:`** Minimum length of region in meters.
- **`Thickness:`** Minimum thickness of region in meters.
- **`Area:`** Area of region in square meter.
- **`Overlap:`** Overlap of distribution with values outside region.
- **`Threshold:`** Minimum value of all pixels inside a region.

Smoother module

Performs smoothing by convolution.

Requirements

- Depth datagrams, Dep0Datagram, see Bottom detection module, if parameter `MaskBottom` is selected.
- Noise datagrams, Nqp0Datagram, see NoiseQuantificationModule, if parameter `MaskNoise` is selected.
- Region border datagrams, RegionBorderDatagram, see School detection module, if parameter `MaskRegion` is selected.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`OnlyLastChannel`**: If selected only the last channel is processed, otherwise all channels are processed.
- **`MaskPelagic`**: Exclude samples above bottom.
- **`MaskBottom`**: Exclude samples below bottom.
- **`MaskSecondBottom`**: Exclude samples below 10m above second bottom echo.
- **`MaskNoise`**: Exclude samples below noise threshold.
- **`MaskRegion`**: Exclude samples in connection with regions.
  Allowed values: `{ none, inside, outside }`
- **`MaskTrack`**: Exclude samples in connection with tracks.
  Allowed values: `{ none, inside, outside }`
- **`MinPing`**: Number of pings used ≥ 2*MinPing + 1.
  Allowed values: `MinPing` ≥ 0
- **`MaxPing`**: Number of pings used ≤ 2*MaxPing + 1.
  Allowed values: `MaxPing` ≥ 0
- **`HorizontalKernelType`**: Convolution kernel type horizontally.
  Allowed values: `{ gaussian, tophat }`
- **`VerticalKernelType`**: Convolution kernel type vertically.
  Allowed values: `{ gaussian, tophat }`
- **`HorizontalWidth`**: Convolution kernel width in meters.
  Allowed values: `HorizontalWidth` > 0.0
- **`VerticalWidth`**: Convolution kernel height in meters.
  Allowed values: `VerticalWidth` > 0.0
- **`LogarithmicValues`**: If checked use log(sv), otherwise use (linear) sv.

Towfish module

Merges towfish data with main echosounder data.

For each ping in the input data, the module searches for a corresponding ping in the towfish data. For a corresponding towfish ping to be found, the following criteria must be met:

- A towfish sensor data file folder must be specified
- There must exist towfish sensor data, and the time of the input ping must correspond to sensor data for that time. The sensor data must be on one of the formats specified in sensor data section
- There must exist towfish raw data for the towfish corresponding to the input ping time

If a towfish ping is found, the data from the towfish ping is merged with the input data.

The module attempts to use the data for cable length and depth of the towfish (from the sensor data) to find a matching towfish ping for a given input ping. To do this, there must be a relationship between the cable length *L* and the towfish depth, *D* on one hand and the distance behind the ship on the other hand. Two types of functions are available:

- Pythagoras: The distance behind the ship is given as $sqrt(L^2 - D^2)$
- Straight down: The distance behind the ship is *0*

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Towfish blind zone:`** The blind zone of the towfish vessel. The samples in the blind zone are excluded when merging the towfish data with input data.
- **`Time to add to the towfish echosounder data:`** If there is a known time offset between the towfish echosounder data and the towfish sensor data and the ship mounted echosounder data, the offset can be entered here.
- **`Distance behind function:`** The relationship between the cable length and towfish depth on the one side and the distance behind the ship on the other side.

Sensor data

Currently three types of towfish sensor data can be read by the towfish module. The expected format is decided by the name of the directory where the sensor data file is found. The acoustic data from the towfish should be placed in the configured Towfish directory, and the sensor data directory should be a subdirectory to the Towfish directory.

The following naming conventions for the subdirectory are used:

- Consys: `Consys-metadata`
- Messor: `Messor-metadata`
- Must: `Must-metadata`

The formats and the sensor data directory name are explained below.

Telegram based data

Sensor data directory name: Consys-metadata. In this format the data comes as telegrams in a flat text file. Each line in the file starts with the time and date in the following format:

```
dd.MM.yyyy,HH:mm:ss
```

The relevant telegrams are:

- Pressure: `Pressure = 110.300 dB Temperature = 10.2 dgr C`, where pressure is in decibars and temperature is in degrees Celsius

- Cable length: `L=102.9m S=13.4m/m`, where L is cable length [meters] and S is winch speed [meters/minute]
- Ship vessel log: `$EKDEA,D2,19360507,1823.9,6,1,0,*5c`, where the third parameter is the time/data and the fourth parameter is the vessel log in nmi

Line based data

In this format each line is a complete set of parameters for the towfish. The first line should be headers giving the name of the columns. For all following lines, each line starts with the time and date, and contains vessel log and depth. All numbers use . as decimal character. There are currently two variants of line based data available based on sensor data files from two different tow-bodies (MUST and MESSOR).

Sensor data directory name: Messor-metadata. The relevant columns of MESSOR separated by tab are:

- Column 0 - Date and time: format `ddMMyyyyHHmmss`
- Column tag "EK_Log-Dist" - Vessel log: The vessel log in nmi
- Column tag "CTD_Depth(m)" - towfish depth: The depth of the towfish in meters

Sensor data directory name: Must-metadata. The relevant columns of MUST separated by " , " (space-comma-space) are:

- Column 0 - Date and time: format `yyyyMMddHHmmssSS`
- Column tag "ShipER60VesselLog" - Vessel log: The vessel log in nmi
- Column tag "CTDDepth" - towfish depth: The depth of the towfish in meters

Tracking module

Tracks single targets. Tracking is performed on a single frequency using a chosen tracker type, in the following steps:

1. target detection
2. target filtering
3. track initialization
4. track association
5. track acceptance/rejection

Settings

General settings

- **`TrackerType`**: The tracker to be used
- **`Frequency [kHz]`**: The frequency on which to do tracking.

