Modal Filter Reverberation

Juri Lukkarila
Aalto University
School of Electrical Engineering
Department of Signal Processing and Acoustics

juri.lukkarila@aalto.fi

Abstract

A relatively new artificial reverberation algorithm based on modal filters is covered. Using modal analysis to produce a paraller filter structure where each filter represents one acoustic mode, any number of acoustic spaces and objects, both real and imaginary, can be simulated accurately. An acoustic mode can be modeled with a digital filter by matching the filter resonance frequency and damping to the mode frequency and decay time. The paraller filter structure can then be used for synthesizing high quality artificial reverberation. The desired reverberation time and frequency response can be designed directly with the algorithm, and each mode can be controlled precisely and interactively. A modal density of 1000 to 2000 modes is sufficient to model most acoustic systems, leading to low memory requirements. A practical implementation of the algorithm done using MATLAB is presented to validate and examine the algorithm.

1 Introduction

Reverberation is an acoustic phenomenon, in which reflections cause the sound to be prolonged and ring even after the sound source stops [1]. In an enclosed space, the direct sound is heard first, which is followed by the so called early reflections from nearby surfaces [2, p. 526-527]. As the amount of reflections increases, these isolated arrivals become indistinguishable from each other and combine to create a diffuse sound field that typically decays exponentially [2, p. 526-530]. This dense tail of the impulse response is generally referred as late reverberation, and the time it takes for it to decay 60 decibels in amplitude is called the reverberation time T_{60} [1, 3]. A generic structure of reverberation is illustrated in figure 1.

Since reverberation is a natural phenomenon present in all situations where there is sound and sound reflecting surfaces, all environments have their own particular reverberation, and therefore, it directly affects the sense of space in that environment. Artificial reverberation refers to creating and simulating this acoustic reverberation mechanically, electrically or digitally, which first became necessary with the advent of music recording and broadcasting. Previously, specially constructed echochambers and various electromechanical devices were used to produce reverberation artificially, but as with most audio

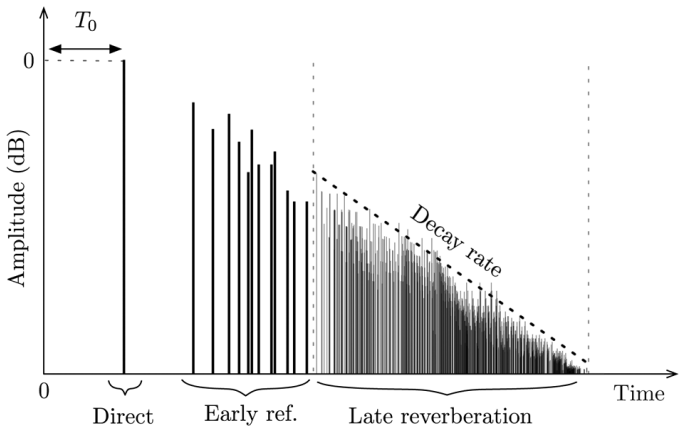


Figure 1: Example of a generic room impulse response, image from [1].

technology, digital technology has become the predominant method for artificial reverberation, and it offers many practical advantages over the analog methods. Today, digital reverberation algorithms are used extensively in audio production, as well as in game and virtual environment applications. [1, 4]

Digital reverberation algorithms were first proposed in the early 1960s, and research into new algorithms and improvements has been ongoing ever since. Välimäki et al. divided the currently known algorithms into four broad categories [4]:

- **delay networks**, which utilize delay lines, filters and feedback;
- **convolution algorithms**, which colvolve the input signal with a real or simulated impulse response;
- **computational acoustics**, were the propagation of sound waves is simulated;
- **virtual analog models**, which simulate the old analog and electromechanical devices.

This paper covers a relatively new method for artificial reverberation called modal reverberation or modal filter reverberation. The idea was mentioned by Karjalainen and Järvinen in 2001 [3], and it was fully developed into an algorithm by Abel, Coffin and Spratt in a paper published in 2014 [5]. In the modal reverb algorithm, the acoustic behaviour of a space is described with modal analysis, which is then used to synthesize artificial reverberation. The modes of a system, which in this case can be either a real space or any imaginary object, are implemented as a paraller combination of resonant filters, where each filter corresponds to one mode. [3, 5]

A mode can be simulated with a filter by matching the filter resonance frequency and damping to the modal frequency and decay time. With a sufficiently large number of modeled modes, any acoustic space can be accurately simulated. As each filter operates and can be adjusted separately, the modal filter structure enables explicit and precise interactive control of the reverberation in respect to both time and frequency. One of the

main advantages of the method is that the desired parametric behavior, like reverberation time and magnitude response as a function of frequency can be designed easily. Furthermore, the method allows for the movement of the source and listener positions, as well as morphing between different acoustic spaces. [3, 4, 5]

The modal analysis of acoustic spaces is overviewed in section 2. Modal decomposition is described in section 3. Section 4 presents the modal reverberation algorithm and practical implementation examples, and section 5 concludes the paper. The MATLAB code used for the figures and implementation of the algorithm is included in appendix A.

