The Open Master Hearing Aid (openMHA)

4.16.0

Documentation of openMHA plugins (openMHA)

HörTech
Kompetenzzentrum für Hörgeräte-Systemtechnik

The Open Master Hearing Aid (openMHA) – Documentation of openMHA plugins (open-MHA)

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

Contents

1 Plugin category 'adaptive'

1.1 gsc_adaptive_stage

Frequency-domain block-adaptive filter specialised for usage as gsc adaptive stage

1.1.1 Detailed description

Implements the FIR-filter block-adaptation scheme based on the NLMS optimization found in John J. Shynk, Frequency-Domain and Multirate Adaptive Filtering, IEEE Signal Processing Magazine, 1992, with specialisations to be used as the adaptive filter stage of an adaptive MVDR beamformer as described in Baumgaertel, et al. 2015. Comparing Binaural Pre-processing Strategies I: Instrumental Evaluation. Trends in hearing, 19, 2331216515617916.

1.1.2 Supported domains

The MHA plugin `gsc_adaptive_stage` supports these signal domains:

- waveform to waveform

1.1.3 Plugin Tags

adaptive filter

1.1.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
lenOldSamps	int	how many old samples to buffer **Range:** [0,5000]	1024
doCircularComp	bool	whether to compensate for circular convolution	no
mu	float	step size for gradient computation **Range:** [0,2]	0.2
alp	float	autoregressive coefficient for estimating PSD **Range:** [0,1]	0.5
useVAD	bool	whether to use the VAD given in AC-variable	no
vadName	string	Name of VAD AC-variable	VAD

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 Plugin category 'beamforming'

2.1 delaysum_spec

simple delay and sum with single channel output

2.1.1 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.

2.1.2 Supported domains

The MHA plugin `delaysum_spec` supports these signal domains:

- spectrum to spectrum

2.1.3 Plugin Tags

beamforming directional multichannel

2.1.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
groupdelay	vector<float>	Group delay in seconds. Positive values represent a delay. One entry for each audio channel	[0 0]
gain	vector<float>	weights of channels. Each entry is multiplied to its respective channel. Needs one entry per channel.	[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)

2.2 rohBeam

Rohdenburg binaural beamformer

2.2.1 Detailed description

Implements the binaural beamformer found in the PhD thesis of Thomas Rohdenburg. Please see Chpt. 4,"Multi-Channel Noise Reduction Schemes with Binaural Output- Performance Evaluation and Optimization" for an overview.

Rohdenburg's strategy is to use standard mono beamforming techniques, but to use the mono output to estimate a linear time-varying filter that approximately equalizes PSD of the reference input channels to the beamformer output. This beamformer is implemented as a spectral domain plugin in MHA. This algorithm has three main parts:

1. Fixed monoaural superdirectional beamformer based on Minimum Variance Distortionless Response (MVDR). This uses three ingredients to determine the fixed filter spectrum in each input channels:

a. Azimuth angle of the speech source relative to the listener.

b. Propogation vector that models the transfer function from source to each mic. There are two options, head model HM1, or loading a HRTF from a WAV file.

c. Noise cross-correlation matrix, allows superdirectivity.

d. To avoid overamplification of noise in MVDR, you can either use a fixed diagonal loading constant or an iterative per-frequency WNG limited algorithm to choose the loading constants.

2. Adaptive Generalized Sidechain (GSC) adaptive beamformer.

This step estimates cross-correlations of output of (1) with inputs,using a blocking strategy to distinguish signal from noise. This component probably doesn't work as expected and might be dangerous; please disable this unless you really know what you are doing.

3. Binaural output adaptation:

a. The preferred strategy, Binaural Postfilter, estimates PSD of mono beamformed output and reference LR channels, then chooses and applies an equalizing filter that splits the difference between LR channels.

b. One of the other strategies, phase reconstruction, is implemented for comparison.

c. It is also an option to simply output the monaural beamformer output.

d. The third strategy, bilateral beamforming is not supported, but can be easily implemented by configuring two rohBeams.

Plausible features not implemented (but could be added) are:

a. Free Field for propogation vector. However you could do this by rendering your own arbitrary HRTF and loading it via the "sampled" option for the propogation vector.

b. Null directions for the MVDR recipe.

2.2.2 Supported domains

The MHA plugin `rohBeam` supports these signal domains:

- spectrum to spectrum

2.2.3 Plugin Tags

beamforming binaural

2.2.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
prop_type	keyword_list	Propogation vector type. **Range:** [hm1 sampled]	hm1
sampled_hrir_path	string	Path for a sampled hrir for the propogation vector.	
source_azimuth_degrees	float	Azimuth angle for the speech source. **Range:** [-180,180]	0
mic_azimuth_degrees_vec	vector<float>	Azimuth angle for each mic, as vector. **Range:** [-180,180]	[0 0 0 0 0 0]
head_model_sphere_radius_cm	float	Radius size of head model in meter **Range:** [0,100]	8.2
intermic_distance_cm	matrix<float>	Intermic distances in cm **Range:** [0,100]	[[0 0 0 0 0 0];[0 0 0 0 0 0
noise_field_model	keyword_list	Noise field model **Range:** [uncorr diff2D diff3D intHRTF]	uncorr
enable_adaptive_beam	bool	Whether adaptive beamformer is active.	no
binaural_type	keyword_list	Binaural adaptation type **Range:** [mono bin_pr bin_pf]	bin_pf
diag_loading_mu	float	Diagonal loading constant mu for design of fixed beamformer. **Range:** [0,2]	0.1
enable_export	bool	Whether filter design is exported as AC variables.	no
enable_wng_optimization	bool	Whether beamform design uses white noise gain optimization.	no
tau_postfilter_ms	float	Smoothing time constant for postfilter in milliseconds. **Range:** [1,5000]	100
tau_blocking_XkXi_ms	float	Time constant for estimation of noise cross-PSD in blocked signal. **Range:** [1,5000]	30
tau_blocking_XkY_ms	float	Time constant for estimation of filtered with blocked noise cross-PSD. **Range:** [1,5000]	30

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 'compression'

3.1 dc

dynamic compression

3.1.1 Detailed description

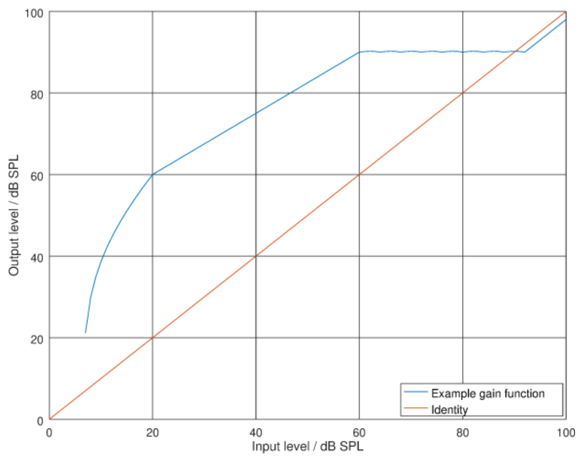


Figure 1 Input-output function of one channel in the dc dynamic compression algorithm.

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, which separates broadband audio channels into multiple frequency bands.

The input-level dependent gain function is configured by a gain table containing the gain values in dB applied in different channels and frequency bands. When a gain table is read by the *dc* plugin the gains contained in the gain table are converted from dB gains to linear factors. The variable `log_interp` controls if the gain values are interpolated linearly or logarithmically, the default is linear interpolation. For input levels outside of the range covered by the gain table an extrapolation depending on the variable `log_interp` on the two nearest points is used.

The linear interpolation of gains originally given in dB can have undesired interpolation effects especially for large step sizes `gtstep`. For step sizes `gtstep` significantly larger than 1dB, these undesired interpolation effects should be avoided by switching `log_interp`. We provide the mfile tool `dc_plot_io.m` which can be used to plot the resulting input/output characteristic resulting from a *dc* gain table configuration with Octave/Matlab, refer to the inline help in that mfile.

The following configuration fragment e.g. reproduces the I/O characteristic shown in Fig. 1. To keep the example readable, a gtstep size of 4 dB was used, reducing the amount of numbers to give here and avoiding fractional dB gains. An actual configuration should use a step size of 1 dB and not avoid fractions. The fitting GUI can be used to configure the dc plugins, and it uses a 1 dB step size. Refer to the fitting GUI manual openMHA_gui_manual.pdf.

```
algos = [fftfilterbank dc combinechannels]
fftfilterbank.ftype = center
fftfilterbank.f = [200 2000]
fftfilterbank.ovltype = rect
dc.gtmin=[16 16]
dc.gtstep=[4 4]
dc.gtdata = [[37 40 39 38 37 36 35 34 33 32 31 30 26 22 18 14 10 6 2 -2 -2];...
             [37 40 39 38 37 36 35 34 33 32 31 30 26 22 18 14 10 6 2 -2 -2];]
dc.tau_attack = [0.005 0.005]
dc.tau_decay =  [0.060 0.060]
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. This configuration applies 40 dB gain at 20 dB SPL input level, and 30 dB at 60 dB input level. Above 60 dB input level, it limits the output level to 90 dB SPL until the output level is softer than the input level. At low input levels below 20 dB SPL, it applies a noise gate. This example configuration is not a fitting recommendation. Established fitting rules should be used to derive individual fittings for test subjects depending on the individual hearing impairment.

dc allows to specify band-specific input level adjustments that are applied to the measured input levels in the respective bands through the configuration variable `level_offset`. Using this variable, it is e.g. possible to compute the gain table in LTASS levels rather than physical band levels.

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.

$$L_a = \langle 20\log_{10}(L_{st})\rangle_{\tau_{attack}} \tag{1}$$
$$L_{in} = \max(L_a, \langle L_a\rangle_{\tau_{release}}) \tag{2}$$

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 $x_n = gtmin + gtstep \cdot n$.

3.1.2 Supported domains

The MHA plugin `dc` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

3.1.3 Plugin Tags

compression level-modification

3.1.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
gtmin	vector<float>	a vector containing one entry for each channel/band which is the input sound level in dB SPL for which first gain in the corresponding row of gain table gtdata is applied to amplify the signal	[]
gtstep	vector<float>	input sound level difference in dB between table columns in the corresponding row of gain table gtdata. I.e. the first entry in each gtdata row is applied when input level is gtmin dB, the second entry is applied when input level is gtmin+gtstep dB, etc. A small step size (e.g. 1 dB) is recommended to avoid undesired effects of the linear interpolation	[]
gtdata	matrix<float>	gaintable data with gains in dB. Each row in this matrix contains gains for one channel or band. Internally, the dB gains of this table are converted to linear gain factors, and interpolated and extrapolated linearly between mesh points.	[[]]
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	[]
level_offset	vector<float>	level offset for each band in dB. If this vector is non-empty, the computed input level are adjusted by the offset values in this vector before the gains are looked up in the gaintable.	[]
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
log_interp	bool	use logarithmic interpolation of gaintable entries	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 of last block 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)

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 dc_simple

Simple dynamic compression scheme

3.2.1 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.

$$L_a = \langle 20\log_{10}(L_{st})\rangle_{\tau_{attack}} \tag{3}$$
$$L_{in} = \max(L_a, \langle L_a\rangle_{\tau_{release}}) \tag{4}$$

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:

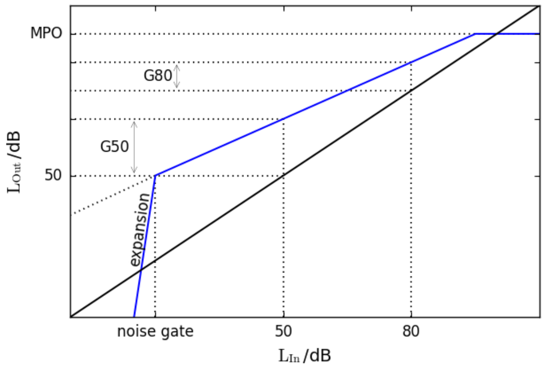


Figure 2 Input-output function of one channel in the *dc_simple* dynamic compression algorithm.

```
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.

3.2.2 Supported domains

The MHA plugin `dc_simple` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

3.2.3 Plugin Tags

compression level-modification

3.2.4 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)

3.3 softclip

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

3.3.1 Supported domains

The MHA plugin `softclip` supports these signal domains:

- waveform to waveform

3.3.2 Plugin Tags

compression limiter level-modification

3.3.3 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 **Range:** [0,[	0.05
tau_attack	float	time constant of attack filter **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) **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)

4 Plugin category 'data-export'

4.1 ac2lsl

Send AC variables as LSL messages.

4.1.1 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.

4.1.2 Supported domains

The MHA plugin `ac2lsl` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

4.1.3 Plugin Tags

data-export network-communication lab-streaming-layer

4.1.4 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`:

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 ac2osc

Send AC variables as OSC messages over udp.

