The Open Master Hearing Aid (openMHA)

4.5.5

Documentation of openMHA plugins (openMHA)

Contents

1 Plugin category 'AC-variables'

1.1 ac2wave

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

Spectral input is discarded and replaced by a zero signal.

1.1.0.1 Supported domains

The MHA plugin `ac2wave` supportes these signal domains:

- waveform to waveform

- spectrum to waveform

1.1.0.2 Categories

AC-variables

1.1.0.3 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)

1.2 acConcat_wave

Concatenating two or more waveforms into one

1.2.0.1 Detailed description

This plugin concatenates two or more waveforms in the given order into a new waveform all living in the AC space.

1.2.0.2 Supported domains

The MHA plugin `acConcat_wave` supportes these signal domains:

- waveform to waveform

1.2.0.3 Categories

AC-variables

1.2.0.4 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_conAC	string	Name of the concatenated waveform	vGCC_con_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)

1.3   acmon

This algorithm converts AC variables into parsable monitor variables.

1.3.0.1   Supported domains

The MHA plugin `acmon` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

1.3.0.2   Categories

AC-variables

1.3.0.3   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)

1.4 acPooling_wave

Pooling of several consecutive time frames

1.4.0.1 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.

1.4.0.2 Supported domains

The MHA plugin `acPooling_wave` supportes these signal domains:

- waveform to waveform

1.4.0.3 Categories

AC-variables

1.4.0.4 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:**]0,360]	33
pooling_wndlen	int	This parameter determines the length of the pooling window in msec. **Range:** [0,5000]	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:** [-1,360]	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
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

Variables of sub-parser `mhaconfig_in`:

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

Variables of sub-parser `mhaconfig_out`:

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

1.5 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.

1.5.0.1 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.

1.5.0.2 Supported domains

The MHA plugin `acsave` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

1.5.0.3 Categories

AC-variables

1.5.0.4 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)

1.6 acSteer

Steering Vector Loading Plugin

1.6.0.1 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

$$nsteerchan = nrefmix * nchan * nangle \tag{1}$$

1.6.0.2 Supported domains

The MHA plugin `acSteer` supportes these signal domains:

- spectrum to spectrum

1.6.0.3 Categories

AC-variables

1.6.0.4 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)

1.7   save_spec

Save signal spectrum to AC variable

1.7.0.1   Supported domains

The MHA plugin `save_spec` supportes these signal domains:

- spectrum to spectrum

1.7.0.2   Categories

AC-variables

1.7.0.3   Configuration variables

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

Variables of sub-parser `mhaconfig_in`:

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

Variables of sub-parser `mhaconfig_out`:

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

1.8 save_wave

Save signal waveform to AC variable

1.8.0.1 Supported domains

The MHA plugin `save_wave` supportes these signal domains:

- waveform to waveform

1.8.0.2 Categories

AC-variables

1.8.0.3 Configuration variables

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

Variables of sub-parser `mhaconfig_in`:

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

Variables of sub-parser `mhaconfig_out`:

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

2 Plugin category 'adaptive'

2.1 lpc

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

2.1.0.1 Detailed description

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

2.1.0.2 Supported domains

The MHA plugin `lpc` supportes these signal domains:

- waveform to waveform

2.1.0.3 Categories

adaptive

2.1.0.4 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)

2.2 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.

2.2.0.1 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

$$f(m) \quad = \quad f(m-1) + \kappa(m,2) * b(m-1,2) \tag{2}$$
$$b(m,1) \quad = \quad b(m-1,2) + \kappa(m,2) * f(m-1) \tag{3}$$

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

2.2.0.2 Supported domains

The MHA plugin `lpc_bl_predictor` supportes these signal domains:

- waveform to waveform

2.2.0.3 Categories

adaptive

2.2.0.4   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)

2.3   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.