2 Modal Analysis

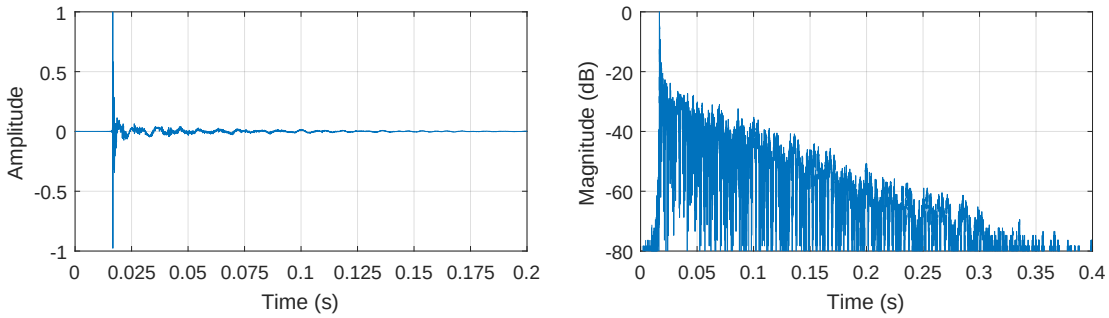
Acoustic modes can be understood as the resonance frequencies of the acoustic system. The resonances are created by constructive and destructive interference of the sound waves propagating in the space. At low frequencies, a perfect constructive interference between sound waves traveling between two or more room boundaries creates standing waves at relatively well isolated frequencies related to the dimensions of the space, which cause large variations in the sound pressure at different points. At higher frequencies, instead of standing waves, the numerous reflections resulting from the corresponding smaller wavelengths combine to create an irregular and complex pattern of interferences. Therefore, as the frequency increases, a small number of distinctive modes at lower frequencies morph into a complex combination of an increasing amount of overlapping modes. [6, p. 56-57, 198-203]

Modes cause the following acoustic behavior [6, p. 199]:

- Narrow-band peaks and attenuation in the frequency response.
- Ringing in the time domain.
- Variation in both of the above depending on location.

The acoustics of a space can be described by the time and frequency response of the space. An example room impulse response and the corresponding frequency response, measured in an average-sized living room, are presented in figures 2 and 3. The exponential decay of the impulse response as a function of time can be seen clearly in 2b, in which the impulse response has been squared and is presented on a logarithmic scale. Exponential decay corresponds to a straight line on a logarithmic scale. [2, p. 525-526]

In the frequency response, distinctive peaks and valleys can be observed on the low frequencies up to around 150 Hz, with a particularly strong narrow-band peak at approximately 100 Hz. At higher frequencies, a large number of small variations characterize the response curve in accordance with the theory. The resonant, ringing behavior of modes and the overall time-decaying nature of the room response can be seen in figure 4, where the example frequency response is plotted in relation to time as a waterfall plot. It should be noted that the example frequency response measurement also includes the frequency response of the loudspeakers used in the measurement, and as such is not fully descriptive of the modal behavior of the room, especially in the low frequencies where loudspeaker performance was the limiting factor.



(a) Linear scale. **(b)** Log squared.

Figure 2: Example impulse response.

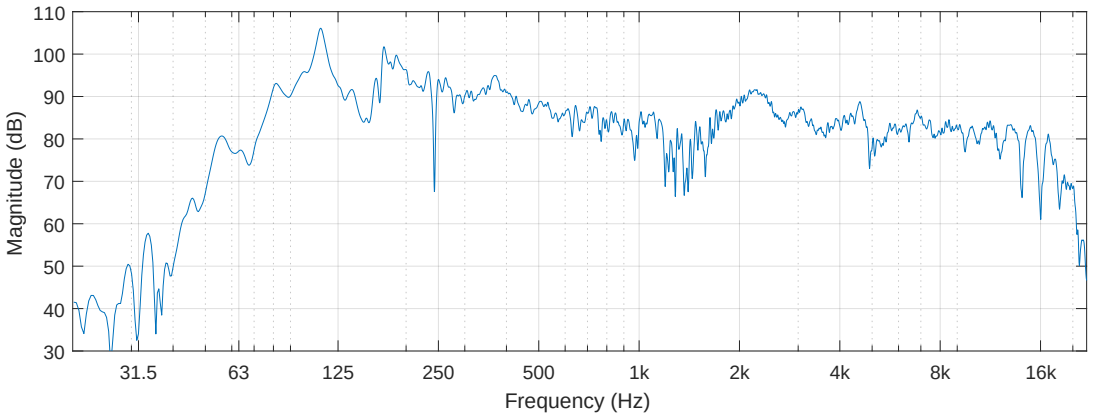


Figure 3: Example of a room frequency response, $1/96$ octave band smoothing.