4.2.1 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]
```

4.2.2 Supported domains

The MHA plugin `ac2osc` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

4.2.3 Plugin Tags

data-export network-communication open-sound-control

4.2.4 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 **Range:** [rec pause]	pause
skip	int	number of frames to skip after sending **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)

4.3 acmon

This algorithm converts AC variables into parsable monitor variables.

4.3.1 Supported domains

The MHA plugin `acmon` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

4.3.2 Plugin Tags

data-export network-communication

4.3.3 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 **Range:** [vector matrix]	vector
recmode	keyword_list	record mode **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)

4.4 acrec

ac variable file recorder

4.4.1 Detailed description

AC variable file recorder plugin. This plugin writes the values of an ac variable to a file in a thread-safe manner. It always records the value that is current when its process callback is called, i.e. if an ac variable is written to by multiple plugins, only the final values are committed to file, intermediary values are lost. A new data file is opened every time the record variable is set to yes. The file is closed on any of "cmd=release", "cmd=quit" or "record=no". Note that "cmd=stop" does not close the data file. After the the close command is given, it can take an unspecified, but usually small amount amount of time until the file is actually closed and ready for further processing. The name (and path) of the output file is chosen by the prefix configuration variable. By default the current date and time and the file name extension ".dat" are appended to the file name, this behaviour can be influenced by the "use_date" variable. The date format follows ISO 8601 extended format omitting colons and time zone information, so e.g. November 5, 1994, 8:15:30 corresponds to 1994-11-05T081530. Only AC variables of numeric types can be stored into a file by this plugin. Regardless of the data type of the AC variable, the data is converted to data type double and stored as binary data in host byte order. Complex data are stored storing real part and imaginary part consecutively. No metadata is stored in the file.

The AC variable may change the number of elements that it contains from one process call to the next, but its stride (e.g. number of channels or number of bins) must remain constant.

If more data arrives through the process callback than can be written to disk in the same time, then some of the incoming data will have to be discarded before writing to disk continues. This may e.g. happen with slow disks like network drives or SD cards, or with very high data rates.

The "fifolen" and "minwrite" variables control the behaviour of the fifo buffer and should usually remain unchanged.

4.4.2 Supported domains

The MHA plugin `acrec` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

4.4.3 Plugin Tags

data-export disk-files

4.4.4 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 **Range:** [2,]	262144
minwrite	int	Minimal write length (must be less then fifolen) **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
varname	string	Name of AC variable	
record	bool	Recording session. Each write access will finalize the previous recording session. Each write access with value "yes" will start a new recording session into a new or re-opened output file.	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)

4.5 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.

4.5.1 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.

4.5.2 Supported domains

The MHA plugin `acsave` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

4.5.3 Plugin Tags

data-export disk-files

4.5.4 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 **Range:** [txt mat4 m]	txt
name	string	output file name	
reclen	float	maximal recording length in seconds **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`:

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)

4.6 wavrec

wav file recorder

4.6.1 Detailed description

Wave file recorder plugin. This plugin writes the current audio signal to a wave file in a thread-safe manner. A new wave file is opened every time the record variable is set to yes. The file is closed on any of "cmd=release", "cmd=quit" or "record=no". Note that "cmd=stop" does not close the wave file. After the the close command is given, it can take an unspecified, but usually small amount amount of time until the file is actually closed and ready for further processing. The name (and path) of the output file is chosen by the prefix configuration variable. By default the current date and time are appended to the file name, this behaviour can be controlled by the "use_date" variable. The "fifolen" and "minwrite" variables control the behaviour of the fifo buffer and should usually remain unchanged.

4.6.2 Supported domains

The MHA plugin `wavrec` supports these signal domains:

- waveform to waveform

4.6.3 Plugin Tags

data-export disk-files

4.6.4 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 **Range:** [2,]	262144
minwrite	int	Minimal write length (must be less then fifolen) **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
output_sample_format	keyword_list	Output sample format **Range:** [32_bit_float Signed_8_bit_PCM Signed_16_bit_PCM Signed_24_bit_PCM Signed_32_bit_PCM Unsigned_8_bit_PCM 32_bit_float 64_bit_float U-Law A-Law IMA_ADPCM Microsoft_ADPCM GSM_6.10 32kbs_G721_ADPCM 24kbs_G723_ADPCM 12_bit_DWVW 16_bit_DWVW 24_bit_DWVW VOX_ADPCM 16_bit_DPCM 8_bit_DPCM Vorbis 16_bit_ALAC 20_bit_ALAC 24_bit_ALAC 32_bit_ALAC]	32_bit_float

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 Plugin category 'data-flow'

5.1 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. The AC variable and the input waveform have to have the same dimensions.

Spectral input is discarded and replaced by a zero signal.

5.1.1 Supported domains

The MHA plugin `ac2wave` supports these signal domains:

- waveform to waveform
- spectrum to waveform

5.1.2 Plugin Tags

data-flow algorithm-communication

5.1.3 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 **Range:** [0,[	0
delay_ac	int	Delay of AC input signal in fragments **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)

5.2 acConcat_wave

Concatenating two or more waveforms into one

5.2.1 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.

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.

5.2.2 Supported domains

The MHA plugin `acConcat_wave` supports these signal domains:

- waveform to waveform

5.2.3 Plugin Tags

data-flow algorithm-communication

5.2.4 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 **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 **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)

5.3 acPooling_wave

Pooling of several consecutive time frames

5.3.1 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.

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.

```
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
```

5.3.2 Supported domains

The MHA plugin `acPooling_wave` supports these signal domains:

- waveform to waveform

5.3.3 Plugin Tags

data-flow feature-extraction algorithm-communication

5.3.4 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

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)

5.4 ac_mul

Element-wise multiplication expression in style "a * b", where "a" and "b" are AC variables.

5.4.1 Detailed description

The `ac_mul` plugin can be used to multiply two AC variables element-wise and store the result in another AC variable. The result AC variable is named after the configured name of this plugin (see example below). The matrix dimensions of both input AC variables must be identical, and must not change during signal processing. Multiplying two real-valued input matrices results in a real-valued output matrix. If at least one of the input matrices is complex-valued, then the result is also complex-valued.

The names of the AC variables to multiply are extracted from the expression string assigned to this plugin.

Example: in order to compute the squared samples of a time-domain input signal with this plugin, one could include the following excerpt in an openMHA configuration:

```
mhachain.algos = [save_wave:signal ac_mul:squared_signal]
mhachain.squared_signal = signal * signal
```

which will store the squared samples of the signal in AC variable `squared_signal`.

5.4.2 Supported domains

The MHA plugin `ac_mul` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

5.4.3 Plugin Tags

data-flow math algorithm-communication

5.4.4 Configuration

The plugin represents a variable node in the MHA configuration hierarchy.

Type	Description	Default
string	Element-wise multiplication expression in style "a * b", where "a" and "b" are AC variables.	a * b

5.5 ac_proc

AC variable processor.

5.5.1 Detailed description

This plugin can use the content of a real-valued AC variable (scalar, vector or matrix) as a time-domain input of a plugin. A sub-graph for this plugin is created. The return value of the plugin is stored as an AC variable.

A typical usage of this plugin is feature analysis and processing, e.g., for level compression.

5.5.2 Supported domains

The MHA plugin `ac_proc` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

5.5.3 Plugin Tags

data-flow algorithm-communication feature-extraction

5.5.4 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	
input	string	Name of AC variable to use as input (must exist during prepare)	
permute	bool	Permute AC variable?	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)

5.6 combinechannels

Channel combiner

5.6.1 Detailed description

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

The input signal is by default expected to have a non-interleaved channel order, i.e., first all bands of first output channel, then all bands of second channel, etc. This behaviour can be controlled by the "interleaved" configuration variable. It is also possible to apply independent channel-wise and element-wise gains from AC variables to the signal before summation. This can be done by setting the configuration variables "element_gain_name" and "channel_gain_name" variables.

5.6.2 Supported domains

The MHA plugin `combinechannels` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

5.6.3 Plugin Tags

data-flow audio-channels filterbank

5.6.4 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 **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)

5.7 db

Synchronous double buffer plugin.

5.7.1 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.

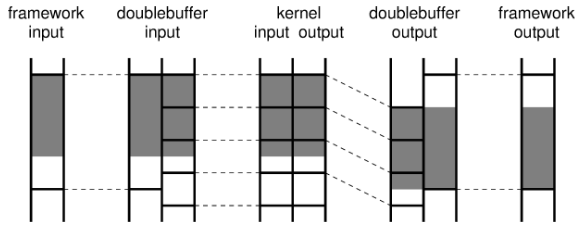


Figure 3 Concept of the double buffer plugin. The outer fragments, provided by the framework, are split up into smaller fragments for processing in the kernel. For a continuous output stream, an extra delay is needed, i.e. the first fragment is filled with zeros at the beginning.

5.7.1.1 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 5.8) 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 17.3).

5.7.2 Supported domains

The MHA plugin `db` supports these signal domains:

- waveform to waveform

5.7.3 Plugin Tags

data-flow signal-transformation

5.7.4 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 **Range:** [0,]	200

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)

5.8 dbasync

Bidirectional fragment size adaptor (double buffer) with asynchronous processing

5.8.1 Detailed description

Bidirectional fragment size adaptor (double buffer) with asynchronous processing

5.8.2 Supported domains

The MHA plugin `dbasync` supports these signal domains:

- waveform to waveform

5.8.3 Plugin Tags

data-flow signal-transformation

5.8.4 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 inner plugin **Range:** [1,]	200
delay	int	algorithmic delay present after bidirectional fragment size adaptation (minimum is inner_fragment_size - gcd(inner_fragment_size, outer_fragment_size) **Range:** [0,]	0
worker_thread_priority	int	Priority assigned to worker threads. Suggested setting is: Something minimally less important than the priority of the framework processing thread (see framework_thread_priority after preparing the MHA). The default thread priority given here is invalid. No attempt will be made to set the priority of the worker thread if this value remains unchanged.	999999999
worker_thread_scheduler	keyword_list	Scheduler used for worker thread. 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
framework_thread_priority	int	Priority of the frameworks processing thread. Only valid after first signal processing callback.	(monitor)
framework_thread_scheduler	string	Scheduler used by the framework's processing thread. Only valid after first signal processing callback.	(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)

5.9 delay

Delay line

5.9.1 Detailed description

Delays the signal by an integer number of samples which is configurable on a per-channel basis.

5.9.2 Supported domains

The MHA plugin `delay` supports these signal domains:

- waveform to waveform

5.9.3 Plugin Tags

data-flow audio-channels signal-transformation

5.9.4 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 **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)

5.10 fader_spec

fader

5.10.1 Supported domains

The MHA plugin `fader_spec` supports these signal domains:

- spectrum to spectrum

5.10.2 Plugin Tags

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

5.10.3 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 **Range:** [0,[	1
gains	vector<float>		[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)

5.11 fader_wave

Apply level

5.11.1 Supported domains

The MHA plugin `fader_wave` supports these signal domains:

- waveform to waveform

5.11.2 Plugin Tags

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

5.11.3 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 **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)

5.12 matrixmixer

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

5.12.1 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`:

```
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`.

[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.

[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.

5.12.2 Supported domains

The MHA plugin `matrixmixer` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

5.12.3 Plugin Tags

data-flow audio-channels

5.12.4 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)

5.13 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.

5.13.1 Supported domains

The MHA plugin `route` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

5.13.2 Plugin Tags

data-flow audio-channels algorithm-communication

5.13.3 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)

5.14 save_spec

Save signal spectrum to AC variable

5.14.1 Detailed description

This plugin saves the spectral signal to an AC variable. The name of the variable is the same as the name of the plugin and can be changed by assigning an alias to the plugin with the usual plugin_name:alias_name syntax.

5.14.2 Supported domains

The MHA plugin `save_spec` supports these signal domains:

- spectrum to spectrum

5.14.3 Plugin Tags

data-flow algorithm-communication

5.14.4 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)

5.15 save_wave

Save signal waveform to AC variable

5.15.1 Detailed description

This plugin saves the waveform signal to an AC variable. The name of the variable is the same as the name of the plugin and can be changed by assigning an alias to the plugin with the usual plugin_name:alias_name syntax.

5.15.2 Supported domains

The MHA plugin `save_wave` supports these signal domains:

- waveform to waveform

5.15.3 Plugin Tags

data-flow algorithm-communication

5.15.4 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)

5.16 shadowfilter_begin

Save signal spectrum to AC variable

5.16.1 Detailed description

The plugins 'shadowfilter_begin' and 'shadowfilter_end' (section 5.17) 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.

5.16.2 Supported domains

The MHA plugin `shadowfilter_begin` supports these signal domains:

- spectrum to spectrum

5.16.3 Plugin Tags

data-flow feature-extraction filter

5.16.4 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 **Range:** [1,[	1
ntracks	int	number of input sources, each with nch audio channels **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)

5.17 shadowfilter_end

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

5.17.1 Detailed description

See section 5.16 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.

5.17.2 Supported domains

The MHA plugin `shadowfilter_end` supports these signal domains:

- spectrum to spectrum

5.17.3 Plugin Tags

data-flow feature-extraction filter

5.17.4 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)

6 Plugin category 'data-import'

6.1 acSteer

Steering Vector Loading Plugin

6.1.1 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

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

6.1.2 Supported domains

The MHA plugin `acSteer` supports these signal domains:

- spectrum to spectrum

6.1.3 Plugin Tags

data-import disk-files beamformer binaural adaptive

6.1.4 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)

6.2 addsndfile

Add sound data from a sound file to the MHA audio channels.

The sound file is read into memory and scaled as determined by configuration variables "level" and "levelmode".

Changing any parameter except "mode" will start playing the file from the beginning. addsndfile stores the complete sound file in RAM memory before starting playback, this limits the total duration of sound files that can be read by addsndfile.

6.2.1 Detailed description

The *addsndfile* plugin modifies the audio signal by either mixing the sound from a sound file to the openMHA audio signal that reaches the *addsndfile* plugin, or by replacing the openMHA audio signal completely with the sound from a sound file. The *addsndfile* plugin to does not change the number of openMHA audio channels.

The playback level of the sound inside the openMHA and the meaning of the `level` variable depend on the settings in the `levelmode` field:

`levelmode=relative` Each sound file channel plays back inside the openMHA at a level of $(\mathrm{level} + L_{fs})$ where level is the value of configuration variable `level` and L_{fs} is the level of the respective channel in dB re full scale inside the sound file.

`levelmode=peak` If `peak` is selected, the level denotes the peak level of the input file: The sound file will be scaled so that the maximum magnitude sound sample will be mapped to an amplitude $10^{\frac{\mathrm{level}}{20}} \cdot 2 \cdot 10^{-5}\mathrm{Pa}$, which is the amplitude of a rectangular wave with that level in dB SPL.

`levelmode=rms` The sound file will be scaled so that it plays back with the level defined by level (in dB SPL) inside openMHA. openMHA uses the same scaling factor for all audio channels of the sound file based on the RMS level across all channels. Therefore, if different channels inside the sound file have different levels re full scale, then the scaling will result in some channels playing back at a softer level, and some at a higher level than level. The inter-channel level difference in openMHA playback levels will be the same as in the sound file.

The sound file may be stored with a different sampling rate than the the sampling rate of the openMHA at the point where the addsndfile plugin is loaded into the processing chain. The parameter `resamplingmode` controls how addsndfile behaves when differing sampling rates are detected in the sound file and in openMHA:

`resamplingmode=dont_resample_permissive` The sound samples from the sound file are played inside openMHA without resampling. The file will play back at too high or too low pitches if the sampling rates in the file and in openMHA differ.

`resamplingmode=dont_resample_strict` The sound from the sound file is only played back inside openMHA if the sampling rates in sound file and openMHA match exactly. If the sampling rates differ, an error is triggered.

`resamplingmode=do_resample` The sound from the sound file is resampled to the openMHA sampling rate if the sampling rates in sound file and openMHA differ.

6.2.2 Supported domains

The MHA plugin `addsndfile` supports these signal domains:

- waveform to waveform

6.2.3 Plugin Tags

data-import disk-files signal-generator

6.2.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
path	string	The directory containing the sound file to read. Should end in the system-specific directory separator, e.g. /, if non-empty. If empty, then the current working directory of the mha process is used. path can be given absolute or relative to the current working directory of the mha process.	
filename	string	File name of the sound file. If empty, addsndfile does not modify the sound.	
loop	bool	Infinitely loop sound file playback	yes
level	float	Level in dB. Exact meaning of this parameter depends on levelmode.	0
levelmode	keyword_list	relative - scaling relative to original level peak - scaling to maximum absolute magnitude rms - scaling to long-term rms level Refer to the detailed description subsection in the plugins manual for details. **Range:** [relative peak rms]	relative
resamplingmode	keyword_list	Resampling mode, for details refer to the detailed description subsection in the plugins manual. **Range:** [dont_resample_permissive dont_resample_strict do_resample]	do_resample
channels	vector<int>	Indices of MHA channels in which to store each of the sound file channels. Indices start from 0. If the sound file has fewer channels than the "channels" vector has elements, then the channels of the file will be duplicated. Note: The addsndfile plugin does not change the number of MHA audio channels. If you specify an MHA channel index >= the number of MHA channels, then that channel from the sound file will not be used. **Range:** [0,]	[0]
mode	keyword_list	add: combine sounds from file and MHA input, replace: discard MHA input, play only file, input: leave MHA input unmodified, mute: no sound output. This parameter can be changed during playback without causing rewind **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 if sound currently plays 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`:

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 double2acvar

Converts configuration variable of type string containing a decimal floating point number literal to algorithm communication variable of type double. Name of the AC variable is the configured algorithm name.

