# The Open Master Hearing Aid (openMHA)

**4.8.1**

## Documentation of openMHA plugins (openMHA)


© 2005-2019 by HörTech gGmbH, Marie-Curie-Str. 2, D–26129 Oldenburg, Germany

**The Open Master Hearing Aid (openMHA) – Documentation of openMHA plugins (openMHA)**

HörTech gGmbH
Marie-Curie-Str. 2
D–26129 Oldenburg

## LICENSE AGREEMENT

This file is part of the HörTech Open Master Hearing Aid (openMHA)
Copyright © 2005 2006 2007 2008 2009 2010 2012 2013 2014 2015 2016 HörTech gGmbH.
Copyright © 2017 2018 2019 HörTech gGmbH.

openMHA is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, version 3 of the License.

openMHA is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License, version 3 for more details.

You should have received a copy of the GNU Affero General Public License, version 3 along with openMHA. If not, see <http://www.gnu.org/licenses/>.

# Contents

© 2005-2019 HörTech gGmbH, Oldenburg

© 2005-2019 HörTech gGmbH, Oldenburg

© 2005-2019 HörTech gGmbH, Oldenburg

# 1 Plugin category 'compression'

## 1.1 dc

dynamic compression

### 1.1.1 Detailed description

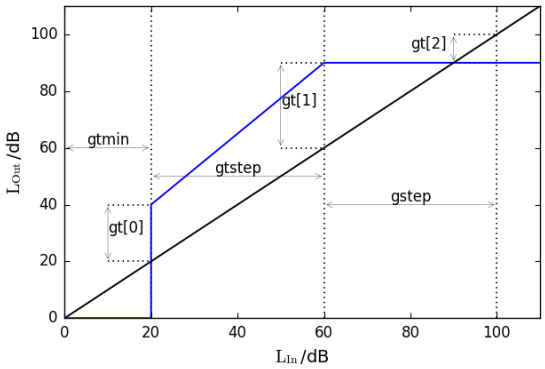


**Figure 1 Input-output function of one channel in the dc dynamic compression algorithm.**

The plugin *dc* is a multiband dynamic range compression plugin. One compression function (input-output function) is applied to each audio channel. Frequency-dependent compression can be achieved by using the *fftfilterbank* plugin in conjunction with this plugin. The input-level dependent gain function is determined by a gain table containing the gain values applied in different channels and frequency bands. Between the points of the gain table linear interpolation is used. For gains outside of the range of the gaintable a linear extrapolation based on the two nearest points is used. If spectral processing is used, the input level (abscissa of the input-output function, cf. Fig. 1) is determined by an attack- and release-filter of the short time RMS level $L_{st}$, given in dB (SPL). The attack filter $L_a$ is a first order low pass filter. The release filter $L_{in}$ is a maximum tracker, i.e.

$$L_a = \langle 20 \log_{10}(L_{st}) \rangle_{\tau_{attack}} \quad (1)$$

$$L_{in} = \max(L_a, \langle L_a \rangle_{\tau_{release}}) \quad (2)$$

The gain table is given as a matrix with $n_f \cdot n_{ch}$ rows, where $n_f$ is the number of frequency bands and $n_{ch}$ is the number of channels. The order of row indices is $0...n_f...n_f \cdot n_{ch}$. The x-values for the n-th column of the gain table are given as $x_n = gtmin + gtstep \cdot n$. See also Fig. 1. A configuration fragment reproducing Fig. 1 could be:

```
algos = [fftfilterbank dc combinechannels]
fftfilterbank.ftype = center
fftfilterbank.f = [200 2000]
fftfilterbank.ovltype = rect
fftfilterbank.fscale = bark
dc.gtmin=[20 20]
dc.gtstep=[40 40]
```

© 2005-2019 HörTech gGmbH, Oldenburg

```
dc.gtdata = [[40 30 -10];[40 30 -10];]
dc.tau_attack = [0.005 0.005]
dc.tau_rmslev = [0.005 0.005]
dc.tau_decay = [0.015 0.015]
dc.combinechannels.name = fftfilterbank_channels
```

In this configuration it is assumed that one audio channel is configured. All variables of *dc* have two entries, one for each frequency band.

#### 1.1.2 Supported domains

The MHA plugin `dc` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

#### 1.1.3 Plugin Tags

*compression* level-modification

#### 1.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| gtdata | matrix<float> | gaintable data in dB gains | [[]] |
| gtmin | vector<float> | input level for first gain entry in dB SPL | [] |
| gtstep | vector<float> | level step size in dB | [] |
| tau_rmslev | vector<float> | RMS level averaging time constant in s | [] |
| tau_attack | vector<float> | attack time constant in s | [] |
| tau_decay | vector<float> | decay time constant in s | [] |
| fb | string | Name of fftfilterbank plugin. Used to extract frequency information. | fftfilterbank |
| chname | string | name of audio channel number variable (empty: broadband) | |
| bypass | bool | bypass dynamic compression | no |
| clientid | string | Client ID of last fit | |
| gainrule | string | Gain rule of last fit | |
| preset | string | Preset name of last fit | |
| modified | int | Flag if configuration has been modified | (monitor) |
| level_in | vector<float> | input level of last block / dB SPL | (monitor) |
| level_in_filtered | vector<float> | input level after time-constant filters / dB SPL | (monitor) |
| cf | vector<float> | nominal center frequencies of filterbank bands | (monitor) |
| ef | vector<float> | edge frequencies of filterbank bands | (monitor) |
| band_weights | vector<float> | Weights of the individual frequency bands. Computed as (sum of squared fft-bin-weigths) / num_frames. | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 1.2 dc_simple

Simple dynamic compression scheme

### 1.2.1 Detailed description

The plugin *dc_simple* is a multiband dynamic compression. One compression function (input-output function) is applied to each audio channel; multiple frequency bands can be used via the *fftfilterbank* plugin. The level dependent gain function is determined by the gains at 50 and 80 dB (G50 and G80). To reduce noise, an expansion is applied below a noise gate level. See also Fig. 2.

If spectral processing is used, the input level ($x$-axis of the input-output function) is determined by an attack- and release-filter of the short time RMS level $L_{st}$ given in dB (SPL)The attack filter is a first order low pass filter. The release filter is a maximum tracker, i.e.

$$L_a = \langle 20 \log_{10}(L_{st}) \rangle_{\tau_{attack}} \quad (3)$$

$$L_{in} = \max(L_a, \langle L_a \rangle_{\tau_{release}}) \quad (4)$$

The input level is divided into three sections. In each section the input level $L_{in}$ is transformed linearly into a gain $G$ on a log-log scale: $G_{dB} = (m-1)L_{in} + n$, where $m$ is the slope of the input-output function, and $n$ is an offset. Between expansion threshold and limiter threshold, $m$ and $n$ are given by the gain at 50 and 80 dB. In the section below the expansion threshold, $m$ is the expansion slope, and above the limiter threshold, $m$ is zero. $n$ is chosen to result in a continuous input-output function.

All variables are vectors with one entry for each input channel (number of audio channels times number of frequency bands).

An example configuration of a chain with dynamic compression could be:

© 2005-2019 HörTech gGmbH, Oldenburg

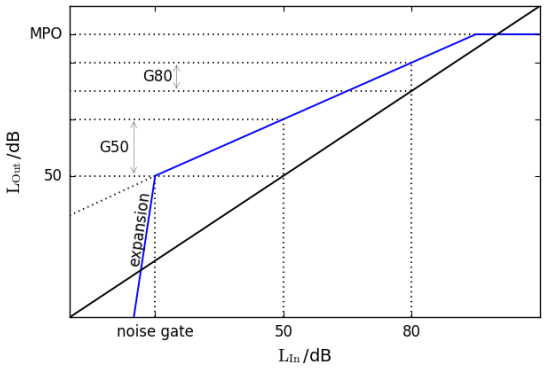


**Figure 2 Input-output function of one channel in the *dc_simple* dynamic compression algorithm.**

```
algos = [fftfilterbank dc_simple combinechannels]
fftfilterbank.ftype = center
fftfilterbank.f = [250 1000 4000]
fftfilterbank.ovltype = rect
fftfilterbank.fscale = bark
dc_simple.g50 = [10 25 40 11 31 55]
dc_simple.g80 = [5 15 10 5 21 19]
dc_simple.expansion_threshold = [20 20 20 20 20 20]
dc_simple.expansion_slope = [4 4 4 4 4 4]
dc_simple.limiter_threshold = [120 120 120 120 120 120]
dc_simple.tau_attack = [0.005 0.005 0.005 0.005 0.005 0.005]
dc_simple.tau_decay = [0.015 0.015 0.015 0.015 0.015 0.015]
combinechannels.name = fftfilterbank_channels
```

In this configuration it is assumed that two audio channels are configured, i.e. all variables of *dc_simple* have three entries for the first audio channel and three for the second audio channel.

### 1.2.2 Supported domains

The MHA plugin `dc_simple` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 1.2.3 Plugin Tags

*compression* level-modification

© 2005-2019 HörTech gGmbH, Oldenburg

### 1.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| g50 | vector<float> | Gain in dB at 50 dB input level **Range:** [-80,80] | [0] |
| g80 | vector<float> | Gain in dB at 80 dB input level **Range:** [-80,80] | [0] |
| maxgain | vector<float> | Maximal amplification in dB | [80] |
| expansion_threshold | vector<float> | expansion threshold in dB | [0] |
| expansion_slope | vector<float> | expansion slope of input-output function in dB/dB **Range:** ]0,10] | [1] |
| limiter_threshold | vector<float> | limiter threshold in dB | [100] |
| tau_attack | vector<float> | attack time constant in s **Range:** [0,] | [0.005] |
| tau_decay | vector<float> | decay time constant in s **Range:** [0,] | [0.05] |
| bypass | bool | bypass dynamic compression | no |
| clientid | string | Client ID of last fit | |
| gainrule | string | Gain rule of last fit | |
| preset | string | Preset name of last fit | |
| modified | int | Flag if configuration has been modified | (monitor) |
| level | vector<float> | Input level in dB | (monitor) |
| gain | vector<float> | Applied gain in dB | (monitor) |
| filterbank | string | Name of fftfilterbank plugin. Used to extract frequency information. | |
| cf | vector<float> | center frequencies of the frequency bands [Hz] | (monitor) |
| ef | vector<float> | edge frequencies of the frequency bands [Hz] | (monitor) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

## 1.3 softclip

The softclipper implements a broad band dynamic compression above a given level (Compression limiting).

### 1.3.1 Supported domains

The MHA plugin `softclip` supports these signal domains:

- waveform to waveform

### 1.3.2 Plugin Tags

*compression* limiter level-modification

### 1.3.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| tau_decay | float | time constant of decay filter<br>**Range:** [0,[ | 0.05 |
| tau_attack | float | time constant of attack filter<br>**Range:** [0,[ | 0.002 |
| start | float | entry point of time domain soft clipping (dB) | 110 |
| slope | float | slope of input-output table above start (dB/dB)<br>**Range:** [0,] | 0.125 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

# 2 Plugin category 'data-export'

## 2.1 ac2lsl

Send AC variables as LSL messages.

### 2.1.1 Detailed description

This plugin provides a mechanism to send ac variables over the network using the lab streaming layer (lsl). If no source id is set, recovery of the stream after changing channel count, data type, or any configuration variable is not possible. Sending data over the network is not real-time safe and processing will be aborted if this plugin is used in a real-time thread without user override. Currently no user-defined types are supported.

### 2.1.2 Supported domains

The MHA plugin `ac2lsl` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 2.1.3 Plugin Tags

*data-export* network-communication lab-streaming-layer

### 2.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| vars | vector<string> | List of AC variables to be saved, empty for all. | [] |
| source_id | string | Unique source id for the stream outlet. | |
| rt_strict | bool | Abort if used in real-time thread? | yes |
| activate | bool | Send frames to network? | yes |
| skip | int | Number of frames to skip after sending **Range:** [0,] | 0 |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 2.2 ac2osc

Send AC variables as OSC messages over udp.

### 2.2.1 Detailed description

Send AC variables as OSCvariables using the UDP transport layer. The variable "vars" can be used to select variables from the AC space for sending. The sending of variables can be verified using the open source tool "dump_osc". When selecting an AC variable, a target path can be specified using the colon delimiter, e.g.:

```
vars = [level:/mhalevels]
```

### 2.2.2 Supported domains

The MHA plugin `ac2osc` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 2.2.3 Plugin Tags

*data-export* network-communication open-sound-control

© 2005-2019 HörTech gGmbH, Oldenburg

### 2.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| host | string | OSC server host name | localhost |
| port | string | OSC server port | 7777 |
| ttl | int | Time-to-live of UDP packages (1 = subnet) | 1 |
| vars | vector<string> | List of AC variables to be saved, empty for all. A colon may be used to specify target address. | [] |
| mode | keyword_list | record mode<br>**Range:** [rec pause] | pause |
| skip | int | number of frames to skip after sending<br>**Range:** [0,] | 0 |
| rt_strict | bool | abort if used in real-time thread? | yes |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 2.3 acmon

This algorithm converts AC variables into parsable monitor variables.

### 2.3.1 Supported domains

The MHA plugin `acmon` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

© 2005-2019 HörTech gGmbH, Oldenburg

### 2.3.2 Plugin Tags

*data-export* network-communication

### 2.3.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| varlist | vector<string> | complete list of variables | (monitor) |
| dimensions | vector<string> | variable dimensions in AC space | (monitor) |
| dispmode | keyword_list | display mode of variables<br>**Range:** [vector matrix] | vector |
| recmode | keyword_list | record mode<br>**Range:** [cont snapshot] | cont |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 2.4 acsave

Save chain data to text or Matlab 4 files.

Usage:

1. set up file name and type

2. set the maximal length. This will start the recording.

3. Set "flush" to yes to save recorded frames. This will overwrite previously written data.

File name and type can be changed at any time and has to be valid when sending the flush command. Changing the list of variables also starts the recording with the currently configured recording length (previously recorded data might be overwritten). Issueing the 'flush' command frees allocated memory.

© 2005-2019 HörTech gGmbH, Oldenburg

### 2.4.1 Detailed description

The 'acsave' plugin can save numeric algorithm communication variables (AC variables) into files. The files can have plain text, MATLAB 4.x or MATLAB script format. Each signal frame represents a row. The number of columns is gathered at preparation time. If a variable size is increased after preparation, only the part available at preparation time is stored. If the size is decreased, it is zero-padded to the original size.

To save the data to disk, first set up file name and type. Then setting the maximal length will start the recording. At any time, set 'flush' to yes in order to save the recorded frames. This will overwrite previously written data.

File name and type can be changed at any time and have to be valid when sending the flush command.

### 2.4.2 Supported domains

The MHA plugin `acsave` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 2.4.3 Plugin Tags

*data-export* disk-files

### 2.4.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| fileformat | keyword_list | file format of output file<br>**Range:** [txt mat4 m] | txt |
| name | string | output file name | |
| reclen | float | maximal recording length in seconds<br>**Range:** [0,] | 10 |
| flush | bool | flush the buffers to disk | no |
| vars | vector<string> | list of variables to be saved (empty: save all) | [] |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 2.5 wavrec

wav file recorder

### 2.5.1 Supported domains

The MHA plugin `wavrec` supports these signal domains:

- waveform to waveform

### 2.5.2 Plugin Tags

*data-export* disk-files

### 2.5.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| fifolen | int | Length of FIFO in samples<br>**Range:** [2,] | 262144 |
| minwrite | int | Minimal write length (must be less then fifolen)<br>**Range:** [1,] | 65536 |
| prefix | string | Path (including path delimiter) and file prefix | |
| use_date | bool | Use date and time (yes), or only prefix (no) | yes |
| record | bool | Record session. Each write access with argument "yes" will start a new file with current time and date as a name. | no |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

# 3 Plugin category 'data-flow'

## 3.1 ac2wave

Mix the main input signal with a waveform stored into AC variables. Main and AC signal can be attenuated or delayed by integer fragments.

Spectral input is discarded and replaced by a zero signal.

### 3.1.1 Supported domains

The MHA plugin `ac2wave` supports these signal domains:

- waveform to waveform
- spectrum to waveform

### 3.1.2 Plugin Tags

*data-flow* algorithm-communication

© 2005-2019 HörTech gGmbH, Oldenburg

#### 3.1.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| name | string | AC variable name | |
| gain_in | float | Linear gain for main input signal | 0 |
| gain_ac | float | Linear gain for AC input signal | 1 |
| delay_in | int | Delay of main input signal in fragments<br>**Range:** [0,[ | 0 |
| delay_ac | int | Delay of AC input signal in fragments<br>**Range:** [0,[ | 0 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

### 3.2 acConcat_wave

Concatenating two or more waveforms into one

#### 3.2.1 Detailed description

This plugin concatenates two or more waveforms in the given order into a new waveform all living in the AC space. The waveforms to be concatenated as well as the concatenated waveforms must have the same number of channels. However the lengths of the waveforms to be concatenated may differ.

The waveforms to be concatenated should have been created in advance by some other plugin. This plugin creates an AC variable for the concatenated waveform and puts it into the AC space.