Tracker types

The following tracker types are available:

- **`Aggregation`**
- **`SED`**
- **`Broadband`**
- **`Narrowband`**

The choice of the tracker affects the target detection and track initialisation. For more information about different target detection of each tracker type see below. All the other stages of the tracking algorithm are independent of this choice.

Platform settings

The platform settings allow to specify the type of movement the platform undergoes by setting **`PlatformMotionType`** to

- **`Stationary`**,
- **`Floating`**, or
- **`Moving`**.

`Stationary`

This setting assumes that changes in the position and orientation of the platform from ping to ping are negligible. Tracking is performed in the transducer's coordinate system.

`Floating`

This setting assumes that vertical changes in the position of the platform from ping to ping are negligible. Tracking is performed in a local coordinate system, after correcting for heading, pitch, roll and heave.

`Moving`

This setting assumes that changes in the position and orientation of the platform from ping to ping are non-negligible. Tracking is performed in a global coordinate system.

Target detector settings

Common parameters for all trackers

- **`MinTS`**: Minimum TS value to be detected [dB]
- **`MaxGainCompensation`**: Maximum (one way) gain compensation [dB]

In addition to the common settings, the following parameters can be set for **`Aggregation`**, **`Broadband`**, and **`Narrowband`** trackers:

- **`PulseLengthDeterminationLevel`**: Pulse length determination level [dB]
- **`MinEchoLength`**: Minimum echo length (relative to pulse length)
- **`MaxEchoLength`**: Maximum echo length (relative to pulse length)
- **`MaxGainCompensation`**: maximal allowed gain compensation to be applied [dB]
- **`DoPhaseDeviationCheck`**: Check this if phase deviation check should be performed
- **`MaxPhaseDevSteps`**: Max phase deviation [phase steps]

The recommended presets for the target detection settings of the chosen tracker chan be loaded clicking on the **`SetPreset`** button.

Target filtering settings

All targets not satisfying the here set condition are rejected post detection. The settings independent of the chosen tracker.

- **`MaxTS`**: Maximum corrected TS value [dB]
- **`MaxDepth`**: Maximum target depth at peak position [m]
- **`MaxAlongshipAngle`**: Maximum alongship angle [deg]
- **`MaxAthwartshipAngle`**: Maximum athwartship angle [deg]

Track initialization settings

Settings governing the initialisation of new tracks. In this stage close-by targets may also be merged.

`Aggregation`, `Broadband`, and `Narrowband` tracker:

- **`InitiationGateFunction`**: Definition of gate for merging of detected targets
- **`InitiationMinLength`**: Minimal number of targets to be merged for new track

`SED` tracker:

No settings. All filtered detected targets that cannot be associated with an existing track are initialized as new tracks without merging.

For details on the gate function, see below.

Track association settings

Settings governing the construction and termination of tracks. The settings independent of the chosen tracker.

- **`GateFunction`**: Definition of gate for tack association
- **`AlphaBetaEstimator`**: Definition predictor-corrector scheme for track association
- **N_m**: Maximum number of missing pings in a row before termination
- **N_e**: Maximum number of missing samples in a single ping before termination

For further details on gate function see below.

Track acceptance settings

Setting for final acceptance/rejection of a terminated track.

- **`MN`**: Maximum acceptable ratio of missing pings to track length.
- **`TL`**: Minimum acceptable track length.
- **`NL`**: Minimum acceptable ratio of total number of samples to track length.

Background

Tracker types

`Aggregation`

The **`Aggregation`** tracker detects targets in each ping based on target strength, extracting all samples with target strength above the defined minimum (**`MinTS`**), provided their gain compensation is less than **`MaxGainCompensation`**.

`SED`

The **`SED`** tracker detects targets in each ping based on target strength (TS). Targets are defined as connected signal sequences with

- TS values greater or equal a certain minimal threshold (based on **`MaxGainCompensation`** and **`PulseLengthDeterminationLevel`**), and
- the location of the maximal TS is the only local maximum.

All targets fulfilling additional criteria on minimal corrected target strength (**`MinTS`**) and maximal gain compensation (**`MaxGainCompensation`**), in addition to falling in the specified extend range (defined by **`MinEchoLength`** and **`MaxEchoLength`**), are accepted for tracking. Furthermore, target not showing coherent placement with respect the centre of the echo beam (threshold specified by **`MaxPhaseDevSteps`**) may be rejected (enabled by **`DoPhaseDeviationCheck`**).

`Broadband`

The **`Broadband`** tracker detects targets in each ping based on the squared absolute value of the (sector averaged) pulse compressed raw signal, i.e. in linear domain. A sample is said to be a target if

- it is a local maximum,
- it value is larger than the value of other local maxima within a certain number of samples

The threshold for the required separation from other equally strong (or stronger) local maxima is determined using Ridler and Calvard's (1978) classical iterative threshold selection method.

The extension of the target is determined as the sample range spanning from the target sample to the next local maximum at least **`PulseLengthDeterminationLevel`** dB weaker then the than the signal at the peak sample, measured in corrected target strength.

All targets fulfilling additional criteria on minimal corrected target strength at peak (**`MinTS`**) and maximal gain compensation (**`MaxGainCompensation`**), in addition to falling in the specified extend range (defined by **`MinEchoLength`** and **`MaxEchoLength`**), are accepted for tracking. Filtering on phase deviation (enabled by **`DoPhaseDeviationCheck`**) is not recommended for broadband data since angle information is only sensible for main peaks, but not on the side lobes.