6.3.1 Detailed description

Publishes an AC variable of type `double`. Because the openMHA configuration language does not have configuration variables of type `double`, we are using a configuration variable of type string. The string is converted to double by the C function `atof()` which means:

1. The number format uses the standard C locale.
2. Leading white space is skipped.
3. The string is converted up to the first non-conversible character.
4. Empty strings and strings with only con-conversible characters convert to value 0.0.

6.3.2 Supported domains

The MHA plugin `double2acvar` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

6.3.3 Plugin Tags

data-import

6.3.4 Configuration

The plugin represents a variable node in the MHA configuration hierarchy.

Type	Description	Default
string	Converts configuration variable of type string containing a decimal floating point number literal to algorithm communication variable of type double. Name of the AC variable is the configured algorithm name.	0

6.4 lsl2ac

Receive LSL streams and copy them to AC variables.

6.4.1 Detailed description

This plugin provides a mechanism to receive lsl streams and convert them to ac variables. It currently only supports MHA_AC_FLOAT-type variables. Type conversions from other types of stream should be handled in the background. This is a beta version of the plugin. It is probably real-time safe. An LSL stream named NAME results in the follwing AC variables: NAME containing the data, NAME_ts containing the time stamps, NAME_ts containing the offset between receiver and sender clocks, and NAME_new containing the number of new samples per channel since the last process callback. The size of the AC variables is configurable via the nchannels and nsamples configuration variables. nchannels controls the stride of the AC variable and must be equal to the number of channels of the AC variables or be left on default to accept any number of channels of the LSL stream. nsamples corresponds to the number of samples per channel of the AC variable. Leaving nsamples on default means that the AC variable will be resized according to the number of samples received, up to a maximum of chunksize samples. If the size of the AC variable is fixed and there are less samples available in the LSL buffers than are needed to fill the AC variable, the oldest samples are overwritten and the contents of the AC variable are reordered so that the oldest samples come first. On overrun, i.e. there are more samples available than fit in the AC variable, the user can decide if all samples but the newest should be discarded or if the overrun should be ignored and only the oldest samples should be saved to AC, leaving newer samples in the LSL buffers. Warning: If the overrun behavior is set to discard, the plugin pulls new samples as long as samples are ready for pickup in the LSL buffers. If the sender is considerably faster than the receiver this may cause the plugin to hang indefinitely. The buffer length and chunk size of the LSL inlet are configurable. For more details on the meaning of these variables please consult the LSL documentation. The configuration regarding the AC variable size and the LSL stream inlet applies plugin wide. To use per-stream configuration this plugin must be instantiated multiple times. This implementation should only be used for evaluation purposes and is not yet intended for production use.

6.4.2 Supported domains

The MHA plugin `lsl2ac` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

6.4.3 Plugin Tags

data-import network-communication

6.4.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
streams	vector<string>	List of LSL streams to be saved, empty for all.	[]
activate	bool	Receive from network?	yes
overrun_behavior	keyword_list	How to handle overrun **Range:** [discard ignore]	discard
nchannels	int	The number of channels to expect of the LSL stream, also determines stride of AC variable. Default means sender decides **Range:** [0,2.1475e+09]	0
buffersize	int	The maximum amount of data for LSL to buffer. In seconds if there is a nominal sampling rate, otherwise x100 in samples **Range:** [0,]	360
chunksize	int	The maximum granularity, in samples, at which chunks are transmitted by LSL. Default means sender decides **Range:** [0,]	0
nsamples	int	The number of samples per channel to be stored in the AC variable. Default means the AC variable will be resized as needed to accommodate the samples received, up to a maximum of chunksize. **Range:** [0,]	0
available_streams	vector<string>	List of all available LSL streams	(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)

6.5 osc2ac

Receive OSC messages and convert them to AC variables. Only data type float can be received.

6.5.1 Detailed description

Receive open sound control (OSC) messages and mirror their data in AC variables. This plugin can receive OSC messages provided that they contain only numbers (scalars or vectors).

The configuration variable `vars` can be used to control which OSC messages to receive, and to define the names of the AC variables in which the received data is mirrored inside the openMHA, e.g.:

```
# Example 1:
osc2ac.vars = [spatial/source/0/sphericalCoords spatial/setParam/headRadius]
osc2ac.size = [3 1]
```

Example 1 shows an *osc2ac* configuration that receives OSC messages with addresses `/spatial/source/0/sphericalCoords` and `/spatial/setParam/headRadius`, the first of which is expected to contain 3 floating point values (a vector), while messages with the latter address are expected to contain only 1 floating point value (a scalar) in each received message. The data received at these addresses will be mirrored in the AC variables `spatial/source/0/sphericalCoords` and `spatial/setParam/headRadius`, respectively. Note that the openMHA configuration and the AC variable name do not contain the leading slash `/`. This leading slash is prepended by the *osc2ac* plugin when constructing the OSC address for which messages are received from the `vars` entry.

It is also possible to give AC variable name and OSC address separately by separating them with a colon, e.g.: colon delimiter, e.g.:

```
# Example 2:
vars = [level:/mhalevels]
```

In example 2, data received in OSC messages with address `/mhalevels` is mirrored in the AC variable `level`. When `size` is not set in this example, the default value 1 for scalars is used for all AC variables and OSC messages.

6.5.2 Supported domains

The MHA plugin `osc2ac` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

6.5.3 Plugin Tags

data-import network-communication open-sound-control

6.5.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
host	string	multicast adress (empty for unicast)	
port	string	server port	7777
vars	vector<string>	List of AC variables to provide as receivers of OSC messages. Each AC variable will mirror the latest received OSC message with address /NAME where each NAME is the name of the mirroring AC variable as given here. For more details, please refer to the detailed description subsection in the manual.	[]
size	vector<int>	Number of floats to receive with the AC variables from OSC. Each entry here corresponds to the entry in vars with the same index and determines the length of the float vector that will be allocated to receive the OSC messages with the corresponding address. **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)

7 Plugin category 'example'

7.1 attenuate20

This plugin attenuates by 20dB

7.1.1 Detailed description

Plugin attenuate20 attenuates the input signal by 20dB.

7.1.2 Supported domains

The MHA plugin `attenuate20` supports these signal domains:

- waveform to waveform

7.1.3 Plugin Tags

example level-modification

7.1.4 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)

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 example1

7.2.1 Detailed description

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

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

7.2.2 Supported domains

The MHA plugin `example1` supports these signal domains:

- waveform to waveform

7.2.3 Plugin Tags

example level-modification audio-channels

7.2.4 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)

7.3 example2

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

7.3.1 Supported domains

The MHA plugin `example2` supports these signal domains:

- waveform to waveform

7.3.2 Plugin Tags

example level-modification audio-channels

7.3.3 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. **Range:** [0,[	0
factor	float	The scaling factor that is applied to the selected channel. **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)

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)

7.4 example3

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

7.4.1 Supported domains

The MHA plugin `example3` supports these signal domains:

- waveform to waveform

7.4.2 Plugin Tags

example level-modification audio-channels

7.4.3 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)

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)

7.5 example4

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

7.5.1 Detailed description

This plugin scales one channel of the input signal, working in the spectral domain. The scale factor and the scaled channel number is made accessible to the configuration structure.

7.5.2 Supported domains

The MHA plugin `example4` supports these signal domains:

- spectrum to spectrum

7.5.3 Plugin Tags

example level-modification audio-channels

7.5.4 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`:

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)

7.6 example5

example plugin scaling a spectral signal

7.6.1 Detailed description

This plugin scales one channel of the input signal, working in the spectral domain. The scale factor and the scaled channel number is made accessible to the configuration structure.

7.6.2 Supported domains

The MHA plugin `example5` supports these signal domains:

- spectrum to spectrum

7.6.3 Plugin Tags

example level-modification audio-channels

7.6.4 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 **Range:** [0,[	0
factor	float	scale factor **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)

7.7 example6

Example rms level meter plugin

7.7.1 Detailed description

This plugin calculates the RMS level of a given channel of the input signal, working in the time domain. The channel number is made accessible to the configuration structure and the result is stored into a algorithm communication variable (AC variable).

7.7.2 Supported domains

The MHA plugin `example6` supports these signal domains:

- waveform to waveform

7.7.3 Plugin Tags

example feature-extraction algorithm-communication

7.7.4 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 **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)

7.8 example7

7.8.1 Detailed description

The is again example1 but split into .hh and .cpp-file in order to provide the class declaration to the Unit-TestThis plugin scales one channel of the input signal, working in the time domain.

7.8.2 Supported domains

The MHA plugin `example7` supports these signal domains:

- waveform to waveform

7.8.3 Plugin Tags

example level-modification audio-channels unit-testing

7.8.4 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)

8 Plugin category 'feedback-suppression'

8.1 adaptive_feedback_canceller

Prediction error method for adaptive feedback cancellation

8.1.1 Detailed description

This plugin implements the prediction error method to perform adaptive feedback cancellation.The prediction error method estimates the feedback path by minimizing the error between the desired and the predicted output signals. The algorithm handles each input sample within each input channel separately.

The plugin computes not only the least mean squares (NLMS) estimate of the feedback path but also performs the necessary steps for delaying the input and output signals as well as the prewhitening using the linear predictive coding (LPC) coefficients. The flow graph of the whole processing chain is shown in Figure Fig. 4.

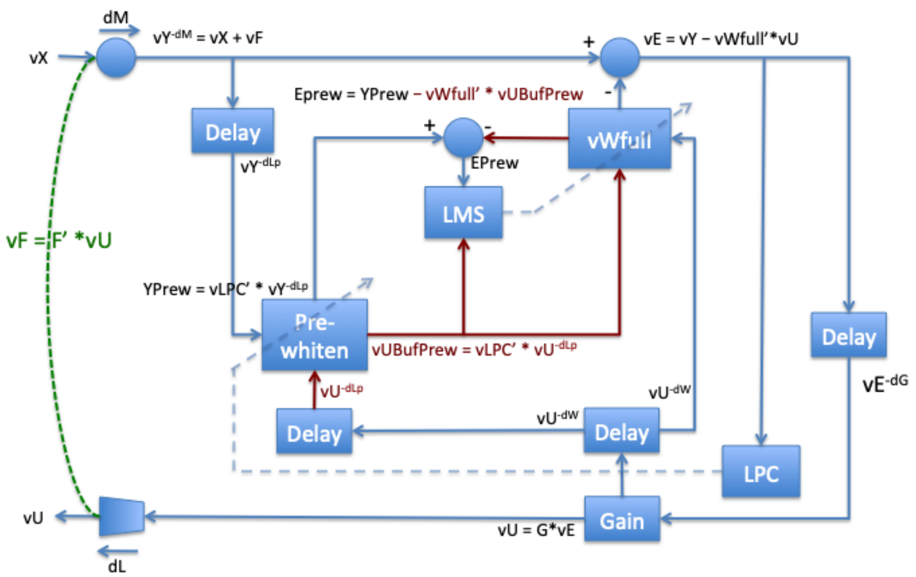


Figure 4 This figure shows the signal flow and the applied operations on the signal to perform adaptive feedback cancellation.

In the forward path, the error signal *vE* is estimated by subtracting the estimated feedback signal of the output signal *vU* of the last iteration from the current input signal *vY* as shown in Equation 6. This error signal is delayed a configurable number of taps, which is set using the configuration variable **pred_err_delay** (*dG* in the figure). Subsequently a configurable amount of gain is applied to the delayed error signal to compute the output signal. The amount of gain can be set using the configuration variable **gains**.

$$e[k] = y[k] - \mathbf{vWfull}^T[k-1] \cdot \mathbf{u}[k-1], \tag{6}$$

In order to compute the feedback signal (the second term in Equation 6), the output signal is delayed a configurable amount of taps, filtered using the last estimate of the feedback path *vWfull*. The aforementioned delay is set using the configuration variable **delay_w** (*dW* in the figure).

In the closed loop identification of the feedback path, the LPC coefficients are used for prewhitening the input signal as well as the output signal. The LPC coefficients are estimated in the plugin called LPC and saved in the AC space, which the prediction_error plugin reads in each iteration (*vLPC* in the figure). The name of the corresponding AC variable must be set using the configuration variable **name_lpc**. Furthermore, the dimensions of the LPC coefficients must be set using the configuration variable **lpc_order**. Prior to the prewhitening step, the input as well as the output signal are delayed the same amount of taps, which can be set by the configuration variable **delay_d** (*dLp* in the figure).

In the following step, the prewhitened error signal is computed as shown in Equation 6. For this, the prewhitened output signal is filtered by the last estimate of the feedback path and subtracted from the prewhitened input signal.

By using the prewhitened output signal and error signal, the NLMS algorithm re-estimates the coefficients of the feedback path 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** (*vWfull* in the figure).

The estimation of the filter coefficients of the feedback path is performed using the update rule given as in the following:

$$\mathbf{vWfull}[k] = \mathbf{vWfull}[k-1] + \rho \cdot \frac{\mathbf{u}[k-1]}{(|\mathbf{u}^T[k-1] \cdot \mathbf{u}[k-1]| + c)} \cdot e[k], \tag{7}$$

where e is the prewhitened error (*EPrew*) signal, and u (*vUBufPrew*) is the output signal. The step size **rho** (ρ in the equation) and the regularization parameter **c** can be set using the corresponding configuration variables.

The default values of the numeric configuration variables are optimized for a sampling rate of 16KHz. They have to be adjusted for other sampling rates for optimal feedback cancellation performance.

8.1.2 Supported domains

The MHA plugin `adaptive_feedback_canceller` supports these signal domains:

- waveform to waveform

8.1.3 Plugin Tags

feedback-suppression adaptive

8.1.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
rho	float	Step size **Range:**]0,2]	0.01
c	float	Regularization parameter **Range:**]0,]	1e-05
ntaps	int	Length of the feedback path filter in taps **Range:**]0,]	32
gains	vector<float>	Gain in dB **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 in taps **Range:**]0,1024]	20
afc_delay	vector<int>	Delay in the forward path in taps **Range:**]0,[	[96]
delay_w	vector<int>	Delay in the adaptive filtering path due to the microphone and loudspeaker transducers in taps **Range:** [0,[	[130]
delay_d	vector<int>	Delay in the adaptive filtering path for the LPC in taps **Range:** [0,[	[161]
n_no_update	int	Number of iterations without updating the filter coefficients **Range:** [0,1024[	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)

8.2 fshift

8.2.1 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.