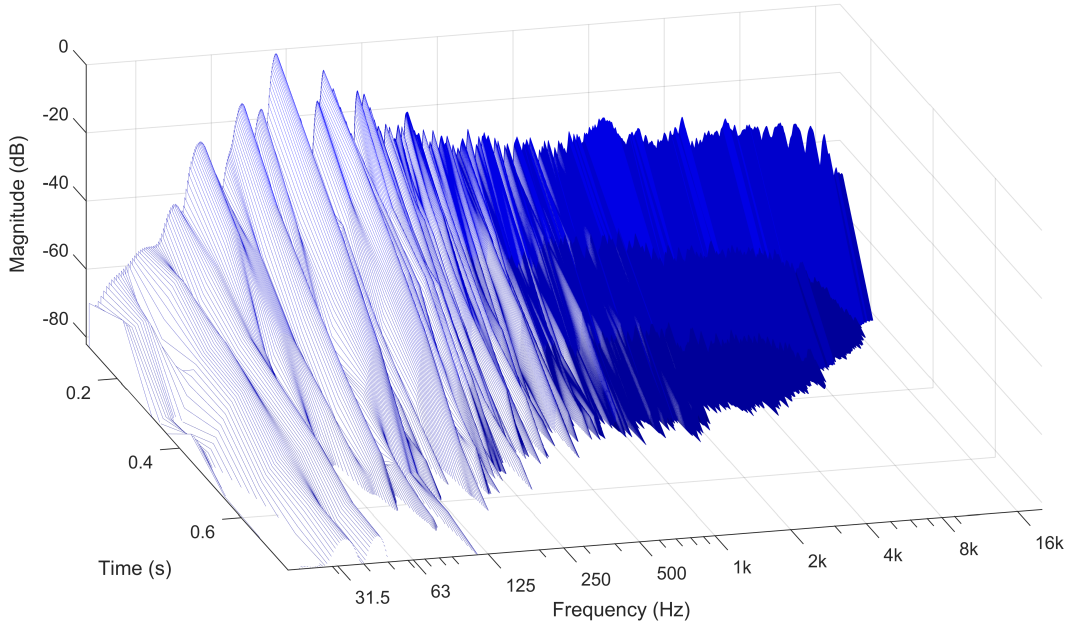
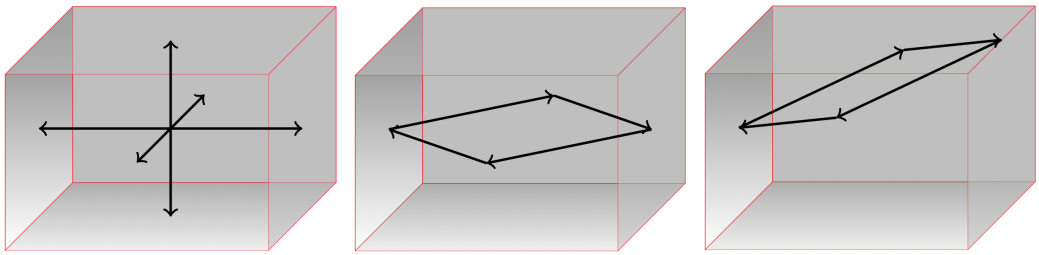


Figure 4: Waterfall plot of the example room frequency response.

Modes in rectangular enclosures can be divided in to three different cases, which are presented in figure 5. Axial modes (5a) are the simples, and typically most significant in terms of the room response. Axial modes form between two opposing surfaces, which corresponds to the length, height and width axis in a rectangular room. Tangential modes include four surfaces and oblique modes use all six surfaces. The lowest axial mode forms betweeen the longest axial dimension on a frequency, for which the wavelength is double the room dimension. The following modes for the same axial dimension form at the integer multiples of that wavelength. [2, p. 565-567] [6, p. 197-206] The mode frequencies in an



(a) Axial mode **(b)** Tangential mode **(c)** Oblique mode

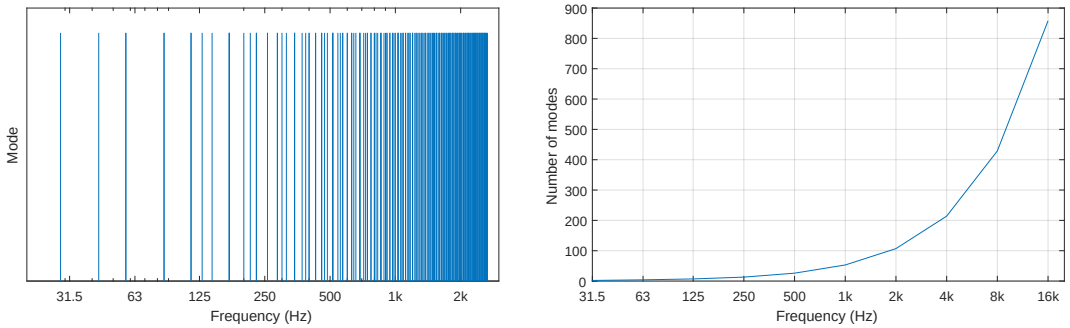
Figure 5: The different types of room modes in rectangular spaces, image source: *Pakarinen, J, S-89.3310 lecture material, 2011.*

ideally reflecting rectangular enclosure with dimensions L, W and H can be calculated with the equation

$$f_{lmn} = \frac{c}{2}\sqrt{\left(\frac{l}{L}\right)^2 + \left(\frac{m}{W}\right)^2 + \left(\frac{n}{H}\right)^2}, \tag{1}$$

where c is the speed of sound and l, m and n are integers $0, 1, 2, \ldots$ [2, p. 566] [6, p. 203]

Using eq. 1, the modes were calculated for a rectangular space with dimensions of six, four and three meters in length, width and height respectively, which corresponds approximately to the size of the example room in figures 2, 3 and 4. The first 200 axial modes starting from the lowest mode frequency, and the number of axial modes in each octave band from 31.5 Hz to 16 kHz are presented in figure 6. Both figures show an exponential increase in the number of modes in relation to increasing frequency.



(a) The first 200 axial modes. **(b)** Number of modes in octave bands.