`Narrowband`

The **`Narrowband`** tracker has the same behaviour as the **`Broadband`** tracker, apart from

- operating on the (sector averaged) match filtered signal, and
- using the whole signal, not only local maxima for determining the target extent.

The filter used in the matched filtering is the sent pulse. Note that this processing is equivalent to pulse compression for broadband data.

Gating function

The difference between two targets *i* and *j* is calculated as:

$$\varepsilon_{ij} = y_i - y_j$$

where $y_i = [\ \alpha\ \beta\ r\ TS\ ]$, α = alongship angle, β = athwartship angle, r = range, TS = target strength.

The gate function defines a distance between two targets as

$$d(y_i, y_j) = d_{ij} = \varepsilon_{ij}\, G\, \varepsilon_{ij}^T$$

where

$$G = \begin{bmatrix} \alpha_G^2 & 0 & 0 & 0 \\ 0 & \beta_G^2 & 0 & 0 \\ 0 & 0 & r_G^2 & 0 \\ 0 & 0 & 0 & I_G^2 \end{bmatrix}^{-1}$$

For association, a sample is said to be inside the gate if $d_{ij} \leq 1$.

References

Handegard, N.O. (2007). "Observing individual fish behavior in fish aggregations: Tracking in dense aggregations", J. Acoust. Soc. Am, Vol 122(1), pp 177-187

Handegard, N.O., Patel, R., and Hjellvik, V. (2005). "Tracking individual fish from a moving platform using a split-beam transducer", J. Acoust. Soc. Am., Vol 118(4), pp 2210-2223

Ridler, T. W., and S. Calvard (1978). "Picture thresholding using an iterative selection method." IEEE trans. on Systems, Man and Cybernetics, Vol 8(8), pp 630-632.

New datagrams

Tbr0Datagram, Tnf0Datagram, Ttc0Datagram.

Bottom detection module

Module for detecting bottom depth.

Three algorithms are available:

- EK500: An algorithm similar to the one used in the EK500 system.
- Gradient: Finds some bottom candidates and gives them points according to signal strength, distance from transducer, and distance from previously found bottom.
- Threshold: Returns the depth where for the first time the signal strength is above a pre-specified threshold.

Finding a channel dependent bottom depth

The algorithms above finds one bottom depth per ping common for all channels. The module then also finds a bottom depth for each of the channels according to the algorithm below:

For all channels:

1. Find the sample number corresponding to the detected bottom depth.
2. Create a moving average filter with a length of about 0.2 m. If the sample distance is above 0.1 m, a moving average filter of length 1 is used.
3. From the sample number found in 1., find the actual peak by using the moving average filter, and moving maximally 10 m in each direction.
4. Back-stepping: From the peak value step in the direction of the transducer until the value (according to the moving average filter) is less than **`MinDepthValueFraction`** of the peak value found in 3. The depth of this value is the channel dependent bottom depth.

Coordinated bottom

The following algorithm is used for finding the coordinated bottom:

1. Use the bottom depths detected on all frequencies higher than or equal to **`Minimum kHz`** and less than or equal to **`Maximum kHz`**.
2. Find the maximum depth.
3. Clamp the remaining depths to be no less than than **`MinDepthLimit`** above than the maximum depth.
4. Reject the depths less than **`MinimumDepthThresholdFactor`** times the maximum depth.
5. The coordinated depth is the smallest of the remaining depths.

The bottom datagrams, BOT0, that may accompany the raw data files are used to detect the case that no bottom can be found. If the accompanied bottom datagrams exist and have detected zero-depth for all channels, this is used as an indication that there is no detectable bottom. In this case the coordinated depth is set to the maximum range of the ping.

The coordinated bottom is used in other preprocessing modules.

This module re-detects the bottom depth every time it is used in the processing setup. Only the coordinated bottom depth is used in the other preprocessing modules. The channel dependent bottom depths are written to a separate file which can be read and used by LSSS.

New datagrams

- Depth datagrams, Dep0Datagram. These datagrams (one per ping) represent the coordinated depth.
- Bottom datagrams, Bot0Datagram. These datagrams (one per ping) represent the channel dependent depths (contains one depth per channel).

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Algorithm:`** Specifies the algorithm to be used in the bottom calculation.
  Allowed values: `{ EK500, Gradient, Threshold }`
- **`MinDepthLimit:`** Specifies the maximal distance above bottom accepted as the minimal bottom distance.
- **`MinDepthValueFraction:`** For the minimal bottom calculation. The module backsteps from the depth maximal bottom echo strength, until the echo strength is less than this fraction of the maximal bottom echo strength.
- **`SignalStrengthThreshold:`** The detected bottom must have signal strength larger than this value.
- **`MinimumDepthThresholdFactor:`** Shallowest acceptable depth relative to max depth.
- **`MaxRangeFactor:`** Factor to multiply with given transducer ranges for ranges used in depth detection.
- **`Always detect bottom:`** Always try to detect bottom even if echosounder has zero bottom at all frequencies.
- **`Minimum bottom depth:`** The detected bottom must be minimum this value.
- **`Maximum bottom depth:`** The detected bottom must be maximum this value.
- **`Minimum kHz:`** Only use channels with at least this frequency.
- **`Maximum kHz:`** Only use channels with at most this frequency.