8.2.2 Supported domains

The MHA plugin `fshift` supports these signal domains:

- spectrum to spectrum

8.2.3 Plugin Tags

feedback-suppression frequency-modification

8.2.4 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 **Range:** [0,]	4000
fmax	float	upper boundary for frequency shifter in Hz **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)

8.3 fshift_hilbert

Pitch shifter

8.3.1 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.

8.3.2 Supported domains

The MHA plugin `fshift_hilbert` supports these signal domains:

- spectrum to spectrum

8.3.3 Plugin Tags

feedback-suppression frequency-modification

8.3.4 Configuration variables

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

Variables of sub-parser `mhaconfig_in`:

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

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)

8.4 lpc

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

8.4.1 Detailed description

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

8.4.2 Supported domains

The MHA plugin `lpc` supports these signal domains:

- waveform to waveform

8.4.3 Plugin Tags

feedback-suppression adaptive

8.4.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
lpc_order	int	LPC filter order **Range:** [0,500]	20
lpc_buffer_size	int	Size of the buffer in samples for which the autocorrelation matrix will be computed **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 **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)

8.5 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.

8.5.1 Detailed description

This plugin computes the forward and backward LPC estimates using the Burg-Lattice algorithm given the κ (sometimes also called μ) 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

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

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

. In this implementation κ from the previous is used. Note that the second index of κ is 2.

8.5.2 Supported domains

The MHA plugin `lpc_bl_predictor` supports these signal domains:

- waveform to waveform

8.5.3 Plugin Tags

feedback-suppression adaptive

8.5.4 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 **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`:

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)

8.6 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.

8.6.1 Detailed description

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

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

$$dm(m-1) = \lambda * dm(m-1) + (1-\lambda) * (f(m-1)^2 + b(m-1,2)^2) \quad (10)$$

$$nm(m-1) = \lambda * nm(m-1) + (1-\lambda) * -2 * f(m-1) * b(m-1,2) \quad (11)$$

$$km(m,1) = \frac{nm(m-1)}{dm(m-1)}. \quad (12)$$

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

8.6.2 Supported domains

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

- waveform to waveform

8.6.3 Plugin Tags

feedback-suppression adaptive

8.6.4 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_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 **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)

8.7 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.

8.7.1 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:

$$\begin{aligned} e[k] &= y[k-1] - f[k-1]u[k-1] && (13)\\ f[k] &= f[k-1] + rho/(|u|^2 + c)u[k-1]e[k], && (14) \end{aligned}$$

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.

8.7.2 Supported domains

The MHA plugin `nlms_wave` supports these signal domains:

- waveform to waveform

8.7.3 Plugin Tags

feedback-suppression adaptive

8.7.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
rho	float	convergence coefficient **Range:**]0,2]	0.01
c	float	stabilization parameter **Range:**]0,]	1e-05
ntaps	int	number of taps in filter **Range:**]0,]	32
name_u	string	Name of input signal U	
name_d	string	Name of desired signal D	
normtype	keyword_list	Normalization type **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. **Range:** [previous current]	previous
lambda_smoothing_power	float	Recursive smoothing constant for sum normalization **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)

9 Plugin category 'filter'

9.1 equalize

Equalizer plugin applies configurable gains to all bins of the spectrum

9.1.1 Detailed description

High resolution gain structure. This plugin allows to apply a bin-wise gain to every bin of the spectrum.

9.1.2 Supported domains

The MHA plugin `equalize` supports these signal domains:

- spectrum to spectrum

9.1.3 Plugin Tags

filter level-modification

9.1.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
gains	matrix<float>	gains in FFT resolution (FFT length/2+1 entries required per row) as linear factors, one row per audio channel **Range:** [0,[	[[]]
id	string	Access to the id feature of the equalize plugin. Usually the equalize plugin exposes no plugin id. This variable allows to set a plugin id. If set to "equalize", then this plugin can be fitted with a linear hearing aid fitting rule	

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)

9.2 fftfilter

FFT based FIR filter

9.2.1 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.

The 'fftfilter' plugin does not introduce additional delay. Regardless of fragsize, length of impulse response, or fft length, the computed output of plugin 'fftfilter' is the same as if the output had been computed by performing the convolution on the same signal blocks in the time domain, except for numerical errors.

9.2.2 Supported domains

The MHA plugin `fftfilter` supports these signal domains:

- waveform to waveform

9.2.3 Plugin Tags

filter

9.2.4 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). **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)

9.3 iirfilter

IIR filter

9.3.1 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.

9.3.2 Supported domains

The MHA plugin `iirfilter` supports these signal domains:

- waveform to waveform

9.3.3 Plugin Tags

filter

9.3.4 Configuration variables

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

9.4 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.

9.4.1 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.

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

9.4.2 Supported domains

The MHA plugin `mconv` supports these signal domains:

- waveform to waveform

9.4.3 Plugin Tags

filter

9.4.4 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)

9.5 steerbf

Steerable Beamformer

9.5.1 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**.

9.5.2 Supported domains

The MHA plugin `steerbf` supports these signal domains:

- spectrum to spectrum

9.5.3 Plugin Tags

filter spatial audio-channels beamformer binaural

9.5.4 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`:

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)

9.6 transducers

Signal level calibration plugin.

9.6.1 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. 115) shows the same level.
4. Create a test tone in the MHA (e.g. with 'noise', p. 143, or 'sine', p. 146) 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:

$$y = \mathrm{floor}(2^{(N-1)}x)2^{-(N-1)} \tag{15}$$

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

9.6.2 Supported domains

The MHA plugin `transducers` supports these signal domains:

- waveform to waveform

9.6.3 Plugin Tags

filter limiter calibration level-meter

9.6.4 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`:

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 **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 **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 **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 **Range:** [0,120]	80
channels	vector<int>	Channels where to playb speech noise signal **Range:** [0,]	[]

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 **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 **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`:

Name	Type	Description	Default
mode	keyword_list	Playback mode and level of speech shaped noise **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 **Range:** [0,120]	80
channels	vector<int>	Channels where to playb speech noise signal **Range:** [0,]	[]

Variables of sub-parser `softclip`:

Name	Type	Description	Default
tau_attack	float	attack filter time constant / s **Range:** [0,]	0.002
tau_decay	float	decay filter time constant / s **Range:** [0,]	0.005
threshold	float	start point on linear scale (hard clipping at 1.0) **Range:** [0,]	0.6
hardlimit	float	hard limit **Range:**]0,]	1
slope	float	compression factor **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 **Range:** [0,]	1
clipped	float	clipped ratio	(monitor)
max_clipped	float	maximum allowed clipped ratio **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)

10 Plugin category 'filterbank'

10.1 fftfbpow

FFT based filterbank analysis with overlapping filters

10.1.1 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 10.2 on page 92).

10.1.2 Supported domains

The MHA plugin `fftfbpow` supports these signal domains:

- spectrum to spectrum

10.1.3 Plugin Tags

filterbank feature-extraction level-meter

10.1.4 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 **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 **Range:** [linear bark log erb ERB_Glasberg1990]	linear
ovltype	keyword_list	filter overlap type **Range:** [rect linear hanning exp gauss]	rect
plateau	float	relative plateau width **Range:** [0,1[	0
ftype	keyword_list	frequency entry type **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
flag_allow_empty_bands	bool	Set true to allow bands where all STFT-bin-gains equal zero.	no
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)

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 fftfilterbank

FFT based filterbank with overlapping filters

10.2.1 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. 149) to get the waveform signal of the filterbank output. The *matrixmixer* (p. 39) plugin or *combinechannels* (p. 31) 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 neightbour 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.

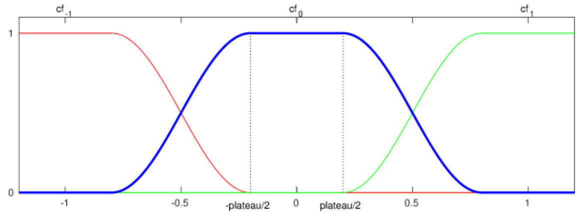


Figure 5 Schematic plot of overlapping filters

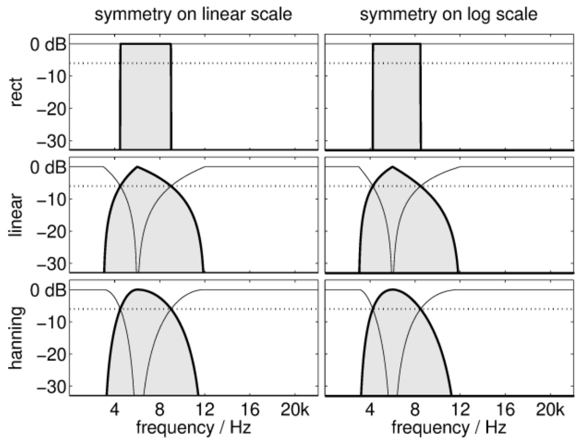


Figure 6 Example filter shapes with center frequencies configured

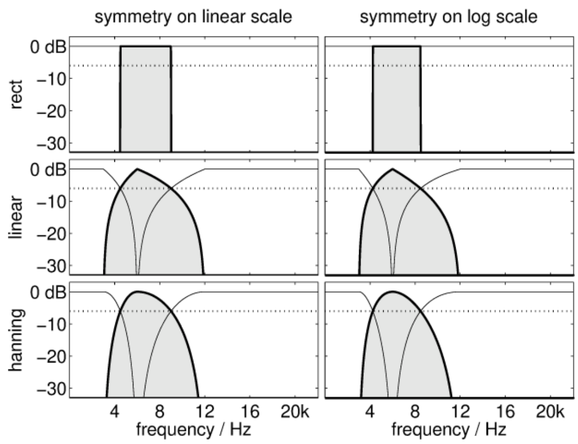


Figure 7 Example filter shapes with edge frequencies configured

10.2.2 Supported domains

The MHA plugin `fftfilterbank` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

10.2.3 Plugin Tags

filterbank

10.2.4 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 **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 **Range:** [linear bark log erb ERB_Glasberg1990]	linear
ovltype	keyword_list	filter overlap type **Range:** [rect linear hanning exp gauss]	rect
plateau	float	relative plateau width **Range:** [0,1[	0
ftype	keyword_list	frequency entry type **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
flag_allow_empty_bands	bool	Set true to allow bands where all STFT-bin-gains equal zero.	no
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) **Range:** [2,]	128
phasemodel	keyword_list	Phase model (affects time domain only) **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

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).	[]

10.3 gtfb_analyzer

Gammatone Filterbank Analyzer

10.3.1 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.

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

10.3.2 Supported domains

The MHA plugin `gtfb_analyzer` supports these signal domains:

- waveform to waveform

10.3.3 Plugin Tags

filterbank

10.3.4 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 **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)

10.4 gtfb_simd

Gammatone Filterbank Analyzer

10.4.1 Detailed description

gtfb_simd implements the same gammatone filterbank as plugin gtfb_analyzer. The gammatone filtering is performed using built-in vector operations of x86.The total number of bands (audio channels x filterbank frequencies) has to bea multiple of 4.

This plugin should be regarded as a proof-of-concept how Single-Instruction-Multiple-Data (SIMD) can be used inside openMHA. For practical gammatone filtering applications, the plugins gtfb_analyzer and gtfb_simple_bridge should be used instead.

10.4.2 Supported domains

The MHA plugin `gtfb_simd` supports these signal domains:

- waveform to waveform

10.4.3 Plugin Tags

filterbank

10.4.4 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 **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)

10.5 gtfb_simple_bridge

Simple gammatone filterbank

10.5.1 Detailed description

Simple gammatone filterbank plugin. Computes complex-valued gammatone filterbank signal from the real-valued broad-band signal, processes the filterbank signal with the plugin loaded via plugin_name, and resynthesizes the output signal again to a real-valued broadband output signal.

The signal in each band can be restricted to the respective frequency band by applying additional gammatone filter stages to the output signal of the loaded plugin.

Gammatone filterbank is implemented after Hohmann 2002 and produces complex-valued analytic output in each frequency band. Frequency bands are presented as audio channels to the loaded plugin. The order of bands is: All bands created from the first input channel form the first nbands audio channel, followed by all bands created from the second input channel, etc.

Real and imaginary signal are presented separately to the loaded plugin: The real part is transferred as the regular MHA audio input signal, while the imaginary part is transferred through an AC variable with the same name as the configured name of this filterbank plugin.

This plugin does not support changing the configuration at run time.

This plugin creates the following AC variables during preparation:

gtfb_simple_bridge_imag waveform matrix, contains the imaginary part of the filtered signal to be processed by the loaded plugin in-place. Size: (fragsize x channels*bands)

gtfb_simple_bridge_cf vector containing the center frequencies of the gammatone filter bands in Hz. Size: (1 x bands)

gtfb_simple_bridge_bw vector containing the bandwidths of the gammatone filter bands in Hz. Size: (1 x bands)

gtfb_simple_bridge_cLTASS vector containing negative LTASS correction values in dB for the gammatone filter bands. Size: (1 x channels*bands)

gtfb_simple_bridge_resyncgain vector containing the per-band resynthesis gains computed by the Hohmann 2002 method (linear factors, applied during resynthesis). Size: (1 x channels*bands)

If the plugin is assigned a different name than gtfb_simple_bridge, then the first parts of the above AC variable names change accordingly.

This plugin can make use of an AC variables with the name given to configutation variable element_gain_name: It expects a real matrix with size (fragsize x channels*bands). If this name is given, then the values given in this matrix are multiplied element-wise with the real and imaginary output signals of the loaded plugin before the filterbank resynthesis is performed.

10.5.2 Supported domains

The MHA plugin `gtfb_simple_bridge` supports these signal domains:

- waveform to waveform

10.5.3 Plugin Tags

filterbank

10.5.4 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	
unit	keyword_list	Frequency unit **Range:** [Hz kHz Oct Oct/3 Bark Erb ERB_Glasberg1990]	Hz
f	vector<float>	Frequencies	[]
f_hz	vector<float>	Frequencies in Hz	(monitor)
bw	vector<float>	Bandwidth **Range:**]0,]	[]
bw_hz	vector<float>	Bandwidth in Hz	(monitor)
order	int	Filterbank order **Range:** [1,]	4
prestages	int	Number of stages to be processed before the plugin **Range:** [0,]	3
desired_delay	int	Desired delay in samples **Range:** [0,]	0
element_gain_name	string	Name of element wise gain AC variable (looked up during waveform process, can be empty)	
cLTASS	vector<float>	Vector of band-specific negative LTASS level correction values in dB: Of a broadband LTASS signal with level X dB, X+cLTASS dB will fall into each band of the gammatone filterbank	(monitor)
resynthesis_gain	vector<float>	Linear gains for resynthesis.	(monitor)
gf_internals	string	internal coefficients of the gammatone filterbank	(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)

10.6 multibandcompressor

Multiband compressor framework based on level in overlapping filter bands.