© 2005-2019 HörTech gGmbH, Oldenburg

The configuration variable **num_AC** defines the number of waveforms, which will be concatenated into one waveform. The waveforms to be concatenated obey the same naming convention followed by a numeric suffix defining the order of concatenation. This order begins with 1 and a whole in the numeric order is not allowed. This naming convention is defined by the user by setting the configuration variable **prefix_names_AC**. The lengths of each waveform to be concatenated is defined by the configuration variable **samples_AC**. This vector variable must contain an integer value corresponding to each of the waveforms to be concatenated defining their lengths respectively. The name of the concatenated waveform is defined by the configuration variable **name_con_AC**.

This plugin is typically used together with the plugins `doasvm_feature_extraction` instantiated several times for computing the cross correlation between all combinations of input channels and with `doasvm_classification`, which uses the concatenated cross correlation vectors for each channel combination to estimate the arrival direction of audio signals.

As an example, if there are six waveforms, which are supposed to be concatenated into one waveform, this plugin can be configured as shown in the following:

```
acConcat_wave.num_AC = 6
acConcat_wave.samples_AC = [161 17 161 161 17 161]
acConcat_wave.prefix_names_AC = "vGCC"
acConcat_wave.name_con_AC = "vGCCcon"
```

In this case, the six waveforms to be concatenated should be called `vGCC_1`, `vGCC_2`, `vGCC_3`, `vGCC_4`, `vGCC_5` and `vGCC_6`. Note that in the localization context, for a setup of four microphones, there are six different combinations of two microphones.

### 3.2.2 Supported domains

The MHA plugin `acConcat_wave` supports these signal domains:

- waveform to waveform

### 3.2.3 Plugin Tags

*data-flow* algorithm-communication

© 2005-2019 HörTech gGmbH, Oldenburg

#### 3.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| num_AC | int | Number of waveforms to be concatenated<br>**Range:** [1,28] | 15 |
| prefix_names_AC | string | Prefix of the names of the waveforms to be concatenated | vGCC_ac |
| samples_AC | vector<int> | Lengths of the waveforms to be concatenated | [] |
| name_con_AC | string | Name of the concatenated waveform | vGCC_con_AC |
| numchannels | int | Number of channels in each waveform to be concatenated<br>**Range:** [1,[ | 1 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

### 3.3 acPooling_wave

Pooling of several consecutive time frames

#### 3.3.1 Detailed description

This plugin computes an average over several consecutive frames using several different approaches (max, sum, mean, ...). Subsequently, the maximum of the average frame is delivered as well. The plugin receives the frames through the AC space and deliveres the averaged frame as well as its maximum to the AC space.

© 2005-2019 HörTech gGmbH, Oldenburg

This plugin is typically used together with the plugins `doasvm_feature_extraction` and `doasvm_classification` for estimating the arrival direction of an audio signal. Within this context, this plugin smooths the vector of estimated arrival directions over time using several estimation vectors. However, it can also used within other contexts for smoothing purposes of e.g. waveforms.

The length of a frame is given by the configuration variable **numsamples**. In the context of localization, for an angular resolution of 5 degrees, a total number of 37 estimation values for the interval of possible arrival directions [-90, 90] would be produced by the `doasvm_classification` plugin.

The frames to be smoothed have to be created by some other plugin in advance, e.g. `doasvm_classification` plugin for the localization, and should be available in the AC space to be read by this plugin. The name of the corresponding AC variable is defined using the configuration variable **p_name**.

At each iteration, this plugin reads the AC variable corresponding to the frame to be smoothed and smooths it by averaging a number of such frames, which have been read in the past iterations and saved in a pool. The length of this pool in msec is defined using the configuration variable **pooling_wndlen**. Depending on the frame rate used within the current MHA configuration, the exact number of iterations, which fall into this pool is computed during the preparation of this plugin. Note that the length of this pool should be chosen carefully so as to make sure that more than one frame falls in.

This plugin implements several pooling methods for smoothing the frames saved in the pool. The alternatives are **max, sum** and **mean**. The pooling method to be used is defined using the configuration variable **pooling_type**. For each value in the frame (e.g. for each possible arrival direction in the context of localization), the **max** pooling takes the maximum between the iterations. The **sum** pooling sums these values up. Finally, the **mean** pooling computes the arithmetic mean of these values. Once the pooling step has been completed, a smoothed vector of frames of the same size with a single frame from each iteration has been constucted. This vector is saved in another AC variable, which is defined by using the configuration variable **pool_name**.

Optionally, after the smoothing step, certain areas within the smoothed frame can be weighted differently. In the localization context, this optional step can correspond to a prior probability to favour certain possible arrival directions more compared to others. For instance, a hearing aid waerer can expect that the person who he / she is talking to is infront of him / her. In that case, the prior probabilities for the frontal directions can be set higher than the other possible arrival directions. This can be defined by setting tha configuration variable `prob_bias`. The default values of this variable are all set to 1, hence uniform distribution. The size of this variable should be equal to the frame length given in the configuration variable `numsamples`. The smoothed and weighted frame is saved in yet another AC variable, which is defined by using the configuration variable `p_biased_name`.

After the smoothed frame has been computed, the maximum value of this frame is found and saved in another AC variable. In the localization context, this maximum corresponds to the arrival direction of an audio signal. The name of this AC variable is defined using the configuration variable **max_pool_ind_name**.

In the following, an example configuration within a localization context is given. In this configuration, an angular resolution of 5 degrees for the whole circle, namely the interval of [-180, 180] is considered. In that case, there are in total of 73 possible arrival directions.

© 2005-2019 HörTech gGmbH, Oldenburg

```
acPooling_wave.p_name = p
acPooling_wave.pool_name = pool
acPooling_wave.max_pool_ind_name = pool_max
acPooling_wave.numsamples = 73
acPooling_wave.pooling_wndlen = 300
acPooling_wave.pooling_type = mean
```

### 3.3.2 Supported domains

The MHA plugin `acPooling_wave` supports these signal domains:

- waveform to waveform

### 3.3.3 Plugin Tags

*data-flow* feature-extraction algorithm-communication

© 2005-2019 HörTech gGmbH, Oldenburg

### 3.3.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| numsamples | int | This parameter determines the length of the wave to be pooled in samples **Range:** ]0,] | 37 |
| pooling_wndlen | int | This parameter determines the length of the pooling window in msec. **Range:** ]0,] | 300 |
| pooling_type | keyword_list | This parameter determines the pooling method applied to the pooling window. **Range:** [max sum mean] | mean |
| upper_threshold | float | This parameter sets a threshold for finding the maximum probability. If the maximum is above this threshold, it is taken, even if it is not in the neighbourhood of the last estimated direction. **Range:** [0,1] | 0.75 |
| lower_threshold | float | This parameter sets a threshold for finding the minimum probability. If the maximum probability is below this threshold, the estimated direction of the last iteration is taken. **Range:** [0,1] | 0 |
| neighbourhood | int | This parameter defines the neighbourhood of the allowed change of the estimated direction between iterations. -1 means no neighbourhood. **Range:** [-1,] | 2 |
| alpha | float | This parameter simulates the forgetting effect by weighting the frames within the pooling window, e.g. p(n + 1) = (1 - alpha) * p(n) + alpha * p_new. 0 means no weighting. **Range:** [0,1] | 0.1 |
| p_name | string | The name of the AC variable of the frame, which is going to be pooled. | p |
| p_biased_name | string | The name of the AC variable of the biased frame, after pooling. | prob_biased |
| pool_name | string | The name of the AC variable of the averaged (pooled) frame. | pool |
| max_pool_ind_name | string | The name of the AC variable for the index of the maximum of the averaged frames | pool_max |
| like_ratio_name | string | The name of the AC variable for the likelihood ratios of the averaged frames | like_ratio |
| prob_bias | vector<float> | A multiplicative probability bias | [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 3.4 combinechannels

Channel combiner

### 3.4.1 Detailed description

Several filter channels can be combined into one or more output channels by summing the input channels. This plugin is intended as a filter resynthesis of linear-phase filter banks.

The input signal is expected to have a non-interleaved channel order, i.e., first all bands of first output channel, then all bands of second channel, etc.

### 3.4.2 Supported domains

The MHA plugin `combinechannels` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 3.4.3 Plugin Tags

*data-flow* audio-channels filterbank

© 2005-2019 HörTech gGmbH, Oldenburg

### 3.4.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| outchannels | int | Number of output channels<br>**Range:** [1,] | 1 |
| interleaved | bool | Input signal has interleaved channel order? | no |
| channel_gain_name | string | Name of channel gain AC variable (looked up during prepare, can be empty) | |
| element_gain_name | string | Name of element wise gain AC variable (looked up during waveform process, can be empty) | |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 3.5 db

Synchronous double buffer plugin.

### 3.5.1 Detailed description

The double buffer plugin allows changes of fragment size. It has an outer layer (e.g. framework) and an inner layer (e.g. MHA kernel, plugin). A configurable fragment size is used on the inner side, which is independent from the outer fragment size. The input data is buffered, and the data is processed when enough samples are available.

Please note that double buffering adds an extra delay of the audio stream. If both fragment sizes are identical, the double buffering is bypassed.

© 2005-2019 HörTech gGmbH, Oldenburg

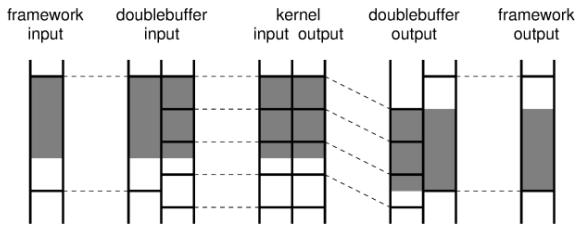


**Figure 3 Concept of the double buffer plugin. The outer fragments, provided by the framework, are split up into smaller fragments for processing in the kernel. For a continuous output stream, an extra delay is needed, i.e. the first fragment is filled with zeros at the beginning.**

#### 3.5.1.1 Warning:

If the inner fragment size is larger than the outer fragment size, the maximal processing time is limited by the shorter fragment size. This results in a maximal processor usage determined by the ratio of outer to inner fragment size. This problem holds not for offline processing. As an alternative, the asynchronous double-buffer plugin `dbasync` (section **??**) can be used, which processes the double-buffered signal in a separate thread. That plugin should be preferred for real-time processing. If the inner thread should only be used for signal analysis, please refer to the plugin `analysispath` (section 13.2).

### 3.5.2 Supported domains

The MHA plugin `db` supports these signal domains:

- waveform to waveform

### 3.5.3 Plugin Tags

*data-flow* signal-transformation

### 3.5.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| plugin_name | string | Plugin name | |
| fragsize | int | fragment size of client plugin<br>**Range:** [0,] | 200 |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 3.6 delay

Delay line

### 3.6.1 Supported domains

The MHA plugin `delay` supports these signal domains:

- waveform to waveform

### 3.6.2 Plugin Tags

*data-flow* audio-channels signal-transformation

### 3.6.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| delay | vector<int> | delay in samples, one entry for each channel<br>**Range:** [0,[ | [0 0] |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 3.7 fader_spec

fader

### 3.7.1 Supported domains

The MHA plugin `fader_spec` supports these signal domains:

- spectrum to spectrum

### 3.7.2 Plugin Tags

*data-flow* audio-channels cross-fade level-modification

### 3.7.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| tau | float | fader duration in seconds<br>**Range:** [0,[ | 1 |
| gains | vector<float> | | [1 1] |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 3.8 fader_wave

Apply level

### 3.8.1 Supported domains

The MHA plugin `fader_wave` supports these signal domains:

- waveform to waveform

### 3.8.2 Plugin Tags

*data-flow* audio-channels cross-fade level-modification

### 3.8.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| gain | vector<float> | Gain (linear) | [1] |
| ramplen | float | Length of hanning ramp at gain changes in seconds<br>**Range:** [0,] | 0 |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 3.9 matrixmixer

Matrix mixer plugin, can mix multiple input channels into any number of output channels with configurable weights.

### 3.9.1 Detailed description

The `matrixmixer` plugin can combine the signal from multiple input channels into any number of output channels, with defined mixing weights.

Example: To combine the two channels of a stereo signal into a single (mono) channel, configure the `matrixmixer` plugin configuration variable `m` as

```
m = [[1.0 1.0]]
```

which causes the first and the second channel to be multiplied with a weight of 1 before they are mixed (by adding them together) to form a single output channel.

It is also possible to mix the channels with weights different from 1:

```
m = [[1 0.5]]
```

This attenuates the second channel by multiplying all samples in that channel with 0.5 before mixing it with the first channel. The configuration variable `m` expects a matrix of float values. The examples above showed a matrix with only one row, which resulted in only one output channel being produced by the `matrixmixer` plugin. To produce more output channels, more rows (separated by semicolons)[1] can be specified for matrix `m`:

[1] In openMHA configuration, a matrix is specified as a vector of vectors, where the subsequent row vectors are separated by semicolons. For details, refer to the subsection on multidimensional variables in the openMHA application manual.

© 2005-2019 HörTech gGmbH, Oldenburg

```
m = [[1 0];[0 1]]
```

This is the identity matrix for two channels. This matrix does not change the signal.

The following setting demonstrates how matrixmixer can be used to change the order of audio channels in a multi-channel signal. This example swaps the first two channels:

```
m = [[0 1];[1 0]]
```

The next setting creates a 4-channel signal output from a stereo signal, where the first two channels are the original stereo channels, the third is the sum of the two stereo channels, and the fourth output channel is the difference of the two stereo channels[2]:

```
m = [[0 1];[1 0];[1 1];[1 -1]]
```

The following example duplicates a single input channel to two output channels:

```
m = [[1];[1]]
```

To summarize, you need to configure the variable `m` with a matrix with float values. The matrix needs to have as many columns as the `matrixmixer` receives input channels, and as many rows as you want `matrixmixer` to produce output channels.

Example configurations and example input files are contained in the matrixmixer examples directory. Please refer to the README file in this directory for an explanation of the different examples.

A matlab/octave test exercising the matrixmixer plugin in six different configurations can be found in the mhatest directory in file `test_matrixmixer.m`. This test file is executed together with the other system-level tests when invoking `make test`.

### 3.9.2 Supported domains

The MHA plugin `matrixmixer` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 3.9.3 Plugin Tags

*data-flow* audio-channels

[2] The combination of the sum and the difference of the two channels of a stereo signal is known as the mid-side signal and used for stereo transmission in FM radio. We combine it here with the orignial stereo signal for the sole purpose of demonstrating the creation of more output channels than input channels with the `matrixmixer` plugin.

© 2005-2019 HörTech gGmbH, Oldenburg

#### 3.9.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| m | matrix<float> | Mixer matrix, one row vector for each output channel. The number of columns must match the number of input channels. | [[1 0];[0 1]] |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

### 3.10 route

Signal router plugin.

Arguments are the input signal source names (AC variables) followed by a colon, followed by the channel number, starting at zero. Empty names correspond to the direct input.

An AC variable will be created if the AC output dimension is not zero. Example: out = [:0 :1 x:0 x:1] ac = [:2 :3] returns a four channel output signal containing first two direct input channels, and the first two channels of the AC variable "x". An AC variable is created with the third and fourth channel of the direct input.

#### 3.10.1 Supported domains

The MHA plugin `route` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

© 2005-2019 HörTech gGmbH, Oldenburg

### 3.10.2 Plugin Tags

*data-flow* audio-channels algorithm-communication

### 3.10.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| out | vector<string> | direct output | [] |
| ac | vector<string> | AC output | [] |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 3.11 save_spec

Save signal spectrum to AC variable

### 3.11.1 Supported domains

The MHA plugin `save_spec` supports these signal domains:

- spectrum to spectrum

### 3.11.2 Plugin Tags

*data-flow* algorithm-communication

© 2005-2019 HörTech gGmbH, Oldenburg

#### 3.11.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

### 3.12 save_wave

Save signal waveform to AC variable

#### 3.12.1 Supported domains

The MHA plugin `save_wave` supports these signal domains:

- waveform to waveform

#### 3.12.2 Plugin Tags

*data-flow* algorithm-communication

© 2005-2019 HörTech gGmbH, Oldenburg

### 3.12.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 3.13 shadowfilter_begin

Save signal spectrum to AC variable

### 3.13.1 Detailed description

The plugins 'shadowfilter_begin' and 'shadowfilter_end' (section 3.14) are designed to measure the gains produced by any spectral plugins and apply those gains to audio channels not passed to the algorithm. This method can be used to process a mixed signal, but apply the same gains to the unmixed signal parts seperately. For a stereo mixed signal, this can be done by reading the mixed signal from channels 1 and 2, the desired signal from channels 3 and 4, and the competing signal from channels 5 and 6. The 'shadowfilter_begin' plugin hides channels 3 to 6 from the plugin, and remembers the input spectrae for all channels. The 'shadowfilter_end' plugin compares the processed output signal (channels 1 and 2) with its input spectrum and derives complex gains produced by the algorithm (without any knowledge of the algorithm). The same gains are applied to channels 3 to 6.

### 3.13.2 Supported domains

The MHA plugin `shadowfilter_begin` supports these signal domains:

- spectrum to spectrum

© 2005-2019 HörTech gGmbH, Oldenburg

### 3.13.3 Plugin Tags

*data-flow* feature-extraction filter

### 3.13.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| nch | int | number of processing channels<br>**Range:** [1,[ | 1 |
| ntracks | int | number of input sources, each with nch audio channels<br>**Range:** [1,[ | 1 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 3.14 shadowfilter_end

Compute spectral gains seen since shadowfilter_begin, apply gains to other tracks

### 3.14.1 Detailed description

See section 3.13 for a description of the shadow filter method. The 'shadowfilter_end' plugin creates an AC variable shadowfilter_gains, which contains the complex gains created by the algorithm.