2.3.0.1 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

$$dm(m-1) = \lambda * dm(m-1) + (1-\lambda) * (f(m-1)^2 + b(m-1,2)^2) \tag{4}$$

$$nm(m-1) = \lambda * nm(m-1) + (1-\lambda) * -2 * f(m-1) * b(m-1,2) \tag{5}$$

$$km(m,1) = \frac{nm(m-1)}{dm(m-1)}. \tag{6}$$

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

2.3.0.2 Supported domains

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

- waveform to waveform

2.3.0.3 Categories

adaptive

2.3.0.4 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)

3 Plugin category 'analysis'

3.1 acTransform_wave

Transform Plugin Between Coordinate Systems for Waveforms

3.1.0.1 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.

3.1.0.2 Supported domains

The MHA plugin `acTransform_wave` supportes these signal domains:

- waveform to waveform

3.1.0.3 Categories

analysis

3.1.0.4 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)

3.2 droptect

Plugin detects dropouts in channels that have a constant spectrum

3.2.0.1   Supported domains

The MHA plugin `droptect` supportes these signal domains:

- spectrum to spectrum

3.2.0.2   Categories

analysis

3.2.0.3   Configuration variables

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

Variables of sub-parser `mhaconfig_in`:

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

Variables of sub-parser `mhaconfig_out`:

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

4 Plugin category 'beamforming'

4.1 adm

Adaptive differential microphone

4.1.0.1 Detailed description

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

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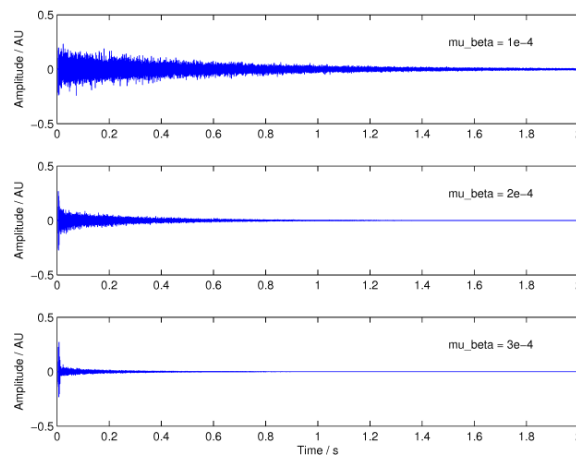

Figure 1 Output signals illustrating convergence of the ADM algorithm for three different values of `mu_beta` (input signal: white Gaussian noise exactly from behind)

4.1.0.2 Supported domains

The MHA plugin `adm` supportes these signal domains:

- waveform to waveform

4.1.0.3 Categories

beamforming multichannel

4.1.0.4 Configuration variables

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

Variables of sub-parser `mhaconfig_in`:

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

Variables of sub-parser `mhaconfig_out`:

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

4.2 steerbf

Steerable Beamformer

4.2.0.1 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**.

4.2.0.2 Supported domains

The MHA plugin `steerbf` supportes these signal domains:

- spectrum to spectrum

4.2.0.3 Categories

beamforming

4.2.0.4 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)

5 Plugin category 'binaural'

5.1 coherence

Coherence filter

5.1.0.1 Supported domains

The MHA plugin `coherence` supportes these signal domains:

- spectrum to spectrum

5.1.0.2 Categories

binaural filter noise-reduction

5.1.0.3 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
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)

6 Plugin category 'core'

6.1 identity

6.1.0.1 Detailed description

The simplest openMHA plugin.

This plugin does nothing.

6.1.0.2 Supported domains

The MHA plugin `identity` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

6.1.0.3 Categories

core

6.1.0.4 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 Plugin category 'delay'

7.1 delaysum

delay and sum plugin

7.1.0.1 Supported domains

The MHA plugin `delaysum` supportes these signal domains:

- waveform to waveform

7.1.0.2 Categories

delay and sum

7.1.0.3 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 **Range:** [0,[	[1 1]
delay	vector<int>	delay in number of frames **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)

8 Plugin category 'directional'

8.1 doasvm_classification

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

8.1.0.1 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.

8.1.0.2 Supported domains

The MHA plugin `doasvm_classification` supportes these signal domains:

- waveform to waveform

8.1.0.3 Categories

directional

8.1.0.4 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)

8.2 doasvm_feature_extraction

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

8.2.0.1   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.