10.6.1 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.

10.6.2 Supported domains

The MHA plugin `multibandcompressor` supports these signal domains:

- spectrum to spectrum

10.6.3 Plugin Tags

filterbank compression feature-extraction level-modification level-meter

10.6.4 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 **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 **Range:** [linear bark log erb ERB_Glasberg1990]	linear
ovltype	keyword_list	filter overlap type **Range:** [rect linear hanning exp gauss]	rect
plateau	float	relative plateau width **Range:** [0,1[	0
ftype	keyword_list	frequency entry type **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
flag_allow_empty_bands	bool	Set true to allow bands where all STFT-bin-gains equal zero.	no
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`:

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

11 Plugin category 'io'

11.1 MHAIOFile

Sound file IO client.

Read from input files and write to files of same format.

11.1.1 Detailed description

The plugin 'MHAIOFile' provides file to file processing with openMHA. It uses libsndfile to read from and write to sound files. Input and output file name can be configured. After the openMHA host application is started (cmd=start), the whole input file will be processed and the processed data will be written to the output file. The start command will block until the processing is finished. The files are opened when preparing the openMHA host application and closed when releasing the openMHA host application. The output file format is inherited from the input file and the data format of the output file is set by the output_sample_format. By default the format of the input file is also used for the output file, e.g. if the input file is a 32 bit WAVE file, the output file will also be 32-bit WAVE. The plugin supports most commonly used file formats. Nota bene: When writing to WAVE files in integer formats, MHAIOFile clips all values above +1.0 and below -1.0. To avoid clipping use floating point WAVE files.

11.1.2 Supported domains

The MHA plugin `MHAIOFile` supports these signal domains:

- waveform to waveform

11.1.3 Plugin Tags

io

11.1.4 Configuration variables

Name	Type	Description	Default
in	string	Input sound file name	
out	string	Output sound file name	
output_sample_format	keyword_list	Output sample format, or 'input' to copy from input file **Range:** [input Signed_8_bit_PCM Signed_16_bit_PCM Signed_24_bit_PCM Signed_32_bit_PCM Unsigned_8_bit_PCM 32_bit_float 64_bit_float U-Law A-Law IMA_ADPCM Microsoft_ADPCM GSM_6.10 32kbs_G721_ADPCM 24kbs_G723_ADPCM 12_bit_DWVW 16_bit_DWVW 24_bit_DWVW VOX_ADPCM 16_bit_DPCM 8_bit_DPCM Vorbis 16_bit_ALAC 20_bit_ALAC 24_bit_ALAC 32_bit_ALAC]	input
startsample	int	First sample to be processed. **Range:** [0,]	0
length	int	Number of samples to be processed by one start command, or zero for all. **Range:** [0,]	0
strict_channel_match	bool	Require same channel count in MHA and sound file.	yes
strict_srate_match	bool	Require same sample rate in MHA and sound file.	yes

11.2 MHAIOJack

JACK client

11.2.1 Detailed description

JACK client

11.2.2 Supported domains

The MHA plugin `MHAIOJack` supports these signal domains:

- waveform to waveform

11.2.3 Plugin Tags

io

11.2.4 Configuration variables

Name	Type	Description	Default
servername	string	Name of JACK server	default
name	string	Name of JACK client	MHA
con_in	vector<string>	Connections for input ports	[]
delays_in	vector<int>	Input delay in samples as reported by JACK. Only valid after prepare.	(monitor)
con_out	vector<string>	Connections for output ports	[]
delays_out	vector<int>	Output delay in samples as reported by JACK. Only valid after prepare.	(monitor)
names_in	vector<string>	Names of input ports (empty for automatic names)	[]
names_out	vector<string>	Names of output ports (empty for automatic names)	[]
ports	parser	Jack ports	(see below)
state	parser	Jack state.	(see below)

Variables of sub-parser `ports`:

Name	Type	Description	Default
physical_inputs	vector<string>	Physical (hardware) input ports	(monitor)
physical_outputs	vector<string>	Physical (hardware) output ports	(monitor)
all_inputs	vector<string>	All input ports (software/hardware)	(monitor)
all_outputs	vector<string>	All output ports (software/hardware)	(monitor)

Variables of sub-parser `state`:

Name	Type	Description	Default
xruns	int	Number of xruns since first connection of MHA to jack.	(monitor)
cpuload	float	Current CPU load in Jack.	(monitor)
priority	int	Jack thread priority.	(monitor)
scheduler	string	Jack thread scheduler model.	(monitor)

11.3 MHAIOJackdb

JACK client

11.3.1 Detailed description

JACK client

11.3.2 Supported domains

The MHA plugin `MHAIOJackdb` supports these signal domains:

- waveform to waveform

11.3.3 Plugin Tags

io

11.3.4 Configuration variables

Name	Type	Description	Default
servername	string	Name of JACK server	default
name	string	Name of JACK client	MHA
con_in	vector<string>	Connections for input ports	[]
con_out	vector<string>	Connections for output ports	[]
names_in	vector<string>	Names of input ports (empty for automatic names)	[]
names_out	vector<string>	Names of output ports (empty for automatic names)	[]
use_jack_transport	bool	Use jack transport for start/stop control (changes after first start may result in undefined behavior)	no
fail_on_async_jackerr	bool	Should MHA fail to operate if jack reports asynchronouos errors?	yes
locate	float	jack transport locate command (time in seconds) **Range:** [0,]	0
ports	parser	Jack ports	(see below)
server_srate	float	Sampling rate of Jack server.	(monitor)
server_fragsize	int	Fragment size rate of Jack server.	(monitor)
state	parser	Jack state.	(see below)

Variables of sub-parser `ports`:

Name	Type	Description	Default
physical_inputs	vector<string>	Physical (hardware) input ports	(monitor)
physical_outputs	vector<string>	Physical (hardware) output ports	(monitor)
all_inputs	vector<string>	All input ports (software/hardware)	(monitor)
all_outputs	vector<string>	All output ports (software/hardware)	(monitor)

Variables of sub-parser `state`:

Name	Type	Description	Default
xruns	int	Number of xruns since first connection of MHA to jack.	(monitor)
cpuload	float	Current CPU load in Jack.	(monitor)
priority	int	Jack thread priority.	(monitor)
scheduler	string	Jack thread scheduler model.	(monitor)

11.4 MHAIOParser

process data from parser input

11.4.1 Detailed description

process data from parser input

11.4.2 Supported domains

The MHA plugin `MHAIOParser` supports these signal domains:

- waveform to waveform

11.4.3 Plugin Tags

io

11.4.4 Configuration variables

Name	Type	Description	Default
input	matrix<float>	input signal buffer (size: nchannels x fragsize)	[[]]
output	matrix<float>	output signal buffer	(monitor)

11.5 MHAIOPortAudio

MHA IO library for portaudio V19 backend

11.5.1 Detailed description

MHA IO library for portaudio V19 backend

11.5.2 Supported domains

The MHA plugin `MHAIOPortAudio` supports these signal domains:

- waveform to waveform

11.5.3 Plugin Tags

io

11.5.4 Configuration variables

Name	Type	Description	Default
device_info	parser	PortAudio's information about the sound devices present on this system	(see below)
stream_info	parser	PortAudio stream info	(see below)
device_name_in	string	Variable to load device by name. This name has to match the portaudio device name, exactly or as a substring. Exact matches take precedence over substring matches. Thereafter, matches at lower device indices are preferred. device_index_in will be updated when a match is found.	default
device_index_in	int	Variable to load device by index. Upon setting device_index_in, device_name_in will be updated to the full portaudio name of this device. **Range:** [0,0[	0
device_name_out	string	Variable to load device by name. This name has to match the portaudio device name, exactly or as a substring. Exact matches take precedence over substring matches. Thereafter, matches at lower device indices are preferred. device_index_out will be updated when a match is found.	default
device_index_out	int	Variable to load device by index. Upon setting device_index_out, device_name_out will be updated to the full portaudio name of this device. **Range:** [0,0[	0
suggested_input_latency	float	The desired input latency in seconds. **Range:** [0,]	0
suggested_output_latency	float	The desired output latency in seconds. **Range:** [0,]	0

Variables of sub-parser `device_info`:

Name	Type	Description	Default
numDevices	int	Number of sound devices on system as recognized by PortAudio v19	(monitor)
structVersion	vector<int>	PortAudio has no documentation for this field	(monitor)
name	vector<string>	PortAudio has no documentation for this field	(monitor)
hostApi	vector<int>	The type used to enumerate to host APIs at runtime. Values of this type range from 0 to (Pa_GetHostApiCount()-1).	(monitor)
maxInputChannels	vector<int>	PortAudio has no documentation for this field	(monitor)
maxOutputChannels	vector<int>	PortAudio has no documentation for this field	(monitor)
defaultLowInputLatency	vector<float>	Default latency values for interactive performance. Time values in seconds.	(monitor)
defaultLowOutputLatency	vector<float>	PortAudio has no documentation for this field	(monitor)
defaultHighInputLatency	vector<float>	Default latency values for robust non-interactive applications. Time values in seconds.	(monitor)
defaultHighOutputLatency	vector<float>	PortAudio has no documentation for this field	(monitor)
defaultSampleRate	vector<float>	PortAudio has no documentation for this field	(monitor)

Variables of sub-parser `stream_info`:

Name	Type	Description	Default
paInputLatency	float	The input latency of the stream in seconds. This value provides the most accurate estimate of input latency available to the implementation. It may differ significantly from the suggestedLatency value passed to Pa_OpenStream(). The value of this field will be zero (0.) for output-only streams.	(monitor)
paOutputLatency	float	The output latency of the stream in seconds. This value provides the most accurate estimate of output latency available to the implementation. It may differ significantly from the suggestedLatency value passed to Pa_OpenStream(). The value of this field will be zero (0.) for input-only streams.	(monitor)
paSampleRate	float	The sample rate of the stream in Hertz (samples per second). In cases where the hardware sample rate is inaccurate and PortAudio is aware of it, the value of this field may be different from the sampleRate parameter passed to Pa_OpenStream(). If information about the actual hardware sample rate is not available, this field will have the same value as the sampleRate parameter passed to Pa_OpenStream().	(monitor)

11.6 MHAIOTCP

TCP IO-lib exchanges sound samples as interleaved binary float32 data in network-byte-order (big endian) over a TCP connection

11.6.1 Detailed description

TCP IO-lib exchanges sound samples as interleaved binary float32 data in network-byte-order (big endian) over a TCP connection

11.6.2 Supported domains

The MHA plugin `MHAIOTCP` supports these signal domains:

- waveform to waveform

11.6.3 Plugin Tags

io

11.6.4 Configuration variables

Name	Type	Description	Default
server_port_open	int	Status of local server port	(monitor)
connected	int	Status of tcp connection	(monitor)
peer_address	string	IP address of remote computer	(monitor)
peer_port	int	Remote tcp port of connection	(monitor)
address	string	Local address, determines interfaces	0.0.0.0
port	int	TCP Server Port for sound data exchange **Range:** [0,65535]	33338
debug_filename	string	debug messages of MHAIOTCP will be written to this file if non-empty	

11.7 MHAIOalsa

ALSA client

11.7.1 Detailed description

ALSA client

11.7.2 Supported domains

The MHA plugin `MHAIOalsa` supports these signal domains:

- waveform to waveform

11.7.3 Plugin Tags

io

11.7.4 Configuration variables

Name	Type	Description	Default
in	parser		(see below)
out	parser		(see below)
priority	int	Set SCHED_FIFO priority of processing thread or -1 for no realtime scheduling **Range:** [-1,99]	-1
format	keyword_list	PCM sample format **Range:** [S32_LE S16_LE]	S32_LE
link	bool	link PCM devices	yes
alsa_start_counter	int	alsa is started on startup and for every dropout.	(monitor)

Variables of sub-parser `in`:

Name	Type	Description	Default
device	string	device name of the alsa PCM device for audio capture	
nperiods	int	number of periods in alsa buffer **Range:** [2,]	2

Variables of sub-parser `out`:

Name	Type	Description	Default
device	string	device name of the alsa PCM device for audio playback	
nperiods	int	number of periods in alsa buffer **Range:** [2,]	2

12 Plugin category 'level'

12.1 level_matching

Input level matching

12.1.1 Detailed description

This plugin implements automatic pairwaise matching of input levels. This algorithm can be used to e.g. compensate for microphone gain drift. Microphone gain matching relies on the assumption that the input signal on both microphones has the same level. This assumption breaks down if the mic distance is small compared to the sound wavelength. To exclude high frequencies from the gain matching, if used in the time domain the signal is filtered by a lowpass filter before the mismatch is calculated. In the spectral domain the user is responsible for the restriction of the matching to sensible frequencies, e.g. by usage of a fft filterbank and careful selection of channels for the matching algorithm. As gain drift usually happens on a time scale large compared to the block size the mismatch is also lowpass filtered. Currently this plugin can only match pairs of microphones. The microphone pairings are passed as a matrix, with each row containing the indices of two microphones. The first entry is taken as reference mic. Its signal will not be changed. The signal of the other microphone will be scaled so that the average rms levels match.

12.1.2 Supported domains

The MHA plugin `level_matching` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

12.1.3 Plugin Tags

level

12.1.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
channels	matrix<int>	channels	[[0 1]]
lp_signal_fpass	float	Upper edge of the lp pass band for the signal in Hz **Range:** [0,[	4000
lp_signal_fstop	float	Stop band lower edge frequency for the signal in Hz **Range:** [0,[	8000
lp_level_tau	float	Low pass time constant for the mismatch in s **Range:** [0,[	1
range	float	Maximum matching range in dB **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)

12.2 levelmeter

Broadband level meter.

12.2.1 Detailed description

This level meter calculates the RMS level of the input signal in the last *tau* seconds.

12.2.2 Supported domains

The MHA plugin `levelmeter` supports these signal domains:

- waveform to waveform

12.2.3 Plugin Tags

level compression

12.2.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
tau	float	RMS time constant / s **Range:** [0,]	0.1
mode	keyword_list	Level scale **Range:** [Pa FS]	Pa
rms	vector<float>	RMS level / dB	(monitor)
peak	vector<float>	Peak level / dB	(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 Plugin category 'level-meter'

13.1 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 The 'peak' variables are only available during waveform processing.

