SOFA/AmbiX Binaural Rendering (SABRE) Toolkit

Joseph G. Tylka
josephgt@princeton.edu

v0.11 – October 26th, 2017

Summary

The SOFA/ambiX binaural rendering (SABRE) toolkit is an open-source collection of MATLAB functions for generating custom binaural decoders for the ambiX binaural plug-in, which renders higher-order ambisonics to binaural. This document describes the methods implemented in the toolkit and provides instructions for its use.

1 Introduction

This document is structured as follows. In Section 2, we describe the ambiX mathematical conventions and subsequently, in Section 3, we describe the ambisonics decoding approaches implemented in this toolkit. Then, in Section 4, we discuss the various processing options that can be applied to the HRTFs. This information was originally published by Tylka and Choueiri [1], but may be updated here to reflect the current version of the toolkit. Finally, in Section 5, we describe the various functions included in the toolkit and how to use them.

1.1 Contents

The toolkit consists of the following items:

1. the source LaTeX files for this manual (`/doc/`);
2. some DSP-related MATLAB functions (`/dsp/`);
3. a MATLAB file containing examples of how to use the toolkit (`/examples.m`);
4. some MAT files containing spherical grids and corresponding quadrature weights (`/grids/`);
5. some SOFA-formatted HRTFs (`/hrtfs/`); and
6. the core set of MATLAB functions for the toolkit (`/SABRE_*.m`).

2 Conventions

In accordance with the ambiX specification [2], we use real-valued spherical harmonics as given by

$$Y_l^m(\theta,\phi) = N_l^{|m|} P_l^{|m|}(\sin\theta) \times \begin{cases} \cos m\phi & \text{for } m \geq 0, \\ \sin |m|\phi & \text{for } m < 0, \end{cases}$$

where P_l^m is the associated Legendre polynomial of degree l and order m, as defined in the MATLAB `legendre` function by[1]

$$P_l^m(x) = (-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}P_l(x), \qquad \text{with} \qquad P_l(x) = \frac{1}{2^l l!}\left[\frac{d^l}{dx^l}(x^2-1)^l\right],$$

and N_l^m is a normalization term which, for the Schmidt seminormalized (SN3D) spherical harmonics with Condon-Shortley phase,[2] is given by [2]

$$N_l^m = (-1)^m\sqrt{\frac{2-\delta_m}{4\pi}\frac{(l-m)!}{(l+m)!}},$$

where δ_m is the Kronecker delta. With an inner product defined by integrating over all directions, the squared-norm of these spherical harmonics is thus given by

$$\|Y_l^m\|^2 = \frac{1}{2l+1}.$$

We also adopt the ambisonics channel numbering (ACN) convention [2] such that for a spherical harmonic function of degree $l \in [0,\infty)$ and order $m \in [-l,l]$, the ACN index n is given by $n = l(l+1)+m$ and we denote the spherical harmonic function by $Y_n \equiv Y_l^m$.

3 Decoding Ambisonics

Implemented in this toolkit are several basic methods of decoding ambisonics (described below), but the Ambisonic Decoder Toolbox (ADT)[3] is a much more comprehensive tool for creating state-of-the-art ambisonic decoders [3]. Consequently, one intended use of this toolkit is to add custom HRTFs to an existing ambiX decoder preset (such as those generated using the ADT).[4] Note that doing so requires the user to specify the grid of speaker positions, as that information is not explicitly contained in ambiX decoder presets.

Generally, the decoding matrix, $\mathbf{D}$, determines the loudspeaker signals, x_q, given by

$$\mathbf{x} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_Q(t) \end{bmatrix} = \mathbf{D}\cdot\begin{bmatrix} a_0(t) \\ a_1(t) \\ \vdots \\ a_{N-1}(t) \end{bmatrix} = \mathbf{D}\cdot\mathbf{a}, \tag{1}$$

where Q is the number of loudspeakers and a_n is the ambisonic signal for ACN index n. For a thorough review of ambisonic decoding theory and practice, the reader is referred to the works of Heller et al. [3, 4].