Figure 6: Axial mode distribution.

For mid and high frequencies, the statistical modal density N_{mode} (modes/Hz) in a room can be calculated by

$$N_{mode}(f) = \frac{4\pi V f^2}{c^3}, \tag{2}$$

where V is the volume of the room and c is the speed of sound [3, 7]. Evaluating the equation shows, together with figure 6, that the total number of modes grows quickly to a large number when high frequencies are included. However, Karjalainen et al. noted that for medium-to-large spaces the modal density exceeds the spectral and temporal resolution of the human auditory system. Therefore, in many cases it is not necessary to model every single mode for perceptually good late reverberation, which can lead to considerable savings in computational complexity. [3]

Karjalainen et al. analyzed the perception of reverberation using auditory models, followed by listening experiments. While it had been previously known that a too low mode density in artificial reverberation produces a "metallic" sound, Karjalainen et al. found this to be the result of low frequency modulation components, which are perceived as metallic roughness and quasi-periodicity in the signal, of which the latter can lead to tonal qualities in the response. Increasing the modal density makes the periodic fluctuation and metallic sound less pronounced. The perceived periodicity can also be reduced with randomness in the mode frequencies. For a high enough modal density the auditory perception of reverberation becomes spectrally dense and temporally diffuse, which a further increase in the modal density doesn't improve noticeably. In the listening experiments, 1500 modes were statistically considered to be as good as the reference of 5000 modes for all reverberation times used in the study. [3]

3 Modal Decomposition

Modal decomposition assumes a linear and time-invariant (LTI) system [7, 8]. For a linear system with a transfer function $h(t)$, the system output $y(n)$ with input $x(t)$ will be the convolution of the input and transfer function

$$y(t) = h(t) * x(t). \tag{3}$$

The transfer function for acoustic spaces and vibrating objects can be expressed as a linear combination of modes

$$h(t) = \sum_{m=1}^{M} h_m(t), \tag{4}$$

where M is the number of modes and h_m being the mth mode response. Therefore, the system output can be expressed as the sum of the mode outputs

$$y(t) = \sum_{m=1}^{M} y_m(t) = \sum_{m=1}^{M} h_m(t) * x(t). \tag{5}$$

For digital systems, the continues time variable t is changed to the discrete time sample index n

$$y(n) = \sum_{m=1}^{M} h_m(n) * x(n). \tag{6}$$

A mode response $h_m(t)$ is described by a mode frequency ω_m, mode damping α_m, and complex amplitude γ_m

$$h_m(t) = \gamma_m \, \mathrm{e}^{(j\omega_m - \alpha_m)t}, \quad \omega_m = 2\pi f_m / f_s. \tag{7}$$

The amplitude is determined by the positions of the sound source and listeners, as the standing waves of the modes are excited differently from different locations, and have different amplitudes in different positions [6, p. 197-206]. The mode frequency and damping are physical properties of the acoustic system, and describe the resonance frequency and decay time of the mode. [5]

3.1 Room Response Modeling

The general form of a discrete-time LTI system transfer function in the z-domain is

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 + a_P z^{-P} + \cdots + b_N z^{-N}} = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 + \sum_{k=1}^{P} a_k z^{-k}}. \tag{8}$$

The most straigthforward way to implement a measured impulse response $h(n)$ of length N is to use the values of the impulse response directly as the coefficients b_k in eq. 8, thereby forming a finite impulse response (FIR) filter

$$H(z) = \sum_{k=0}^{N} b_k z^{-k} = \sum_{k=0}^{N} h(k) z^{-k}. \tag{9}$$

However, for complicated systems the length of the resulting FIR filter can be unpractical and computationally too complex especially for real-time applications. While FIR modeling can be applied efficiently with frequency-domain convolution, it is not applicable for producing the wanted modal filter structure. [8, 7]

Infinite impulse response (IIR) filters are well suited for modeling modal decay behavior. IIR filters can be divided into two forms: all-pole models, where the numerator in eq. 8 is reduced to a constant gain value b_0, and pole-zero models, where both the numerator and denominator in eq. 8 are non-trivial polynomials of z. The challenge has been in dealing with the high filter orders needed for accurate modeling. Autoregressive (AR) and autoregressive moving-average (ARMA) modeling techniques, which model the target response as a whole by minimizing a chosen error criterion, typically a least squares fit, can be used for obtaining efficient IIR filters for modeling resonant and reverberant systems. In the AR method, linear prediction is used to model the target response, resulting in a minimum-phase all-pole filter. While high filter orders can provide a good perceptual match, it cannot model the temporal details (decay) accurately because of the minimum-phase property. ARMA estimation is an iterative method requiring nonlinear optimization, producing a pole-zero filter with more modeling capability than the AR method. Combining ARMA modeling with frequency-zooming, where the modeled target response is partitioned into subbands, enables high-resolution modeling of complex modal and reverberant behavior. [7, 8]

4 Modal Filter Reverberator

In this section the architecture, design methods, and an example practical implementation of the modal filter reverberation algorithm is presented.

