Technical University of Denmark

RABAT Documentation

February 25, 2013

David Jamal Duhalde Rahbæk
Mathias Immanuel Nielsen
Oliver Ackermann Lylloff

1 Introduction

1.1 Background

This toolbox was developed to implement the necesarry functions for making room acoustic measurements with *MATLAB*. The functions can be used as building blocks to make measurements with any computer that has *MATLAB* and a soundcard.

About the authors

The toolbox was developed by three students from *Technical University of Denmark*. All three are master students in engineering acoustics with an interest in room and build acoustics. The development of this toolbox started with the need for a simple and freely available tool to perform room acoustic measurements.

1.2 Installation

The newest version of this toolbox can be downloaded from `https://github.com/RBAtoolbox`. After downloading the toolbox the path should be added in *MATLAB*. This can be done through the menu *File Set Path* or using the `pathtool`-command.

Performing measurements requires that PsychToolbox[1] is installed. PsychToolbox contains functionality that allows for low-latency asynchrounous playback and recording of audio signal on multiple platforms.

[1] See `http://psychtoolbox.org/HomePage`, for information on how to install it

2 List of functions

2.1 Basic Functions

Circular convolution	`rbaCircularConv`
Linear convolution	`rbaLinearConv`
Crop impulse response	`rbaCropIR`
Cross correlation	`rbaCrossCorr`
Stadardized octave bands	`rbaGetFreqs`

2.2 Signal Generation

Generate excitation signals	`rbaGenerateSignal`

2.3 Measurement

Measurement procedure	`rbaMeasurement`

2.4 Deconvolution

Deconvolve sweep	`sweepdeconv`
Deconvovle mls	`mlsdeconv`

2.5 Post-processing

Centre time T_s	`rbaCentreTime`
Clarity C	`rbaClarity`
Definition D	`rbaDefinition`
Early decay time	`rbaEDT`
Filter bank	`rbaFilterBank`
Filter impulse response into bands	`rbaIR2OctaveBands`
Lundeby's method	`rbaLundeby`
Schröders backwards integration	`rbaSchroeder`

3 rbaLinearConv

3.1 Purpose

A faster replacement of the built-in function for linear convolution.

3.2 Syntax

```
y = rbaLinearConv(f,h)
```

3.3 Algorithm

The function uses that a convolution in the time domain is equal to a multiplication in the frequency domain.

$$\mathcal{F}\{h_1 * h_2\} = \mathcal{F}\{h_1\}\mathcal{F}\{h_2\}$$

3.4 Description

This function does convolution faster than the built-in matlab function `conv`. It can be used to convolve an impulse response with a dry signal to create auralizations.

The output of the function is not normalized.

3.5 Example

3.6 See Also

`rbaCircularConv`

4 rbaCircularConv

4.1 Purpose

A faster replacement of the built-in function for circular convolution.

4.2 Syntax

```
y = rbaCircularConv(f,h)
```

4.3 Algorithm

The function uses that a convolution in the time domain is equal to a multiplication in the frequency domain.

$$\mathcal{F}\{h_1 * h_2\} = \mathcal{F}\{h_1\}\mathcal{F}\{h_2\}$$

4.4 Description

This function does convolution faster than the built-in matlab function `conv`. It can be used to convolve periodic signals, when processing of infinite length (i.e undefined length).

The output of the function is not normalized.

4.5 See Also

`rbaLinearConv`

5 rbaCropIR

5.1 Purpose

Crop impulse response to remove onset and noise.

5.2 Syntax

```
[hCrop,t,Onset,knee] = rbaCropIR(h,fs)
[hCrop,t,Onset,knee] = rbaCropIR(h,fs,'onset')
[hCrop,t,Onset,knee] = rbaCropIR(h,fs,'tight')
```

5.3 Description

Impulse responses need to be cropped in order to produce proper results when computing parameters such as `rbaReverberationTime`. ISO 3382-1 provides a procedure to determine the onset of a broadband impulse response: The onset is the point where the impulse response rises significantly above the background noise but is more than 20 dB below the maximum.

The knee-point where the decay meets the noise floor can be determined by Lundeby's method.

5.4 Optional Input Arguments

- `'onset'`: Crop only onset
- `'tight'`: Crop onset and noise by Lundeby's method

5.5 Example

```
[h,fs] = wavread('ImpulseResponse.wav');
[hCrop,t] = rbaCropIR(h,fs,'onset');
```

5.6 See Also

`rbaLundeby`, `rbaReverberationTime`

5.7 References

ISO 3382-1.