Combination module

Module for generating combination echograms.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`FirstOperandKhz`**: Frequency of the first operand.
  Allowed values: `FirstOperandKhz` >= 0
- **`SecondOperandKhz`**: Frequency of the second operand.
  Allowed values: `SecondOperandKhz` >= 0
- **`FirstOperandChannel`**: Channel of the first operand (used if `FirstOperandKhz` = 0).
  Allowed values: `FirstOperandChannel` >= 1
- **`SecondOperandChannel`**: Channel of the second operand (used if `SecondOperandKhz` = 0).
  Allowed values: `SecondOperandChannel` >= 1
- **`LogarithmicOperands`**: If checked use log(sv), otherwise use (linear) sv.
- **`Operation`**: Specifies how the operands are combined.
  Allowed values: `{ -, +, *, /, mean, harmonic mean, min, max, zero, copy1stOperand }`

Comment module

This module can be used for writing a comment about the module configuration.

Expression module

Creates a new channel defined as an expression of the existing channels.

Parameters

- **`Active`**: If the module is not active then all datagrams pass unaltered through it.
- **`Expression`**: An expression of the (linear) s_V values. Use C1, C2, ..., or F18, F38, ...
  If multiple expressions are specified then the first expression with all variables available is used.
  All functions in java.lang.Math are available.
  Examples:
  C1
  (C1 + C2) / 2
  C1 / C5 + F70 - C4
  C6 * log10(C2) / pow(C3, F364)

Plugin module

Uses the Korona API for module computations. The parameter `Implementation` implements the ModuleComputation interface, and is compiled and used directly by Korona. This provides a simple way to getting started and experimenting with the API.

Details about the Korona API can be found in the Javadoc.

Parameters

- `Active`: If the module is not active then all datagrams pass unaltered through it.
- `Description`: A description of what the implementation does.
- `Example`: Example implementations.
- `Implementation`: An implementation of the ModuleComputation interface.

Temporary computations begin module

This module marks the start of temporary computations. For each ping, copies of the Raw0Datagrams are made. Subsequent changes to Raw0Datagrams are reverted by a TemporaryComputationsEndModule. New datagrams, however, from for example seabed detection, region detection or categorization, are kept.

Temporary computations end module

This module marks the end of temporary computations. Changes made to Raw0Datagrams since a previous TemporaryComputationsBeginModule are reverted. New datagrams, however, from for example seabed detection, region detection or categorization, are kept.

Datagrams

Bot0Datagram
Cac0Datagram
Cad0Datagram
Cas0Datagram
Cds0Datagram
Con0Datagram
Dep0Datagram
Idx0Datagram
Nme0Datagram
Nqp0Datagram
Pic0Datagram
Pid0Datagram
Raw0Datagram
RegionBorderDatagram
RegionInfoDatagram
RegionTableOfContentsDatagram
Tbr0Datagram
Tnf0Datagram
Ttc0Datagram
UnknownDatagram

Bot0Datagram

The ER60 bottom datagram.

Occurrences

- In the bot file.

Creator

- ER60.

Contents

- Bottom depth.

Cac0Datagram

Categorization configuration datagram. Identifies all categories referred to later by categorization datagrams, Cad0Datagram.

Occurrences

- Once only, after Con0Datagram and before the first Raw0Datagram.

Creator

- CategorizationModule.

Contents

- **`NumberOfCategories`**: The number of categories.
- **`Categories`**: A list of properties for each category:
  - **`CategoryName`**: The name of a category.
  - **`CategoryLegend`**: The legend (short name) of a category.
  - **`CategoryNumber`**: The number of a category.
  - **`CategoryColor`**: The color (24 bit rgb) of a category.

Cad0Datagram

Categorization datagram that contains categorization data for each pixel in each ping. Refers to categories identified in the initial categorization configuration datagram, Cac0Datagram.

Occurrences

- Once per ping.

Creator

- CategorizationModule.

Contents

- **`NCategories`**: The number of categories included for each pixel.
- **`Count`**: The number of pixels, which is the same as the Raw0Datagram for the reference channel.
- **`SampleDistance`**: Vertical distance [m] between pixels.
- **`FirstDepth`**: Depth [m] of the first pixel.
- **`Categories`**: Array of category numbers that loops over all included categories and all pixels. The innermost loop is over the `Count` pixels. If a pixel has fewer than `NCategories` categories, the unknown category is used for remaining entries.
- **`Discriminants`**: Array of discriminant values. Same ordering as for `Categories`.
- **`Probabilities`**: Array of probabilities. Same ordering as for `Categories`.

Cas0Datagram

Result of school categorization. Refers to categories identified in the initial categorization configuration datagram, Cac0Datagram. The datagram keeps a reference to the RegionInfoDatagram it was created from.

Occurrences

- Once per region which has not been identified as multi-species.

Creator

- SchoolCategorizationModule.

Contents

- **`NCategories`**: The number of categories included for each region.
- **`RegionID`**: The id of the corresponding the RegionInfoDatagram.
- **`Categories`**: Array of category numbers that loops over all included categories.
- **`Discriminants`**: Array of discriminant values. Same ordering as for `Categories`.
- **`Probabilities`**: Array of probabilities. Same ordering as for `Categories`.

Cds0Datagram

Datagram for module configuration. Contains complete description of the module configuration used when processing the file which this datagram is read from.

Occurrences

- Once only, after Con0Datagram and before the first Raw0Datagram.

Creator

- WriterModule.

Contents

- **`ModuleConfiguration:`** Module configuration as XML.

Con0Datagram

The EK60 echo sounder configuration datagram.

Occurrences

- Initially, as the first datagram.

Creator

- EK60.

Contents

- Echo sounder configuration.

Dep0Datagram

Datagram for depth data.

Occurrences

- Once per ping.

Creator

- Bottom detection module.