4.1 Architecture

Each individual mode is modeled with a resonant filter by matching the filter resonance frequency to the modal frequency and the filter damping to the modal decay. Therefore, each filter produces one specific mode response in the modal reverberation architecture, and the total response is directly the sum of every filter output. The paraller filter structure is presented in figure 7. The paraller structure and independent operation of the filters

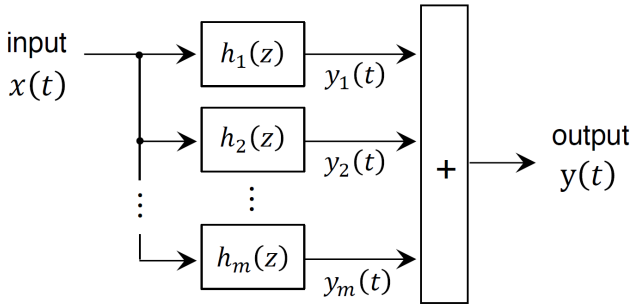


Figure 7: Paraller filter structure, figure from [5].

allows for separate control and adjustment of each mode. The filters can be implemented using any number of different methods, provided they are numerically stable. [5]

Abel et al. [5] proposed using phasor filters, where each mode is represented by a complex first-order update

$$y_m(n) = \gamma_m x(n) + \mathrm{e}^{(j\omega_m - \alpha_m)}\, y_m(n-1), \tag{10}$$

where $x(n)$ is the discrete-time input signal. Other possible filter designs include second-order IIR filters [3, 9, 10], higher-order all-pole and pole-zero (IIR) filters [8], and Kautz filters, which are a special class of fixed-pole IIR filters [7], all of which can be used for modeling real acoustic systems.

While delay network algorithms can be more efficient to implement than the modal reverberator, since they generate a multitude of modes with each feedback loop, it is difficult to accurately design arbitary target responses with them, whereas in the modal reverberator architecture it is trivial. One of the special features of the modal architecture is that the length of the reverberation time does not affect the computational complexity. As only 1000 to 2000 modes are needed to sufficiently model most spaces and objects, the memory required can be a fraction of what is typically needed for producing high quality artificial reverberation. [3, 5, 7].

4.2 Design

The desired reverberation is designed by specifying the number of modes and their frequencies, dampings, and amplitudes. Abel et al. [5] presented three different approaches for choosing these parameters:

- **Behavioral:** fitting the parameters to measurements of real spaces and objects.
- **Analytical:** derive the parameters analytically from system physics.
- **Perceptual:** set parameters for a desired frequency response and reverberation time.

In the behavioral fitting design method, the mode parameters are obtained from an impulse response of the space or object being simulated. The mode frequencies can be estimated from the magnitude spectrum, where peaks occur approximately at the mode frequencies. The desired amount of perceptually important modes can be selected by choosing only the spectral peaks which exceed the critical-band smoothed version of the magnitude spectrum by a certain ratio. The mth mode damping can be set using the reverberation time $T_{30}(\omega_m)$ for that mode frequency

$$\alpha_m = \frac{ln(1000)}{T_{30}(\omega_m) f_s}, \tag{11}$$

where f_s is the sampling rate. The mode amplitudes can be set by a least squares fit to the impulse response being modeled. Denoting the column of impulse response samples by $\boldsymbol{h}$, the complex mode amplitude column $\boldsymbol{\gamma}$ can be obtained for mode count M by

$$\boldsymbol{\gamma} = (\boldsymbol{G}^T \boldsymbol{W} \boldsymbol{G})^{-1} \boldsymbol{G}^T \boldsymbol{W} \boldsymbol{h}, \tag{12}$$

where $\boldsymbol{W}$ is a positive-definite weighting matrix, $T+1$ the length of the impulse response and $\boldsymbol{G}$ a M-column matrix of complex mode responses:

$$\boldsymbol{G} = \begin{bmatrix} 1 & \dots & 1 \\ \mathrm{e}^{(j\omega_1 - \alpha_1)} & \dots & \mathrm{e}^{(j\omega_M - \alpha_M)} \\ \vdots & & \vdots \\ \mathrm{e}^{(j\omega_1 - \alpha_1)T} & \dots & \mathrm{e}^{(j\omega_M - \alpha_M)T} \end{bmatrix}. \tag{13}$$

Using a sufficient mode count, the modeled response can be closely matched to the original response. [5]

In the perceptual design method, a desired set of M mode frequencies and the reverberation time profile (reverberation time as a function of frequency) are generated first. The mode frequencies can be randomly generated according to a desired modal distribution over a chosen frequency range. The mode dampings can then be set directly using eq. 11. The magnitude of each complex mode gain γ_m is set according to a chosen initial equalization $Q(\omega_m)$ scaled with the local modal density $p(\omega_m)$ of modes per Hz

$$|\gamma_m| = \frac{Q(\omega_m)}{\sqrt{p(\omega_m)}}. \tag{14}$$

The complex mode gain can then be set with desired mode phase ϕ_m

$$\gamma_m = |\gamma_m| \, \mathrm{e}^{j\phi_m} \, . \tag{15}$$

With random mode phases, the impulse response is immediately dense and maintains an exponential curve. Nonrandom phases generate a distinctive initial echo pattern, which can be desirable in some cases, especially when modeling a real space. [3, 5]

Based on informal listening tests conducted by Abel et al., the choice of a mode distribution (including uniform, linear, quadratic and exponential) showed only a small perceptual difference when a sufficient modal density was used. It is also suggested that adding some random variation into the mode gains and dampings can lead to a more natural response, as in real acoustic spaces nearby modes can have noticeably different amplitudes and decay times. [5]

4.3 Implementation

The modal filter reverberation algorithm was implemented in MATLAB as it was presented by Abel et al. in [5]. The perceptual design method was used to examine the effect of different numbers of modes used for synthesizing the reverberation. The numbers of modes chosen are presented in table 1. The modes were distributed to eight octave bands with center frequencies from 63 Hz to 8000 Hz, with each octave band containing twice the amount of modes as the previous band, and scaled so that the sum of all the modes on the octave bands matched the number of modes used. The mode frequencies inside each band were generated randomly using an uniform distribution. The octave band limits are presented in table 2, and the exponential increase in modes per octave is illustrated in figure 8.