6 rbaCrossCorr

6.1 Purpose

Calculate cross correlation

6.2 Syntax

```
[c, lags] = rbaCrossCorr(f, g)
```

6.3 Algorithm

The definition of cross correlation is

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f^*[m]g[n+m]$$

The function makes use of the following: That a convolution in time, is a multiplication in frequency.

$$\mathcal{F}\{f \star g\} = \mathcal{F}\{f\}\mathcal{F}\{g\}$$

6.4 Description

Since Matlabs built-in cross correlation sends calculations to `fftfilt`, when `length(f)>=10*length(h)`, and `fftfilt` does not support longer signals than $2^{20} = 1048576$, the need for a more robust implementation was born. This function is written with the purpose of determining the exact position of a single sweep response in a room acoustic measurement, which consists of the response of several sweeps. The cross correlation searches for patterns, similar to h, in signal f. It is closely related to the convolution function.

`c` is the result of the cross-correlation for n `lags` is a vector containing the lags n

6.5 See Also

`rbaLinearConv`, `rbaCircularConv`.

7 rbaSchroeder

7.1 Purpose

Compute decay curves from Schröders backward integration method.

7.2 Syntax

```
R = rbaSchroeder(h,fs,flag)
R = rbaSchroeder(h,fs,flag,'Lundeby')
R = rbaSchroeder(h,fs,flag,knee)
```

- `flag = {1,0}`: Enables noise compensation (tested but not implemented yet).

7.3 Algorithm

7.4 Description

The decay curve

$$R(t) = 10 \cdot \log_{10} \left(\int_{\infty}^{t} h^2(t) \, \mathrm{d}(-\tau) \right) \tag{1}$$

$$R(t) = 10 \cdot \log_{10} \left(\int_{t_1}^{t} h^2(t) \, \mathrm{d}(-\tau) + \int_{t_1}^{\infty} h^2_{exp}(t) \, \mathrm{d}(-\tau) \right) \tag{2}$$

The latter integral is an optional correction factor to minimize the influece of noise (enabled by setting `flag = 1`). It is calculted under the assumption of exponential decay of energy with the same slope as $h^2(t)$ between t_0 and t_1, where t_0 is the time corresponding to a level 10 dB higher than the level at t_1. The slope is calculated from the time-averaged energy decay and a linear least-squares fit is found to $y(t) = 10 \cdot \log_{10} \left(h^2_{exp}(t) \right)$, with

$$h^2_{exp}(t) = E_0 \mathrm{e}^{At}. \tag{3}$$

The coefficients of the linear fit $y_{fit}(t) = at + b$ is connected to the exponential coefficients E_0 and A, by

$$E_0 = 10^{b/10} \, , \quad A = \ln \left(10^{a/10} \right) \tag{4}$$

7.5 Optional Input Arguments

- `'Lundeby'`: Use Lundeby's method to calculate the knee-point where the exponential decay meets the noise floor (default).

- `knee`: Index value of the knee-point found by e.g. a graphical approach.

7.6 Example

7.7 See Also

`rbaLundeby`, `rbaCropIR`

7.8 References

8 rbaGenerateSignal

8.1 Purpose

Generate measurement signal.

8.2 Syntax

```
[y,t] = rbaGenerateSignal('logsin',fs,f1,f2,length_sig[,zero_pad,amp,phase])
[y,t] = rbaGenerateSignal('linsin',fs,f1,f2,length_sig[,zero_pad,amp,phase])
[y,t] = rbaGenerateSignal('sin',fs,f0,length_sig[,amp,phase])
[y,t] = rbaGenerateSignal('mls',fs, n[, amp, zero_pad, offset])
```

8.3 Description

`rbaGenerateSignal` is used to generate the measurement signal. The available types of signal are

- Logarithmic sine sweep
- Linear sine sweep
- Single sine tone
- Maximum-length sequence

Each signal type requires different input arguments. There is no time-window applied to the generated signal. If af window is needed, e.g. for real world measurements using sweep signal, use `sweepwin`.

When applying a time-window using `sweepwin` it is necesarry to generate a sweep with a wider frequency range than the desired measurement range.

8.4 Optional Input Arguments

- `zero_pad`: duration in seconds, of silence inserted after the generated signal.
- `amp`: the default amplitude of each signal is 1. Use this argument if another is needed.
- (Sweep)`phase`: Initial phase in radians of the signal. Default is 0.
- (MLS)`offset`: Needed offset in MLS sequence.