[1] See: `https://www.mathworks.com/help/matlab/ref/legendre.html`

[2] Note that including Condon-Shortley phase in the normalization term cancels it in the associated Legendre term.

[3] The ADT is available online at: `https://bitbucket.org/ambidecodertoolbox/adt.git`

[4] At present, the SABRE toolkit is only compatible with single-band decoders. Consequently, multi-band decoders should be implemented in parallel (as a bank of single-band decoders), downstream of a crossover network.

Assuming a free field and ideal loudspeakers that are equidistant from the listener, the resulting binaural pressure signals are given by

$$p^{\mathrm{L,R}}(t) = \mathbf{h}^{\mathrm{L,R}} * \mathbf{x} = \sum_{q=1}^{Q} h_q^{\mathrm{L,R}}(t) * x_q(t), \tag{2}$$

where '$*$' denotes convolution and

$$\mathbf{h}^{\mathrm{L,R}} = \begin{bmatrix} h_1^{\mathrm{L,R}}(t) & h_2^{\mathrm{L,R}}(t) & \cdots & h_Q^{\mathrm{L,R}}(t) \end{bmatrix} \tag{3}$$

is a vector containing the head-related impulse responses (HRIRs) for a given listener and for the directions of each loudspeaker. The superscript "L,R" denotes the response at the left or right ear, respectively.

3.1 Basic Decoding

Following traditional ambisonic theory, we consider the ambisonic signals produced at the center of the loudspeaker array in response to the loudspeaker signals, given by

$$a_n(t) = \sum_{q=1}^{Q} Y_n(\hat{v}_q) x_q(t). \tag{4}$$

Equivalently, in matrix form, we have $\mathbf{a} = \mathbf{Y} \cdot \mathbf{x}$, where

$$\mathbf{Y} = \begin{bmatrix} Y_0(\hat{v}_1) & Y_0(\hat{v}_2) & \cdots & Y_0(\hat{v}_Q) \\ Y_1(\hat{v}_1) & Y_1(\hat{v}_2) & \cdots & Y_1(\hat{v}_Q) \\ \vdots & \vdots & \ddots & \vdots \\ Y_{N-1}(\hat{v}_1) & Y_{N-1}(\hat{v}_2) & \cdots & Y_{N-1}(\hat{v}_Q) \end{bmatrix}. \tag{5}$$

From this formulation, we obtain the basic (pseudoinverse) decoder, given by [4, Appendix A.1]

$$\mathbf{D} = \mathbf{Y}^{+}, \tag{6}$$

where $(\cdot)^{+}$ denotes pseudoinversion.

3.2 Quadrature Decoding

Given higher-order ambisonics signals, a_n, the so-called *signature function*, μ, in the direction $\hat{v}_q$ is given by [5]

$$\mu(t, \hat{v}_q) = \sum_{n=0}^{N-1} a_n(t) \frac{Y_n(\hat{v}_q)}{\|Y_n\|^2}. \tag{7}$$

Equivalently, in matrix form, we have

$$\begin{bmatrix} \mu(t, \hat{v}_1) \\ \mu(t, \hat{v}_2) \\ \vdots \\ \mu(t, \hat{v}_Q) \end{bmatrix} = \mathbf{Y}^{\mathrm{T}} \cdot \mathbf{F}^{-1} \cdot \mathbf{a}, \tag{8}$$

where Q is now the number of plane-wave terms and $\mathbf{F}$ is a diagonal matrix given by

$$\mathbf{F} = \mathrm{diag}\left\{ \begin{bmatrix} \|Y_0\|^2 & \|Y_1\|^2 & \cdots & \|Y_{N-1}\|^2 \end{bmatrix} \right\}. \tag{9}$$