© 2005-2019 HörTech gGmbH, Oldenburg

### 3.14.2 Supported domains

The MHA plugin `shadowfilter_end` supports these signal domains:

- spectrum to spectrum

### 3.14.3 Plugin Tags

*data-flow* feature-extraction filter

### 3.14.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| basename | string | configuration name of shadowfilter_begin | shadowfilter_begin |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

# 4 Plugin category 'data-import'

## 4.1 acSteer

Steering Vector Loading Plugin

© 2005-2019 HörTech gGmbH, Oldenburg

#### 4.1.1 Detailed description

The `acSteer` plugin loads a file contaning pre-computed steering filters (e.g. MVDR filters) to be used within a beamformer. The steering filters can be monaural (**nrefmic = 1**) or binaural (**nrefmic = 2**). The whole file consists of a column vector of concatenated steering vectors, which are formatted in the order of **angle** and **channel**. This means that the first channel vector of the first angle is followed by the second channel vector of the first angle until the last channel. The channel vectors of the first angle are followed by the channel vectors of the second angle and so an and so forth.

If the steering filters have been computed for two reference microphones, the steering filters of the second reference microphone just follow the ones for the first microphone and have the same format.

This plugin is typically located between a localization plugin (e.g. `doasvm_classification`) and a beamforming plugin (e.g. `steerbf`). The localization plugin estimates the source direction and saves it in an AC variable. This plugin reads the saved direction from the corresponding AC variable and saves the corresponding steering vector to the AC space, which is used by the succeeding beamforming plugin for steering the beam towards that particular direction.

The configuration variable **nrefmic** indicates the number of different reference microphone settings, for which the filters were computed. For each reference microphone and each possible DOA angle and each input channel one filter should be provided so that

$$nsteerchan = nrefmix * nchan * nangle \tag{5}$$

#### 4.1.2 Supported domains

The MHA plugin `acSteer` supports these signal domains:

- spectrum to spectrum

#### 4.1.3 Plugin Tags

*data-import* disk-files beamformer binaural adaptive

© 2005-2019 HörTech gGmbH, Oldenburg

#### 4.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| steerFile | string | Name of the input file where the steering vectors are saved | steerfile.bin |
| acSteerName1 | string | Name of the AC variable where the steering vectors of the first (left) reference microphone are saved | acSteerLeft |
| acSteerName2 | string | Name of the AC variable where the steering vectors of the second (right) reference microphone are saved | acSteerRight |
| nsteerchan | int | Number of channels in each steering vector | 4 |
| nrefmic | int | Number of reference microphones **Range:** ]0,2] | 1 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

### 4.2 addsndfile

Add data from a sound file to some channels of input signal.

The sound file is read into memory and scaled to a given peak level, i.e. the RMS level of an abstract signal with 0 dB full scale. Changing any parameter will start playing the file from the beginning. Playback of the file starts only after the complete file has been read. If the sound file has fewer channels then the 'channels' vector has elements, then the file channels will be repeated.

© 2005-2019 HörTech gGmbH, Oldenburg

#### 4.2.1 Detailed description

Add data from a sound file to some channels of input signal.

The sound file is read into memory and scaled to a given peak level, i.e. the RMS level of an abstract signal with 0 dB full scale. Changing any parameter will start playing the file from the beginning. Playback of the file starts only after the complete file has been read. If the sound file has fewer channels then the 'channels' vector has elements, then the file channels will be repeated. The addsndfile plugin does not change the number of MHA audio channels. If you specify a channel index $\geq$ the number of MHA audio channels, then the channel from the sound file will not be used.

#### 4.2.2 Supported domains

The MHA plugin `addsndfile` supports these signal domains:

- waveform to waveform

#### 4.2.3 Plugin Tags

*data-import* disk-files signal-generator

© 2005-2019 HörTech gGmbH, Oldenburg

### 4.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| path | string | Optional path to sound file | |
| filename | string | File name of the sound file. | |
| loop | bool | Infinitely loop the playback of the sound file | yes |
| level | float | Level in dB (SPL) of the input file | 0 |
| levelmode | keyword_list | Level mode **Range:** [relative peak rms rms_limit40] | relative |
| resamplingmode | keyword_list | Resampling mode **Range:** [dont_resample_permissive dont_resample_strict do_resample] | do_resample |
| channels | vector<int> | Output signal channels in which to store the individual sound file channels. Output channel indices start from 0. Note: The addsndfile plugin does not change the number of MHA audio channels. If you specify a channel index >= the number of MHA audio channels, then that channel from the sound file will not be used. **Range:** [0,] | [0] |
| mode | keyword_list | Playback mode **Range:** [add replace input mute] | add |
| ramplen | float | Length of hanning ramp at level changes in seconds **Range:** [0,] | 0 |
| startpos | int | Starting position in samples, loop will begin from zero **Range:** [0,] | 0 |
| mapping | vector<int> | Channel mapping | (monitor) |
| filechannels | int | Number of channels in current file | (monitor) |
| mhachannels | int | Number of MHA channels at plugin position | (monitor) |
| active | int | indicates whether sound is currently played back | (monitor) |
| search_pattern | string | Search pattern for file list | *.wav |
| files | vector<string> | Available files | (monitor) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

# 5 Plugin category 'example'

## 5.1 example1

### 5.1.1 Detailed description

The **simplest** example of an openMHA plugin.

This plugin scales one channel of the input signal, working in the time domain.

### 5.1.2 Supported domains

The MHA plugin `example1` supports these signal domains:

- waveform to waveform

### 5.1.3 Plugin Tags

*example* level-modification audio-channels

### 5.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 5.2 example2

This plugin multiplies the sound signal in one audio channel by a factor

### 5.2.1 Supported domains

The MHA plugin `example2` supports these signal domains:

- waveform to waveform

### 5.2.2 Plugin Tags

*example* level-modification audio-channels

### 5.2.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| channel | int | Index of audio channel to scale. Indices start from 0.<br>**Range:** [0,[ | 0 |
| factor | float | The scaling factor that is applied to the selected channel.<br>**Range:** [0,[ | 0.1 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 5.3 example3

This plugin multiplies the sound signal in one audio channel by a factor

### 5.3.1 Supported domains

The MHA plugin `example3` supports these signal domains:

- waveform to waveform

### 5.3.2 Plugin Tags

*example* level-modification audio-channels

### 5.3.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| channel | int | Index of audio channel to scale. Indices start from 0. Only channels with even indices may be scaled. **Range:** [0,[ | 0 |
| factor | float | The scaling factor that is applied to the selected channel. **Range:** [0,[ | 0.1 |
| prepared | int | State of this plugin: 0 = unprepared, 1 = prepared | (monitor) |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 5.4 example4

This plugin multiplies the sound signal in one audio channel by a factor. It works in the spectral domain.

### 5.4.1 Supported domains

The MHA plugin `example4` supports these signal domains:

- spectrum to spectrum

### 5.4.2 Plugin Tags

*example* level-modification audio-channels

### 5.4.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| channel | int | Index of audio channel to scale. Indices start from 0. Only channels with even indices may be scaled. **Range:** [0,[ | 0 |
| factor | float | The scaling factor that is applied to the selected channel. **Range:** [0,[ | 0.1 |
| prepared | int | State of this plugin: 0 = unprepared, 1 = prepared | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 5.5 example5

example plugin configuration structure

### 5.5.1 Supported domains

The MHA plugin `example5` supports these signal domains:

- spectrum to spectrum

### 5.5.2 Plugin Tags

*example* level-modification audio-channels

© 2005-2019 HörTech gGmbH, Oldenburg

### 5.5.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| channel | int | channel number to be scaled<br>**Range:** [0,[ | 0 |
| factor | float | scale factor<br>**Range:** [0,2] | 1 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 5.6 example6

example plugin configuration structure

### 5.6.1 Supported domains

The MHA plugin `example6` supports these signal domains:

- waveform to waveform

### 5.6.2 Plugin Tags

*example* feature-extraction algorithm-communication

© 2005-2019 HörTech gGmbH, Oldenburg

#### 5.6.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| channel | int | channel in which the RMS level is measured<br>**Range:** [0,[ | 0 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

# 6 Plugin category 'feedback-suppression'

## 6.1 fshift

### 6.1.1 Detailed description

Performs a quantized frequency shift on the selected frequency interval. The frequency band between (originally) `fmin` and `fmax` (frequencies in Hz) is shifted by `df` (desired frequency change in Hz). Positive `df` shifts the selected band to higher frequencies, negative `df` shifts to lower frequencies.

The shifted and the unshifted parts of the input signal are split at the STFT bin boundaries nearest to the. The frequency shift `df` is rounded to the nearest bin as well. The parts of the spectrum that would be shifted below 0 Hz or above the Nyquist frequency are discarded.

### 6.1.2 Supported domains

The MHA plugin `fshift` supports these signal domains:

- spectrum to spectrum

© 2005-2019 HörTech gGmbH, Oldenburg

### 6.1.3 Plugin Tags

*feedback-suppression* frequency-modification

### 6.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| fmin | float | lower boundary for frequency shifter in Hz<br>**Range:** [0,] | 4000 |
| fmax | float | upper boundary for frequency shifter in Hz<br>**Range:** [0,] | 16000 |
| df | float | shift frequency in Hz | 40 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 6.2 fshift_hilbert

Pitch shifter

© 2005-2019 HörTech gGmbH, Oldenburg

### 6.2.1 Detailed description

Performs a frequency shift on the selected frequency interval. The frequency band between (originally) `fmin` and `fmax` (frequencies in Hz) is shifted by `df` (desired frequency change in Hz). Positive `df` shifts the selected band to higher frequencies, negative `df` shifts to lower frequencies.

The frequency shift on the sub-band is performed by splitting the input signal's spectrum into 2 parts: the band to be shifted, and the rest. The band to be shifted is multiplied in the time domain with a complex sinusoid of frequency `df` Hz (see Wardle(1998)[3]) before it is recombined in the spectral domain with the unshifted signal part.

By default the shifted and the unshifted parts of the input signal are split at STFT bin boundaries. The resulting rectangular transitions between shifted and unshifted parts can be smoothed if desired by setting `phasemode` to `linear` or `minimal`, and choosing a longer impulse response length than the default of 1 sample. Our experience with hearing aid applications so far suggests that smoothing these boundaries is not necessary.

### 6.2.2 Supported domains

The MHA plugin `fshift_hilbert` supports these signal domains:

- spectrum to spectrum

### 6.2.3 Plugin Tags

*feedback-suppression* frequency-modification

### 6.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| df | vector<float> | frequency to shift the bins / Hz | [40] |
| fmin | float | lower boundary for frequency shifter<br>**Range:** [0,] | 4000 |
| fmax | float | upper boundary for frequency shifter<br>**Range:** [0,] | 16000 |
| irslen | int | Bandpass: maximum length of cut off filter response<br>**Range:** [1,] | 1 |
| phasemode | keyword_list | Bandpass: mode of gain smoothing<br>**Range:** [none linear minimal] | none |

[3] Scott Wardle. A hilbert-transformer frequency shifter Proc. DAFX98 Workshop on Digital Audio Effects, pages 25–29, Barcelona, 1998.

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 6.3 lpc

This plugin implements the linear predictive coding analysis (LPC) by using the Levinson-Durbin recursion.

### 6.3.1 Detailed description

This plugin estimates the autocorrelation of each block. It then produces the inverse filter using the Levinson-Durbin recursion.

### 6.3.2 Supported domains

The MHA plugin `lpc` supports these signal domains:

- waveform to waveform

### 6.3.3 Plugin Tags

*feedback-suppression* adaptive

© 2005-2019 HörTech gGmbH, Oldenburg

### 6.3.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| lpc_order | int | LPC filter order<br>**Range:** [0,500] | 20 |
| lpc_buffer_size | int | Size of the buffer in samples for which the autocorrelation matrix will be computed<br>**Range:** ]0,501] | 21 |
| shift | bool | Refill the LPC buffer completely with new input signal by ignoring the old samples (no) or shift the old buffer as large as the block size of the input signal and read in the current input signal (yes). | yes |
| comp_each_iter | int | Reestimate the LPC coefficients each <comp_each_iter> iterations, default value is 1<br>**Range:** ]0,] | 1 |
| norm | bool | Normalize the auto correlation matrix with the LPC order | no |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 6.4 lpc_bl_predictor

This plugin performs forward and backward linear prediction using the Burg - Lattice algorithm for computing the next value of a given time series.

The estimated forward and backward linear predictionn parameters are saved in th AC space.

© 2005-2019 HörTech gGmbH, Oldenburg

### 6.4.1 Detailed description

This plugin computes the forward and backward LPC estimates using the Burg-Lattice algorithm given the $\kappa$ (sometimes also called $\mu$) parameter precomputed using the `lpc_burg-lattice` plugin. The estimation of the forward and backward linear prediction parameters is performed using the following equations: For each forward and backward linear prediction parameter $f(m)$ and $b(m)$, where $m$ in $[2 \cdots P]$, $P$ being the lpc order

$$f(m) = f(m-1) + \kappa(m,2) * b(m-1,2) \quad (6)$$

$$b(m,1) = b(m-1,2) + \kappa(m,2) * f(m-1) \quad (7)$$

. In this implementation $\kappa$ from the previous is used. Note that the second index of $\kappa$ is $2$.

### 6.4.2 Supported domains

The MHA plugin `lpc_bl_predictor` supports these signal domains:

- waveform to waveform

### 6.4.3 Plugin Tags

*feedback-suppression* adaptive

### 6.4.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| lpc_order | int | LPC order defines the number of coffecients to be estimated **Range:** ]0,] | 21 |
| name_kappa | string | Name of the kappa parameter of the Burg-Lattice algorithm in the AC domain to be used for the joint estimation of more than one time series | km |
| name_lpc_f | string | Name of the forward LPC estimate of the Burg-Lattice algorithm in the AC domain | name_lpc_f |
| name_lpc_b | string | Name of the backward LPC estimate of the Burg-Lattice algorithm in the AC domain | name_lpc_b |
| name_f | string | Name of the forward linear prediction parameter | |
| name_b | string | Name of the backward linear prediction parameter | |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 6.5 lpc_burg-lattice

This plugin estimates the linear predictive coding coefficients for estimating the next sample value of a time series using the Burg-Lattice approach.

The estimated parameters are saved in the AC space.

### 6.5.1 Detailed description

This plugin estimates the parameters for the forward and backward linear prediction using the Burg - Lattice algorithm. The previous estimate of the $\kappa$ parameter is saved in the AC space for future use in the `lpc_bl_predictor` plugin to estimate several time-series sharing the same $\kappa$ values.

For the estimation of $\kappa$ the following series of equations are used: For each $\kappa$ in $[2 \cdots P]$, $P$ being the lpc order

$$dm(m-1) = \lambda * dm(m-1) + (1-\lambda) * (f(m-1)^2 + b(m-1,2)^2) \quad (8)$$

$$nm(m-1) = \lambda * nm(m-1) + (1-\lambda) * -2 * f(m-1) * b(m-1,2) \quad (9)$$

$$km(m,1) = \frac{nm(m-1)}{dm(m-1)}. \quad (10)$$

Note that the previous estimate of $\kappa$, which is given by $\kappa(m,2)$ is saved in the AC space.

### 6.5.2 Supported domains

The MHA plugin `lpc_burg-lattice` supports these signal domains:

- waveform to waveform

© 2005-2019 HörTech gGmbH, Oldenburg

### 6.5.3 Plugin Tags

*feedback-suppression* adaptive

### 6.5.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| lpc_order | int | LPC order defines the number of coffecients to be estimated<br>**Range:** ]0,] | 21 |
| name_kappa | string | Name of the kappa parameter of the Burg-Lattice algorithm in the AC domain to be used for the joint estimation of more than one time series | km |
| name_f | string | Name of the forward linear prediction parameter | |
| name_b | string | Name of the backward linear prediction parameter | |
| lambda | float | Forgetting factor for the linear predictor<br>**Range:** [0,1] | 0.99375 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 6.6 nlms_wave

This plugin adaptively estimates the coefficients of a filter by means of the NLMS algorithm.

The estimated filter is stored into an AC variable named by the algorithm configuration name or by the configuration variable name_f and the input signal is filtered by the current filter and returned as the output signal of the plugin.

© 2005-2019 HörTech gGmbH, Oldenburg

### 6.6.1 Detailed description

This plugin implements the NLMS algorithm for re-estimating the coefficients of an adaptive filter in each iteration. The estimated filter coefficients are saved in an AC variable having the same name as the plugin in the current configuration. The name of this AC variable can also be set differently by setting the configuration variable **name_f**. The input signal is filtered by the filter estimated in the current iteration and returned as the current output of the plugin from within the processing callback. The estimation of the filter coefficients is performed using the update rule given as in the following:

$$e[k] = y[k-1] - f[k-1]u[k-1] \quad (11)$$
$$f[k] = f[k-1] + rho/(|u|^2 + c)u[k-1]e[k], \quad (12)$$

where $e$ is the error signal, $y$ is the desired signal and $u$ is the input signal. All three signals are read from the AC space. For this, the configuration variables **name_e**, **name_d** and **name_u** should be set. The error signal can also be computed within the plugin given the other two signals, when the corresponding configuration variable is left empty. The plugin can be configured to use also the current sample $u[k]$ of the input signal in the estimation by asigning the configuration variable **estimtype** to the value *current*. However in the default case (*previous*), the previous values as long as the filter (**ntaps**) but the current one are used.