13.1.1 Detailed description

This plugin computes the rms level and peak level of the current fragment and provides them as AC and monitor variables rms level in W/m^2 and peak level in Pascal. The values are provided in linear (variable names: level and peak) and logarithmic scale (level_db and peak_db). The default unit for the logarithmic scale is dB(SPL), but conversion to dB(HL) as per ISO 389-7:2005 (freefield) can be activated in the spectral domain. The correction values for frequencies above 16 kHz are extrapolated.

13.1.2 Supported domains

The MHA plugin `rmslevel` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

13.1.3 Plugin Tags

level-meter feature-extraction

13.1.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
unit	keyword_list	Use dB(SPL) or dB(HL) **Range:** [spl hl]	spl

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 Plugin category 'level-modification'

14.1 gain

Gain plugin:

Apply a gain to each channel

14.1.1 Detailed description

This plugin applies a configurable gain to each channel. The number of entries in the gain vector must be either one per channel or 1 (same gains for all channels)

For security reasons, the gain is limited to the range given by `min` and `max` which are pre-configured to -16dB and +16dB, respectively. Maximum and minimum gains are themselves configurable and need to be adjusted before gains exceeding the range [-16,+16] can be set through variables `gains`.

14.1.2 Supported domains

The MHA plugin `gain` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

14.1.3 Plugin Tags

level-modification

14.1.4 Configuration variables

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

14.2 smoothgains_bridge

Gain smoothing for reduction of filter length

14.2.1 Detailed description

The overlap-add framework allows finite impulse response filter lengths up to 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, thereby 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. Using a minimal phase can lead to undesired interference between subsequent overlapping synthesized frames, also introducing unwanted sound artifacts. It should only be used if the filter applied in the STFT domain does not change or only changes very slowly.

14.2.2 Supported domains

The MHA plugin `smoothgains_bridge` supports these signal domains:

- spectrum to spectrum

14.2.3 Plugin Tags

level-modification filter data-flow overlap-add

14.2.4 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 Note: Appropriate settings of window position are required (linear_phase: 0.5, minimal_phase: 0) **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) **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. **Range:** [rect hanning hamming blackman bartlett user]	hanning
user	vector<float>	User provided window (used if window type==user).	[]

15 Plugin category 'math'

15.1 acTransform_wave

Transform Plugin Between Coordinate Systems for Waveforms

15.1.1 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.

15.1.2 Supported domains

The MHA plugin `acTransform_wave` supports these signal domains:

- waveform to waveform

15.1.3 Plugin Tags

math linear-algebra

15.1.4 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

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 'noise-suppression'

16.1 noise_psd_estimator

Noise power estimator after Gerkmann (2012).

16.1.1 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 with the same name as the plugin 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.

16.1.2 Supported domains

The MHA plugin `noise_psd_estimator` supports these signal domains:

- spectrum to spectrum

16.1.3 Plugin Tags

noise-suppression feature-extraction adaptive

16.1.4 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)

16.2 smooth_cepstrum

Cepstral smoothing single-channel noise reduction

16.2.1 Detailed description

Single-channel noise reduction applying cepstral smoothing based on 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' plugin 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.

16.2.2 Supported domains

The MHA plugin `smooth_cepstrum` supports these signal domains:

- spectrum to spectrum

16.2.3 Plugin Tags

noise-suppression signal-enhancement adaptive

16.2.4 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) **Range:** [-50,50]	-27
f0_low	float	Lower limit for F0 detection in Hz **Range:** [0,400]	70
f0_high	float	Upper limit for F0 detection in Hz **Range:** [0,400]	300
delta_pitch	float	Quefrency half-width of pitch-set in samps **Range:** [0,20]	2
lambda_thresh	float	Pitch detection threshold for smooth cepstrum in magnitude **Range:** [0,3]	0.2
alpha_pitch	float	Alpha value to set for pitch range **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 **Range:** [0,1]	0.2886
gain_min_db	float	Minimum gain in dB for a frequency bin **Range:** [-30,0]	-17
win_f0	vector<float>	Window coefficients for cepstral smoothing window **Range:** [0,1]	[0.0207 0.0656 0.1664 0.2473 0.24
alpha_const_vals	vector<float>	Piecewise values for steady-state alphas **Range:** [0,2]	[0.2 0.4 0.92]
alpha_const_limits_hz	vector<float>	Limits for steady-state alphas given in Hz **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`:

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 **Range:** [0,2]	0.5
xi_opt_db	float	xiOpt in dB for computing GLR and SPP from local SNR **Range:** [0,40]	15

16.3 windnoise

This plugin detects which microphone channels are affected by wind noise and replaces their signal with signal from unaffected channels.

16.3.1 Detailed description

The windnoise plugins smoothes power spectra over time and detects wind noise in the audio by computing the ratio of sound energy at low frequencies with respect to the overall energy in the smoothed power spectra. The presence of wind noise is then detected when this ratio exceeds a configurable threshold criterium. This criterium as well as the cut-off frequency are configurable, and may need to be adapted to the microphones used. Users can inspect the monitor variable lowpass quotient which is also published as an AC variable for downstream plugins to check the value of the low frequency energy ratio and derive a suitable threshold for their acoustic configuration.

As an example for setting a suitable threshold, a voice sample was generated using an AKG K-501 microphone with the waveform shown top of Fig. 8. Therefore, the value of LowPass-Fraction needs to be 0.96 which translates to -0.3 dB in decibel scale and this value is set using the configuration variable. With this set value, wind noise is correctly detected as can be seen bottom of Fig. 8

Artificial test signals can be used to test the technical feature extraction performed by the wind-noise algorithm: The low-the pass quotient can be influenced by adding low-frequency or high-frequency sinusoids, and the smoothing over time of the power spectra can be observed by introducing transient changes into an otherwise constant signal and then observing the extracted features over time. An artificial sine wave is generated as shown top of Fig. 9. A low frequency signal (40 Hz) is followed by a high frequency (1 kHz) which is above the threshold of 500 Hz. It can be observed in bottom of Fig. 9 how the Low pass ratio immediately drops the moment a high frequency signal comes in, thereby not reducing speech intelligibility.

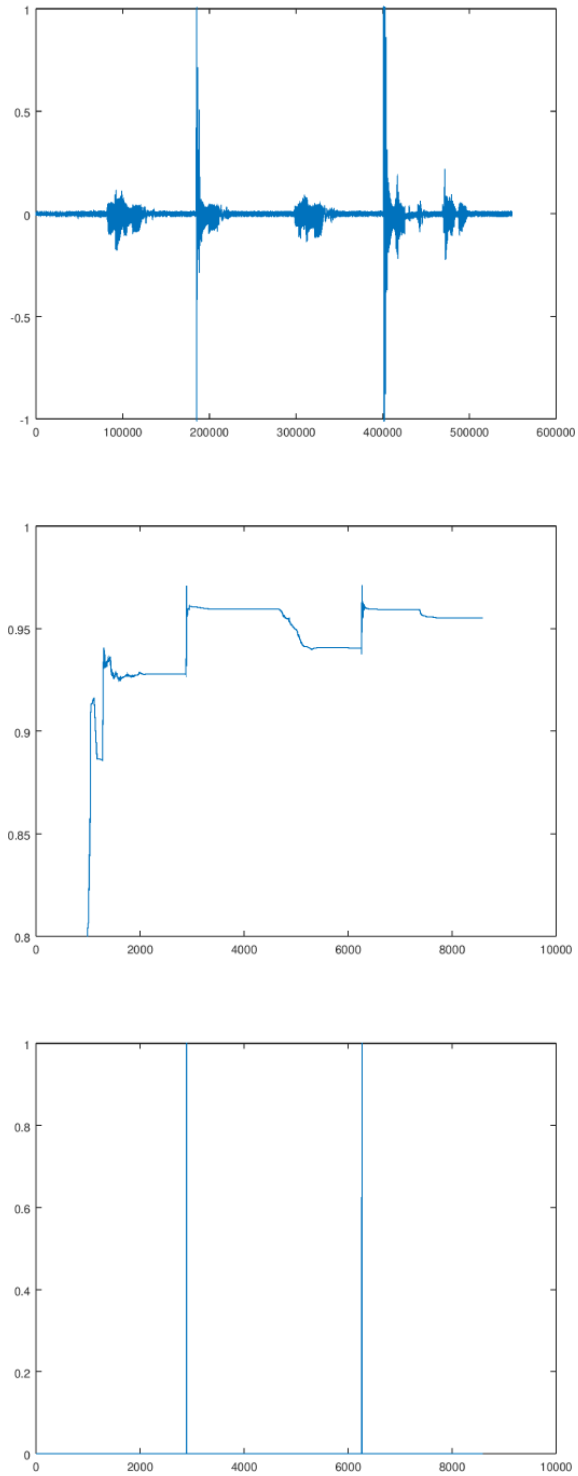
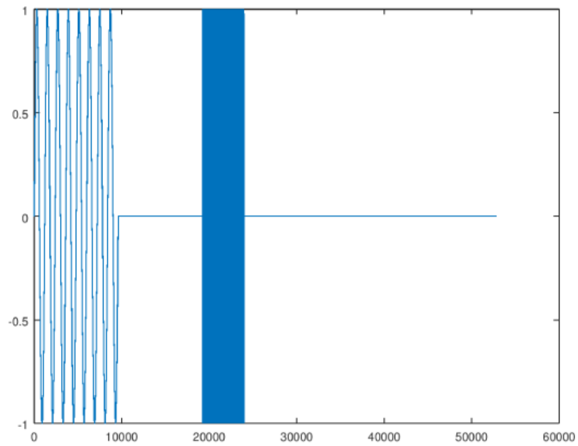


Figure 8 Top: Waveform of test file Middle: Low pass quotient generated from AC varialbes. Bottom: Detected AC variable with windnoise showing as peaks.



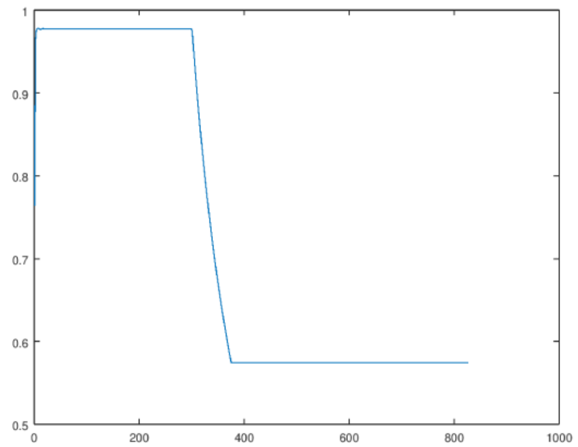


Figure 9 Top: Waveform of sine wave generated with 40 Hz at start and 1 kHz later. Bottom: Low pass quotient AC variable.

16.3.2 Supported domains

The MHA plugin `windnoise` supports these signal domains:

- spectrum to spectrum

16.3.3 Plugin Tags

noise-suppression feature-extraction

16.3.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
UseChannel_LF_attenuation	bool	switch for channelwise LF-attenuation (yes=on, no=off)	no
tau_Lowpass	float	low-pass filter time constant for filtering spectral power / s **Range:** [0,1]	1
LowPassCutOffFrequency	float	cut-off frequency of the spectral weighting windnoise detector **Range:** [0,]	500
LowPassFraction	float	level difference threshold / dB between low and high band. If the level difference between low and high band exceeds this threshold in dB, then wind noise is detected. **Range:** [-10,10]	-1
LowPassWindGain	float	Gain / dB applied to low frequency part when wind noise is detected and not compensated signal replacement **Range:** [-100,0]	-10
WindNoiseDetector	keyword_list	type of windnoise detector to apply **Range:** [psd diffsum msc none]	diffsum
detected	vector<int>	windnoise detector state, one element (with value 0 or 1) per audio channel	(monitor)
lowpass_quotient	vector<float>	ratio of intensity at frequencies $<$ LowPassCutOffFrequency to broadband intensity as quotient of intensities after smoothing the power spectrum with tau_Lowpass	(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)

17 Plugin category 'plugin-arrangement'

17.1 altconfig

Alternative configurations for a plugin

17.1.1 Detailed description

This plugin loads another MHA plugin for signal processing when user assigns configuration variable "plugin_name" and allows sending stored configuration commands to the loaded plugin at run time by selecting one of the pre-configured alternative configuration commands. Users can create names for an arbitrary number of alternative configurations by assigning a list of names to configuration variable "algos". New configuration variables with these names are then created by the plugin. Users can then store arbitrary configuration commands in these new configuration variables. Execution of these configuration commands can later be triggered by selecting them, i.e. assigning their name to configuration variable "select". The selected configuration command will be executed in the context of the loaded plugin. Empty configuration commands will be ignored.

17.1.2 Supported domains

The MHA plugin `altconfig` supports these signal domains:

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

17.1.3 Plugin Tags

plugin-arrangement data-flow

17.1.4 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	
algos	vector<string>	List of names for plugin configurations. Assigning names to algos creates configuration variables with the given names which can be used to store configuration commands for the loaded plugin.	[]
select	keyword_list	Select a configuration for parsing. Assigning one of the names to select causes execution of the configuration command stored under that name in the context of the loaded plugin. **Range:** [(none)]	(none)
selectall	bool	Iterate through all configuration options (for validation)	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)

17.2 altplugs

Configure alternative plugins.

17.2.1 Detailed description

The plugin altplugs allows configuration of alternative plugins. Plugins can either be registered en-bloc via the plugs variable or one by one by repeated assignment to the add variable. Plugins can be removed via delete. Registered plugins are configured as sub parsers of altplugs. The plugin to be used for processing can be selected via the select variable at any time. If the plugin output is in the time domain the newly selected plugin can optionally be faded in, ramplen controlling the ramp length, the old plugin is always switched off instantaneously. 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 renamed using the ":" operator.

A module for the mhacontrol graphical user interface is provided.

17.2.2 Supported domains

The MHA plugin altplugs supports these signal domains:

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

17.2.3 Plugin Tags

plugin-arrangement data-flow

17.2.4 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)

17.3 analysispath

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

17.3.1 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.

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.

17.3.2 Supported domains

The MHA plugin `analysispath` supports these signal domains:

- waveform to waveform

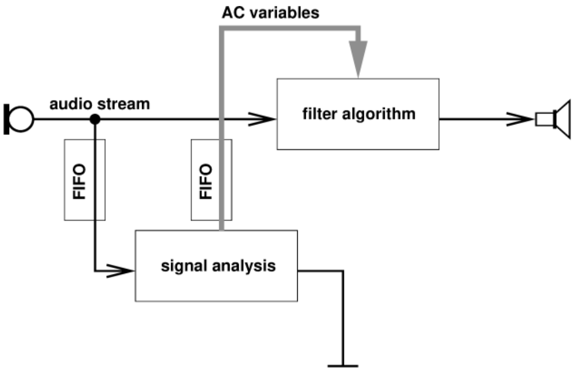


Figure 10 Schematic signal flow in the analysis path scenario.