The signature function represents the coefficients of a plane-wave decomposition of the sound field, such that the binaural pressure signals can be approximately reconstructed using a finite number of plane-waves, given by [5]

$$p^{\mathrm{L,R}}(t) \approx \sum_{q=1}^{Q} h_q^{\mathrm{L,R}}(t) * (w_q \mu(t, \hat{v}_q)) = \mathbf{h}^{\mathrm{L,R}} * \begin{bmatrix} w_1 \mu(t, \hat{v}_1) \\ w_2 \mu(t, \hat{v}_2) \\ \vdots \\ w_Q \mu(t, \hat{v}_Q) \end{bmatrix}, \tag{10}$$

where w_q is the quadrature weight of the q^{th} plane-wave term and is dependent on the chosen grid of directions. Rearranging, we arrive at the quadrature decoder, given by

$$\mathbf{D} = \mathrm{diag}\left\{ \begin{bmatrix} w_1 & w_2 & \cdots & w_Q \end{bmatrix} \right\} \cdot \mathbf{Y}^{\mathrm{T}} \cdot \mathbf{F}^{-1}. \tag{11}$$

3.3 Compact Decoding

Now that we have established the typical decoding and binaural rendering signal chain, we can derive an equally valid but more computationally efficient approach. We first combine the HRIRs by performing the matrix multiplication with the decoder matrix, which yields a vector of N compacted HRIRs

$$\tilde{\mathbf{h}}^{\mathrm{L,R}} = \begin{bmatrix} \tilde{h}_0^{\mathrm{L,R}}(t) & \tilde{h}_1^{\mathrm{L,R}}(t) & \cdots & \tilde{h}_{N-1}^{\mathrm{L,R}}(t) \end{bmatrix} = \mathbf{h}^{\mathrm{L,R}} \cdot \mathbf{D}. \tag{12}$$

This process simply combines the decoding matrix and per-direction HRIRs into a single set of filters.[5] The corresponding "compact" decoder is simply an $N \times N$ identity matrix, i.e., $\tilde{\mathbf{D}} = \mathbf{I}_{(N \times N)}$, such that $\tilde{\mathbf{h}}^{\mathrm{L,R}} \cdot \tilde{\mathbf{D}} = \mathbf{h}^{\mathrm{L,R}} \cdot \mathbf{D}$. It's worth noting that, in the case of the basic pseudoinverse decoder, given by Eq. (6), the compacting process described by Eq. (12) is equivalent to to computing the spherical-harmonics coefficients of the HRTFs, as done by Rafaely and Avni [6, Sec. IV].

[5] Conceptually, each compacted HRIR $\tilde{h}_n^{\mathrm{L,R}}(t)$ represents the signals at the ears in response to an impulse sent through the n^{th} HOA channel.

3.4 Normalization

For each HRIR (compacted or not), we first compute the maximum gain, α_q, across the entire frequency response. Subsequently, we attenuate each HRIR and amplify the corresponding row of the decoder matrix by that gain, i.e.,

$$\hat{\mathbf{h}}^{\mathrm{L,R}} = \mathbf{h}^{\mathrm{L,R}} \cdot \mathbf{G}^{-1}, \qquad \text{and} \qquad \hat{\mathbf{D}} = \mathbf{G} \cdot \mathbf{D}, \tag{13}$$

where

$$\mathbf{G} = \mathrm{diag}\left\{ \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_Q \end{bmatrix} \right\}, \tag{14}$$

such that $\hat{\mathbf{h}}^{\mathrm{L,R}} \cdot \hat{\mathbf{D}} = \mathbf{h}^{\mathrm{L,R}} \cdot \mathbf{D}$. Finally, we normalize the overall decoder matrix such that the maximum absolute value of any element in matrix is unity.

4 HRTF Processing

This toolkit requires HRTFs to be stored stored in SOFA format [7]. The HRTF files contain, among other things, the measured HRIRs and the grid of corresponding measurement positions. Depending on the decoder used, the HRTFs may need to be interpolated, and depending on the intended playback system (e.g., type of headphones), the HRTFs may need to be equalized. Consequently, the SABRE toolkit contains several options for carrying out these processes.