### 6.6.2 Supported domains

The MHA plugin `nlms_wave` supports these signal domains:

- waveform to waveform

### 6.6.3 Plugin Tags

*feedback-suppression* adaptive

© 2005-2019 HörTech gGmbH, Oldenburg

### 6.6.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| rho | float | convergence coefficient<br>**Range:** ]0,2] | 0.01 |
| c | float | stabilization parameter<br>**Range:** ]0,] | 1e-05 |
| ntaps | int | number of taps in filter<br>**Range:** ]0,] | 32 |
| name_u | string | Name of input signal U | |
| name_d | string | Name of desired signal D | |
| normtype | keyword_list | Normalization type<br>**Range:** [none default sum] | default |
| estimtype | keyword_list | Estimation type defined whether the current value of the input signal u[k] will be incorporated in the estimation of the filter coefficients or not. Default value (previous) does not.<br>**Range:** [previous current] | previous |
| lambda_smoothing_power | float | Recursive smoothing constant for sum normalization<br>**Range:** [0,1[ | 0.9 |
| name_e | string | Name of error signal E | |
| name_f | string | Name of the AC variable for saving the adapive filter | |
| n_no_update | int | Number of iterations without updating the filter coefficients | 0 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 6.7 prediction_error

Prediction error model for adaptive feedback cancellation

© 2005-2019 HörTech gGmbH, Oldenburg

### 6.7.1 Detailed description

This plugin computes the prediction error model to perform adaptive feedback cancellation. The prediction error method produces and estimate of the feedback path by minimizing the measured and the predicted output signals.

### 6.7.2 Supported domains

The MHA plugin `prediction_error` supports these signal domains:

- waveform to waveform

### 6.7.3 Plugin Tags

*feedback-suppression* adaptive

### 6.7.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| rho | float | convergence coefficient<br>**Range:** ]0,2] | 0.01 |
| c | float | stabilization parameter<br>**Range:** ]0,] | 1e-05 |
| ntaps | int | number of taps in filter<br>**Range:** ]0,] | 32 |
| gains | vector<float> | Gain in dB<br>**Range:** [-60,60] | [0] |
| name_e | string | Name of the AC variable for saving the prediction error | E |
| name_f | string | Name of the AC variable for saving the adapive filter | F |
| name_lpc | string | Name of the AC variable for the LPC coefficients | lpc |
| lpc_order | int | Length of the lpc filter<br>**Range:** ]0,1024] | 20 |
| pred_err_delay | vector<int> | Delay in the forward path<br>**Range:** ]0,[ | [96] |
| delay_w | vector<int> | Delay in the adaptive filtering path due to the microphone and loudspeaker transducers<br>**Range:** [0,[ | [130] |
| delay_d | vector<int> | Delay in the adaptive filtering path for the LPC<br>**Range:** [0,[ | [161] |
| n_no_update | int | Number of iterations without updating the filter coefficients<br>**Range:** [0,1024[ | 0 |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

# 7 Plugin category 'filter'

## 7.1 fftfilter

FFT based FIR filter

### 7.1.1 Detailed description

The 'fftfilter' plugin implements a generic FFT-based FIR filter. The overlap-save method is used to apply the impulse response to each block of the signal. The default FFT length used is computed from the fragsize and the inpulse response length and set to the minimum required FFT length to perform the overlap-save operation (see documentation of configuration variable `fftlen`). If this is not a power of two, the computation may be inefficient, and it should be considered to increase it to the next power of two larger than the required minimum.

### 7.1.2 Supported domains

The MHA plugin `fftfilter` supports these signal domains:

- waveform to waveform

### 7.1.3 Plugin Tags

*filter*

© 2005-2019 HörTech gGmbH, Oldenburg

#### 7.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| irs | matrix<float> | Impulse responses, one row for each channel (or single row to use in all channels) | [[1]] |
| fftlen | int | FFT length used for FIR filter. If zero, the FFT length is fragsize + impulse response length - 1 (assuming that the discrete Dirac delta function has the IRS length 1).<br>**Range:** [0,] | 0 |
| fftlen_final | int | FFT length used by FFT filter (computed during prepare) | (monitor) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

### 7.2 iirfilter

IIR filter

#### 7.2.1 Detailed description

The 'iirfilter' plugin implements a generic IIR filter (direct form II). The coefficients have the same names as in MATLAB. Due to different internal implementations and numeric resolutions, filters may be instable with coeffients which are stable in MATLAB.

© 2005-2019 HörTech gGmbH, Oldenburg

#### 7.2.2 Supported domains

The MHA plugin `iirfilter` supports these signal domains:

- waveform to waveform

#### 7.2.3 Plugin Tags

*filter*

#### 7.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| A | vector<float> | recursive filter coefficients | [1] |
| B | vector<float> | non-recursive filter coefficients | [1] |

### 7.3 mconv

FFT based FIR filter using partitioned convolution This plugin filters its input channels using partitioned fast convolution. The variables in this plugin define a sparse matrix of impulse responses. The number of elements in the vectors inch and outch and the number of rows in irs have to be equal.

#### 7.3.1 Detailed description

The plugin *mconv* performs partitioned convolution, using a sparse matrix of impulse responses.

The partition size used for the partitioned convolution is equal to fragsize, the number of samples per channel in one block of audio. The impulse responses are separated into partitions, and each partition is applied with the appropriate delay. Each partition is applied using theoverlap-save method. The FFT length used is 2*fragsize.For efficiency reasons, fragsize should be a power of two.

Partitioned convolution is used to reduce the overall algorithmic delay of the convolution with a long impulse response. The overall computational cost of using partitioned convolution is higher when compared to convolving the complete impulse response in a single overlap-save operation, because longer FFTs are more efficient.

This implementation discards impulse response partitions where the coefficients are all zero.

© 2005-2019 HörTech gGmbH, Oldenburg

### 7.3.2 Supported domains

The MHA plugin `mconv` supports these signal domains:

- waveform to waveform

### 7.3.3 Plugin Tags

*filter*

### 7.3.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| nchannels_out | int | Number of output channels to produce **Range:** [0,[ | 1 |
| inch | vector<int> | Vector of input channel indices. Each element in this vector identifies the input channel to which to apply the corresponding impulse response in irs. **Range:** [0,[ | [0] |
| outch | vector<int> | Vector of output channel indices. Each element in this vector identifies the output channel to which the result of filtering with the corresponding impulse response in irs is mixed. **Range:** [0,[ | [0] |
| irs | matrix<float> | Impulse responses, one per row. For each row, the corresponding element of inch identifies the source channel, and the corresponding element of outch identifies the target channel. | [[1]] |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

## 7.4 steerbf

Steerable Beamformer

### 7.4.1 Detailed description

Implements frequency-domain beamformer processing (filter and sum) using externally provided filters. A plugin called `acSteer` can be used to provide the filter coefficients. The filter coefficients to be read are saved as a waveform object in the AC space. Each channel of this object corresponds to a different steering angle. The steering angle is typically determined in real-time by a localization plugin (e.g. `doasvm_classification`). In this case, the index to the corresponding steering direction is read from the AC space. Note that the number of available filters should be consistent with the number of possible steering directions to be estimated. The configuration variable **angle_src** keeps the name of the AC variable for the estimated steering direction. The steering angle can also be fixed in the configuration time using the configuration variable **angle_ind**.

### 7.4.2 Supported domains

The MHA plugin `steerbf` supports these signal domains:

- spectrum to spectrum

### 7.4.3 Plugin Tags

*filter* spatial audio-channels beamformer binaural

### 7.4.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| bf_src | string | Provides the beamforming filters encoded as a block matrix: [chanXnangle,nfreq]. | |
| angle_ind | int | Sets the steering angle in filtering. **Range:** [0,1000] | 0 |
| angle_src | string | If initialized, provides an int-AC variable of steering index. | |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 7.5 transducers

Signal level calibration plugin.

### 7.5.1 Detailed description

Some plugins in the MHA expect the input signal to be calibrated to sound pressure level in Pascal. This plugin converts AD and DA converter levels to SPL in Pa and also allows for a FIR filters for mircophone and receiver equalization.

#### A schematic calibration rule for the MHA

1. Measure frequency response of hearing aid microphones and receiver.
2. Create FIR filter coefficients for frequency response equalization for microphones and receiver, configure the FIR coefficients of this plugin correspondingly.
3. Play an acoustic reference signal of a known SPL level to the microphone, adjust the 'calib_in.peaklevel' variable until the internal level meter (e.g. rmslevel, p. 73) shows the same level.
4. Create a test tone in the MHA (e.g. with 'noise', p. 92, or 'sine', p. 95) of a given level, and adjust the variable 'calib_out.peaklevel' until the same acoustic level is measured at the receiver.

Besides the signal calibration, this plugin also contains a soft-limiter in the output path, and a quantization module. The soft-limiter acts as a fast broadband compressor, and can be configured correspondingly. The quantisation module limits the signal to the interval $[-1, 1]$ and optionally reduces the resolution, by this quantization rule:

$$y = \mathrm{floor}(2^{(N-1)}x)2^{-(N-1)} \tag{13}$$

$N$ is the number of bits, $x$ the input signal and $y$ the output signal.

© 2005-2019 HörTech gGmbH, Oldenburg

### 7.5.2 Supported domains

The MHA plugin `transducers` supports these signal domains:

- waveform to waveform

### 7.5.3 Plugin Tags

*filter* limiter calibration level-meter

### 7.5.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| plugin_name | string | Plugin name | |
| calib_in | parser | calibration module | (see below) |
| calib_out | parser | calibration module | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `calib_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| nbits | int | Number of bits to simulate, or zero for limiting only<br>**Range:** [0,32] | 0 |
| fir | matrix<float> | FIR filter coefficients, one row for each channel | [[]] |
| peaklevel | vector<float> | Reference peak level in dB (0 dB FS corresponds to this SPL level) | [] |
| speechnoise | parser | | (see below) |
| tau_level | float | Time constant in seconds for RMS level meter<br>**Range:** ]0,10] | 0.125 |
| rmslevel | vector<float> | RMS level in dB at input (after calibration or addition of noise) | (monitor) |
| config | parser | | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `speechnoise`:

| Name | Type | Description | Default |
|---|---|---|---|
| mode | keyword_list | Playback mode and level of speech shaped noise<br>**Range:** [off on olnoise LTASS_combined LTASS_female LTASS_male white pink brown TEN_SPL TEN_SPL_250_8k TEN_SPL_50_16k sin125 sin250 sin500 sin1k sin2k sin4k sin8k] | off |
| level | float | Test signal level in dB SPL<br>**Range:** [0,120] | 80 |
| channels | vector<int> | Channels where to playb speech noise signal<br>**Range:** [0,] | [] |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `config`:

| Name | Type | Description | Default |
|---|---|---|---|
| srate | float | Actual sampling rate / Hz | (monitor) |
| fragsize | int | Actual fragment size / samples | (monitor) |
| channels | int | Actual number of channels | (monitor) |

Variables of sub-parser `calib_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| speechnoise | parser | | (see below) |
| peaklevel | vector<float> | Reference peak level in dB (0 dB FS corresponds to this SPL level) | [] |
| fir | matrix<float> | FIR filter coefficients, one row for each channel | [[]] |
| softclip | parser | 'Hardware' softclipper | (see below) |
| nbits | int | Number of bits to simulate, or zero for limiting only<br>**Range:** [0,32] | 0 |
| do_clipping | bool | Will the soft/ hard clipping be executed | no |
| tau_level | float | Time constant in seconds for RMS level meter<br>**Range:** ]0,10] | 0.125 |
| rmslevel | vector<float> | RMS level in dB at output (before calibration or addition of noise) | (monitor) |
| config | parser | | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `speechnoise`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| mode | keyword_list | Playback mode and level of speech shaped noise<br>**Range:** [off on olnoise LTASS_combined LTASS_female LTASS_male white pink brown TEN_SPL TEN_SPL_250_8k TEN_SPL_50_16k sin125 sin250 sin500 sin1k sin2k sin4k sin8k] | off |
| level | float | Test signal level in dB SPL<br>**Range:** [0,120] | 80 |
| channels | vector<int> | Channels where to playb speech noise signal<br>**Range:** [0,] | [] |

Variables of sub-parser `softclip`:

| Name | Type | Description | Default |
|---|---|---|---|
| tau_attack | float | attack filter time constant / s<br>**Range:** [0,] | 0.002 |
| tau_decay | float | decay filter time constant / s<br>**Range:** [0,] | 0.005 |
| threshold | float | start point on linear scale (hard clipping at 1.0)<br>**Range:** [0,] | 0.6 |
| hardlimit | float | hard limit<br>**Range:** ]0,] | 1 |
| slope | float | compression factor<br>**Range:** [0,1] | 0.5 |
| linear | bool | input/output function is linear on linear (yes) or logarithmic (no) scale | no |
| tau_clip | float | clipping meter time constant / s<br>**Range:** [0,] | 1 |
| clipped | float | clipped ratio | (monitor) |
| max_clipped | float | maximum allowed clipped ratio<br>**Range:** [0,1] | 1 |

Variables of sub-parser `config`:

| Name | Type | Description | Default |
|---|---|---|---|
| srate | float | Actual sampling rate / Hz | (monitor) |
| fragsize | int | Actual fragment size / samples | (monitor) |
| channels | int | Actual number of channels | (monitor) |

# 8 Plugin category 'filterbank'

## 8.1 fftfbpow

FFT based filterbank analysis with overlapping filters

© 2005-2019 HörTech gGmbH, Oldenburg

### 8.1.1 Detailed description

This plugin implements a filterbank based on FFT spectrum. The power in each filter bank channel is calculated and stored into an AC variable. The input signal is passed through unmodified.

For details on the filter shapes, please see description of plugin `fftfilterbank` (section 8.2 on page 66).

### 8.1.2 Supported domains

The MHA plugin `fftfbpow` supports these signal domains:

- spectrum to spectrum

### 8.1.3 Plugin Tags

*filterbank* feature-extraction level-meter

### 8.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| unit | keyword_list | Frequency unit<br>**Range:** [Hz kHz Oct Oct/3 Bark Erb ERB_Glasberg1990] | Hz |
| f | vector<float> | Frequencies | [] |
| f_hz | vector<float> | Frequencies in Hz | (monitor) |
| fscale | keyword_list | frequency scale of filter bank<br>**Range:** [linear bark log erb ERB_Glasberg1990] | linear |
| ovltype | keyword_list | filter overlap type<br>**Range:** [rect linear hanning exp gauss] | rect |
| plateau | float | relative plateau width<br>**Range:** [0,1[ | 0 |
| ftype | keyword_list | frequency entry type<br>**Range:** [center edge] | center |
| normalize | bool | normalize broadband output amplitude | no |
| fail_on_nonmonotonic | bool | Fail if frequency entries are non-monotonic (otherwise sort) | yes |
| fail_on_unique_bins | bool | Fail if center frequencies share the same FFT bin. | yes |
| cf | vector<float> | final center frequencies in Hz | (monitor) |
| ef | vector<float> | final edge frequencies in Hz | (monitor) |
| cLTASS | vector<float> | Bandwidth level correction for LTASS noise in dB | (monitor) |
| shapes | matrix<float> | Frequency band shapes | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 8.2 fftfilterbank

FFT based filterbank with overlapping filters

### 8.2.1 Detailed description

This plugin implements a linear phase filterbank based on FFT spectrum. Each filter bank channel is stored into an own audio channel. The number of output channels of this plugin is the number of frequency bands times the number of input channels.

Please use the iFFT plugin *spec2wave* (p. 98) to get the waveform signal of the filterbank output. The *matrixmixer* (p. 26) plugin or *combinechannels* (p. 20) can be used for resynthesis.

The filters are calculated by applying filter weights to each FFT bin. These weights (filter shapes) depend on the settings of the `ftype` variable. If `center` is selected, the frequency interval between the lower neighbour center frequency and the desired center frequency is mapped to the interval [-1,0] and between the desired center frequency and the upper neighbour to the interval [0,1]. These mappings are linear on the given frequency scale so that a value of 0.5 denotes the middle between two neighboured center frequencies on the given frequency scale. The filter weights are calculated with the configured crossing function on this interval, see next figure for details. Please note that the filters are not necessarily symmetric (symmetry is achieved only if the center frequencies are equally spaced on the desired frequency scale). The lowest and highest filter channels include the full range from zero to the center frequency or from the center frequency to the nyquist frequency, respectively.

If `edge` is selected, then the frequency axis is transformed to be linear on the desired frequency scale. The interval between two edge frequencies is mapped to [-0.5,0.5]. Now, the filter shape function (rectangular, linear/sawtooth, hanning) is applied to the frequency axis. This results in symmetric filters on the desired frequency scale.