Contents

- **`Depth`**: The detected bottom depth.
- **`MinimumDepth`**: The minimum bottom depth, see Bottom detection module.

Idx0Datagram

The ER60 index datagram.

Occurrences

- In the index file.

Creator

- ER60.

Contents

- Ping number, vessel distance, latitude, longitude, file offset.

Nme0Datagram

NMEA datagram.

Occurrences

- Unspecified.

Creator

- EK60.

Contents

- Positioning data.

Nqp0Datagram

Datagram for Noise Quantification Parameters. Contains parameters that quantify the noise level for a channel.

Occurrences

- Once per ping.

Creator

- NoiseQuantificationModule.

Contents

- **`Channel`**: The channel to which this datagram corresponds.
- **`Average`**: Average noise level, N_E.
- **`UpperLimit`**: Upper limit for noise values, N_H.
- **`Quality`**: The quality of the noise parameters, ranging from 0 to 100.

Pic0Datagram

Plankton inversion configuration datagram. Defines all plankton categories to be referred to later in the plankton inversion datagrams Pid0Datagram.

Occurrences

- Once only, after Con0Datagram and before the first Raw0Datagram.

Contents

```
Pic0Datagram // "PIC"=Plankton Inversion Configuration
{
   int numberOfCategories;
   PlanktonCategory[] planktonCategories;
}
PlanktonCategory
{
   string name;
   string legend;
   byte number;
   unsigned byte red;
   unsigned byte green;
   unsigned byte blue;
}
```

Pid0Datagram

Plankton inversion datagram that contains plankton category and associated length distribution of the plankton.

Occurrences

- Once per ping.

Contents

```
Pid0Datagram // "PID"=Plankton Inversion Datagram
{
   int count; // The number of inverted samples.
   float sampleDistance; // The distance between the samples [m].
   float firstDepth; // The depth of the first sample [m].
   int numberOfSamples; // The number of stored planktonSamples objects.
   PlanktonSample[] planktonSamples;
}
PlanktonSample
{
   int startSampleNumber; // The first sample number in the original RAW datagram this plankt
   int numberOfPlanktonDatas; //The number of plankton datas for this sample.
   PlanktonData[] planktonDatas; // Should be numberOfPlanktonDatas of these.
}
PlanktonData
{
   byte planktonNumber;
   float fraction; // The fraction of the s_A this data unit represents.
   float residual; // The computed residual from the inversion algorithm.
   int numberOfBinsInLengthDistribution;
   LengthDistribution lengthDistribution; // Only present if numberOfBinsInLengthDistributio
}
LengthDistribution
{
   float[] binDividerPositions; // Should be numberOfBins + 1 of these.
   float[] abundanceForBins; // Should be numberOfBins of these.
}
```

Raw0Datagram

The EK60 echo sounder sample datagram.

Occurrences

- Once per ping per channel.

Creator

- EK60.

Contents

- Sample data.

RegionBorderDatagram

Datagram for regionborder data. This datagram describes where the regions start and stop for one ping thereby describing a vertical slice of a region. It contains a list of `BorderInfo` objects describing all regions cross-sections with this ping.

Occurrences

- Irregularly: from none per ping to as many per ping as there have been TraceBorderModules in front of the stream.

Creator

- TraceBorderModule, also School detection module since it contains one or more TraceBorderModules.

Contents

- **`Channel`:** On what channel this regionslice was detected.
- **`Threshold`:** The threshold for the regionslice.
- **`BorderSize`:** Number of regionslices (twice the number of borders) for this ping.
- **`BorderInfo`:** A list of `BorderInfo` objects, each being a description of a region's cross-section with this ping, the objects are in increasing order with respect to start and end values: if BorderInfo a is before BorderInfo b then a.mEnd < b.mStart.

`BorderInfo` object

Describes a region's position in a ping in terms of its start index and end index, also contains its id and its 'average pixelvalue': (sum of Sv values in region so far) / (sum of pixels in region so far).

Contents

- **`mID`:** Id of the region cross section, is guaranteed to be unique among all currently open regions within one program session.
- **`mStart`:** Start index of the region, the region cross section is [mStart,mEnd].
- **`mEnd`:** End index of the region, the region cross section is [mStart,mEnd].
- **`mProbability`:** (sum of Sv values in region so far) / (sum of pixels in region so far), i.e. the average strength of a pixel in the region up till now.

Comments

- Start and end values within one RegionBorderDatagram will never overlap each other.
- If there are multiple RegionBorderDatagrams in a ping (created by multiple TraceBorderModules) and all were traced on the same channel then the regionslices for higher thresholds will be contained inside regionslices for lower thresholds.
- Any interval from any two datagrams on same channel and ping will either be completely disjoint, equal or one will be contained inside the other, they will never 'partially overlap'.

RegionInfoDatagram

Datagram for region data. This datagram describes different properties of a region and comes at the end of the region.

Occurrences

- Irregularly: Pings can have none, one or many of this datagram.

Creator

- TraceBorderModule, also School detection module since it contains one or more TraceBorderModules.