4.1 Interpolation

When measured HRTFs are not available at the desired grid positions, interpolation is performed through one of the following methods.

Nearest Neighbor: By default, we perform nearest-neighbor interpolation. This is carried out by minimizing the ℓ^2 norm (Euclidean distance) between the desired position $\vec{v}_{q'}$ and each measurement position $\vec{u}_q$.

Time Domain: Alternatively, we can perform weighted-average interpolation of the HRIRs for three different interpolation schemes: natural neighbor, linear, and spherical-harmonic.[6] Generally, for some function f_q measured at positions $\vec{u}_q$, the interpolated values, $f'_{q'}$, for all desired positions $\vec{v}_{q'}$, are given by

$$\begin{bmatrix} f'_1 & f'_2 & \cdots & f'_{Q'} \end{bmatrix} = \begin{bmatrix} f_1 & f_2 & \cdots & f_Q \end{bmatrix} \cdot \mathbf{W}, \tag{15}$$

where each element, $w_{q,q'}$, of $\mathbf{W}$ is the interpolation weight from measurement position $\vec{u}_q$ to the desired position $\vec{v}_{q'}$. Before interpolating, we first measure the onset delays, $\tau_q^{\mathrm{L,R}}$, using a 10%

[6] For spherical-harmonic interpolation, the interpolation weights are given as a matrix by $\mathbf{W} = \mathbf{Y}_Q^{+}\mathbf{Y}_{Q'}$, where $\mathbf{Y}_Q$ is given by Eq. (5) for all measurement positions and up to some maximum order (by default, $L = 4$), and $\mathbf{Y}_{Q'}$ is the same for all desired positions.

(−20 dB) threshold for each impulse response. We then align all of the impulse responses such that their onsets coincide at the earliest onset, and separately store the relative delays, given by

$$d_q^{\mathrm{L,R}} = \tau_q^{\mathrm{L,R}} - \min\left(\min_q \tau_q^{\mathrm{L}},\ \min_q \tau_q^{\mathrm{R}}\right). \tag{16}$$

Then we compute interpolation weights from each measurement position to each desired position and interpolate, using Eq. (15), the time-aligned impulse responses and the relative delays. Finally, we introduce the interpolated time delays to each interpolated impulse response.

Frequency Domain: We can also interpolate by first decomposing the HRTFs into magnitude spectra and time delays. The magnitude spectra are given (in dB) by

$$M_q^{\mathrm{L,R}}(f) = 10\log_{10}\left(\left|H_q^{\mathrm{L,R}}(f)\right|^2\right), \tag{17}$$

where H denotes the Fourier transform of h. We then interpolate the magnitude responses and time delays using Eq. (15). The interpolated magnitude responses are then converted into minimum-phase impulse responses, and the interpolated onset delays are introduced to yield the final interpolated HRIRs.

4.1.1 Interpolation Threshold

Optionally, we may apply a threshold to determine which desired positions are close enough to a measurement position such that they do not require interpolation. For each desired position, we find the nearest measurement position and compute the angular distance between the two, given by

$$\psi = \cos^{-1}\left(\frac{\vec{u}_q \cdot \vec{v}_{q'}}{\|\vec{u}_q\| \cdot \|\vec{v}_{q'}\|}\right), \tag{18}$$

where $\|\cdot\|$ denotes the ℓ^2 norm of a vector. If this angular distance exceeds a user-specified threshold, then the selected interpolation method is carried out. Otherwise, nearest-neighbor interpolation is used.

4.2 Equalization

For optimal binaural playback, one should use individually equalized headphones [8]. As this may not always be possible, we provide several methods of equalization so that the user may try to compensate for the equalization of the headphones. We design the equalization filters using the full set of measured HRTFs and apply them to the (possibly interpolated) HRTFs for the desired positions.