© 2005-2019 HörTech gGmbH, Oldenburg

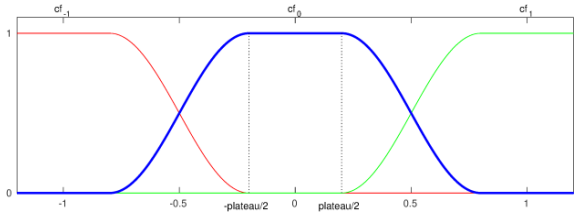


**Figure 4 Schematic plot of overlapping filters**

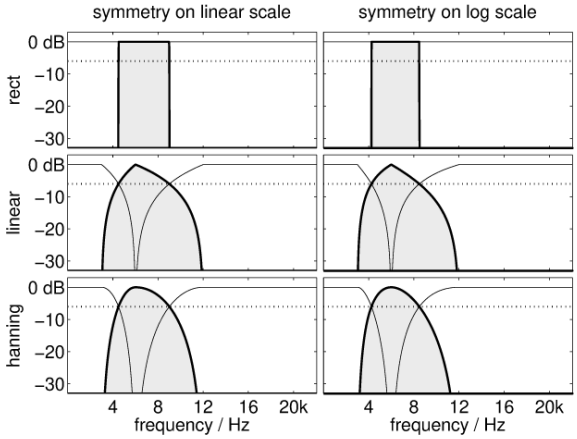


**Figure 5 Example filter shapes with center frequencies configured**

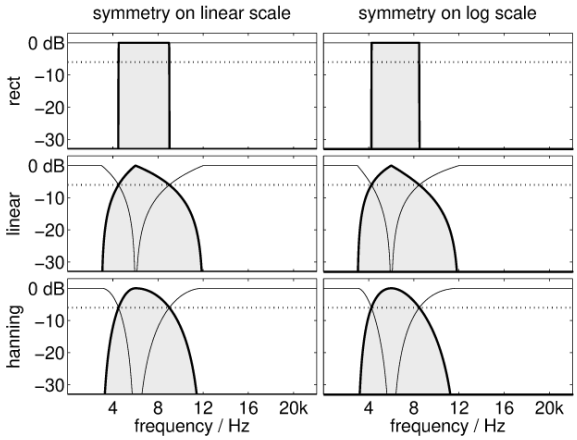


**Figure 6 Example filter shapes with edge frequencies configured**

© 2005-2019 HörTech gGmbH, Oldenburg

### 8.2.2 Supported domains

The MHA plugin `fftfilterbank` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 8.2.3 Plugin Tags

*filterbank*

### 8.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| unit | keyword_list | Frequency unit<br>**Range:** [Hz kHz Oct Oct/3 Bark Erb ERB_Glasberg1990] | Hz |
| f | vector<float> | Frequencies | [] |
| f_hz | vector<float> | Frequencies in Hz | (monitor) |
| fscale | keyword_list | frequency scale of filter bank<br>**Range:** [linear bark log erb ERB_Glasberg1990] | linear |
| ovltype | keyword_list | filter overlap type<br>**Range:** [rect linear hanning exp gauss] | rect |
| plateau | float | relative plateau width<br>**Range:** [0,1[ | 0 |
| ftype | keyword_list | frequency entry type<br>**Range:** [center edge] | center |
| normalize | bool | normalize broadband output amplitude | no |
| fail_on_nonmonotonic | bool | Fail if frequency entries are non-monotonic (otherwise sort) | yes |
| fail_on_unique_bins | bool | Fail if center frequencies share the same FFT bin. | yes |
| cf | vector<float> | final center frequencies in Hz | (monitor) |
| ef | vector<float> | final edge frequencies in Hz | (monitor) |
| cLTASS | vector<float> | Bandwidth level correction for LTASS noise in dB | (monitor) |
| shapes | matrix<float> | Frequency band shapes | (monitor) |
| fftlen | int | FFT length of filterbank (affects time domain only)<br>**Range:** [2,] | 128 |
| phasemodel | keyword_list | Phase model (affects time domain only)<br>**Range:** [minimal linear] | linear |
| irswnd | parser | IRS window function (affects time domain only) | (see below) |
| return_imag | bool | Return imaginary part? Results are stored in AC variable '<plugname>_imag'. | no |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `irswnd`:

| Name | Type | Description | Default |
|---|---|---|---|
| type | keyword_list | Window type. **Range:** [rect hanning hamming blackman bartlett user] | hanning |
| user | vector<float> | User provided window (used if window type==user). | [] |

## 8.3 gtfb_analyzer

Gammatone Filterbank Analyzer

### 8.3.1 Detailed description

Implements a complex-valued gammatone filterbank using cascaded first-order filters as described in Hohmann(2002)[4], and Herzke and Hohmann(2007)[5]. Set the parameter order to the desired gammatone filter order. The coeff is a vector of complex filter coefficients, one for each filterbank frequency band. The complex coefficients need to be computed outside of the MHA, e.g. with the help of the matlab implementation of the gammatone filterbank which can be downloaded from https://uol.de/mediphysik/downloads/. Similarly, the combination of normalization factors and phases also have to be computed outside of the MHA, e.g. also with the same matlab implementation of this gammatone filterbank.

[4] Volker Hohmann, Frequency analysis and synthesis using a Gammatone filterbank. Acta Acustica united with Acustica 88(3), pp. 433-442, 2002.

[5] Tobias Herzke and Volker Hohmann, Improved Numerical Methods for Gammatone Filterbank Analysis and Synthesis. Acta Acustica united with Acustica 93(3), pp. 498-500, 2007.

© 2005-2019 HörTech gGmbH, Oldenburg

The output signal produced by this plugin contains the complex output signal produced by the cascaded gammatone filters in each band. Because the MHA time domain signal representation does not support storing of complex values, real and imaginary parts are stored in different output channels.

Example: If the input has 2 channels (ch0, ch1), and `gtfb_analyzer` splits into 3 bands (b0, b1, b2), then the order of output channels produced by `gtfb_analyzer` is: ch0_b0_real, ch0_b0_imag, ch0_b1_real, ch0_b1_imag, ch0_b2_real, ch0_b2_imag, ch1_b0_real, ch1_b1_imag, ch1_b1_real, ch1_b1_imag, ch1_b2_real, ch1_b2_imag

#### 8.3.2 Supported domains

The MHA plugin `gtfb_analyzer` supports these signal domains:

- waveform to waveform

#### 8.3.3 Plugin Tags

*filterbank*

#### 8.3.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| coeff | vector<complex> | Filter coefficients of gammatone filters | [] |
| norm_phase | vector<complex> | Normalization & phase correction factors | [] |
| order | int | Order of gammatone filters<br>**Range:** [0,[ | 4 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

## 8.4 multibandcompressor

Multiband compressor framework based on level in overlapping filter bands.

### 8.4.1 Detailed description

multibandcompressor provides a complete framework for dynamic range compression in multiple frequency bands.

It contains the same filterbank as the fftfilterbank plugin (see there for documentation of the filterbank) and combines the frequency bands again after the compression.

For the actual dynamic range compression, multibandcompressor can load any other plugin using the field plugin_name. Common choices for this plugin would be dc_simple or dc.

Note that the dynamic range compression receives a pseudo time signal where the sampling rate is the rate of the block processing, i.e. in each channel and band, there is exactly 1 signal sample for every block. These samples are a sparse, non-negative representation of the actual signal in the respective frequency band: The magnitude of each sample is chosen by multibandcompressor so that the level computed from this sparse signal is the same as the level computed from the full signal for this frequency band.

The dynamic range compression will then apply gain (or attenuation) to the sparse signal in each frequency band. The gain applied to the sparse signal is measured by multibandcompressor and eventually applied to the respective full signal.

Before the compressor gain is applied to the full signal, it may be modified by the after-burner built into the multibandcompressor plugin (sub-parser 'burn'): The purpose of the after-burner is to enforce a configurable Maximum Power Output (MPO) for each frequency band, and to compensate for drains and confluxes of sound energy through vents and open fittings. Note that compensating for drains in this way can easily lead to feedback howling and should be done with caution. The after-burner can be disabled by setting burn.bypass=yes.

### 8.4.2 Supported domains

The MHA plugin `multibandcompressor` supports these signal domains:

- spectrum to spectrum

### 8.4.3 Plugin Tags

*filterbank* compression feature-extraction level-modification level-meter

© 2005-2019 HörTech gGmbH, Oldenburg

### 8.4.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| unit | keyword_list | Frequency unit<br>**Range:** [Hz kHz Oct Oct/3 Bark Erb ERB_Glasberg1990] | Hz |
| f | vector<float> | Frequencies | [] |
| f_hz | vector<float> | Frequencies in Hz | (monitor) |
| fscale | keyword_list | frequency scale of filter bank<br>**Range:** [linear bark log erb ERB_Glasberg1990] | linear |
| ovltype | keyword_list | filter overlap type<br>**Range:** [rect linear hanning exp gauss] | rect |
| plateau | float | relative plateau width<br>**Range:** [0,1[ | 0 |
| ftype | keyword_list | frequency entry type<br>**Range:** [center edge] | center |
| normalize | bool | normalize broadband output amplitude | no |
| fail_on_nonmonotonic | bool | Fail if frequency entries are non-monotonic (otherwise sort) | yes |
| fail_on_unique_bins | bool | Fail if center frequencies share the same FFT bin. | yes |
| cf | vector<float> | final center frequencies in Hz | (monitor) |
| ef | vector<float> | final edge frequencies in Hz | (monitor) |
| cLTASS | vector<float> | Bandwidth level correction for LTASS noise in dB | (monitor) |
| shapes | matrix<float> | Frequency band shapes | (monitor) |
| plugin_name | string | Plugin name | |
| burn | parser | | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `burn`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| f | vector<float> | Sample frequencies of data / Hz. **Range:** [0,] | [1000] |
| drain | vector<float> | Drain caused by vent / dB. | [0] |
| conflux | vector<float> | Conflux caused by vent / dB. | [-120] |
| maxgain | vector<float> | Maximum allowed gain / dB. | [80] |
| mpo | vector<float> | Maximum allowed output level / dB SPL (see notes in plugin doc). | [120] |
| taugain | float | Time constant of afterburn gain modifier lowpass / s. **Range:** [0,] | 0.2 |
| commit | keyword_list | Commit changes of configuration variables. **Range:** [commit] | commit |
| bypass | bool | Bypass afterburn stage. | no |

# 9 Plugin category 'level-meter'

## 9.1 rmslevel

This algorithm displays block based RMS level informations. Results are stored in these AC variables (replace 'rmslevel' by the configured plugin name):

rmslevel_level_db rmslevel_peak_db rmslevel_level rmslevel_peak

### 9.1.1 Supported domains

The MHA plugin `rmslevel` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 9.1.2 Plugin Tags

*level-meter* feature-extraction

© 2005-2019 HörTech gGmbH, Oldenburg

#### 9.1.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 10 Plugin category 'level-modification'

### 10.1 gain

Gain plugin:

Apply a gain to each channel

#### 10.1.1 Supported domains

The MHA plugin `gain` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

#### 10.1.2 Plugin Tags

*level-modification*

© 2005-2019 HörTech gGmbH, Oldenburg

### 10.1.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| min | float | Minimal gain<br>**Range:** [,0] | -16 |
| max | float | Maximal gain<br>**Range:** [0,] | 16 |
| bbgain | float | Broadband gain in dB (setting of broad band gain overrides band gain)<br>**Range:** [-16,16] | 0 |
| gains | vector<float> | Gain in dB<br>**Range:** [-16,16] | [0] |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 10.2 smoothgains_bridge

Gain smoothing for reduction of filter length

### 10.2.1 Detailed description

The overlap-add framework allows filter lengths of the zero padding length. Longer filters will result in artifacts caused by circular aliasing. Artifacts can be reduced by either applying Hanning ramps to the zero-padded blocks after filtering, or by shortening the impulse response of the filter, and thus implicitely reducing the frequency resolution. This plugin reduces the filter length to match exactly the zero-padding length. It can either keep the phase (mode=linear_phase), and reduce causal and a-causal parts of the impulse response, or apply a minimum phase filter phase, and cut the causal part of the filter. The window position in the overlap-add framework has to be configured appropriately: For linear phase mode, a symmetric window position is required, i.e., wnd.pos=0.5. To allow minimal phase filters, an asymmetric window position (wnd.pos=0) is needed. Using minimal phase filters will destroy the phase, but reduces the algorithmic delay.

© 2005-2019 HörTech gGmbH, Oldenburg

#### 10.2.2 Supported domains

The MHA plugin `smoothgains_bridge` supports these signal domains:

- spectrum to spectrum

#### 10.2.3 Plugin Tags

*level-modification* filter data-flow overlap-add

#### 10.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| plugin_name | string | Plugin name | |
| mode | keyword_list | Gain smoothing mode<br>Note: Appropriate settings of window position are required (linear_phase: 0.5, minimal_phase: 0)<br>**Range:** [off linear_phase minimal_phase] | linear_phase |
| irswnd | parser | Impulse response window function | (see below) |
| epsilon | float | Epsilon for safe division by zero (avoid inf)<br>**Range:** [1.1e-19,] | 1e-18 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `irswnd`:

| Name | Type | Description | Default |
|---|---|---|---|
| type | keyword_list | Window type.<br>**Range:** [rect hanning hamming blackman bartlett user] | hanning |
| user | vector<float> | User provided window (used if window type==user). | [] |

© 2005-2019 HörTech gGmbH, Oldenburg

# 11 Plugin category 'math'

## 11.1 acTransform_wave

Transform Plugin Between Coordinate Systems for Waveforms

### 11.1.1 Detailed description

This plugin transforms a waveform in the AC space from one coordinate system into another. For this it receives an angle also saved in the AC space. Then, the plugin rotates the axes into the direction of the given angle.

### 11.1.2 Supported domains

The MHA plugin `acTransform_wave` supports these signal domains:

- waveform to waveform

### 11.1.3 Plugin Tags

*math* linear-algebra

### 11.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| ang_name | string | This parameter has the name of the AC variable having the rotation angle | head_ang |
| raw_p_name | string | This parameter has the name of the AC variable having the waveform to be rotated | p |
| raw_p_max_name | string | This parameter has the name of the AC variable having the maximum of the waveform to be rotated | p_max |
| rotated_p_name | string | This parameter has the name of the AC variable having the waveform after rotation | rotated_p |
| rotated_p_max_name | string | This parameter has the name of the AC variable having the maximum of the waveform after rotation | rotated_p_max |
| numsamples | int | This parameter determines the length of the wave to be pooled in samples. **Range:** ]0,360] | 73 |
| to_from | bool | This parameter tells whether the rotation will be performed to the given angle or from it | yes |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

# 12 Plugin category 'noise-suppression'

## 12.1 noise_psd_estimator

Noise power estimator after Gerkmann (2012).

### 12.1.1 Detailed description

Noise power spectral density (PSD) estimator based on a cepstral-domain speech production model using estimated speech presence probability.

The noise PSD estimate is stored into an AC variable named by the algorithm configuration name.

Reference:

Timo Gerkmann, Richard C. Hendriks, "Unbiased MMSE-based Noise Power Estimation with Low Complexity and Low Tracking Delay", IEEE Trans. Audio, Speech and Language Processing, Vol. 20, No. 4, pp. 1383 - 1393, May 2012.

Patent:

Timo Gerkmann and Rainer Martin: "Method for Determining Unbiased Signal Amplitude Estimates After Cepstral Variance Modification", United States Patent US8208666B2, granted Jun. 2012.

© 2005-2019 HörTech gGmbH, Oldenburg

### 12.1.2 Supported domains

The MHA plugin `noise_psd_estimator` supports these signal domains:

- spectrum to spectrum

### 12.1.3 Plugin Tags

*noise-suppression* feature-extraction adaptive

### 12.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| alphaPH1mean | float | low pass filter coefficient for PH1mean **Range:** [0,1[ | 0.9 |
| alphaPSD | float | low pass filter coefficient for PSD **Range:** [0,1[ | 0.8 |
| q | float | a priori probability of speech presence **Range:** [0,1] | 0.5 |
| xiOptDb | float | optimal fixed a priori SNR for SPP estimation | 15 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 12.2 smooth_cepstrum

Cepstral smoothing single-channel noise reduction

© 2005-2019 HörTech gGmbH, Oldenburg

#### 12.2.1 Detailed description

Single-channel noise reduction applying cepstral smoothing based noise power spectral density (PSD). The PSD must be provided by another plugin as an AC variable. The PSD computed by the 'noise_psd_estimator' is compatible with this plugin. The name of the AC variable to read the PSD can be changed in the parameter *noisePow_name*.

References:

Colin Breithaupt, Timo Gerkmann, Rainer Martin, "A Novel A Priori SNR Estimation Approach Based on Selective Cepstro-Temporal Smoothing", IEEE Int. Conf. Acoustics, Speech, Signal Processing, Las Vegas, NV, USA, Apr. 2008.

Timo Gerkmann, Rainer Martin, "On the Statistics of Spectral Amplitudes After Variance Reduction by Temporal Cepstrum Smoothing and Cepstral Nulling", IEEE Trans. Signal Processing, Vol. 57, No. 11, pp. 4165-4174, Nov. 2009.

Patent:

Colin Breithaupt, Timo Gerkmann, and Rainer Martin: "Spectral Smoothing Method for Noisy Signals", European Patent EP2158588B1, granted Oct. 2010, Danish Patent DK2158588T3, granted Feb. 2011, US Patent US8892431B2, granted Nov. 2014.