8.5 Example

```
% Generate logarithmic sine sweep from 100 Hz to 10 kHz of length 5 s
sig_type = 'logsin';
fs = 48e3;
f1 = 100;
f2 = 10e3;
length_sig = 5;
[y, t] = rbaGenerateSignal(sig_type, fs, f1, f2, length_sig);
```

8.6 See Also

`rbaMeasurement`

9 rbaMeasurement

9.1 Purpose

Perform measurement. This function requires PsychToolbox `http://psychtoolbox.org/HomePage`

9.2 Syntax

```
y = rbaMeasurement(signal, fs, N, estimatedRT)
```

9.3 Description

Connections are established to the default input and output of the computers operating system, and a system or room response is recorded. The input `signal` is used to excite the room. The measurements are averaged over `N` recordings.
Each excitation is followed by a quiet time of `estimatedRT` seconds. This method is called a transient measurement, and while it takes slightly longer, than with a sound field build-up, it is very reliable and produces robust results. `estimatedRT` should be an estimate of the reverberation time.

9.4 Example

```
% Generate logarithmic sine sweep
fs = 48000;          % Sampling frequency
f1 = 1000;           % Lower frequency
f2 = 8000;           % Upper frequency
length_sig = 5;      % Duration of sweep in seconds

[sweep,t] = rbaGenerateSignal('logsin',fs,f1,f2,length_sig);

% Start measurement
RT = 1;              % Estimated reverberation time of room
N = 4;               % number of sweeps to average over
transient = 0;
y = rbaMeasurement(sweep,fs,N,RT,transient);
% Compute impulse response
h = sweepdeconv(sweep,y,f1,f2,fs);

figure('Name','Spectrogram','Position',[0 400 400 400])
specgram(y,min(256,length(y)),fs)
figure('Name','Impulse response','Position',[420 400 400 400])
plot(h)
```

9.5 See Also

`rbaGenerateSignal`

10 mlsdeconv

10.1 Purpose

Computes the circular crosscorrelation function between signals `x` and `y`.

10.2 Syntax

```
[h,t] = mlsdeconv(x,y,fs[,'AmpCorr'])
```

10.3 Description

The circular crosscorrelation function between input and output signals is a good approximation of the impulse response of a system. Given `y` is the output signal of a *linear* system, and `x` is the output, this is used to do real-time analysis on mls measurements over long periods of time.

`mlsdeconv` uses the utility function `flip` to flip the elements of `x`.

10.4 Optional Input Arguments

- `'AmpCorr'`: The use of this string enables the option of correcting the impulse response with a scale factor.

10.5 See Also

`sweepdeconv`, `rbaLinearConv`, `rbaCircularConv`.

10.6 References

11 sweepdeconv

11.1 Purpose

Performs a linear deconvolution of input and output of a system. This can be a room measurement, being correlated with the excitation signal, which ends up in the room impulse response.

11.2 Syntax

```
[h,t] = sweepdeconv(x,y,f1,f2,fs)
```

11.3 Description

A deconvolution is basically a cross correlation between measurement and excitation signal. When doing measurements with logarithmic sweeps, the linear deconvolution should be used, since the sweep response will have a decay, that longens the signal.

11.4 See Also

`mlsdeconv`, `rbaLinearConv`, `rbaCircularConv`

12 rbaCentreTime

12.1 Purpose

Calculates the room acoustic parameter Centre Time, according to ISO 3382-1:2009.

12.2 Syntax

```
Ts = rbaCentreTime(h,fs)
```

12.3 Algorithm

The parameter Centre Time is defined as follows:

$$T_S = \frac{\int_0^\infty t h(t)\,\mathrm{d}t}{\int_0^\infty h^2\,\mathrm{d}t} \tag{5}$$

where $h(t)$ is the room impulse response of interest.

12.4 See Also

`rbaClarity`, `rbaDefinition`, `rbaEDT`, `rbaReverberationTime`.

12.5 References

ISO 3382-1:2009

13 rbaClarity

13.1 Purpose

Calculates Clarity (e.g C_{30} or C_{80}) from impulse response, according to ISO 3382-1:2009

13.2 Syntax

```
C = rbaClarity(h,fs,time)
```

13.3 Algorithm

The clarity parameter is calculated according to ISO-3382-1

$$C_{t_e} = 10 \log \frac{\int_0^{t_e} h^2(t) \mathrm{d}t}{\int_{t_e}^{\infty} h^2(t) \mathrm{d}t} \mathrm{dB} \tag{6}$$

where t_e is the input time constant, and $h(t)$ is the room implulse response of interest.