8.2.0.2   Supported domains

The MHA plugin `doasvm_feature_extraction` supportes these signal domains:

- waveform to waveform

8.2.0.3   Categories

directional

8.2.0.4   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)

9 Plugin category 'example'

9.1 example1

9.1.0.1 Detailed description

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

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

9.1.0.2 Supported domains

The MHA plugin `example1` supportes these signal domains:

- waveform to waveform

9.1.0.3 Categories

example

9.1.0.4 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)

9.2   example2

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

9.2.0.1   Supported domains

The MHA plugin `example2` supportes these signal domains:

- waveform to waveform

9.2.0.2   Categories

example

9.2.0.3   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)

9.3   example3

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

9.3.0.1 Supported domains

The MHA plugin `example3` supportes these signal domains:

- waveform to waveform

9.3.0.2 Categories

example

9.3.0.3 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)

9.4 example4

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

9.4.0.1 Supported domains

The MHA plugin `example4` supportes these signal domains:

- spectrum to spectrum

9.4.0.2 Categories

example

9.4.0.3 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)

9.5 example5

example plugin configuration structure

9.5.0.1 Supported domains

The MHA plugin `example5` supportes these signal domains:

- spectrum to spectrum

9.5.0.2 Categories

example

9.5.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
channel	int	channel number to be scaled **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)

9.6 example6

example plugin configuration structure

9.6.0.1 Supported domains

The MHA plugin `example6` supportes these signal domains:

- waveform to waveform

9.6.0.2 Categories

example

9.6.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
channel	int	channel in which the RMS level is measured **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)

10 Plugin category 'feedback'

10.1 fshift_hilbert

Pitch shifter

10.1.0.1 Supported domains

The MHA plugin `fshift_hilbert` supportes these signal domains:

- spectrum to spectrum

10.1.0.2 Categories

feedback

10.1.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
df	vector<float>	frequency to shift the bins / Hz	[40]
fmin	float	lower boundary for frequency shifter **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`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(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 prediction_error

Prediction error model for adaptive feedback cancellation

10.2.0.1 Detailed description

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

10.2.0.2 Supported domains

The MHA plugin `prediction_error` supportes these signal domains:

- waveform to waveform

10.2.0.3 Categories

feedback

10.2.0.4 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
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 **Range:**]0,1024]	20
pred_err_delay	vector<int>	Delay in the forward path **Range:**]0,[	[96]
delay_w	vector<int>	Delay in the adaptive filtering path due to the microphone and loudspeaker transducers **Range:** [0,[	[130]
delay_d	vector<int>	Delay in the adaptive filtering path for the LPC **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)

11 Plugin category 'filter'

11.1 combinechannels

Channel combiner

11.1.0.1 Detailed description

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

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

11.1.0.2 Supported domains

The MHA plugin `combinechannels` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

11.1.0.3 Categories

filter

11.1.0.4 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)

11.2 fftfilterbank

FFT based filterbank with overlapping filters

11.2.0.1 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. 64) to get the waveform signal of the filterbank output. The *matrixmixer* (p. 73) plugin or *combinechannels* (p. 36) 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 `use_edges` 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.
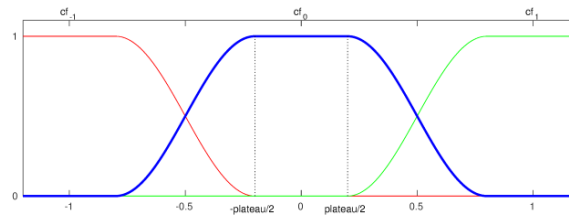

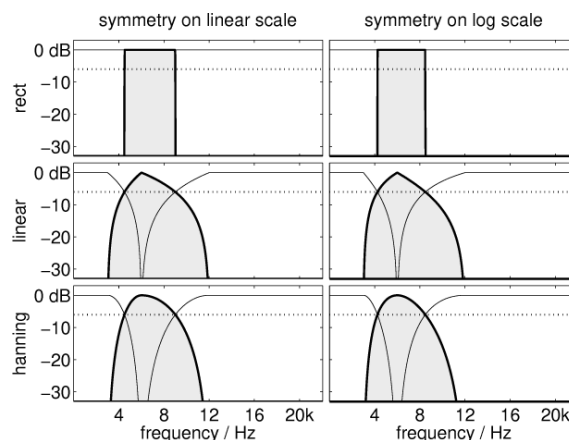**Figure 2 Schematic plot of overlapping filters**