#### 12.2.2 Supported domains

The MHA plugin `smooth_cepstrum` supports these signal domains:

- spectrum to spectrum

#### 12.2.3 Plugin Tags

*noise-suppression* signal-enhancement adaptive

© 2005-2019 HörTech gGmbH, Oldenburg

### 12.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| xi_min_db | float | Minimum a priori SNR for a bin in dB(power)<br>**Range:** [-50,50] | -27 |
| f0_low | float | Lower limit for F0 detection in Hz<br>**Range:** [0,400] | 70 |
| f0_high | float | Upper limit for F0 detection in Hz<br>**Range:** [0,400] | 300 |
| delta_pitch | float | Quefrency half-width of pitch-set in samps<br>**Range:** [0,20] | 2 |
| lambda_thresh | float | Pitch detection threshold for smooth cepstrum in magnitude<br>**Range:** [0,3] | 0.2 |
| alpha_pitch | float | Alpha value to set for pitch range<br>**Range:** [0,4] | 0.15 |
| beta_const | float | AR coeff for smoothing of alphas(smoothing-factors) | 0.96 |
| kappa_const | float | Exponential bias correction constant for a priori SNR estimate<br>**Range:** [0,1] | 0.2886 |
| gain_min_db | float | Minimum gain in dB for a frequency bin<br>**Range:** [-30,0] | -17 |
| win_f0 | vector<float> | Window coefficients for cepstral smoothing window<br>**Range:** [0,1] | [0.0207 0.0656 0.1664 0.2473 0.24 |
| alpha_const_vals | vector<float> | Piecewise values for steady-state alphas<br>**Range:** [0,2] | [0.2 0.4 0.92] |
| alpha_const_limits_hz | vector<float> | Limits for steady-state alphas given in Hz<br>**Range:** [0,10000] | [93.75 625] |
| noisePow_name | string | Name of est. noise spectrum in AC space | noise_psd_estimator |
| spp | parser | Subparser for exporting SPP | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `spp`:

| Name | Type | Description | Default |
|---|---|---|---|
| prior_q | float | priorQ for computing GLR and SPP from local SNR<br>**Range:** [0,2] | 0.5 |
| xi_opt_db | float | xiOpt in dB for computing GLR and SPP from local SNR<br>**Range:** [0,40] | 15 |

# 13 Plugin category 'plugin-arrangement'

## 13.1 altplugs

Configure alternative plugins.

### 13.1.1 Detailed description

The plugin `altplugs` allows configuration of alternative plugins. The plugin used for processing can be selected via the `select` variable at any time. Any plugins can be used as alternative plugins, with the only limitations that input and output domain and signal dimension is equal for all alternative plugins. Plugins can be renamed using the ":" operator.

This plugin is automatically located by the graphical user interface  t mhacontrol and used for an algorithm selection panel.

### 13.1.2 Supported domains

The MHA plugin `altplugs` supports these signal domains:

- waveform to waveform
- waveform to spectrum
- spectrum to waveform
- spectrum to spectrum

© 2005-2019 HörTech gGmbH, Oldenburg

### 13.1.3 Plugin Tags

*plugin-arrangement* data-flow

### 13.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| use_own_ac | bool | Use own AC space for each plug (yes), or share parents space (no). Must be set before plugs. | no |
| plugs | vector<string> | List of plugins | [] |
| add | string | Add a plugin into list | |
| delete | string | Delete a plugin from list | |
| ramplen | float | Ramp length in seconds **Range:** [0,] | 0 |
| select | keyword_list | Select a plugin for processing **Range:** [(none)] | (none) |
| labels | vector<string> | List of plugin labels. | (monitor) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 13.2 analysispath

Split-up of signal analysis and filtering, with asychronous processing of filter path and thread-safe exchange of filter parameters as AC variables.

© 2005-2019 HörTech gGmbH, Oldenburg

### 13.2.1 Detailed description

In many signal processing scenarios, the signal analysis requires larger block sizes and more processing time than the filtering itself. If the filters do not change rapidly, the filter coefficients can be processed independently from the filter process. This is realized in this plugin: A copy of the input signal is stored in a double buffer, which is then processed asynchronously in a thread with lower priority. At the same time, a snapshot of the AC space (or a subset of it) can be transferred from the analysis thread to the main processing thread.

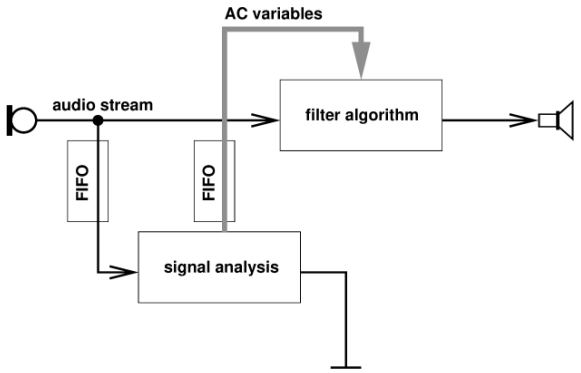


**Figure 7 Schematic signal flow in the analysis path scenario.**

Please note that the AC variables which should be copied to the processing thread must exist after the prepare() callback and should not change their size during run-time.

### 13.2.2 Supported domains

The MHA plugin `analysispath` supports these signal domains:

- waveform to waveform

### 13.2.3 Plugin Tags

*plugin-arrangement* algorithm-communication data-flow

### 13.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| plugname | string | inner plugin name, receives adapted fragment size | |
| fragsize | int | fragment size of inner plugin<br>**Range:** [1,] | 200 |
| fifolen | int | length of double buffer in inner fragment size<br>**Range:** [1,] | 10 |
| priority | int | SCHED_FIFO priority (<0 for no real-time scheduling) | -1 |
| acvars | vector<string> | Names of AC variables to be copied back to processing thread (empty: all) | [] |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 13.3 mhachain

MHA Chain

### 13.3.1 Detailed description

Load a sequence of plugins. During processing, the signal is passed from plugin to plugin, and may change its domain or dimension.

If profiling is switched on, the cumulative time spent in the processing callback of each plugin is stored in a monitor variable.

The complete chain can be replaced by other algorithms during run-time if the default configuration is valid for processing and if the output domain or dimension does not change by replacing the chain.

### 13.3.2 Supported domains

The MHA plugin `mhachain` supports these signal domains:

- waveform to waveform
- waveform to spectrum
- spectrum to waveform
- spectrum to spectrum

© 2005-2019 HörTech gGmbH, Oldenburg

#### 13.3.3 Plugin Tags

*plugin-arrangement* data-flow

#### 13.3.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| use_profiling | bool | use profiling method | no |
| algos | vector<string> | list of plugins | [] |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

### 13.4 overlapadd

Waveform to spectrum overlap add and FFT method. Audio data is collected up to wndlen, than windowed with the given window function, zero padded up to fftlength (symmetric zero padding or asymmetric zero padding possible), and Fast-Fourier-transformed. All parameter changes take effect after the next prepare call.

© 2005-2019 HörTech gGmbH, Oldenburg

### 13.4.1 Detailed description

This plugin transforms fragmented waveform data into short time Fourier transformed audio (STFT), and after processing by spectral processing plugins back to the time domain. This overlap-add mechanism is similar to that from Allen (1977): First, the waveform signal is windowed with a window function, e.g., a Hanning window. In each processing frame, the window is shifted by the fragment size of the input waveform. Missing parts of the signal are taken from the past. The windowed signal is padded with zeros on both sides up to the FFT length, to avoid aliasing when filters are applied in the frequency domain. The impulse response of the applied filter can have the length of the zero padding; if the impulse response is longer, later parts of the impulse response will be mapped to the beginning of the fragment (temporal aliasing). Linear phase filters (real gains in the frequency domain) produce symmetric impulse responses and therefore require symmetric zero padding. The zero padded signal is now fast Fourier transformed. Parameters are FFT length $N$, window length $M$ and the fragment size $P$. Typical values for the window length are $M = 2P$ or $M = 4P$. The Hanning window used in the first step is $w_1(k) = \frac{1}{2}(1 - \cos(2\pi k/M))$, the windowed signal is

$$x_w(m, k) = w_1(k) \cdot x(m \cdot P + k), \tag{14}$$

with $k = 0, \ldots, M - 1$ and the fragment index $m$.

After processing and inverse Fourier transformation, Hanning ramps are applied to the signal to avoid discontinuities in case of temporal aliasing, and thus reducing the artifacts. The length of the Hanning ramps are a fraction $p$ of half the zero-padding length $(N - M)/2$. If $p = 1$, the entire zero-padded parts are smoothed with Hanning ramps. $p = 0$ means, that no Hanning ramps are applied to the signal. This allows an exact reproduction in those cases, where the local impulse response of the filter (represented by all algorithms between FFT and inverse FFT) is shorter than the zero padding length. The windowing in both stages of the overlap-add mechanism is plotted in Fig. 8 for $M = 2P$ (50% overlap).

The total delay between input and output of a real-time system with fragment size $P$ and an overlap-add based linear-phase filter, is the window length plus half the zero-padding length, or $M + (N - M)/2$, plus an additional delay needed for the signal processing, and plus a delay generated by the AD/DA converters (e.g., anti-aliasing filter delay). In an offline system, the complete input signal is available in advance, and thus the delay of the overlap-add method is determined only by the relative shift between output and input signal, which is $(M + N)/2 - P$ (equal to $N/2$ in case of 50% overlap, i.e. $M = 2P$). Contrary to a real-time system, the delay of an offline system depends on the amount of overlap.

### 13.4.2 Supported domains

The MHA plugin `overlapadd` supports these signal domains:

- waveform to waveform

### 13.4.3 Plugin Tags

*plugin-arrangement* signal-transformation overlap-add

© 2005-2019 HörTech gGmbH, Oldenburg

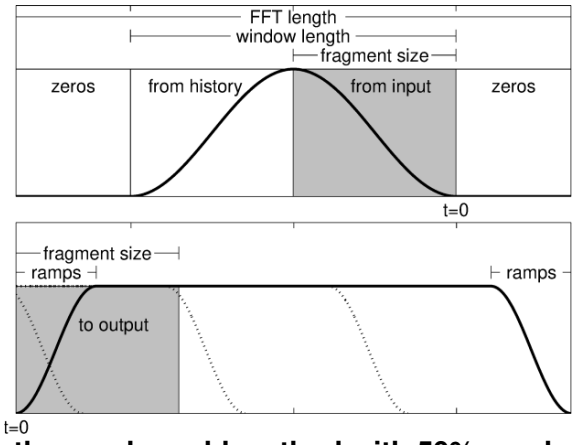


**Figure 8 Windowing in the overlap-add method with 50% overlap and zero-padding. In the upper panel, the windowed input signal before applying the FFT is schematically plotted. In the lower panel, the same time interval after inverse FFT is shown. The shaded segment is the fragment which is read from the input stream (upper panel) and written to the output stream (lower panel) in one processing cycle. The delay between input and output signal is the length of leading zeros plus the window length.**

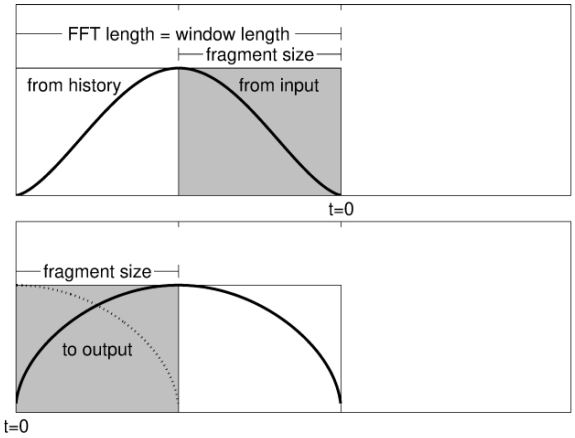


**Figure 9 Windowing in the overlap-add method, as in Fig. 8, but with post-windowing and without zero-padding. In this setup, $W^{\alpha}$ is applied before FFT and $W^{1-\alpha}$ is used for post-windowing. The delay between input and output signal is the window length.**

© 2005-2019 HörTech gGmbH, Oldenburg

### 13.4.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| plugin_name | string | Plugin name | |
| fftlen | int | FFT length<br>**Range:** [1,] | 512 |
| wnd | parser | window type | (see below) |
| zerownd | parser | zero padding post window type | (see below) |
| prescale | float | scaling factor (pre-scaling) | (monitor) |
| postscale | float | scaling factor (post-scaling) | (monitor) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `wnd`:

| Name | Type | Description | Default |
|---|---|---|---|
| type | keyword_list | Window type.<br>**Range:** [rect hanning hamming blackman bartlett user] | hanning |
| user | vector<float> | User provided window (used if window type==user). | [] |
| len | int | window length/samples<br>**Range:** [1,] | 400 |
| pos | float | window position (0 = beginning, 0.5 = symmetric zero padding, 1 = end)<br>**Range:** [0,1] | 0.5 |
| exp | float | window exponent to be applied to all elements of window function | 1 |

Variables of sub-parser `zerownd`:

| Name | Type | Description | Default |
|---|---|---|---|
| type | keyword_list | Window type.<br>**Range:** [rect hanning hamming blackman bartlett user] | rect |
| user | vector<float> | User provided window (used if window type==user). | [] |

© 2005-2019 HörTech gGmbH, Oldenburg

## 13.5 resampling

Synchronous resampling plugin.

### 13.5.1 Detailed description

A bridge type resampling plugin. The signal is converted to target sampling rate and fragment size. The converted signal is processed by the child plugin. The processed signal is then converted back to the original sampling rate and fragment size. The input data is buffered, and the data is processed when enough samples are available.

Please note that double buffering adds an extra delay of the audio stream. If both fragment sizes are identical, the double buffering is bypassed.

#### 13.5.1.1 Warning:

A synchronous resampling ringbuffer such as this causes varying computational loads in the outer processing buffer. It is therefore not real-time safe.

### 13.5.2 Supported domains

The MHA plugin `resampling` supports these signal domains:

- waveform to waveform

### 13.5.3 Plugin Tags

*plugin-arrangement* signal-transformation

### 13.5.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| plugin_name | string | Plugin name | |
| srate | float | sampling rate of client plugin<br>**Range:** ]0,] | 44100 |
| fragsize | int | fragment size of client plugin<br>**Range:** ]0,] | 200 |
| nyquist_ratio | float | lowpass filter cutoff frequency / lower nyquist frequency<br>**Range:** ]0,] | 0.85 |
| irslen_outer2inner | float | filter lenth 1st resampling / sec<br>**Range:** ]0,] | 0.0007 |
| irslen_inner2outer | float | filter lenth 2nd resampling / sec<br>**Range:** ]0,] | 0.0007 |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 13.6 split

Split audio signal into channel groups and have them processed by different plugins in parallel

### 13.6.1 Detailed description

The plugin 'split' takes a multi-channel input signal and splits it up into separate chains of groups of channels. After processing of each chain, the output channels are merged into a multi-channel output signal. The order of the audio channels is left unchanged.

### 13.6.2 Supported domains

The MHA plugin `split` supports these signal domains:

- waveform to waveform
- waveform to spectrum
- spectrum to waveform
- spectrum to spectrum

### 13.6.3 Plugin Tags

*plugin-arrangement* audio-channels data-flow

© 2005-2019 HörTech gGmbH, Oldenburg

### 13.6.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| algos | vector<string> | List of plugins which process the different groups of audio channels. | [] |
| channels | vector<int> | Number of channels processed by the respective plugins. **Range:** [0,[ | [] |
| thread_platform | keyword_list | Thread platform to use. posix is the native linux thread platform, win32 is the native thread platform on windows, dummy means that all processing is performed in a single thread. **Range:** [posix win32 dummy] | posix |
| worker_thread_scheduler | keyword_list | Scheduler used for worker threads. Only used for posix threads. Suggested setting is: The same as present in framework_thread_scheduler after prepare. **Range:** [SCHED_OTHER SCHED_RR SCHED_FIFO] | SCHED_OTHER |
| worker_thread_priority | int | Priority assigned to worker threads. Suggested setting is: The same as present in framework_thread_priority afterpreparing the MHA. The default thread priority given here is invalid. No attempt will be made to set the priority of the threads if this value remains unchanged. | 999999999 |
| framework_thread_scheduler | string | Scheduler used by the framework's processing thread. Only valid after first signal processing callback. | (monitor) |
| framework_thread_priority | int | Priority of the frameworks processing thread. Only valid after first signal processing callback. | (monitor) |
| delay | bool | activates parallel processing of plugins at the cost of one block of additional delay | no |

# 14 Plugin category 'signal-generator'

## 14.1 noise

white noise generator

Waveform and spectral domain are supported. Please note that only in the waveform domain, real continuous white noise is created. In the spectral domain, some modulation and spectral shaping might occur.

© 2005-2019 HörTech gGmbH, Oldenburg

### 14.1.1 Supported domains

The MHA plugin `noise` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 14.1.2 Plugin Tags

*signal-generator*

### 14.1.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| lev | float | noise RMS level in dB SPL | 0 |
| mode | keyword_list | operation mode<br>**Range:** [add replace] | add |
| frozennoise_length | float | Length of frozen noise in s, or 0 for running noise.<br>**Range:** [0,] | 0 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 14.2 plingploing

plingploing algorithm.