Table 1: Numbers of modes used in simulation.

125	250	500	1000	2000	4000

Table 2: Octave band limits and center frequencies used.

Lower Limit (Hz)	Center (Hz)	Upper Limit (Hz)
44	63	88
88	125	176
176	250	353
353	500	707
707	1000	1414
1414	2000	2828
2828	4000	5656
5656	8000	11313

The phase of each complex mode gain was randomized between the interval $[0, 2\pi]$. The mode gain magnitudes and the reverberation time as a function of frequency follow a decreasing curve with some random variation in the values. The resulting reverberation

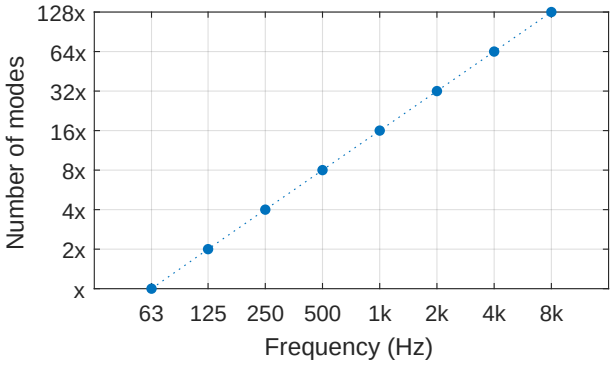


Figure 8: Number of modes per octave band, log-log scale.

time for each case was approximately the same 2,5 seconds, which can be seen in the results. The results containing the modal spacing, mode gain, impulse response, reverberation time profile and spectrogram for each mode count are presented in figures 10, 11, 12, 13, 14 and 15 respectively. On the lowest mode counts, distinctive gaps are seen in the frequency response, whereas higher mode counts produce a smooth frequency distribution. The difference in the spectrograms after 1000 modes is small, suggesting that 1000 modes would be perceptually sufficient in this case.

Synthesizing an extremely long reverberation was also simulated. As previously mentioned, the reverberation time does not affect the computational complexity of the algorithm. The mode count was set to 3000 for highly diffuse and noise-like reverberation, and the other parameters and results are presented in figure 16. The total reverberation time is seen to be approximately 20 seconds.

4.4 Extensions

Interactive control of the reverberation enables smooth morphing between different spaces. For two or more systems having the same number of modes, the mode frequencies of each space can be paired so that every mth mode corresponds to the mth smallest mode frequency in each system. Morphing is then achieved by crossfading separately between the mode parameters (gain, phase, damping and frequency) of each corresponding mode pair or group. Changing the room size can be implemented by increasing or decreasing every mode frequency by a set factor, and adjusting the dampings accordingly. [5]

The mode response convolution in eq. 5 can be rearranged and implemented as a cascade of heterodyning, smoothing and modulation, presented in figure 9, by

$$y_m(t) = \mathrm{e}^{j\omega_m t} \left(\gamma_m \, \mathrm{e}^{-\alpha_m t} * \left[\mathrm{e}^{-j\omega_m t} \, x(t) \right] \right). \tag{16}$$

Using this mode filter implementation, it is possible to create gating and envelope processing, time stretching, pitch manipulation and distortion effects. The structure allows the incorporation of reverberation in to the effects in a novel and unique way, and the effect processing can be applied per mode or mode group for accurate frequency control. [4, 11]

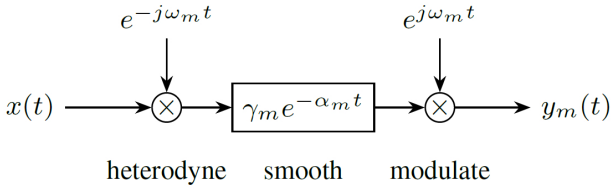
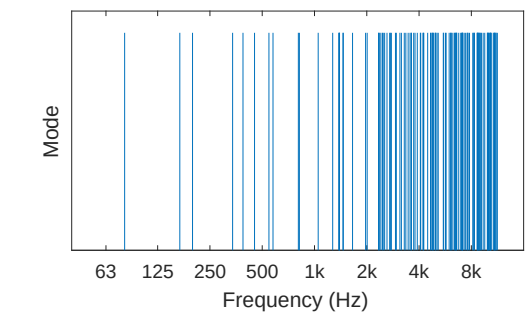


Figure 9: The mode filter implemented as a cascade of heterodyning, smoothing and modulation operations, figure from [11].