Figure 3 Example filter shapes with center frequencies configured

11.2.0.2 Supported domains

The MHA plugin `fftfilterbank` supportes these signal domains:

- waveform to waveform
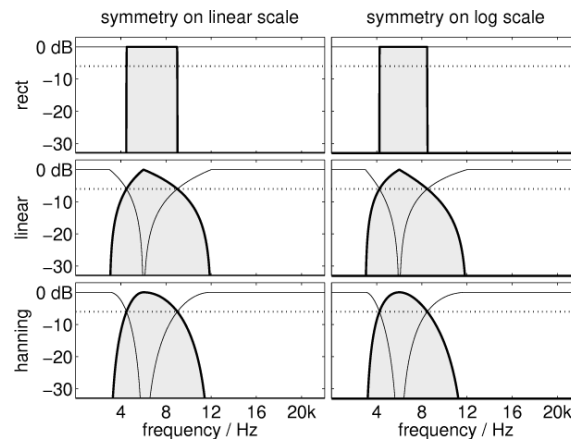
- spectrum to spectrum

11.2.0.3 Categories

filter

Figure 4 Example filter shapes with edge frequencies configured

11.2.0.4 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
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).	[]

11.3 iirfilter

IIR filter

11.3.0.1 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.

11.3.0.2 Supported domains

The MHA plugin `iirfilter` supportes these signal domains:

- waveform to waveform

11.3.0.3 Categories

filter

11.3.0.4   Configuration variables

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

11.4   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.

11.4.0.1   Detailed description

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

$$e[k] \quad = \quad y[k-1] - f[k-1]u[k-1] \tag{7}$$

$$f[k] \quad = \quad f[k-1] + rho/(|u|^2 + c)u[k-1]e[k], \tag{8}$$

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.

11.4.0.2   Supported domains

The MHA plugin `nlms_wave` supportes these signal domains:

- waveform to waveform

11.4.0.3   Categories

filter

11.4.0.4 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)

12   Plugin category 'generator'

12.1   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.

12.1.0.1   Supported domains

The MHA plugin `noise` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

12.1.0.2   Categories

generator

12.1.0.3   Configuration variables

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

12.2 sine

Sine wave generator.

12.2.0.1 Supported domains

The MHA plugin `sine` supportes these signal domains:

- waveform to waveform

12.2.0.2 Categories

generator

12.2.0.3 Configuration variables

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

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(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'

13.1   dc

dynamic compression

13.1.0.1   Supported domains

The MHA plugin `dc` supportes these signal domains:

- waveform to waveform
- spectrum to spectrum

13.1.0.2   Categories

level compression

13.1.0.3   Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
powersum	bool	Input level is summed accross channels	no
gtdata	matrix\<float\>	gaintable data in dB gains	[[]]
gtmin	vector\<float\>	input level for first gain entry in dB SPL	[]
gtstep	vector\<float\>	level step size in dB	[]
tau_rmslev	vector\<float\>	RMS level averaging time constant in s	[]
tau_attack	vector\<float\>	attack time constant in s	[]
tau_decay	vector\<float\>	decay time constant in s	[]
fb	string	Name of fftfilterbank plugin. Used to extract frequency information.	fftfilterbank
chname	string	name of audio channel number variable (empty: broadband)	
bypass	bool	bypass dynamic compression	no
clientid	string	Client ID of last fit	
gainrule	string	Gain rule of last fit	
preset	string	Preset name of last fit	
modified	int	Flag if configuration has been modified	(monitor)
max_level_difference	matrix\<float\>	maximum level difference in dB between adjacent bands **Range:** [0,[	[[]]
level_in	vector\<float\>	input level of last block / dB SPL	(monitor)
level_in_filtered	vector\<float\>	input level after time-constant filters / dB SPL	(monitor)
cf	vector\<float\>	nominal center frequencies of filterbank bands	(monitor)
ef	vector\<float\>	edge frequencies of filterbank bands	(monitor)
band_weights	vector\<float\>	Weights of the individual frequency bands. Computed as (sum of squared fft-bin-weigths) / num_frames.	(monitor)
use_wbinhib	bool	Use wideband inhibition?	no
wbinhib	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 `wbinhib`:

Name	Type	Description	Default
weights	vector<float>	weighting of neighbour frequencies	[1 2 4 4 4 2 1]
dl_map_min	float	mapping of level/broadband level ratio (in dB) into inhibition ratio, lower value means full inhibition	-8
dl_map_max	float	mapping of level/broadband level ratio (in dB) into inhibition ratio, upper value means no inhibition	2
l_min	float	level threshold, inhibition starts above this level	40
g_scale	matrix<float>	scaling factor of inhibition gain	[[1]]

13.2 dc_simple

Simple dynamic compression scheme

13.2.0.1 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. 5.

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}, which is calculated as described

in the documentation of the 'fftfbpow' plugin or in equation 8 of **?**. The attack filter is a first order low pass filter. The release filter is a maximum tracker, i.e.

$$L_a = \langle 20 \log_{10}(L_{st}) \rangle_{\tau_{attack}} \tag{9}$$
$$L_{in} = \max(L_a, \langle L_a \rangle_{\tau_{release}}) \tag{10}$$

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).
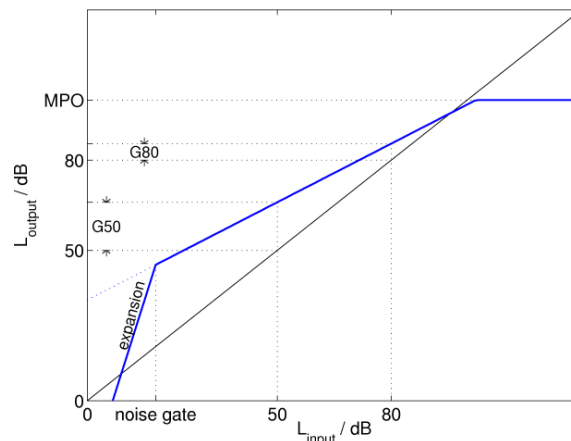
Figure 5 Input-output function of one channel in the *dc_simple* dynamic compression algorithm.