© 2005-2019 HörTech gGmbH, Oldenburg

#### 14.2.1 Detailed description

This plugin creates music (jazz-inspired chord sequence).

#### 14.2.2 Supported domains

The MHA plugin `plingploing` supports these signal domains:

- waveform to waveform

#### 14.2.3 Plugin Tags

*signal-generator* music

#### 14.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| level | float | Output level in dB SPL | 70 |
| pitch | float | Bass pitch in Hz<br>**Range:** [1,] | 415 |
| fun1_key | float | minimum interval of second tone relative to bass, in semitones | 3 |
| fun1_range | float | randomized interval of second tone, added to fun1_key, in semitones<br>**Range:** [0,] | 2 |
| fun2_key | float | minimum interval of third tone relative to bass, in semitones | 5 |
| fun2_range | float | randomized interval of third tone, added to fun2_key, in semitones<br>**Range:** [0,] | 2 |
| bpm | float | beats per minute<br>**Range:** [1,] | 200 |
| minlen | float | minimum note length / beats<br>**Range:** [1,] | 1 |
| maxlen | float | maximum note length / beats<br>**Range:** [1,] | 5 |
| bassmod | float | bass key modulation depth | 5 |
| bassperiod | float | bass key modulation period | 28 |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 14.3 sine

Sine wave generator.

### 14.3.1 Supported domains

The MHA plugin `sine` supports these signal domains:

- waveform to waveform

### 14.3.2 Plugin Tags

*signal-generator*

### 14.3.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| lev | float | sine RMS level in dB SPL FF | 0 |
| f | float | Frequency in Hz<br>**Range:** [0,[ | 0 |
| mode | keyword_list | Replace input signal with tone or mix tone into input signal<br>**Range:** [replace mix] | replace |
| channels | vector<int> | List of audio channels to feed with tone (all other audio channels are not affected) | [] |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

# 15 Plugin category 'signal-transformation'

## 15.1 downsample

Downsampling by integer fractions

### 15.1.1 Detailed description

This plugin performs downsampling by an integer factor named `ratio`. The input fragment size needs to be divisible by `ratio`.

As result of the downsammpling, the output signal has a lower sampling rate ($srate$) as well as a smaller fragment size ($fragsize$) with respect to the input signal of the `downsample` plugin (both are divided by the downsampling factor `ratio`).

The signal duration ($T_{signal}$) of the audio blocks processed in each invocation of the `process` callbacks of openMHAplugins is

$$T_{signal} = \frac{fragsize}{srate} = \frac{fragsize/ratio}{srate/ratio}$$

and is not changed by the `downsample` plugin. The total number of invocations of the process method is not modified for downstream plugins by the downsampling.

The downsampling is performed by copying only every n-th audio sample of the low-pass filtered input signal over to the output signal. A low-pass filter is required to reduce aliasing in the output signal and can be configured through the `antialias` configuration setting.

© 2005-2019 HörTech gGmbH, Oldenburg

### 15.1.2 Supported domains

The MHA plugin `downsample` supports these signal domains:

- waveform to waveform

### 15.1.3 Plugin Tags

*signal-transformation* filter

### 15.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| ratio | int | downsampling ratio<br>**Range:** [1,] | 3 |
| antialias | parser | IIR filter structure | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `antialias`:

| Name | Type | Description | Default |
|---|---|---|---|
| A | vector<float> | recursive filter coefficients | [1] |
| B | vector<float> | non-recursive filter coefficients | [1] |

© 2005-2019 HörTech gGmbH, Oldenburg

## 15.2 spec2wave

spectrum to waveform iFFT plugin Performs inverse FFT, postwindowing, hanning ramps at zero-padding, overlap-add, normalization. Note that normalization only works for mod(wndlen,fragsize)=0. Also note that postwindowing only works for wndpos=0.5. Alway set ramplen=0 here if wndpos!=0 in the corresponding wave2spec.

### 15.2.1 Detailed description

This plugin calculates the inverse FFT and overlap add resynthesis. The parameters are taken from the framework overlap add parameters. After the inverse Fourier transform, hanning window ramps are applied to the previously zero-padded regions.

### 15.2.2 Supported domains

The MHA plugin `spec2wave` supports these signal domains:

- spectrum to waveform

### 15.2.3 Plugin Tags

*signal-transformation* overlap-add

### 15.2.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| ramplen | float | Relative length of post windowing hanning ramps (for centered analysis window)<br>**Range:** [0,1] | 1 |
| wndtype | keyword_list | window type<br>**Range:** [rect bartlett hanning hamming blackman user] | rect |
| wndexp | float | window exponent to be applied to all elements of window function | 1 |
| userwnd | vector<float> | user provided window | [] |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 15.3 upsample

Upsampling by integer fractions

### 15.3.1 Detailed description

This plugin performs upsampling by an integer factor named `ratio`.

As result of the upsampling, the output signal has a higher sampling rate ($srate$) as well as a larger fragment size ($fragsize$) with respect to the input signal of the `upsample` plugin (both are multiplied by the upsampling factor `ratio`).

The signal duration ($T_{signal}$) of the audio blocks processed in each invocation of the `process` callbacks of openMHAplugins

$$T_{signal} = \frac{fragsize}{srate} = \frac{fragsize \cdot ratio}{srate \cdot ratio}$$

is not changed by the `upsample` plugin so that the total number of invocations of the process method is not modified for downstream plugins by the upsampling.

The upsampling is performed by spreading consecutive input audio samples to only every n-th sample of the output signal while setting the output samples in between consecutive input samples to value 0. A low-pass filter is required to reduce aliasing in the output signal and can be configured through the `antialias` configuration setting.

### 15.3.2 Supported domains

The MHA plugin `upsample` supports these signal domains:

- waveform to waveform

© 2005-2019 HörTech gGmbH, Oldenburg

### 15.3.3 Plugin Tags

*signal-transformation* filter

### 15.3.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| ratio | int | upsampling ratio<br>**Range:** [1,] | 3 |
| antialias | parser | IIR filter structure | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `antialias`:

| Name | Type | Description | Default |
|---|---|---|---|
| A | vector<float> | recursive filter coefficients | [1] |
| B | vector<float> | non-recursive filter coefficients | [1] |

## 15.4 wave2spec

Waveform to spectrum overlap add and FFT method.

Audio data is collected up to wndlen, then windowed with the given window function, zero padded up to fftlen (symmetric zero padding or asymmetric zero padding possible), and fast-Fourier-transformed (FFT).

© 2005-2019 HörTech gGmbH, Oldenburg

### 15.4.1 Detailed description

This plugin performs the domain transformation from time-domain waveform signal to short-time Fourier transform (STFT) signal in the spectral domain. This plugin can be used as the analysis part of a complete overlap-add procedure. Audio signal data is collected up to the length of the analysis window. The hop-size is equal to the audio block size that this plugin receives. Window size and FFT length are configurable through the configuration variables of this plugin.

The shape of the analysis window is also configurable: Several pre-defined window shapes can be selected, and also a completely user-defined window shape. In addition, a configurable exponent can be applied to all samples of the window before it is used.

During processing, the input data samples are multiplied with the samples of the analysis window, zero padded to the FFT length and Fourier transformed. Please note that usually the window shift is less than the window length. For this reason, the short time fourier transform does not exactly correspond to the current input waveform block: the analysis window contains samples from the current as well as from previous invocation(s). The absolute window shift is identical to the fragment size, e.g. to achieve a window shift of 50%, configure a fragment size of `wndlen`/2.

A copy of the output spectrum is stored in the AC space in a variable of same name as the configured plugin name. To access the spectrum in AC space from the code of self-developed other plugins, the function `MHA_AC::get_var_spectrum()` can be used. See the openMHAdeveloper manual or the header file `mha_algo_comm.h` for details.

Please refer to section 13.4 for a description of the overlap-add method that is also followed by this plugin, `wave2spec`.

Example configurations for the `wave2spec` plugin are available in the short-time-fourier-transform examples directory, and in the matlab/octave tests exercising this plugin in the `mhatest` directory. These test files are executed together with the other system-level tests when invoking `make test`. Please note that you need to have the signal processing package installed in order to sucessfully execute all tests for this plugin.[6]

### 15.4.2 Supported domains

The MHA plugin `wave2spec` supports these signal domains:

- waveform to waveform
- waveform to spectrum

### 15.4.3 Plugin Tags

*signal-transformation* overlap-add

[6] In octave, the package can be installed with `pkg install -forge control signal` from within octave.

© 2005-2019 HörTech gGmbH, Oldenburg

### 15.4.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| fftlen | int | FFT lengths<br>**Range:** [1,] | 512 |
| wndlen | int | window length/samples<br>**Range:** [1,] | 400 |
| wndpos | float | window position (0 = beginning, 0.5 = symmetric zero padding, 1 = end)<br>**Range:** [0,1] | 0.5 |
| wndtype | keyword_list | window type<br>**Range:** [rect bartlett hanning hamming blackman user] | hanning |
| wndexp | float | window exponent to be applied to all elements of window function | 1 |
| userwnd | vector<float> | user provided window | [] |
| return_wave | bool | return input waveform signal, store spectrum only to AC | no |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

# 16 Plugin category 'spatial'

## 16.1 adm

Adaptive differential microphone

© 2005-2019 HörTech gGmbH, Oldenburg

### 16.1.1 Detailed description

This plugin implements one or more adaptive first-order differential microphones, each based on the output of two omnidirectional microphones, e.g. two hearing-aid microphones (cf. Elko & Nguyen Pong, 1995). This is achieved by first subtracting the outputs of the two omnidirectional microphones with fixed delays to create a forward-facing and a backward-facing cardioid microphone, respectively; then, in a second step, the signal from the backward-facing cardioid is amplified by a variable gain factor and subtracted from the signal from the forward-facing cardioid. Finally, a lowpass filter and a filter compensating for comb-filter effect is applied to the output signal.

The gain factor, `beta`, is determined adaptively such that the power of the output signal is minimized, under the constraint that the null of the ADM is located in the rear half-plane. The adaptation step size, `mu_beta`, can be chosen in order to find the optimal combination of adaptation speed and accuracy.

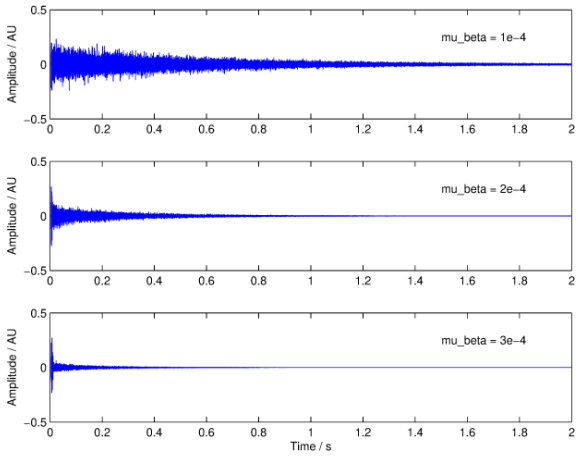


**Figure 10 Output signals illustrating convergence of the ADM algorithm for three different values of `mu_beta` (input signal: white Gaussian noise exactly from behind)**

### 16.1.2 Supported domains

The MHA plugin `adm` supports these signal domains:

- waveform to waveform

### 16.1.3 Plugin Tags

*spatial* signal-enhancement beamformer adaptive

© 2005-2019 HörTech gGmbH, Oldenburg

#### 16.1.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| front_channels | vector<int> | Channel indices for front microphones **Range:** [0,[ | [0 1] |
| rear_channels | vector<int> | Channel indices for rear microphones **Range:** [0,[ | [2 3] |
| distances | vector<float> | Distance between front and rear microphones **Range:** [0.0008,0.08] | [0.0108 0.0108] |
| lp_order | int | Filter order of FIR lowpass filter **Range:** [46,128] | 46 |
| decomb_order | int | Filter order of FIR comb compensation filter **Range:** [46,128] | 54 |
| bypass | int | If 1, output front microphones directly; if 2, output rear microphones directly **Range:** [0,2] | 0 |
| beta | float | Explicit fixed beta (-1 for adaptive filtering) | -1 |
| mu_beta | vector<float> | Adaptation step size for each set of ADMs (e.g. left and right) **Range:** [0,] | [0.0001 0.0001] |
| tau_beta | vector<float> | time constant / s of low pass filter for averaging power of output signal (used for adaptation) **Range:** [0,] | [0.05 0.05] |
| coeff_lp | vector<float> | Lowpass coefficients | (monitor) |
| coeff_decomb | vector<float> | Decomb coefficients | (monitor) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

### 16.2 coherence

Coherence filter

© 2005-2019 HörTech gGmbH, Oldenburg

### 16.2.1 Supported domains

The MHA plugin `coherence` supports these signal domains:

- spectrum to spectrum

### 16.2.2 Plugin Tags

*spatial* signal-enhancement dereverberation adaptive

© 2005-2019 HörTech gGmbH, Oldenburg

### 16.2.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| unit | keyword_list | Frequency unit<br>**Range:** [Hz kHz Oct Oct/3 Bark Erb ERB_Glasberg1990] | Hz |
| f | vector<float> | Frequencies | [] |
| f_hz | vector<float> | Frequencies in Hz | (monitor) |
| fscale | keyword_list | frequency scale of filter bank<br>**Range:** [linear bark log erb ERB_Glasberg1990] | linear |
| ovltype | keyword_list | filter overlap type<br>**Range:** [rect linear hanning exp gauss] | rect |
| plateau | float | relative plateau width<br>**Range:** [0,1[ | 0 |
| ftype | keyword_list | frequency entry type<br>**Range:** [center edge] | center |
| normalize | bool | normalize broadband output amplitude | no |
| fail_on_nonmonotonic | bool | Fail if frequency entries are non-monotonic (otherwise sort) | yes |
| fail_on_unique_bins | bool | Fail if center frequencies share the same FFT bin. | yes |
| cf | vector<float> | final center frequencies in Hz | (monitor) |
| ef | vector<float> | final edge frequencies in Hz | (monitor) |
| cLTASS | vector<float> | Bandwidth level correction for LTASS noise in dB | (monitor) |
| shapes | matrix<float> | Frequency band shapes | (monitor) |
| tau_unit | keyword_list | tau unit<br>**Range:** [seconds periods] | seconds |
| tau | vector<float> | Averaging time constant<br>**Range:** [0,] | [0.04] |
| alpha | vector<float> | Gain exponent<br>**Range:** [0,] | [1] |
| limit | float | gain limit / dB (zero: no limit)<br>**Range:** [,0] | 0 |
| mapping | vector<float> | mapping interval of coherence estimator to coherence (min max)<br>**Range:** [0,1] | [0 1] |
| average | keyword_list | average mode<br>**Range:** [ipd spec] | ipd |
| invert | bool | Invert filter after mapping, before exponent. | no |
| ltgcomp | bool | Long term gain compensation? | no |
| ltgtau | vector<float> | Long term gain estimation time constant / s<br>**Range:** [0,] | [1] |
| staticgain | vector<float> | Static gain in frequency bands / dB | [0] |
| delay | int | Delay between analysis and filter (delay of gains), in fragments.<br>**Range:** [0,] | 0 |

Variables of sub-parser `mhaconfig_in`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 16.3 delaysum

delay and sum plugin. Mixes all channels into a single output channel after applying channel-specific weights and delays.

### 16.3.1 Detailed description

This plugin allows to delay and sum multiple input channels using individual delays and weights. After each channel is delayed it is multiplied with the given weight and then added to the single output channel.

### 16.3.2 Supported domains

The MHA plugin `delaysum` supports these signal domains:

- waveform to waveform

### 16.3.3 Plugin Tags

*spatial* beamformer

© 2005-2019 HörTech gGmbH, Oldenburg

#### 16.3.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| weights | vector<float> | weights of channels. Each entry is multiplied to its respective channel. Needs one entry per channel. | [1 1] |
| delay | vector<int> | delay in number of frames. The nth channel is delayed by the number of frames found in the nth entry. **Range:** [0,] | [0 0] |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

### 16.4 doasvm_classification

Support vector machine (SVM) plugin for computing the direction of arrival (DOA) probabilities

#### 16.4.1 Detailed description

This plugin loads the parameters of a pre-trained SVM and computes the probabilities for given range of directions of arrival (DOA). These probabilities take a value within the interval of $[0, 1]$. Higher probability for a certain DOA indicates higher possibility of a source coming from that particular DOA.

#### 16.4.2 Supported domains

The MHA plugin `doasvm_classification` supports these signal domains:

- waveform to waveform

© 2005-2019 HörTech gGmbH, Oldenburg

### 16.4.3 Plugin Tags

*spatial* classifier binaural

### 16.4.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| angles | vector<float> | The angles for which the SVM model has been trained | [] |
| w | matrix<float> | The separation planes of the model. | [[]] |
| b | vector<float> | The model bias. | [] |
| x | vector<float> | The sigmoid probability mapping parameter x. | [] |
| y | vector<float> | The sigmoid probability mapping parameter y. | [] |
| p_name | string | The name of the AC variable for the vector of probabilities of the DOA estimation. | p |
| max_p_ind_name | string | The name of the AC variable for the index of the maximum probability of the DOA estimation | p_max |
| vGCC_name | string | The name of the AC variable for the GCC matrix, which is computed by another plugin | vGCC_ac |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

## 16.5 doasvm_feature_extraction

Plugin for computing the generalized cross correlation with phase transform (GCC-PHAT)

© 2005-2019 HörTech gGmbH, Oldenburg

#### 16.5.1 Detailed description

This plugin computes the generalized cross correlation with phase transform (GCC-PHAT). The input to this plugin is a stereo time domain signal. The GCC-PHAT matrix is saved into the AC space.