Contents

- **`Channel:`** On what channel the region was detected.
- **`Threshold:`** The threshold for the region.
- **`IDs:`** Ids of all borders in the region.
- **`Area:`** Area of region in square meter.
- **`PixelArea:`** Area of region in pixels.
- **`SumLogSv:`** Sum of LogSv parameter in the area.
- **`LogMeanSv:`** Korona of Mean Sv in the area.
- **`Sa:`** Sa of region.
- **`Perimeter:`** Perimeter of region in meter.
- **`Length:`** Length of region in meter.
- **`MaxHeight:`** Maximum height of region in meter.
- **`Rect:`** A rectangle tightly surrounding the region, in pixels.
- **`MeanRegion:`** Mean LogSv values just on the inside of the region.
- **`StdDevRegion:`** Standard deviation for region values.
- **`MeanOutOfRegion:`** Mean LogSv values just on the outside of the region.
- **`StdDevOutOfRegion:`** Standard deviation for values just outside region.
- **`Crossing:`** Crosspoint (between the mean values) of the two distributions.
- **`Overlap:`** Percent overlap between the gauss curves of the two distributions.
- **`Histogram:`** A histogram over the values inside the region.
- **`HistogramStartValue:`** Min pixelvalue in region, values of the first bin are in [HistogramStartValue,HistogramStartValue+HistogramStep).
- **`HistogramEndValue:`** Max pixelvalue in region, values of the last bin are in [HistogramEndValue,HistogramEndValue+HistogramStep).
- **`HistogramMaxIndex:`** Last index of the Histogram array.
- **`HistogramStep:`** Binsize (stepsize between the bins in the histogram).
- **`Trace of region:`** Consists of a list of points giving the regions outer trace. First the number of points is given followed by the first points x value, the first points y value, and so on for each point. Later the datagram might contain the regions inside the outer region to be subtracted from the outer region so to represent holes in the outer region. The last value of this datagram specifies the number of such regions it describes and is currently 0 as default.

Comments

- Stepsize of histogram can be set in the code by changing `Region.Statistics.Histogram.mHistStep`.
- The values MeanOutOfRegion,StdDevOutOfRegion,Crossing and Overlap requires knowledge of the area just around the detected region to be calculated.

Here is a description of how these values are sampled:

- The band around the region for which out-of-region values are sampled from starts

`Region.OUTREGIONJUMP` meters out in y direction from the border of the region and is `Region.BANDTHICKNESS` meters thick in y direction.

- If thickness of regionslice is less than `BANDTHICKNESS`, no sampling will occur because the region is too thin.

RegionTableOfContentsDatagram

Datagram with references to RegionInfoDatagrams.

Occurrences

- In the last ping of a file. Possibly more than one.

Creator

- TraceBorderModule, also School detection module since it contains one or more TraceBorderModules.

Contents

- A list of times in NTDate format that point to pings containing RegionInfoDatagrams.

Comments

- The same time may be repeated if it refers to a ping with more than one RegionInfoDatagram.

Tbr0Datagram

Tracking info per ping.

Creator

- TrackingModule.

Tnf0Datagram

Tracking info per track.

Creator

- TrackingModule.

Ttc0Datagram

Tracking table of contents.

Creator

- TrackingModule.

UnknownDatagram

Class for unknown datagrams. All datagrams not recognized by the system are represented by this datagram.

Reference

This part contains reference material.

Literature references

Demer, A. D., and Conti, S. G. (2005). New target-strength model indicates more krill in the S
ICES J. Mar. Sci., 62: 25 - 32

Lawson C.L., Hanson R.J. (1974). Solving least squares problems. Prentice-Hall, New Jersey.

Lebourges A. (2005). Plankton/Fish Discrimination and Zooplankton Classification. Chapter 6 i
Species Identification Methods From Acoustic Multi-frequency Information. Contract number (

Stanton T.K. (1989). Simple approximate formulas for backscattering of sound by spherical
and elongated objects. J. Acoust. Soc. Am., 86, 1499-1510.

Stanton T.K., Wiebe P.H., Chu D., Benfield M.C., Scanlon L., Martin L., Eastwood R.L. (1994).
On acoustic estimates of zooplankton biomass. ICES J. Mar. Sci., 51, 505-512.

Conti S.G, Demer D.A. (2006). Improved parameterization of the SDWBA for estimating krill tar
ICES J. Mar. Sci., 63, 928-935.

EK80

This section contains details about data produced by the Simrad EK80 echo sounder.

EK80 broadband data

This section contains details about the processing of broadband data produced by the Simrad EK80 echo sounder.

Computation of S_v

Let y_i be the complex sample value of sector $i \in [1, n]$.
Let $y = (1/n) \sum y_i$ be the average value over all sectors and s the (ideal) transmitted chirp signal.
Let furthermore $y_{pc} = y * \bar{s}_{-1}$ be the pulse compressed received signal, where * denotes the convolution of two signals and $\bar{}_{-1}$ the combined conjugation and time reverse operator.
Let $p = \kappa |y_{pc}|^2$, with $\kappa = (n / 8) \left((R_{wbt,rx} + Z_{trd}) / R_{wbt,rx} \right)^2 / Z_{trd}$.
Then

$$S_v = 10 \log_{10}(p) + C_{S_v} + TVG_{S_v}$$

where

$$C_{S_v} = -2 G_{f_c} - 2 S_{a,corr} - \Psi_{f_c} - 10 \log_{10}\left((p_{tx} \lambda^2 c \tau_{eff}) / (32 \pi^2) \right)$$

- G_{f_c} = Gain at Centre frequency
- $S_{a,corr}$ = S_a correction
- Ψ_{f_c} = Equivalent beam angle at Centre frequency
- p_{tx} = Transmit power
- $\lambda = c / f_c$
- c = Sound speed
- f_c = Centre frequency
- τ_{eff} = Effective pulse duration

$$TVG_{S_v} = 20 \log_{10}(r_{TVG}) + 2 \alpha r_{TVG}$$

- r_{TVG} = Sample range - $\tau c / 4$
- α = Absorption coefficient
- τ = Pulse duration