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

```
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.

13.2.0.2 Supported domains

The MHA plugin `dc_simple` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

13.2.0.3 Categories

level compression

13.2.0.4 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)

13.3 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

13.3.0.1 Supported domains

The MHA plugin `rmslevel` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

13.3.0.2 Categories

level

13.3.0.3 Configuration variables

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

Variables of sub-parser `mhaconfig_in`:

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

Variables of sub-parser `mhaconfig_out`:

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

13.4 softclip

The softclipper implements a broad band dynamic compression above a given level.

13.4.0.1 Supported domains

The MHA plugin `softclip` supportes these signal domains:

- waveform to waveform

13.4.0.2 Categories

level

13.4.0.3 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)

13.5 transducers

Signal level calibration plugin.

13.5.0.1 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. 49) shows the same level.

4. Create a test tone in the MHA (e.g. with 'noise', p. 43, or 'sine', p. 44) 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:

$$y = \mathrm{floor}(2^{(N-1)}x)2^{-(N-1)} \tag{11}$$

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

13.5.0.2 Supported domains

The MHA plugin `transducers` supportes these signal domains:

- waveform to waveform

13.5.0.3 Categories

level

13.5.0.4 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)

14 Plugin category 'noise'

14.1 timoSmooth

Cepstral smoothing single-channel noise reduction

14.1.0.1 Detailed description

Implements the single-channel noise reduction scheme found in Breithaupt, Gerkmann, and Martin, A Novel A Priori SNR Estimation Approach Based on Selective Cepstro-temporal Smoothing.

14.1.0.2 Supported domains

The MHA plugin `timoSmooth` supportes these signal domains:

- spectrum to spectrum

14.1.0.3 Categories

noise reduction

14.1.0.4 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	noisePowProposedScale
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

15 Plugin category 'overlapadd'

15.1 overlapadd

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

15.1.0.1 Detailed description

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

$$x_w(m, k) = w_1(k) \cdot x(m \cdot P + k), \tag{12}$$

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

After processing and inverse Fourier transformation, Hanning ramps are applied to the signal to avoid discontinuities in case of temporal aliasing, and thus reducing the artifacts. The length

of the Hanning ramps are a fraction p of half the zero-padding length $(N - M)/2$. If $p = 1$, the entire zero-padded parts are smoothed with Hanning ramps. $p = 0$ means, that no Hanning ramps are applied to the signal. This allows an exact reproduction in those cases, where the local impulse response of the filter (represented by all algorithms between FFT and inverse FFT) is shorter than the zero padding length. The windowing in both stages of the overlap-add mechanism is plotted in Fig. 6 for $M = 2P$ (50

The total delay between input and output of a real-time system with fragment size P and an overlap-add based linear-phase filter, is the window length plus half the zero-padding length, or $M + (N - M)/2$, plus an additional delay needed for the signal processing, and plus a delay generated by the AD/DA converters (e.g., anti-aliasing filter delay). In an offline system, the complete input signal is available in advance, and thus the delay of the overlap-add method is determined only by the relative shift between output and input signal, which is $(M + N)/2 - P$ (equal to $N/2$ in case of 50% overlap, i.e. $M = 2P$). Contrary to a real-time system, the delay of an offline system depends on the amount of overlap.
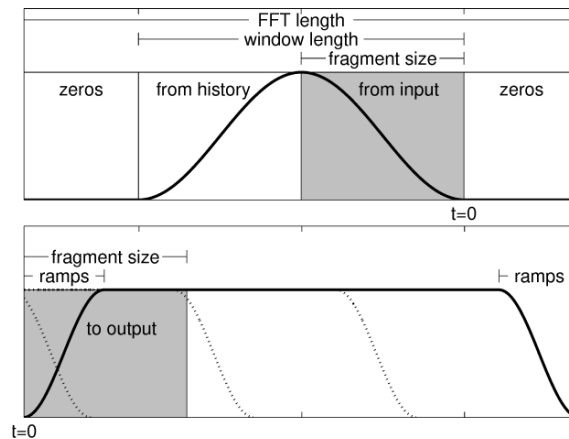
Figure 6 Windowing in the overlap-add method with 50% overlap and zero-padding. In the upper panel, the windowed input signal before applying the FFT is schematically plotted. In the lower panel, the same time interval after inverse FFT is shown. The shaded segment is the fragment which is read from 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.
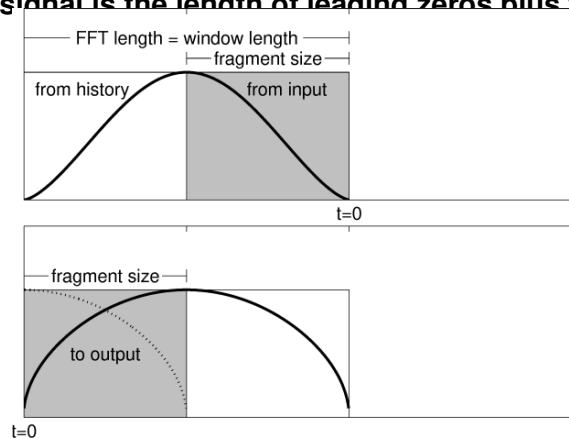
Figure 7 Windowing in the overlap-add method, as in Fig. 6, but with post-windowing and without zero-padding. In this setup, W^α is applied before FFT and $W^{1-\alpha}$ is used for post-windowing. The delay between input and output signal is the window length.

15.1.0.2 Supported domains

The MHA plugin `overlapadd` supportes these signal domains:

- waveform to waveform

15.1.0.3 Categories