None: By default, the HRTFs are not equalized. This option should only be used if the playback headphones will be individually equalized on the user's ears.

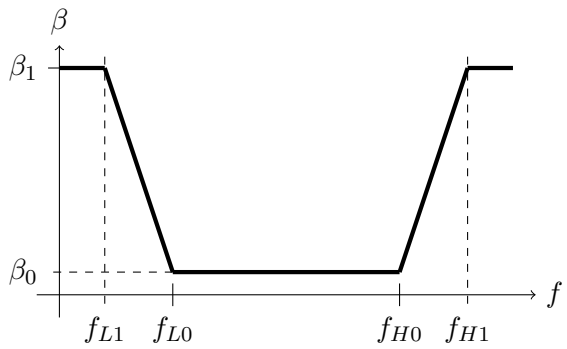

Figure 1: Frequency-dependent regularization function of the inverse filters for HRTF equalization.

Frontal: For headphones that use frontal-incidence "free-field" equalization, we can equalize all HRTFs by the HRTF pair nearest to $(\theta, \phi) = (0, 0)$. The transfer function of the regularized inverse filter is given by [9]

$$Z(f) = \frac{H^*(f)}{H^*(f)H(f) + \beta(f)}, \tag{19}$$

where $(\cdot)^*$ denotes complex conjugation and β is a frequency-dependent regularization function. This function is defined by a set of parameters, which are defined graphically in Fig. 1, and whose default values are given by

$$\begin{aligned} \beta_0 &= 10^{-4}, \quad f_{L0} = 50 \text{ Hz}, \quad f_{H0} = 21 \text{ kHz}, \\ \beta_1 &= 10^{-2}, \quad f_{L1} = 20 \text{ Hz}, \quad f_{H1} = 22 \text{ kHz}. \end{aligned}$$

Diffuse: For headphones that employ diffuse-field equalization, we can equalize the HRTFs by the average magnitude spectrum over all directions. This is computed as the omnidirectional term of the spherical-harmonic decomposition of the HRTF magnitude spectra (in dB), where the decomposition is computed using the pseudoinverse of $\mathbf{Y}$, given by Eq. (5) for all measurement directions and up to order $L = 4$. The equalization filter is then computed for the average magnitude spectrum using Eq. (19).

Horizontal: Alternatively, we can compute an average HRTF over all horizontal-plane directions. The procedure for this is very similar to the diffuse-field equalization, but the average spectrum is computed using only the HRTFs with elevation $|\theta| < 5°$. The equalization filter is then computed for the average magnitude spectrum using Eq. (19).

5 MATLAB Functions

The core MATLAB functions included in the toolkit are listed and described in Table 1.

Function	Description
SABRE_AddHRTFDelays	Introduces specified time delays into HRTF pairs; returns the delayed HRIRs.
SABRE_AmbiXPath	Returns the ambiX binaural decoder preset directory.
SABRE_BasicDecoder	Designs a basic or quadrature ambisonics decoder up to a specified ambisonics order and for a specified grid of positions.
SABRE_BinauralRenderer	Creates and exports a custom ambiX binaural preset. This is the primary function for using the toolkit; most other functions are called within this function.
SABRE_EqualizationFilters	Designs equalization filters for a set of HRTFs.
SABRE_EqualizeHRTFs	Applies equalization filters to a set of HRTFs.
SABRE_GetDecoder	Loads an existing or designs a new ambisonics decoder.
SABRE_InterpolateHRTFs	Interpolates measured HRTFs to a desired grid; returns the interpolated HRTFs.
SABRE_LoadGrid	Loads a spherical grid and quadrature weights from a MAT file.
SABRE_LoadHRTFs	Loads HRTFs from a SOFA file; returns the HRIRs, measurement grid, and sample rate.
SABRE_NormalizeDecoder	Normalizes BIRs and decoder gains.
SABRE_ReadConfigFile	Reads an ambiX binaural preset file; returns the decoder matrix, HRTF filenames, and any global parameters specified in the file.
SABRE_RemoveHRTFDelays	Removes delays from HRTF pairs; returns the time-aligned HRIRs and original delays.
SABRE_RendererSettings	Assembles inline binaural renderer settings into structs `config` and `flags`.
SABRE_SphericalHarmonic	Computes ambiX real-valued spherical harmonics; returns a matrix of spherical harmonics up to a specified ambisonics order and for a specified grid of positions.
SABRE_Start	Starts the SABRE Toolkit and the SOFA API.
SABRE_WriteConfigFile	Writes an ambiX binaural preset file given the decoder matrix, HRTF filenames, and any global parameters.