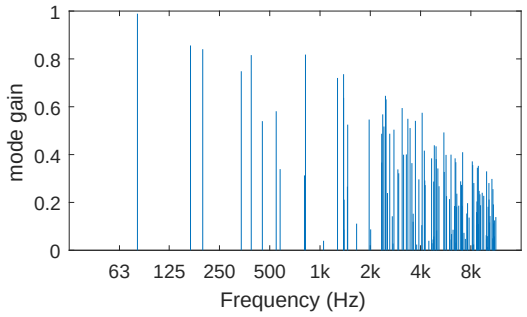
5 Conclusion

In this paper, the modal filter reverberation algorithm was covered, where modal decomposition of an acoustic space or object is used for synthesizing artificial reverberation. The modal analysis of acoustic systems was overviewed, in which the resonance frequencies inherent in acoustic spaces are examined, and more specifically, a room response analyzed from the viewpoint of room modes. Resonant modes cause narrow-band peaks and attenuation in the frequency response, ringing in the time domain and variation in both of the previous depending on location. On low frequencies, separate modes can cause large variations in the sound pressure, while at high frequencies modes overlap creating complex smaller variations in the response. Assuming a linear and time-invariant system, an acoustic system can be decomposed to a sum of separate mode responses, where each mode can be modeled with a resonant filter. Modes increase in number as the frequency increases, with the number of modes in rectangular spaces rising exponentially. The total number of modes can therefore be large for medium-to-large spaces. However, for a perceptually good match only 1000 to 2000 modeled modes are typically sufficient, leading to a low memory cost, which can be a fraction of what is typically needed for high quality artificial reverberation.

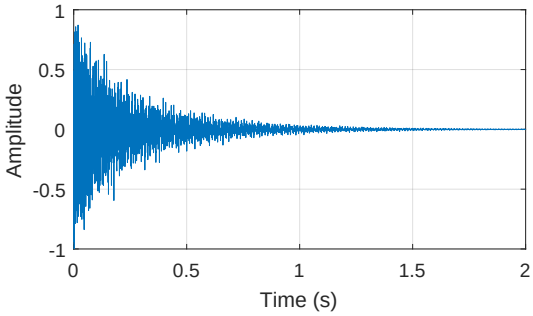
Different filter implementations exists for realizing the modal decomposition. The modal decomposition can be used for synthesizing high quality artificial reverberation, either by matching the mode filter parameters the measurements of real spaces, by deriving the parameters analytically from system physics, or by designing the parameters for a desired perceptual behavior. The reverberation algorithm allows interactive and separate control of each mode, and can also be extended to novel implementations of gating, time stretching, pitch manipulation and distortion processing. A practical implementation of the presented modal reverberation algorithm was done in MATLAB to examine and illustrate the working of the algorithm.



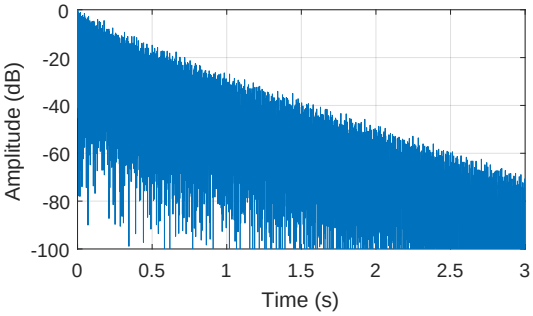
(a) Modal spacing.



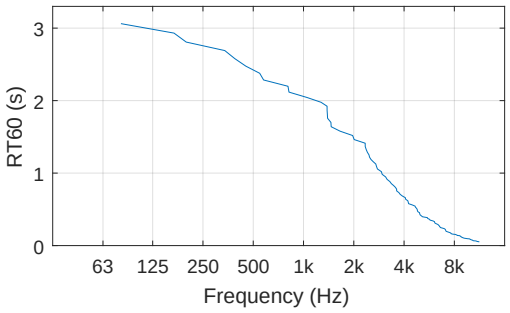
(b) Mode gain.



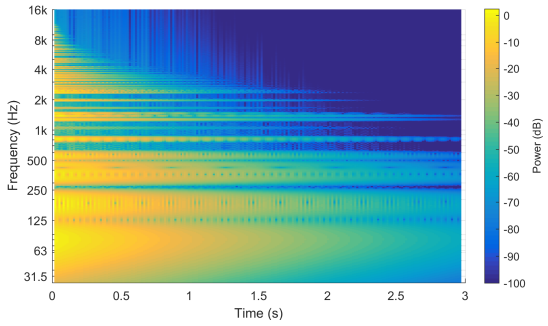
(c) Impulse response.



(d) Impulse, log squared.

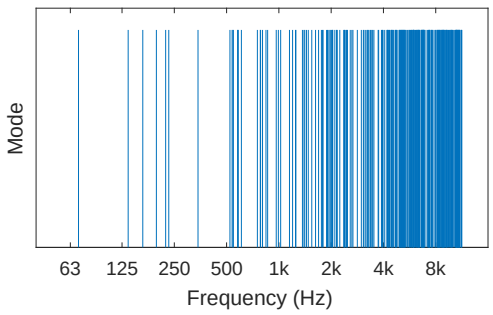


(e) Reverberation time.

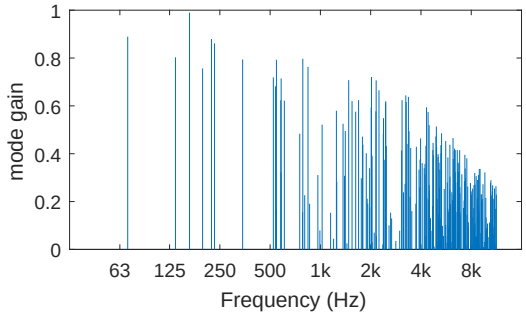


(f) Spectrogram.