13.4 See Also

`rbaCentreTime`, `rbaDefinition`, `rbaEDT`, `rbaReverberationTime`.

13.5 References

ISO 3382-1:2009

14 rbaDefinition

14.1 Purpose

Calculate definition D_50 from impulse response, according to ISO-3382-1:2009.

14.2 Syntax

```
D = rbaDefinition(h,fs,time)
```

14.3 Algorithm

The definition parameter is calculated according to ISO-3382-1

$$D_{t_e} = \frac{\int_0^{t_e} h^2(t)\,\mathrm{d}t}{\int_0^{\infty} h^2(t)\,\mathrm{d}t} \tag{7}$$

where $h(t)$ is the room implulse response of interest.

14.4 See Also

`rbaCentreTime`, `rbaClarity`, `rbaEDT`, `rbaReverberationTime`.

14.5 References

ISO 3382-1:2009

15 rbaEDT

15.1 Purpose

Calculate Early Decay Time (EDT) from decay curve.

15.2 Syntax

```
[EDT,r2p,stdDev] = rbaEDT(h,t)
```

15.3 Description

EDT is defined in ISO-3382-1 and calculated from the slope of the integrated impulse response. The slope is determined from a linear regression of the inital 10 dB decay.

`r2p` is the goodness-of-fit given by the correlation coefficient $1000 \cdot (1 - r^2)$. A correlation coefficient $1000 \cdot (1 - r^2) = 0$ corresponds to a perfect correlation between data and fit. The decay curve is non-linear for $1000 \cdot (1 - r^2) > 10$ according to ISO 3382-1.

`stdDev` is the standard deviation of the difference between the linear regression line and the decay curve.

15.4 Example

```
[h,fs] = wavread('ImpulseResponse.wav');
t = 0:1/fs:length(h)/fs-1/fs;
R = rbaSchroeder(h,fs);
[EDT,r2p,stdDev] = rbaEDT(R,t);
```

15.5 See Also

`rbaReverberationTime`, `rbaSchroeder`, `rbaCropIR`, `rbaLundeby`.

15.6 References

ISO 3382-1:2009

16 rbaIR2OctaveBands

16.1 Purpose

Filter impulse response into 1-octave or 1/3-octave bands

16.2 Syntax

```
H = rbaIR2OctaveBands(h, fs, cfmin, cfmax)
H = rbaIR2OctaveBands(h, fs, cfmin, cfmax, bandsPerOctave)
H = rbaIR2OctaveBands(h, fs, cfmin, cfmax, bandsPerOctave, reverse)
```

16.3 Description

The room acoustic parameters are often computed in octave or third-octave frequency bands. This function will return a matrix `H` containing the filtered impulse response for each octave bands with center frequencies in the range `cfmin;cfmax]`.
The function uses fourth-order Butterworth filters with cut-off frequencies at the edges of the band. This function requires the *Signal Processing Toolbox* included with most versions of *MATLAB*, but also available from the *MathWorks* website.
The function includes the option to filter the time-reversed impulse response. This can be done to minimize the influence of the filter on the decay-curve as described in [] and [].

16.4 Example

```
[h, fs] = wavread('measured_impulse_response');
cfmin = 63;
cfmax = 4000;
H = rbaIR2OctaveBands(h, fs, cfmin, cfmax);
```

16.5 See Also

`rbaFilterBank`, `rbaGetFreqs`

16.6 References

Jacobsen, F. (..). A Note on Acoustic Decay Measurement, 1–8.
Jacobsen, F. (..). Time-reversed decay Measurement. (F. Jacobsen, Ed.), 1–4.

17 rbaLundeby

17.1 Purpose

Determine the sample (knee-point) at which the sound decay of IR meets the noise-floor.

17.2 Syntax

```
[knee, rmsNoise] = rbaLundeby(h,fs)
[knee, rmsNoise] = rbaLundeby(h,fs,maxIter)
[knee, rmsNoise] = rbaLundeby(h,fs,maxIter,avgTime)
[knee, rmsNoise] = rbaLundeby(h,fs,maxIter,avgTime,noiseHeadRoom)
[knee, rmsNoise] = rbaLundeby(h,fs,maxIter,avgTime,noiseHeadRoom,dynRange)
```

Note that the best results are obtained if the impulse response `h` has been cropped at the onset.