17.3.3 Plugin Tags

plugin-arrangement algorithm-communication data-flow

17.3.4 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 **Range:** [1,]	200
fifolen	int	length of double buffer in inner fragment size **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)	[]

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)

17.4 mhachain

MHA Chain

17.4.1 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.

Plugins are loaded by assigning a vector of strings to the configuration variable *algos*. Each entry in this vector has the form *plugin:configured_name<config_file*, where

- *plugin* is the filename of the plugin without path or file extension,
- *:configured_name* optionally assigns a different name to this instance of the plugin. This is useful if multiple instances of the same plugin are loaded into different positions of the processing chain. the colon and the *configured_name* are not specified, then the *configured_name* defaults to *plugin*.
- *<config_file* optionally specifies a configuration file with which the plugin is initially configured. This is only needed when replacing a complete chain while the mha is performing signal processing by reassigning *algos*.

The plugins loaded by assigning to configuration variable *algos* cause creation of sub-parsers named like the *configured_name* in the mhachain plugin configuration and can be configured through these sub-parsers.

17.4.2 Supported domains

The MHA plugin `mhachain` supports these signal domains:

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

17.4.3 Plugin Tags

plugin-arrangement data-flow

17.4.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
use_profiling	bool	Profile the loaded plugins. Needs to be set to true before setting algos.	no
algos	vector<string>	List of plugins to load and arrange in a signal processing chain. Entries are separated by spaces and given in the order of the signal processing. Please refer to the detailed description of this plugin in the plugin manual for more details.	[]

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)

17.5 overlapadd

Waveform to spectrum overlap add and FFT method.

Audio data is collected up to wndlen, then windowed by the given window function, zero padded up to fftlength (symmetric zero padding or asymmetric zero padding possible), and Fast-Fourier-transformed. The configuration variables are locked in the prepare call and must be unlocked by release before a change is possible.

17.5.1 Detailed description

The plugin 'overlapadd' transforms fragmented waveform audio data into short time Fourier transformed (STFT) audio data. Both the forward and the inverse transform are performed. Another plugin which processes the STFT spectra must be loaded by setting `plugin_name`.

The overlap-add mechanism is similar to that from Allen (1977): First the waveform signal is windowed by a window function. The default window shape is the Hanning window, but other pre-defined and user-defined window shapes can be selected. 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.[6] The zero padded signal is then 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 default Hanning window is $w_1(k) = \frac{1}{2}(1 - \cos(2\pi k/M))$, the windowed signal is

$$x_w(m, k) = w_1(k) \cdot x(m \cdot P + k), \tag{16}$$

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

After processing and inverse Fourier transformation, ramps can be applied to the signal to avoid discontinuities in case of temporal aliasing, and thus reducing the artifacts. These ramps are a applied to the zero-padding regions. The shape of the ramps is determined by the window shape *zerownd.type*. Common choices are Hanning ramps or rectangular ramps (i.e. no ramps, the default). 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 shown in Fig. 11 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 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.

The spectral signal produced by this plugin is subject to the following scaling: The attenuation effect of applying the analysis window is compensated by dividing by the RMS (root mean square) of the window. To account for the zero-padding, which would reduce the RMS of the signal block[7], the signal is multiplied with $\sqrt{\mathrm{fftlen}/\mathrm{wndlen}}$. Finally, the forward FFT operation in the MHA will apply a factor $1/\sqrt{\mathrm{fftlen}}$ so that the sum of squared magnitudes of the spectral bins produces the correct level in Pascal.

The purpose of the scaling described in the previous paragraph is to enable spectral algorithms to determine the physical level of the signal in the current STFT block without having to apply correction factors for window shape, zero-padding, overlap, FFT length, etc.

[6] 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.

[7] The same sum of squared samples would be divided by fftlen instead of wndlen to compute the mean after zero-padding

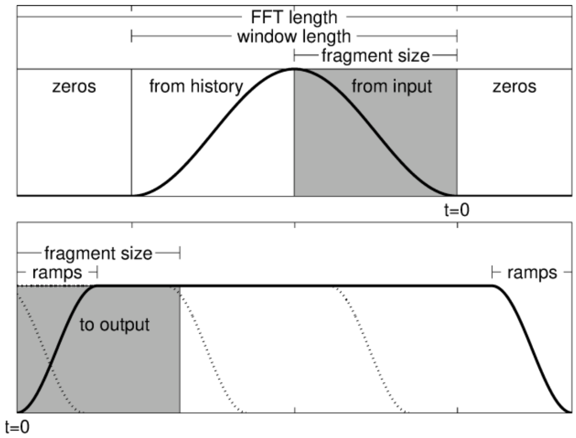


Figure 11 Windowing in the overlap-add method with 50% overlap and zero-padding. In the upper panel, the windowed input signal before applying the FFT is schematically plotted. In the lower panel, the same time interval after inverse FFT is shown. The shaded segment is the fragment which is read from 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.

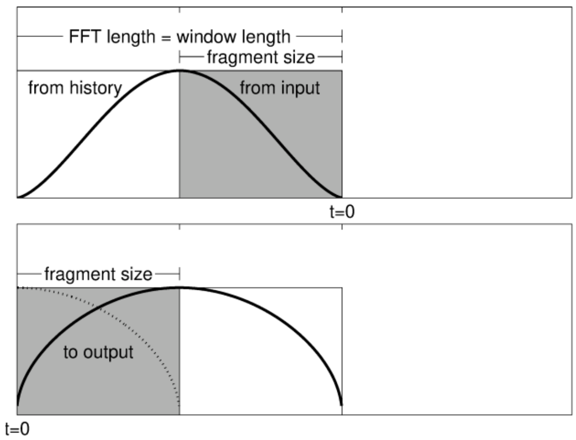


Figure 12 Windowing in the overlap-add method, as in Fig. 11, but with post-windowing and without zero-padding. In this setup, W^{α} is applied before FFT and $W^{1-\alpha}$ is used for post-windowing. The delay between input and output signal is the window length.

17.5.2 Supported domains

The MHA plugin `overlapadd` supports these signal domains:

- waveform to waveform

17.5.3 Plugin Tags

plugin-arrangement signal-transformation overlap-add

17.5.4 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	
fftlen	int	FFT length **Range:** [1,]	512
wnd	parser	window type	(see below)
zerownd	parser	zero padding post window type	(see below)
strict_window_ratio	bool	Disallow window sizes that are not a multiple of the hop size (fragsize) by a power of two.	yes
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. **Range:** [rect hanning hamming blackman bartlett user]	hanning
user	vector<float>	User provided window (used if window type==user).	[]
len	int	window length/samples **Range:** [1,]	400
pos	float	window position (0 = beginning, 0.5 = symmetric zero padding, 1 = end) **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. **Range:** [rect hanning hamming blackman bartlett user]	rect
user	vector<float>	User provided window (used if window type==user).	[]

17.6 resampling

Synchronous resampling plugin.

17.6.1 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.

17.6.1.1 Warning:

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

17.6.2 Supported domains

The MHA plugin `resampling` supports these signal domains:

- waveform to waveform

17.6.3 Plugin Tags

plugin-arrangement signal-transformation

17.6.4 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 **Range:**]0,]	44100
fragsize	int	fragment size of client plugin **Range:**]0,]	200
nyquist_ratio	float	lowpass filter cutoff frequency / lower nyquist frequency **Range:**]0,]	0.85
irslen_outer2inner	float	filter lenth 1st resampling / sec **Range:**]0,]	0.0007
irslen_inner2outer	float	filter lenth 2nd resampling / sec **Range:**]0,]	0.0007

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)

17.7 split

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

17.7.1 Detailed description

The plugin 'split' takes a multi-channel input signal and splits it into separate chains of groups of channels. After processing of each chain, the output channels of each chain are collected into a multi-channel output signal.

By default, all parallel chains are processed sequentially in a single thread. It is also possible to process the different chains in different processing threads, to exploit parallel execution on multi-core CPUs: Set `thread_platform` to `win32` on MS Windows systems, or to `posix` on Linux and macOS.

For real-time processing scenarios, it is important to set up the worker threads' schedulers and priorities to a reasonable value so that they neither starve upstream production or downstream consumption of the processed audio, nor get themselves interrupted by non-audio-related tasks on the same system. A reasonable choice is to use the same scheduler and priority as the framework thread that invokes the processing of this plugin. Unfortunately, this cannot be determined automatically and needs to be set through configuration variables `worker_thread_scheduler` and `worker_thread_priority`. The corresponding settings of the framework thread can be compared by checking the values of `framework_thread_scheduler` and `framework_thread_priority` during processing.

'split' also supports processing all contained chains in parallel to all other signal processing in the MHA by introducing a delay of one audio fragment: In this case, when the split plugin is asked to process an audio fragment, it immediately returns the processed audio fragment from the previous invocation, and simultaneously begins processing the new audio fragment in the worker threads. This mode is activated by setting `delay=yes` and does not work for the '`dummy`' `thread_platform`. Priorities of the worker threads should be set to slightly less important than the priority of the framework thread.

Thread priorities and schedulers are operating system dependent settings. Check the documentation of your operating system for details on the schedulers and priorities, and compare the relative priorities of all processes and threads on your system against expectations with a suitable tool while openMHA is running.

Plugins loaded by split cannot access algorithm communication (AC) variables created outside of split, nor pass on algorithm communication variables created inside of split to the outside, nor can parallel plugins access each others AC variables. Each of the parallel plugins loaded by split receives an isolated and initially empty AC variable space to avoid synchronization overhead.

17.7.2 Supported domains

The MHA plugin `split` supports these signal domains:

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

17.7.3 Plugin Tags

plugin-arrangement audio-channels data-flow

17.7.4 Configuration variables

Name	Type	Description	Default
algos	vector<string>	List of plugins which process the different groups of audio channels. Exactly one plugin per channel group must be given. (Use e.g. [mhachain:chain0 mhachain:chain1 ...] to have more than one processing plugin per group by combining them into a chain.	[]
channels	vector<int>	Number of channels in the respective channel groups to be processed by the corresponding plugins listed in "algos". **Range:** [0,[	[]
thread_platform	keyword_list	Thread platform to use. "posix" is the native Linux and macOS 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]	dummy
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 during processing. **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 during processing. 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 processing of contained plugins outside of the calling processing thread at the cost of one block additional delay	no

18 Plugin category 'signal-generator'

18.1 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.

18.1.1 Detailed description

White noise generator. For each audio channel, statistically independent white noise is added to that channel's input signal. 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.

18.1.2 Supported domains

The MHA plugin `noise` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

18.1.3 Plugin Tags

signal-generator

18.1.4 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 **Range:** [add replace]	add
frozennoise_length	float	Length of frozen noise in s, or 0 for running noise. **Range:** [0,]	0
seed	int	Seed for the random number generator.	-1670791435

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)

18.2 plingploing

plingploing algorithm.

18.2.1 Detailed description

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

18.2.2 Supported domains

The MHA plugin `plingploing` supports these signal domains:

- waveform to waveform

18.2.3 Plugin Tags

signal-generator music

18.2.4 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 **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 **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 **Range:** [0,]	2
bpm	float	beats per minute **Range:** [1,]	200
minlen	float	minimum note length / beats **Range:** [1,]	1
maxlen	float	maximum note length / beats **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)

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)

18.3 sine

Sine wave generator.

18.3.1 Detailed description

Sine generator plugin. Adds a sinusoid with the given RMS level to the configured audio channels. If the amplitude changes from one block of audio to the next, then the amplitude change is spread out linearly across all samples of the audio block that first sees the new level to avoid clicks from discontinuities.

18.3.2 Supported domains

The MHA plugin `sine` supports these signal domains:

- waveform to waveform

18.3.3 Plugin Tags

signal-generator

18.3.4 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 **Range:** [0,[	0
mode	keyword_list	Replace input signal with tone or mix tone into input signal **Range:** [replace mix]	replace
channels	vector<int>	0-based indices of audio channels to feed with tone (all other audio channels are not affected)	[]

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)

19 Plugin category 'signal-transformation'

19.1 downsample

Downsampling by integer fractions

19.1.1 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.

19.1.2 Supported domains

The MHA plugin `downsample` supports these signal domains:

- waveform to waveform

19.1.3 Plugin Tags

signal-transformation filter

19.1.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
ratio	int	downsampling ratio **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]

19.2 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. Always set ramplen=0 here if wndpos!=0 in the corresponding wave2spec.

19.2.1 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.

19.2.2 Supported domains

The MHA plugin `spec2wave` supports these signal domains:

- spectrum to waveform

19.2.3 Plugin Tags

signal-transformation overlap-add

19.2.4 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) **Range:** [0,1]	1
wndtype	keyword_list	window type **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`:

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)

19.3 upsample

Upsampling by integer fractions

19.3.1 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.

19.3.2 Supported domains

The MHA plugin `upsample` supports these signal domains:

- waveform to waveform

19.3.3 Plugin Tags

signal-transformation filter

19.3.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
ratio	int	upsampling ratio **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]

19.4 wave2spec

Waveform to spectrum overlap add and FFT method.

Audio data is collected up to wndlen, then windowed by the given window function, zero padded up to fftlen (symmetric zero padding or asymmetric zero padding possible), and fast-Fourier-transformed. The configuration variables are locked during processing.

19.4.1 Detailed description

The plugin 'wave2spec' transforms time-domain waveform signal to short-time Fourier transform (STFT) signal. It 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.

Several pre-defined window shapes as well as user-defined window shapes are supported. In addition, a configurable exponent can be applied to the window samples.

During processing, the input data samples are multiplied with the samples of the analysis window, zero padded to the FFT length, and Fourier transformed. 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, the function `MHA_AC::get_var_spectrum()` can be used. See the openMHA developer manual or the header file `mha_algo_comm.h` for details.

See section 17.5 for a description of the overlap-add method that is also followed by this plugin.

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.[8]

The plugin performs the following scaling of the signal: The effect on the level of applying the analysis window to the input signal is compensated by dividing by the RMS (root mean square) of the window. To account for the zero-padding, which would reduce the RMS of the signal block[9], the signal is multiplied with $\sqrt{\mathrm{fftlen}/\mathrm{wndlen}}$. Finally, the forward FFT operation in the MHA will apply a factor $1/\sqrt{\mathrm{fftlen}}$ so that algorithms that compute signal level do not have to know the fftlen, but can simply sum squared magnitudes of the STFT bins to compute the RMS of the current block in Pascal.

The purpose of the scaling described in the previous paragraph is to enable spectral algorithms to determine the physical level of the signal in the current STFT block without having to apply correction factors for window shape, zero-padding, overlap, FFT length, etc.