Computation of TS

The target strength is derived from S_v using the following formula

$$TS_u = 10 \log10(p) + C_{TS} + TVG_{TS}$$

for uncorrected target strength, where

$$C_{TS} = -2 G_{f_c} + 40 \log_{10}(r_{TVG}) - 10 \log_{10}\left((p_{tx} \lambda^2 \tau_{eff}) / (16 \pi^2 \tau) \right)$$

$$TVG_{TS} = 40 \log_{10}(r_{TVG}) + 2 \alpha r_{TVG}$$

and

$$TS_c = TS_u + 20 \log_{10}2 \left(\rho_\xi + \rho_\chi - 0.18 \rho_\xi \rho_\chi \right)$$

for beam position corrected target strength, where

- ξ = along angle
- χ = athwart angle
- $\rho_\xi = (2 \xi / bw_{\xi,f_c})^2$
- $\rho_\chi = (2 \chi / bw_{\chi,f_c})^2$
- bw_{ξ,f_c} = along beam width at centre frequency
- bw_{χ,f_c} = athwart beam width at centre frequency

and remaining symbols as already defined

Computation of $S_V(f)$

Let $\left((y_{pc})_\ell\right)_\ell$ be a signal segment with $\ell = \ell_0,\ldots,\ell_n$ the sample index of the signal in a single ping and $m = \ell_n - \ell_0$. Let furthermore s_{pc} be the pulse compressed (or autocorrelated) transmitted chirp signal as above and $\mathcal{F}$ the discrete Fourier transform (DFT) operator.
Then, the S_V-spectrum $S_V(f)$ is computed as follows

$$\begin{aligned} S_V(f_k) &= 10 \log_{10} \left(\kappa |\mathcal{F}\{y_{pc}\}_k / \mathcal{F}\{s_{pc}\}_k|^2 \right) \\ &\quad - 10 \log_{10}\left((p_{tx} c \lambda_{f_k}^2) / (32 \pi^2) \right) \\ &\quad - 10 \log_{10}(m T) \\ &\quad + 2 r \alpha_{f_k,d} - 2 G_{f_k} - \Psi_{f_k} \end{aligned}$$

with

$f_k = k^{th}$ DFT frequency
$\lambda_{f_k} = c / f_k$, wavelength
T = sampling interval
r = range of sample corresponding to signal segment midpoint
d = depth of sample corresponding to signal segment midpoint
$\alpha_{f_k,d}$ = absorption coefficient at f_k and d
G_{f_k} = Gain at f_k
Ψ_{f_k} = Equivalent beam angle at f_k

and remaining symbols as already defined.

Computation of TS(f)

Let $\left((y_{pc})_\ell\right)_\ell$ again be a signal segment with $\ell = \ell_0,\ldots,\ell_n$ the sample index of the signal in a single ping and s_{pc} and $\mathcal{F}$ as above.
Then, the TS-spectrum TS(f) is computed as follows

$$\begin{aligned} TS(f_k) &= 10 \log_{10} \left(\kappa |\mathcal{F}\{y_{pc}\}_k / \mathcal{F}\{s_{pc}\}_k|^2 \right) \\ &\quad - 10 \log_{10}\left((p_{tx} \lambda_{f_k}^2) / (16 \pi^2) \right) \\ &\quad + 40 \log_{10}(r_p) \\ &\quad + 2 r_p \alpha_{f_k,d_p} - 2 G_{f_k} - \Psi_{f_k} \\ &\quad + 20 \log_{10} 2 \left(\rho_{\xi_p} + \rho_{\chi_p} - 0.18 \rho_{\xi_p} \rho_{\chi_p} \right) \end{aligned}$$

with

ξ_p = along angle of sample corresponding to peak of signal
χ_p = athwart angle of sample corresponding to peak of signal
$\rho_{\xi_p} = (2 \xi_p / bw_{\xi,f_c})^2$
$\rho_{\chi_p} = (2 \chi_p / bw_{\chi,f_c})^2$
r_p = range of sample corresponding to peak of signal
d_p = depth of sample corresponding to peak of signal
α_{f_k,d_p} = absorption coefficient at f_k and d_p

and remaining symbols as already defined.

EK80 narrowband data

This section contains details about the processing of narrowband data produced by the Simrad EK80 echo sounder.

Computation of S_V

Let y_i be the complex sample value of sector $i \in [1, n]$.
Let $y = (1/n) \sum y_i$ be the average value over all sectors.
Let $p = \kappa |y|^2$, with $\kappa = (n / 8) \left((R_{wbt,rx} + Z_{trd}) / R_{wbt,rx} \right)^2 / Z_{trd}$.
Then

$S_V = 10 \log_{10}(p) + \text{constant} + TVG$

where

$\text{constant} = -2\,G - 2\,S_{a,corr} - \Psi - 10 \log_{10}\left((p_{tx}\,\lambda^2\,c\,\tau_{eff}) / (32\,\pi^2) \right)$

- G = Gain
- $S_{a,corr}$ = S_a correction
- Ψ = Equivalent beam angle
- p_{tx} = Transmit power
- $\lambda = c / f$
- c = Sound speed
- f = Frequency
- τ_{eff} = Effective pulse duration

$TVG = 20 \log_{10}(r_{TVG}) + 2\,\alpha\,r_{TVG}$

- r_{TVG} = Sample range - $\tau\,c / 4$
- α = Absorption coefficient
- τ = Pulse duration

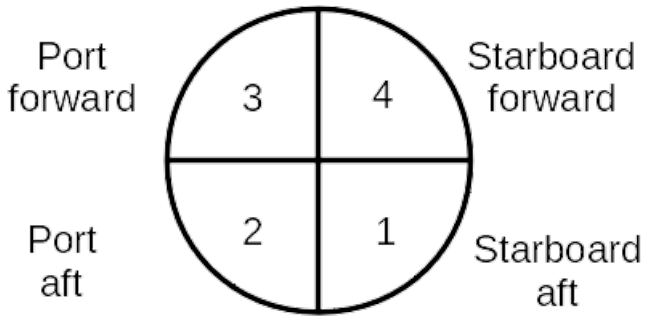
Computation of split beam angles

The complex samples values are used directly for computing split beam angles.