17.3 Algorithm

This function implements Lundeby's method to determine the point where the decay curve meets the noise floor. The algorithm is described in Lundeby's original paper and a newly published Convention paper (see references).

17.4 Description

Lundeby's methods determines the knee-point at which the sound decay of an impulse response meets the noise floor. The algorithm show fast convergence and generally produce stable results. The optional input arguments can be set by the user if the default values does not provide convergence.

17.5 Optional Input Arguments

- `maxIter` is the maximum number of iterations (default = 5)
- `avgTime` is the time averaging in seconds (default = 50ms). Longer intervals are better for lower frequencies
- `noiseHeadRoom` Interval in dB above noise level to be used for fitting curve (default = 10 dB)
- `dynRange` Dynamic range in dB to be used in calculating fit. Larger ranges are better if impulse response allows it (default = 20 dB)

17.6 Example

17.7 See Also

17.8 References

Lundeby, A, T E Vigran, H Bietz, and M Vorlander. *Uncertainties of Measurements in Room Acoustics.* Acta Acustica United with Acustica 81 (4): 344–355. (1995)

Antsalo, P, A Makivirta, V Valimaki, T Peltonen, and M Karjalainen. *Estimation of Modal Decay Parameters From Noisy Response Measurements.* Convention paper. (2012)

18 rbaReverberationTime

18.1 Purpose

Compute T_{20}, T_{30} and T_{60} from decay curves according to ISO 3382-1.

18.2 Syntax

```
[RT, r2p,dynRange,stdDev] = rbaReverberationTime(R,t,'best')
[RT, r2p,dynRange,stdDev] = rbaReverberationTime(R,t,'all')
```

18.3 Algorithm

The slope of the decay curves is determined from MATLAB's `polyfit` function in the intervals defined by T_{60}, T_{30} and T_{20}.

The correlation coefficient squared is given by

$$r^2 = \frac{\sum_{i=1}^{n}(L_i^{\text{fit}} - \bar{L}^{\text{data}})^2}{\sum_{i=1}^{n}(L_i^{\text{data}} - \bar{L}^{\text{data}})^2} \tag{8}$$

where $\bar{L}^{\text{data}} = \frac{1}{n}\sum_{i=1}^{n} L_i^{data}$

18.4 Description

Reverberation time is defined in ISO 3382-1 as the duration required for the space-averaged sound energy density in an enclosure to decrease by 60 dB after the source emission has stopped. This is calculated from the slope of decay curves by reverse-time integration of squared impulse responses, also known as Backward Schröder integration. The slope is determined from a linear regression based on the available dynamic range.

`r2p` is the goodness-of-fit given by the correlation coefficient $1000 \cdot (1 - r^2)$. A correlation coefficient $1000 \cdot (1 - r^2) = 0$ corresponds to a perfect correlation between data and fit. The decay curve is non-linear for $1000 \cdot (1 - r^2) > 10$ according to ISO 3382-1.

`dynRange` is the dynamic range of the decay curve in dB. If `R` is a matrix then `dynRange` is given for each column in `R`.

`stdDev` is the standard deviation of the difference between the linear regression line and the decay curve.

18.5 Optional Input Arguments

- `'best'`: Output only the reverberation time (T_{60},T_{30},T_{20}) with the largest dynamic range.
- `'all'`: Output all available T_{60},T_{30},T_{20} based on the dynamic range. The output is given in a matrix or vector based on the input.

18.6 Example

```
[h,fs] = wavread('ImpulseResponse.wav');
t = 0:1/fs:length(h)/fs-1/fs;    % Construct time vector
H = rbaIR2octBands(h,fs,63,8000,0); % Filter h in octave bands from 63 Hz to 80
R = rbaSchroeder(h,fs);    % Compute decay curves
[RT, r2p,dynRange,stdDev] = rbaReverberationTime(R,t,'all')
```

If a dynamic range of 65 dB is available then T_{60}, T_{30} and T_{20} is computed. For instance,

```
RT =
   5.196  7.447  7.376  5.968  4.976  4.308  3.344  2.349  1.414  0.849
   5.126  7.147  7.476  5.948  4.956  4.368  3.144  2.249  1.214  0.949
   5.296  6.447  7.676  5.948  4.476  4.208  3.944  2.149  1.814  0.747
```

18.7 See Also

`rbaCentreTime`, `rbaClarity`, `rbaDefinition`, `rbaEDT`.