Table 1: Core MATLAB functions in the toolkit.

5.1 Creating a Binaural Renderer Configuration

The function `SABRE_BinauralRenderer` is the primary function used for creating a binaural renderer configuration, and there are two ways to do this.

Option 1: The direct method of creating a binaural renderer is to call `SABRE_BinauralRenderer`, where the first 3 arguments must be

1. the output path,
2. the ambisonics order of the decoder, and
3. the SOFA file path,

followed by any optional settings (see Section 5.2 below). The corresponding MATLAB code for this might look like:

```
configFile = fullfile(SABRE_AmbiXPath,'3D3A-SABRE','example1.config');
maxOrder = 1;
sofaFile = fullfile('hrtfs','Subject2.sofa');
[config, flags] = SABRE_BinauralRenderer(configFile, maxOrder, sofaFile);
```

Option 2: An equivalent but less direct method is to first call `SABRE_RendererSettings`. This function takes as input a single cell array containing pairs of settings, and returns two structs, `config` and `flags`, which are then passed directly to `SABRE_BinauralRenderer`. This approach may be more useful than the first when many options need to be specified. The corresponding MATLAB code for this might look like:

```
temp = {'Output', configFile, 'Order', maxOrder, 'HRTF', sofaFile};
[config, flags] = SABRE_RendererSettings(temp);
[config, flags] = SABRE_BinauralRenderer(config, flags);
```

5.2 Renderer Settings

As discussed in previous sections, the toolkit implements a variety of methods and settings. As alluded to in the previous section, there are two ways of specifying these options when creating the binaural renderer configuration. All possible options are listed in Table 2.

Option 1: The first method is to specify all of the desired options inline when creating the binaural renderer configuration. In this case, after the first 3 required inputs, the options are listed as pairs, for example:

```
SABRE_BinauralRenderer(..., 'equalization', 'diffuse', 'sample rate', 48000);
```

The options specified above would resample (if necessary) the HRTFs to 48 kHz and apply diffuse-field equalization.

Option 2: The second method is to specify the options in the cell array of renderer settings. In this case, the options are again listed as pairs, but may be given in any order. For example:

```
temp = {'equalization', 'diffuse', ..., 'sample rate', 48000};
[config, flags] = SABRE_RendererSettings(temp);
[config, flags] = SABRE_BinauralRenderer(config, flags);
```

Note however that the 3 main settings (output path, decoder order, and SOFA file path) MUST be specified in the same cell array.

Acknowledgements

This toolkit relies on the ambiX ambisonic plug-in suite[7] by M. Kronlachner and requires head-related transfer functions (HRTFs) stored in the SOFA format.[8] Accordingly, the SOFA API[9] for MATLAB is needed to import the HRTF files.

References

[1] Joseph G. Tylka and Edgar Y. Choueiri. A Toolkit for Customizing the ambiX Ambisonics-to-Binaural Renderer. In *Audio Engineering Society Convention 143*, October 2017. URL http://www.aes.org/e-lib/browse.cfm?elib=19351.

[2] Christian Nachbar, Franz Zotter, Etienne Deleflie, and Alois Sontacchi. ambiX - A Suggested Ambisonics Format. In *Proceedings of the 3rd Ambisonics Symposium*, June 2011. URL http://ambisonics.iem.at/proceedings-of-the-ambisonics-symposium-2011/ambix-a-suggested-ambisonics-format.