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

[9] The same sum of squared samples would be divided by fftlen instead of wndlen to compute the mean after zero-padding

19.4.2 Supported domains

The MHA plugin `wave2spec` supports these signal domains:

- waveform to waveform
- waveform to spectrum

19.4.3 Plugin Tags

signal-transformation overlap-add

19.4.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
fftlen	int	FFT lengths **Range:** [1,]	512
wndlen	int	window length/samples **Range:** [1,]	400
wndpos	float	window position (0 = beginning, 0.5 = symmetric zero padding, 1 = end) **Range:** [0,1]	0.5
wndtype	keyword_list	window type **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	[]
strict_window_ratio	bool	Disallow window sizes that are not a multiple of the hop size (fragsize) by power of two.	yes
return_wave	bool	return input waveform signal, store spectrum only to AC	no
zeropadding	vector<float>	Zeropadding in samples before and after the analysis window	(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)

20 Plugin category ’signalflow’

20.1 audiometerbackend

This plugin mimicks an audiometer by playing a signal in a given sound level on a given channel.

20.1.1 Detailed description

This plugin has been designed to perform calibrated listening experiments by playing a signal in a sound level in dB SPL on a user-defined channel both determined by the user. The sound level can be adapted online while the signal is played. By using these features, this plugin can be configured to conduct an audiogram measurement or other audiometric measurements. The sound level in dB SPL can be adjusted by setting the configuration variable **level**. The signal to be played can be selected from a pre-defined list or the input signal to this plugin can be used as well. The choice between the incoming input signal or a signal from the pre-defined list, as well as on which channel the selected signal is played, is made by setting the playback mode. For this, the configuration variable **mode** can be used. This variable is another pre-defined list of four options, as given below:

- **input:** The incoming input signal is played
- **mute:** The selected signal is not played
- **left:** The selected signal is played on the left channel
- **right:** The selected signal is played on the right channel

The list of possible signals is given in the following list:

- **sine:** Sine wave
- **oct3_lnn2:** Third Octave Low-noise Noise, iterated twice
- **oct3_lnn0:** Third octave Low-noise Noise

- **oct_lnn2:** Octave Low-noise Noise, iterated twice
- **oct_lnn0:** Octave Low-noise Noise

In order to be able to select the signal from this list, please set the configuration parameter **sigtype**. For more details about how the low-noise noise (LNN) is generated, please refer to the article Kohlrausch et al 1997. The frequency of the signal to be played is determined by setting the configuration variable **freq**. Finally, a Hanning ramp can be incorporated in order to obtain a smooth transition between level changes. The length of the Hanning ramp in seconds is defined by setting the configuration variable **ramplen**.

20.1.2 Supported domains

The MHA plugin `audiometerbackend` supports these signal domains:

- waveform to waveform

20.1.3 Plugin Tags

signalflow generator audiometer

20.1.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
freq	int	Frequency in Hz.	440
sigtype	keyword_list	Signal type **Range:** [sine oct3_lnn2 oct3_lnn0 oct_lnn2 oct_lnn0]	sine
level	float	Level in dB (SPL) of the input file	0
mode	keyword_list	Playback mode **Range:** [input mute left right]	input
ramplen	float	Length of hanning ramp at level changes in seconds **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)

21 Plugin category 'spatial'

21.1 adm

Adaptive differential microphone

21.1.1 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.

To save cpu time on weak devices the adaptation of `beta` can be performed only every p frames by setting the adaptation_ratio configuration variable to p.

21.1.2 Supported domains

The MHA plugin `adm` supports these signal domains:

- waveform to waveform

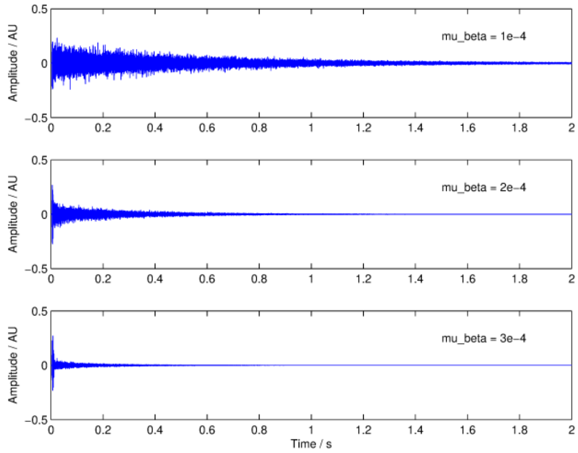


Figure 13 Output signals illustrating convergence of the ADM algorithm for three different values of `mu_beta` (input signal: white Gaussian noise exactly from behind)

21.1.3 Plugin Tags

spatial signal-enhancement beamformer adaptive

21.1.4 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:** [0,128]	46
decomb_order	int	Filter order of FIR comb compensation filter. Values <=1 deactivate filter. **Range:** [0,128]	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)
adaptation_ratio	int	Calculate beta every n frames **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)

21.2 coherence

Coherence filter

21.2.1 Supported domains

The MHA plugin `coherence` supports these signal domains:

- spectrum to spectrum

21.2.2 Plugin Tags

spatial signal-enhancement dereverberation adaptive

21.2.3 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 **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 **Range:** [linear bark log erb ERB_Glasberg1990]	linear
ovltype	keyword_list	filter overlap type **Range:** [rect linear hanning exp gauss]	rect
plateau	float	relative plateau width **Range:** [0,1[	0
ftype	keyword_list	frequency entry type **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
flag_allow_empty_bands	bool	Set true to allow bands where all STFT-bin-gains equal zero.	no
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 **Range:** [seconds periods]	seconds
tau	vector<float>	Averaging time constant **Range:** [0,]	[0.04]
alpha	vector<float>	Gain exponent **Range:** [0,]	[1]
limit	float	gain limit / dB (zero: no limit) **Range:** [,0]	0
mapping	vector<float>	mapping interval of coherence estimator to coherence (min max) **Range:** [0,1]	[0 1]
average	keyword_list	average mode **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 **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. **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)

21.3 delaysum_wave

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

21.3.1 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. This plugin was formerly known as delaysum.

21.3.2 Supported domains

The MHA plugin `delaysum_wave` supports these signal domains:

- waveform to waveform

21.3.3 Plugin Tags

spatial beamformer

21.3.4 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)

21.4 doasvm_classification

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

21.4.1 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.

21.4.2 Supported domains

The MHA plugin `doasvm_classification` supports these signal domains:

- waveform to waveform

21.4.3 Plugin Tags

spatial classifier binaural

21.4.4 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)

21.5 doasvm_feature_extraction

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

21.5.1 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.

21.5.2 Supported domains

The MHA plugin `doasvm_feature_extraction` supports these signal domains:

- waveform to waveform

21.5.3 Plugin Tags

spatial feature-extraction binaural

21.5.4 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 **Range:** [0,[	160
max_lag	int	Maximum lag in samples between microphones (setup-dependent) **Range:** [0,[	20
nupsample	int	The amount the GCC-PHAT spectrum is oversampled **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)

22 Plugin category 'test-tool'

22.1 cpuload

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

22.1.1 Detailed description

This plugin artificially generates cpu load. The achieved CPU load is proportional to number of channels, number of frames, and factor. If use_sine is set, a sine of is calculated, making the load mainly cpu-bound. Alternatively an operation on a variable size table is done, simulatung a memory bound problem.

22.1.2 Supported domains

The MHA plugin `cpuload` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

22.1.3 Plugin Tags

test-tool

22.1.4 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
table_size	int	Size of the lookup table **Range:** [1,]	65536
use_sine	bool	Whether to use the sine function.	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)

22.2 dropgen

22.2.1 Detailed description

This plugin randomly generates dropouts by waiting between 1 and 10 frames in .5

This plugin does not otherwise modify the signal. Do not include this plugin in production setups.

22.2.2 Supported domains

The MHA plugin `dropgen` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

22.2.3 Plugin Tags

test-tool

22.2.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
min_sleep_time	float	minimum sleep time, in s **Range:** [0,[	0
max_sleep_time	float	minimum sleep time, in s **Range:** [0,[	0
chance	float	chance of an artificial dropout **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)

22.3 droptect

Plugin detects dropouts in channels that have a constant spectrum

22.3.1 Detailed description

Plugin to detect dropouts in live audio processing setups. `droptect` expects an input signal with a spectral shape that does not vary over time, e.g. a combination of different sinusoids.

Either feed such a signal from an external source into the sound card used by openMHA, or have openMHA create such a signal downstream of the droptect plugin (e.g. with `sine`), and feed the sound card output back into the sound card input with an audio cable.

`droptect` detects a dropout if

- The broadband level of the current STFT spectrum is below threshold, or
- The level of any bin of the STFT spectrum differs by more than 6dB from the average spectrum.

STFT bins with very low level (35 dB below the broadband threshold) are excluded from the 6dB difference criterium to allow for soft microphone noise.

Detected dropouts are accumulated per audio channel and published in the monitor variable `dropouts`. The count can be reset to 0 by assigning "yes" to `reset`. Some false positive detected dropouts on startup of signal processing are expected. Reset the dropout count after processing has started to remove these false positives.

22.3.2 Supported domains

The MHA plugin `droptect` supports these signal domains:

- spectrum to spectrum

22.3.3 Plugin Tags

test-tool

22.3.4 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. Resets to 0 each time there is no dropout.	(monitor)
blocks	int	Number of blocks processed since last reset or start	(monitor)
reset	bool	Setting to "yes" clears number of dropouts and blocks. Value is reset to "no" when the next spectrum is processed.	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	vector<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)

22.4 identity

22.4.1 Detailed description

The simplest openMHA plugin.

This plugin does not modify the signal.

22.4.2 Supported domains

The MHA plugin `identity` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

22.4.3 Plugin Tags

test-tool

22.4.4 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)

22.5 matlab_wrapper

22.5.1 Supported domains

The MHA plugin `matlab_wrapper` supports these signal domains:

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

22.5.2 Plugin Tags

test-tool

22.5.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
library_name	string	Name of matlab generated library	

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)

22.6 testplugin

loads a plugin for testing

22.6.1 Supported domains

The MHA plugin `testplugin` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

22.6.2 Plugin Tags

test-tool feature-extraction

22.6.3 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)

Variables of sub-parser `config_in`:

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

Variables of sub-parser `config_out`:

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

Variables of sub-parser `ac`:

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)

23 Plugin category 'testing'

23.1 complex_scale_channel

example plugin configuration structure

23.1.1 Supported domains

The MHA plugin `complex_scale_channel` supports these signal domains:

- spectrum to spectrum

23.1.2 Plugin Tags

testing

23.1.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
channel	int	index of channel to be scaled **Range:** [0,[	0
factor	complex	complex scale factor	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)

23.2 proc_counter

Counter for invocations of signal processing callback

23.2.1 Supported domains

The MHA plugin `proc_counter` supports these signal domains:

- waveform to waveform
- spectrum to spectrum

23.2.2 Plugin Tags

testing signalhandling

23.2.3 Configuration

The plugin represents a variable node in the MHA configuration hierarchy.

Type	Description	Default
int	Counter for invocations of signal processing callback	(monitor)

24 All plugins tagged 'adaptive'

25 All plugins tagged 'algorithm-communication'

26 All plugins tagged 'audio-channels'

- *combinechannels*: Section 5.6 on page 31
- *delay*: Section 5.9 on page 36
- *example1*: Section 7.2 on page 59
- *example2*: Section 7.3 on page 60
- *example3*: Section 7.4 on page 61
- *example4*: Section 7.5 on page 62
- *example5*: Section 7.6 on page 64
- *example7*: Section 7.8 on page 66
- *fader_spec*: Section 5.10 on page 37
- *fader_wave*: Section 5.11 on page 38
- *matrixmixer*: Section 5.12 on page 39
- *route*: Section 5.13 on page 42
- *split*: Section 17.7 on page 141
- *steerbf*: Section 9.5 on page 85

27 All plugins tagged 'audiometer'

- *audiometerbackend*: Section 20.1 on page 155

28 All plugins tagged 'beamformer'

- *acSteer*: Section 6.1 on page 47
- *adm*: Section 21.1 on page 157
- *delaysum_wave*: Section 21.3 on page 161
- *steerbf*: Section 9.5 on page 85

29 All plugins tagged 'beamforming'

- **delaysum_spec**: Section 2.1 on page 2
- **rohBeam**: Section 2.2 on page 3

30 All plugins tagged 'binaural'

31 All plugins tagged 'calibration'

32 All plugins tagged 'classifier'

33 All plugins tagged 'compression'

34 All plugins tagged 'cross-fade'

35 All plugins tagged 'data-export'

36 All plugins tagged 'data-flow'

37 All plugins tagged 'data-import'

38 All plugins tagged 'dereverberation'

39 All plugins tagged 'directional'

40 All plugins tagged 'disk-files'

41 All plugins tagged 'example'

42 All plugins tagged ’feature-extraction’

43 All plugins tagged ’feedback-suppression’

44 All plugins tagged ’filter’

45 All plugins tagged 'filterbank'

46 All plugins tagged 'frequency-modification'

47 All plugins tagged 'generator'

48 All plugins tagged 'io'

49 All plugins tagged 'lab-streaming-layer'

50 All plugins tagged 'level'

51 All plugins tagged 'level-meter'

52 All plugins tagged 'level-modification'

53 All plugins tagged 'limiter'

54 All plugins tagged 'linear-algebra'

55 All plugins tagged 'math'

56 All plugins tagged ’multichannel’

- *delaysum_spec*: Section 2.1 on page 2

57 All plugins tagged ’music’

- *plingploing*: Section 18.2 on page 145

58 All plugins tagged ’network-communication’

- *ac2lsl*: Section 4.1 on page 14
- *ac2osc*: Section 4.2 on page 15
- *acmon*: Section 4.3 on page 16
- *lsl2ac*: Section 6.4 on page 54
- *osc2ac*: Section 6.5 on page 56

59 All plugins tagged ’noise-suppression’

- **noise_psd_estimator**: Section 16.1 on page 121
- **smooth_cepstrum**: Section 16.2 on page 123
- **windnoise**: Section 16.3 on page 125

60 All plugins tagged ’open-sound-control’

- *ac2osc*: Section 4.2 on page 15
- *osc2ac*: Section 6.5 on page 56

61 All plugins tagged ’overlap-add’

- *overlapadd*: Section 17.5 on page 135
- *smoothgains_bridge*: Section 14.2 on page 118
- *spec2wave*: Section 19.2 on page 149
- *wave2spec*: Section 19.4 on page 152

62 All plugins tagged 'plugin-arrangement'

63 All plugins tagged 'signal-enhancement'

64 All plugins tagged 'signal-generator'

65 All plugins tagged 'signal-transformation'

66 All plugins tagged 'signalflow'

67 All plugins tagged 'signalhandling'

68 All plugins tagged 'spatial'

69 All plugins tagged 'test-tool'

70 All plugins tagged 'testing'

71 All plugins tagged 'unit-testing'

References

Index