#### 16.5.2 Supported domains

The MHA plugin `doasvm_feature_extraction` supports these signal domains:

- waveform to waveform

#### 16.5.3 Plugin Tags

*spatial* feature-extraction binaural

#### 16.5.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| fftlen | int | The length of the FFT window<br>**Range:** [0,[ | 160 |
| max_lag | int | Maximum lag in samples between microphones (setup-dependent)<br>**Range:** [0,[ | 20 |
| nupsample | int | The amount the GCC-PHAT spectrum is oversampled<br>**Range:** [0,[ | 4 |
| vGCC_name | string | The name of the AC variable for saving the GCC matrix in | vGCC_ac |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

# 17 Plugin category 'test-tool'

## 17.1 cpuload

cpu load generator. CPU load is proportional to number of channels, number of frames, and factor

### 17.1.1 Supported domains

The MHA plugin `cpuload` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 17.1.2 Plugin Tags

*test-tool*

### 17.1.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| factor | float | cpu load factor. Values $>$ 1 increase cpu load, values $<$ 1 decrease it **Range:** [0,] | 1 |
| use_sine | bool | Whether to use the sine function. If not, table interpolation will be used | yes |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

## 17.2 droptect

Plugin detects dropouts in channels that have a constant spectrum

### 17.2.1 Supported domains

The MHA plugin `droptect` supports these signal domains:

- spectrum to spectrum

### 17.2.2 Plugin Tags

*test-tool*

### 17.2.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| dropouts | vector<int> | Number of dropouts detected since last reset or start | (monitor) |
| consecutive_dropouts | vector<int> | Number of consecutive dropouts | (monitor) |
| blocks | int | Number of blocks processed since last reset or start | (monitor) |
| reset | bool | Setting to yes clears number of dropouts and blocks | no |
| threshold | float | Threshold level in dB. All blocks below this threshold are considered to be dropouts | 50 |
| tau | float | Time constant for filtering power spectra | 0.2 |
| filtered_powspec_mon | matrix<float> | Floating average of power spectrum | (monitor) |
| level_mon | float | current level | (monitor) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

## 17.3 identity

### 17.3.1 Detailed description

The simplest openMHA plugin.

This plugin does not modify the signal.

### 17.3.2 Supported domains

The MHA plugin `identity` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 17.3.3 Plugin Tags

*test-tool*

### 17.3.4 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

## 17.4 testplugin

loads a plugin for testing

### 17.4.1 Supported domains

The MHA plugin `testplugin` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

### 17.4.2 Plugin Tags

*test-tool* feature-extraction

### 17.4.3 Configuration variables

| Name | Type | Description | Default |
|---|---|---|---|
| mhaconfig_in | parser | Input configuration | (see below) |
| mhaconfig_out | parser | Output configuration | (see below) |
| plugin_name | string | Plugin name | |
| config_in | parser | signal domain and dimensions | (see below) |
| config_out | parser | signal domain and dimensions | (see below) |
| ac | parser | Insert and retrieve AC variables | (see below) |
| signal | parser | signal input and output | (see below) |
| prepare | bool | for preparing/releasing the loaded plugin | no |

Variables of sub-parser `mhaconfig_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

Variables of sub-parser `mhaconfig_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels | (monitor) |
| domain | string | Signal domain (MHA_WAVEFORM or MHA_SPECTRUM) | (monitor) |
| fragsize | int | Fragment size of waveform data | (monitor) |
| wndlen | int | Window length of spectral data | (monitor) |
| fftlen | int | FFT length of spectral data | (monitor) |
| srate | float | Sampling rate in Hz | (monitor) |

© 2005-2019 HörTech gGmbH, Oldenburg

Variables of sub-parser `config_in`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels<br>**Range:** [0,] | 2 |
| domain | keyword_list | Signal domain<br>**Range:** [MHA_WAVEFORM MHA_SPECTRUM] | MHA_WAVEFORM |
| fragsize | int | Fragment size of waveform data<br>**Range:** [0,] | 200 |
| wndlen | int | Window length of spectral data<br>**Range:** [0,] | 400 |
| fftlen | int | FFT length of spectral data<br>**Range:** [0,] | 512 |
| srate | float | Sampling rate in Hz<br>**Range:** ]0,] | 44100 |

Variables of sub-parser `config_out`:

| Name | Type | Description | Default |
|---|---|---|---|
| channels | int | Number of audio channels<br>**Range:** [0,] | 2 |
| domain | keyword_list | Signal domain<br>**Range:** [MHA_WAVEFORM MHA_SPECTRUM] | MHA_WAVEFORM |
| fragsize | int | Fragment size of waveform data<br>**Range:** [0,] | 200 |
| wndlen | int | Window length of spectral data<br>**Range:** [0,] | 400 |
| fftlen | int | FFT length of spectral data<br>**Range:** [0,] | 512 |
| srate | float | Sampling rate in Hz<br>**Range:** ]0,] | 44100 |

Variables of sub-parser `ac`:

© 2005-2019 HörTech gGmbH, Oldenburg

| Name | Type | Description | Default |
|---|---|---|---|
| insert_var | string | Setting this inserts an AC variable into the AC space | |
| get_var | string | Setting this retrieves an AC variable from the AC space | |
| data_type | keyword_list | Type of data. No support for MHA_AC_USER and MHA_AC_DOUBLE data access **Range:** [MHA_AC_CHAR MHA_AC_INT MHA_AC_MHAREAL MHA_AC_FLOAT MHA_AC_DOUBLE MHA_AC_MHACOMPLEX unknown] | unknown |
| num_entries | int | Number of entries **Range:** [0,] | 1 |
| stride | int | length of one row (C interpretation) or of one column (Fortran interpretation) **Range:** [0,] | 1 |
| char_data | string | data of ac variable if data_type is MHA_AC_CHAR | |
| int_data | vector<int> | data of ac variable if data_type is MHA_AC_INT | [] |
| float_data | vector<float> | data of ac variable if data_type is MHA_AC_FLOAT or MHA_AC_MHAREAL | [] |
| complex_data | vector<complex> | data of ac variable if data_type is MHA_AC_MHACOMPLEX | [] |

Variables of sub-parser `signal`:

| Name | Type | Description | Default |
|---|---|---|---|
| input_wave | matrix<float> | waveform input signal. Writing data will cause processing | [[]] |
| input_spec | matrix<complex> | spectrum input signal. Writing data will cause processing | [[]] |
| output_wave | matrix<float> | waveform output signal from last processing | (monitor) |
| output_spec | matrix<complex> | spectrum output signal from last processing | (monitor) |

# 18 All plugins tagged 'adaptive'

- *acSteer*: Section 4.1 on page 33
- *adm*: Section 16.1 on page 102
- *coherence*: Section 16.2 on page 104
- *lpc*: Section 6.3 on page 47
- *lpc_bl_predictor*: Section 6.4 on page 48
- *lpc_burg-lattice*: Section 6.5 on page 50
- *nlms_wave*: Section 6.6 on page 51
- *noise_psd_estimator*: Section 12.1 on page 78
- *prediction_error*: Section 6.7 on page 53
- *smooth_cepstrum*: Section 12.2 on page 79

© 2005-2019 HörTech gGmbH, Oldenburg

# 19 All plugins tagged 'algorithm-communication'



# 20 All plugins tagged 'audio-channels'



# 21 All plugins tagged 'beamformer'



© 2005-2019 HörTech gGmbH, Oldenburg

## 22 All plugins tagged 'binaural'

- *acSteer*: Section 4.1 on page 33
- *doasvm_classification*: Section 16.4 on page 108
- *doasvm_feature_extraction*: Section 16.5 on page 109
- *steerbf*: Section 7.4 on page 59

## 23 All plugins tagged 'calibration'

- *transducers*: Section 7.5 on page 60

## 24 All plugins tagged 'classifier'

- *doasvm_classification*: Section 16.4 on page 108

## 25 All plugins tagged 'compression'

- **dc**: Section 1.1 on page 1
- **dc_simple**: Section 1.2 on page 3
- *multibandcompressor*: Section 8.4 on page 71
- **softclip**: Section 1.3 on page 6

## 26 All plugins tagged 'cross-fade'

- *fader_spec*: Section 3.7 on page 24
- *fader_wave*: Section 3.8 on page 25

## 27 All plugins tagged 'data-export'

- **ac2lsl**: Section 2.1 on page 7
- **ac2osc**: Section 2.2 on page 8
- **acmon**: Section 2.3 on page 9
- **acsave**: Section 2.4 on page 10
- **wavrec**: Section 2.5 on page 12

© 2005-2019 HörTech gGmbH, Oldenburg

# 28 All plugins tagged 'data-flow'

- **ac2wave**: Section 3.1 on page 13
- **acConcat_wave**: Section 3.2 on page 14
- **acPooling_wave**: Section 3.3 on page 16
- *altplugs*: Section 13.1 on page 82
- *analysispath*: Section 13.2 on page 83
- **combinechannels**: Section 3.4 on page 20
- **db**: Section 3.5 on page 21
- **delay**: Section 3.6 on page 23
- **fader_spec**: Section 3.7 on page 24
- **fader_wave**: Section 3.8 on page 25
- **matrixmixer**: Section 3.9 on page 26
- *mhachain*: Section 13.3 on page 85
- **route**: Section 3.10 on page 28
- **save_spec**: Section 3.11 on page 29
- **save_wave**: Section 3.12 on page 30
- **shadowfilter_begin**: Section 3.13 on page 31
- **shadowfilter_end**: Section 3.14 on page 32
- *smoothgains_bridge*: Section 10.2 on page 75
- *split*: Section 13.6 on page 91

# 29 All plugins tagged 'data-import'

- **acSteer**: Section 4.1 on page 33
- **addsndfile**: Section 4.2 on page 35

# 30 All plugins tagged 'dereverberation'

- *coherence*: Section 16.2 on page 104

© 2005-2019 HörTech gGmbH, Oldenburg

## 31 All plugins tagged 'disk-files'

- *acSteer*: Section 4.1 on page 33
- *acsave*: Section 2.4 on page 10
- *addsndfile*: Section 4.2 on page 35
- *wavrec*: Section 2.5 on page 12

## 32 All plugins tagged 'example'

- **example1**: Section 5.1 on page 38
- **example2**: Section 5.2 on page 39
- **example3**: Section 5.3 on page 40
- **example4**: Section 5.4 on page 41
- **example5**: Section 5.5 on page 42
- **example6**: Section 5.6 on page 43

## 33 All plugins tagged 'feature-extraction'

- *acPooling_wave*: Section 3.3 on page 16
- *doasvm_feature_extraction*: Section 16.5 on page 109
- *example6*: Section 5.6 on page 43
- *fftfbpow*: Section 8.1 on page 64
- *multibandcompressor*: Section 8.4 on page 71
- *noise_psd_estimator*: Section 12.1 on page 78
- *rmslevel*: Section 9.1 on page 73
- *shadowfilter_begin*: Section 3.13 on page 31
- *shadowfilter_end*: Section 3.14 on page 32
- *testplugin*: Section 17.4 on page 114

© 2005-2019 HörTech gGmbH, Oldenburg

## 34 All plugins tagged 'feedback-suppression'

- **fshift**: Section 6.1 on page 44
- **fshift_hilbert**: Section 6.2 on page 45
- **lpc**: Section 6.3 on page 47
- **lpc_bl_predictor**: Section 6.4 on page 48
- **lpc_burg-lattice**: Section 6.5 on page 50
- **nlms_wave**: Section 6.6 on page 51
- **prediction_error**: Section 6.7 on page 53

## 35 All plugins tagged 'filter'

- *downsample*: Section 15.1 on page 96
- **fftfilter**: Section 7.1 on page 55
- **iirfilter**: Section 7.2 on page 56
- **mconv**: Section 7.3 on page 57
- *shadowfilter_begin*: Section 3.13 on page 31
- *shadowfilter_end*: Section 3.14 on page 32
- *smoothgains_bridge*: Section 10.2 on page 75
- **steerbf**: Section 7.4 on page 59
- **transducers**: Section 7.5 on page 60
- *upsample*: Section 15.3 on page 99

## 36 All plugins tagged 'filterbank'

- *combinechannels*: Section 3.4 on page 20
- **fftfbpow**: Section 8.1 on page 64
- **fftfilterbank**: Section 8.2 on page 66
- **gtfb_analyzer**: Section 8.3 on page 69
- **multibandcompressor**: Section 8.4 on page 71

© 2005-2019 HörTech gGmbH, Oldenburg

## 37 All plugins tagged 'frequency-modification'

- *fshift*: Section 6.1 on page 44
- *fshift_hilbert*: Section 6.2 on page 45

## 38 All plugins tagged 'lab-streaming-layer'

- *ac2lsl*: Section 2.1 on page 7

## 39 All plugins tagged 'level-meter'

- *fftfbpow*: Section 8.1 on page 64
- *multibandcompressor*: Section 8.4 on page 71
- **rmslevel**: Section 9.1 on page 73
- *transducers*: Section 7.5 on page 60

## 40 All plugins tagged 'level-modification'

- *dc*: Section 1.1 on page 1
- *dc_simple*: Section 1.2 on page 3
- *example1*: Section 5.1 on page 38
- *example2*: Section 5.2 on page 39
- *example3*: Section 5.3 on page 40
- *example4*: Section 5.4 on page 41
- *example5*: Section 5.5 on page 42
- *fader_spec*: Section 3.7 on page 24
- *fader_wave*: Section 3.8 on page 25
- **gain**: Section 10.1 on page 74
- *multibandcompressor*: Section 8.4 on page 71
- **smoothgains_bridge**: Section 10.2 on page 75
- *softclip*: Section 1.3 on page 6

© 2005-2019 HörTech gGmbH, Oldenburg

## 41 All plugins tagged 'limiter'

- *softclip*: Section 1.3 on page 6
- *transducers*: Section 7.5 on page 60

## 42 All plugins tagged 'linear-algebra'

- *acTransform_wave*: Section 11.1 on page 77

## 43 All plugins tagged 'math'

- **acTransform_wave**: Section 11.1 on page 77

## 44 All plugins tagged 'music'

- *plingploing*: Section 14.2 on page 93

## 45 All plugins tagged 'network-communication'

- *ac2lsl*: Section 2.1 on page 7
- *ac2osc*: Section 2.2 on page 8
- *acmon*: Section 2.3 on page 9

## 46 All plugins tagged 'noise-suppression'

- **noise_psd_estimator**: Section 12.1 on page 78
- **smooth_cepstrum**: Section 12.2 on page 79

## 47 All plugins tagged 'open-sound-control'

- *ac2osc*: Section 2.2 on page 8

© 2005-2019 HörTech gGmbH, Oldenburg

## 48 All plugins tagged 'overlap-add'

- *overlapadd*: Section 13.4 on page 86
- *smoothgains_bridge*: Section 10.2 on page 75
- *spec2wave*: Section 15.2 on page 98
- *wave2spec*: Section 15.4 on page 100

## 49 All plugins tagged 'plugin-arrangement'

- **altplugs**: Section 13.1 on page 82
- **analysispath**: Section 13.2 on page 83
- **mhachain**: Section 13.3 on page 85
- **overlapadd**: Section 13.4 on page 86
- **resampling**: Section 13.5 on page 90
- **split**: Section 13.6 on page 91

## 50 All plugins tagged 'signal-enhancement'

- *adm*: Section 16.1 on page 102
- *coherence*: Section 16.2 on page 104
- *smooth_cepstrum*: Section 12.2 on page 79

## 51 All plugins tagged 'signal-generator'

- *addsndfile*: Section 4.2 on page 35
- **noise**: Section 14.1 on page 92
- **plingploing**: Section 14.2 on page 93
- **sine**: Section 14.3 on page 95

© 2005-2019 HörTech gGmbH, Oldenburg

## 52 All plugins tagged 'signal-transformation'

- *db*: Section 3.5 on page 21
- *delay*: Section 3.6 on page 23
- **downsample**: Section 15.1 on page 96
- *overlapadd*: Section 13.4 on page 86
- *resampling*: Section 13.5 on page 90
- **spec2wave**: Section 15.2 on page 98
- **upsample**: Section 15.3 on page 99
- **wave2spec**: Section 15.4 on page 100

## 53 All plugins tagged 'spatial'

- **adm**: Section 16.1 on page 102
- **coherence**: Section 16.2 on page 104
- **delaysum**: Section 16.3 on page 107
- **doasvm_classification**: Section 16.4 on page 108
- **doasvm_feature_extraction**: Section 16.5 on page 109
- *steerbf*: Section 7.4 on page 59

## 54 All plugins tagged 'test-tool'

- **cpuload**: Section 17.1 on page 111
- **droptect**: Section 17.2 on page 112
- **identity**: Section 17.3 on page 113
- **testplugin**: Section 17.4 on page 114

## References

© 2005-2019 HörTech gGmbH, Oldenburg

## Index

© 2005-2019 HörTech gGmbH, Oldenburg

© 2005-2019 HörTech gGmbH, Oldenburg

© 2005-2019 HörTech gGmbH, Oldenburg