EK80 split beam angles

This section contains details about computing split beam angles.

Four sectors



Let y_i be the complex value for sector i, and let

$y_{forward} = (y_3 + y_4) / 2$
$y_{aft} = (y_2 + y_1) / 2$
$y_{starboard} = (y_4 + y_1) / 2$
$y_{port} = (y_3 + y_2) / 2$

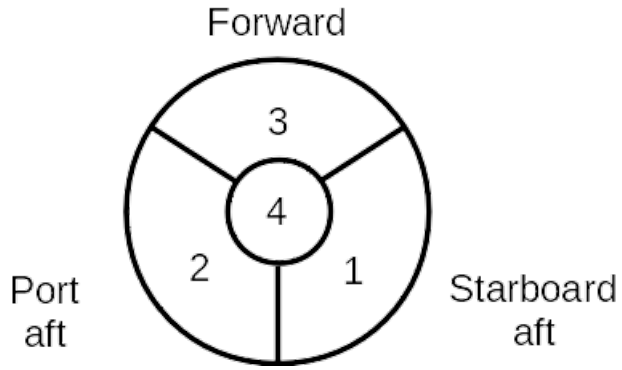
Then

Alongship angle = $1/as \angle(y_{forward} \overline{y_{aft}}) - ao$
Athwartship angle = $1/as \angle(y_{starboard} \overline{y_{port}}) - ao$

where

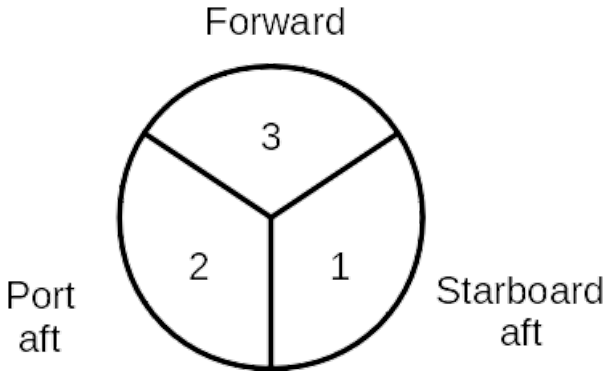
$\angle$ denotes the argument of a complex number
$\overline{\ }$ denotes the complex conjugate operator
as = Angle sensitivity
ao = Angle offset

Three sectors and center



The center sector is added to each of the three other sectors. Then the angles are computed in the same way as for three sectors.

Three sectors



Let y_i be the complex value for sector i, and let

$\phi_{31} = \angle(y_3 \bar{y}_1)$

$\phi_{32} = \angle(y_3 \bar{y}_2)$

where

$\angle$ denotes the argument of a complex number

$\bar{}$ denotes the complex conjugate operator

Then

Alongship angle = $1/(\sqrt{3}\ as)\ (\phi_{31} + \phi_{32})$ - ao

Athwartship angle = $1/as\ (\phi_{32} - \phi_{31})$ - ao

where

as = Angle sensitivity

ao = Angle offset

EK80 transducer impedance

The EK80 system encodes measurements of transmitted (complex) voltage and current for each sector in the beginning of each ping. Thus, it is possible to calculate the impedance per ping/transducer sector i by

$Z_i = |V_i / I_i|$ [Ω]

where V is the transmitted voltage, and I the transmitted current. The numbering of the sectors depends on the transducer and follows the same convention as in the computation of the split beam angles. The impedance is calculated per sample, from the start of each ping, up to the sample corresponding to the end of the sent pulse, yielding an array holding the temporal evolution of impedance during pulse transmission.

For narrowband data, the average of the central part of the array associated with the current sector and ping is used. For broadband data, the impedance at the time of transmission of the center frequency of the used frequency band is used.

Miscellaneous

This section contains miscellaneous reference material.

NMEA

This section contains some details about the NMEA datagrams used.

Index file generation

When idx files are generated, or when a raw file without an idx file is read, these NMEA sentences are used:

- `$GPGGA`: Used for longitude and latitude.
- `$GPGLL`: Used for longitude and latitude, but only if no `$GPGGA` sentences have been read so far.
- `$GPVLW`: Used for vessel distance.
- `$GPVTG`: Speed is used for updating vessel distance, but only if no `$GPVLW` sentences have been read so far.

In some cases when a ping does not contain all necessary NMEA sentences, pings will be buffered while searching for for such sentences. The buffering will be maximum 10 seconds and no more than 100 pings.

- If the first ping does not contain any NMEA sentences with geographical position, then pings will be buffered in search for the next NMEA sentences with geographical position. If found, then that geographical position is used for the first pings, otherwise latitude and longitude are set to 0 which indicates not available.
- If `$GPVLW` sentences have been read so far and a ping does not any `$GPVLW` sentences, then pings will be buffered in search for the next ping with a `$GPVLW` sentence. If found, then the vessel distance will be interpolated linearly, otherwise the vessel distance will remain the same.

Heading

Vessel heading is found through NMEA sentences:

- `$GPHDT`: Used for heading.

If no `$GPHDT` are available for a ping, then heading is interpolated from neighbouring pings.