18.8 References

ISO 3382-1:2009

19 rbaFilterBank

19.1 Purpose

Calculate octave or 3rd-octave band filters according to the ANSI S1.11-2004. Note: The filters are only trustworthy for frequencies at and above 63 Hz at this time.

19.2 Syntax

```
[B,A] = rbaFilterBank(BandsPerOctave,fs,cfmin,cfmax)
```

19.3 Description

Calculates filter coefficients for octave or 1/3-octave 3rd-order Butterworth bandpass filters. The function is used by `rbaIR2OctaveBands` to filter impulse response.

Generally `rbaIR2OctaveBands` should be used to filter impulse response.

19.4 See Also

`rbaIR2OctaveBands`, `rbaGetFreqs`.

19.5 References

20 rbaGetFreqs

20.1 Purpose

Return standardized octave band center frequencies vector for plotting purposes in acoustics.

20.2 Syntax

```
freqs = rbaGetFreqs(cfmin,cfmax,BandsPerOctave)
```

20.3 Description

`cfmin` and `cfmax` determine the range and `BandsPerOctave` should be `1` or `3`.

20.4 Example

```
cfmin = 63;
cfmax = 4e3;
BandsPerOctave = 1;
f = rbaGetFreqs(cfmin, cfmax);
% Plot reverberation time for each octave-band
plot(f, RT)
```

20.5 See Also

`rbaFilterBank`, `rbaIR2OctaveBands`.

21 rbaScaleModel

21.1 Purpose

Convert impulse response measured in a scale model to full-scale impulse response. Note that this function is still under development and further testing is needed in order to verify its functionality.

21.2 Syntax

```
[hFull,tFull] = rbaScaleModel(hModel,fsModel,K,fFull,T,hr,Pa)
```

- `hModel` is the impulse response measured in a $1/K$ scale model
- `fsModel` is the recording sampling rate
- `K` is the scale factor, e.g. 10 or 20
- `fFull` is the full-scale octave bands of interest
- `T` is the temperature in degrees Celcius. Vector containing model values and reference in full-scale.
- `hr` is the relative humidity in %. Vector containing model values and reference in full-scale.
- `Pa` is the ambient atmospheric pressure in kPa. Vector containing model values and reference in full-scale.

21.3 Description

`rbaScaleModel` scales length, time, frequency, sampling rate and calculate the full-scale impulse response. The impulse response measured in a scale model is attenuated (due to sound absorption in air) more at high frequencies. This function calculates the discrepancy between air absorption in the scaled frequency intervals.

Sound absorption in air depends on frequency, temperature, humidity, atmospheric pressure and hence distance of propagation. It follows that the conversion of an impulse response to full scale requires taking the absorption discrepancy into account. If m_{mod} denotes the air attenuation coefficent in the scale model and m_{full} denotes the full scale, then the discrepancy can be determined by applying a time dependent gain factor $H_n(t)$ to the octave

band filtered impulse response around f_n

$$H_n(t) = 10^{b_n t/20} \tag{9}$$

$$b_n = m_{mod}(10 f_n)c' - K m_{full}(f_n)c \tag{10}$$

where c' and c is the speed of sound in the measurement and reference, respectively.

21.4 Optional Input Arguments

21.5 Example

```
[hMod,fsMod] = wavread('modelIR.wav');
K = 10; % Scale factor
fFull = rbaGetFreqs(125,4000,1);  % Octave bands of interest
fMod = fRef*K;  % Scale to model frequencies
% Full scale ambience (reference values)
TFull = 25; % Temperature
hrFull = 40;  % Relative humidity
PaFull = 101.325; % Atmospheric pressure
% Model ambience (measured values)
TMod = 17;  % Temperature
hrMod = 40; % Relative humidity
PaMod = 101.325;  % Atmospheric pressure
[hFull,tFull] = rbaScaleModel(hMod,fsMod,K,fFull,[TFull TMod],...
[hrFull hrMod],[PaFull PaMod]);
```

21.6 See Also

`rbaGetFreqs`

21.7 References

ISO 9613-1.

Boone, M M, and E Braat-Eggen. *Room Acoustic Parameters in a Physical Scale Model of the New Music Centre in Eindhoven: Measurement Method and Results.*. Applied Acoustics 42 (1): 13–28. (1994)