overlapadd

15.1.0.4 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)
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).	[]

15.2 shadowfilter_begin

Save signal spectrum to AC variable

15.2.0.1 Detailed description

The plugins 'shadowfilter_begin' and 'shadowfilter_end' (section 15.3) 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.

15.2.0.2 Supported domains

The MHA plugin `shadowfilter_begin` supportes these signal domains:

- spectrum to spectrum

15.2.0.3 Categories

overlapadd

15.2.0.4   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)

15.3   shadowfilter_end

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

15.3.0.1   Detailed description

See section 15.2 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.

15.3.0.2   Supported domains

The MHA plugin `shadowfilter_end` supportes these signal domains:

- spectrum to spectrum

15.3.0.3 Categories

overlapadd

15.3.0.4 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)

15.4 smoothgains_bridge

Gain smoothing for reduction of filter length

15.4.0.1 Detailed description

The overlap-add framework allows filter lengths of the zero padding length. Longer filters will result in artifacts caused by circular aliasing. Artifacts can be reduced by either applying Hanning ramps to the zero-padded blocks after filtering, or by shortening the impulse response of the filter, and thus implicitly 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.

15.4.0.2   Supported domains

The MHA plugin `smoothgains_bridge` supportes these signal domains:

- spectrum to spectrum

15.4.0.3   Categories

overlapadd filter

15.4.0.4   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).	[]

15.5 spec2wave

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

15.5.0.1 Detailed description

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

15.5.0.2 Supported domains

The MHA plugin `spec2wave` supportes these signal domains:

- spectrum to waveform

15.5.0.3 Categories

overlapadd

15.5.0.4 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)

16 Plugin category 'resample'

16.1 downsample

Downsampling by integer fractions

16.1.0.1 Supported domains

The MHA plugin `downsample` supportes these signal domains:

- waveform to waveform

16.1.0.2 Categories

resample signalflow

16.1.0.3 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]

16.2 upsample

Upsampling by integer fractions

16.2.0.1 Supported domains

The MHA plugin `upsample` supportes these signal domains:

- waveform to waveform

16.2.0.2 Categories

resample signalflow

16.2.0.3 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]

17   Plugin category 'signalflow'

17.1   altplugs

Configure alternative plugins.

17.1.0.1   Detailed description

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

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

17.1.0.2   Supported domains

The MHA plugin `altplugs` supportes these signal domains:

- waveform to waveform

- waveform to spectrum

- spectrum to waveform

- spectrum to spectrum

17.1.0.3   Categories

signalflow

17.1.0.4 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)

17.2 analysispath

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

17.2.0.1 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.
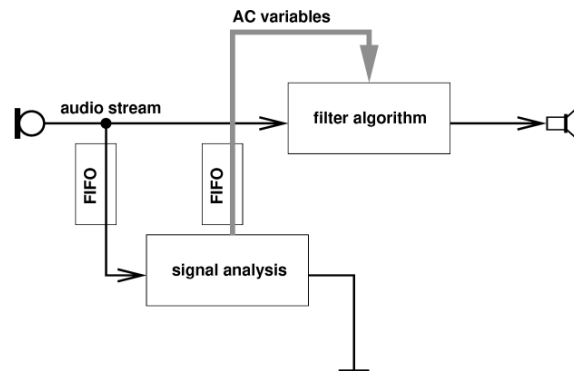
Figure 8 Schematic signal flow in the analsysis path scenario.

17.2.0.2   Supported domains

The MHA plugin `analysispath` supportes these signal domains:

- waveform to waveform

17.2.0.3   Categories

signalflow

17.2.0.4   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)

17.3   db

Synchronous double buffer plugin.

17.3.0.1   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. The configuration of the inner plugin is available via the `plug` prefix.

Please note that double buffering adds an extra delay of the audio stream. If both fragment sizes are identical, the double buffering is bypassed.
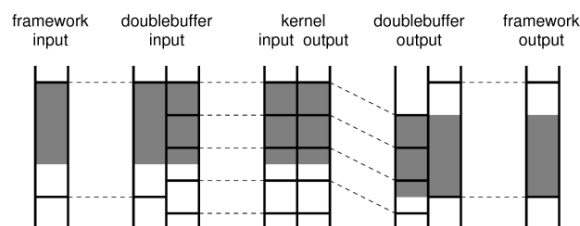
Figure 9 Concept of the double buffer plugin. The outer fragments, provided by the framework, are split up into smaller fragments for processing in the kernel. For a continuous output stream, an extra delay is needed, i.e. the first fragment is filled with zeros at the beginning.

17.3.0.2   Warning:

If the inner fragment size is larger than the outer fragment size, the maximal processing time is limited by the shorter fragment size. This results in a maximal processor usage determined by the ratio of outer to inner fragment size. This problem holds not for offline processing. As an alternative, the asynchronous double-buffer plugin `dbasync` (section **??**) can be used, which processes the double-buffered signal in a separate thread. That plugin should be preferred for real-time processing. If the inner thread should only be used for signal analysis, please refer to the plugin `analysispath` (section 17.2).