[3] Aaron J. Heller, Eric M. Benjamin, and Richard Lee. A Toolkit for the Design of Ambisonic Decoders. In *Linux Audio Conference*, April 2012. URL http://lac.linuxaudio.org/2012/papers/18.pdf.

[4] Aaron Heller, Richard Lee, and Eric Benjamin. Is My Decoder Ambisonic? In *Audio Engineering Society Convention 125*, October 2008. URL http://www.aes.org/e-lib/browse.cfm?elib=14705.

[5] Ramani Duraiswami, Dmitry N. Zotkin, Zhiyun Li, Elena Grassi, Nail A. Gumerov, and Larry S. Davis. High Order Spatial Audio Capture and Its Binaural Head-Tracked Playback Over Headphones with HRTF Cues. In *Audio Engineering Society Convention 119*, October 2005. URL http://www.aes.org/e-lib/browse.cfm?elib=13369.

[7] See: http://www.matthiaskronlachner.com/?p=2015
[8] See: https://www.sofaconventions.org
[9] See: https://github.com/sofacoustics/sofa

Option	Value	Description
'output'	PATH	Specify output file path for the ambiX .config file. NOTE: this input is mandatory.
'order'	L	Specify the ambisonics order of the decoder. NOTE: this input is mandatory.
'hrtf'	PATH	Specify the SOFA file path for the HRTFs. NOTE: this input is mandatory.
'grid'	R	Use a custom speaker grid specified by R. R should be a $P \times 3$ matrix, where each row is a Cartesian vector. By default, the entire measurement grid in the SOFA file is used.
'method'	METHOD	Specify the interpolation METHOD to be used. By default, nearest-neighbor interpolation is used.
'domain'	DOMAIN	Specify the DOMAIN in which to interpolate. Must also specify an interpolation method.
'threshold'	THRESHOLD	Specify the THRESHOLD for interpolation. If a measurement exists within THRESHOLD degrees from a desired position, then the nearest neighbor to that desired position is used.
'equalization'	TYPE	Apply equalization of a given TYPE to the HRTFs. By default, no equalization is applied.
'sample rate'	Fs	Specify desired HRTF sample rate for the decoder. By default, the sample rate of the HRTFs in the SOFA file is used for the decoder impulse responses.
'decoder'	PATH	Load existing ambiX decoder. PATH should be the path to an existing ambiX .config file. NOTE: to use this option, you MUST also specify the corresponding grid.
'weights'	W	Use quadrature-weighted decoder with weights W. W should be a length P vector. NOTE: Weights are ignored if an existing ambiX decoder is used.
'compact'	X	Compact the decoder into a square matrix. X should evaluate to either true or false.
'normalization'	X	Normalize the decoder on a per-channel basis. X should evaluate to either true or false.

Table 2: Options accepted by SABRE_BinauralRenderer or SABRE_RendererSettings to generate the config and flags structs.

[6] Boaz Rafaely and Amir Avni. Interaural cross correlation in a sound field represented by spherical harmonics. *The Journal of the Acoustical Society of America*, 127(2):823–828, 2010. doi: http://dx.doi.org/10.1121/1.3278605. URL `http://scitation.aip.org/content/asa/journal/jasa/127/2/10.1121/1.3278605`.

[7] AES69-2015. AES69-2015: AES standard for file exchange - Spatial acoustic data file format, 2015.

[8] Zora Schärer and Alexander Lindau. Evaluation of Equalization Methods for Binaural Signals. In *Audio Engineering Society Convention 126*, May 2009. URL `http://www.aes.org/e-lib/browse.cfm?elib=14917`.

[9] Angelo Farina. Advancements in Impulse Response Measurements by Sine Sweeps. In *Audio Engineering Society Convention 122*, May 2007. URL `http://www.aes.org/e-lib/browse.cfm?elib=14106`.