17.3.0.3 Supported domains

The MHA plugin `db` supportes these signal domains:

- waveform to waveform

17.3.0.4 Categories

signalflow

17.3.0.5 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)

17.4 delay

Delay line

17.4.0.1 Supported domains

The MHA plugin `delay` supportes these signal domains:

- waveform to waveform

17.4.0.2 Categories

signalflow

17.4.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
delay	vector<int>	delay in samples, one entry for each channel **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)

17.5 fader_wave

Apply level

17.5.0.1 Supported domains

The MHA plugin `fader_wave` supportes these signal domains:

- waveform to waveform

17.5.0.2 Categories

signalflow gain

17.5.0.3 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)

17.6 matrixmixer

Matrix mixer

17.6.0.1 Supported domains

The MHA plugin `matrixmixer` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

17.6.0.2 Categories

signalflow

17.6.0.3 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	[[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)

17.7 mhachain

MHA Chain

17.7.0.1 Detailed description

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

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

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

17.7.0.2   Supported domains

The MHA plugin `mhachain` supportes these signal domains:

- waveform to waveform

- waveform to spectrum

- spectrum to waveform

- spectrum to spectrum

17.7.0.3   Categories

signalflow

17.7.0.4   Configuration variables

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

Variables of sub-parser `mhaconfig_in`:

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

Variables of sub-parser `mhaconfig_out`:

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

17.8   resampling

Synchronous resampling plugin.

17.8.0.1 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.

17.8.0.2 Warning:

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

17.8.0.3 Supported domains

The MHA plugin `resampling` supportes these signal domains:

- waveform to waveform

17.8.0.4 Categories

signalflow

17.8.0.5 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)

17.9   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.

17.9.0.1   Supported domains

The MHA plugin `route` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

17.9.0.2   Categories

signalflow

17.9.0.3   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)

17.10   split

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

17.10.0.1   Detailed description

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

17.10.0.2   Supported domains

The MHA plugin `split` supportes these signal domains:

- waveform to waveform

- waveform to spectrum

- spectrum to waveform

- spectrum to spectrum

17.10.0.3   Categories

signalflow

17.10.0.4   Configuration variables

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

18   Uncategorized plugins

18.1   cpuload

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

18.1.0.1   Supported domains

The MHA plugin `cpuload` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

18.1.0.2 Categories

other

18.1.0.3 Configuration variables

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

Variables of sub-parser `mhaconfig_in`:

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

Variables of sub-parser `mhaconfig_out`:

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

18.2 fader_spec

fader

18.2.0.1 Supported domains

The MHA plugin `fader_spec` supportes these signal domains:

- spectrum to spectrum

18.2.0.2 Categories

other

18.2.0.3 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)

18.3 gain

Gain plugin:

Apply a gain to each channel

18.3.0.1 Supported domains

The MHA plugin `gain` supportes these signal domains:

- waveform to waveform

- spectrum to spectrum

18.3.0.2 Categories

other

18.3.0.3 Configuration variables

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

18.4 noisePowProposedScale

noise power estimator after Gerkmann (2012)

Copyright © 2011 Author: Timo Gerkmann and Richard Hendriks Royal Institute of Technology (KTH) Delft university of Technology

18.4.0.1 Supported domains

The MHA plugin `noisePowProposedScale` supportes these signal domains:

- spectrum to spectrum

18.4.0.2 Categories

other

18.4.0.3 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)

18.5 wavrec

wav file recorder

18.5.0.1 Supported domains

The MHA plugin `wavrec` supportes these signal domains:

- waveform to waveform

18.5.0.2 Categories

other

18.5.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
fifolen	int	Length of FIFO in samples **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

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)

References

Index