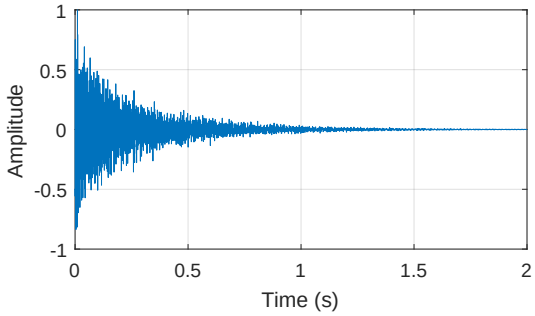
Figure 10: 125 modes.



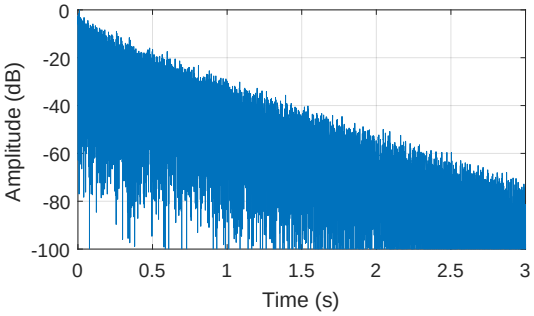
(a) Modal spacing.



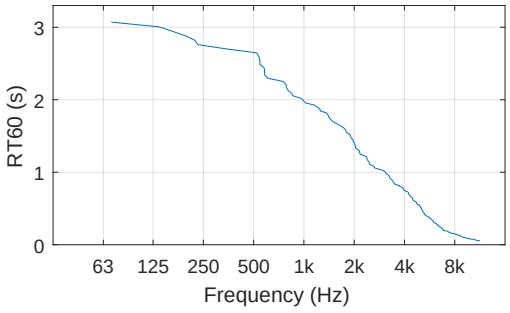
(b) Mode gain.



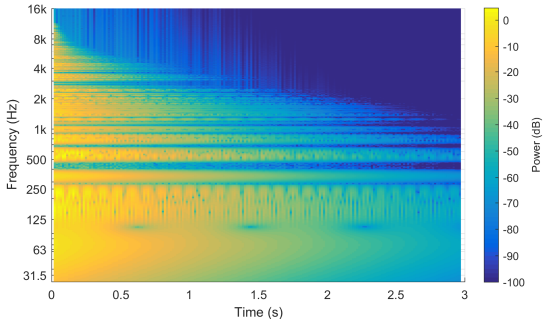
(c) Impulse response.



(d) Impulse, log squared.

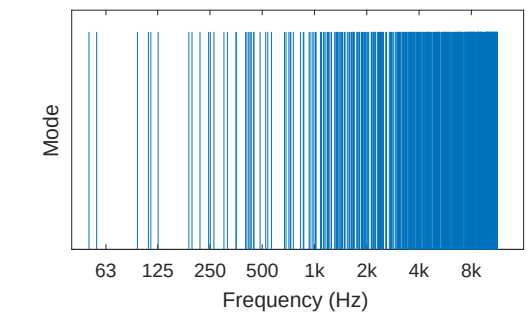


(e) Reverberation time.

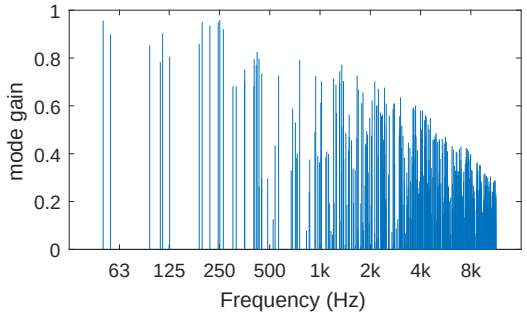


(f) Spectrogram.

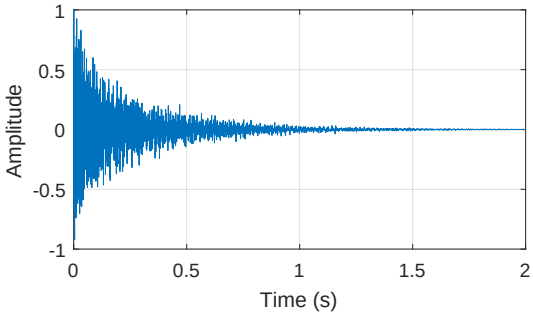
Figure 11: 250 modes.



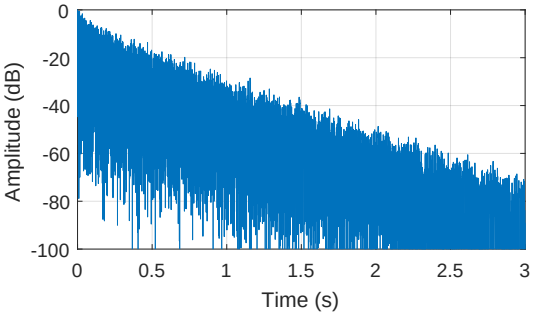
(a) Modal spacing.



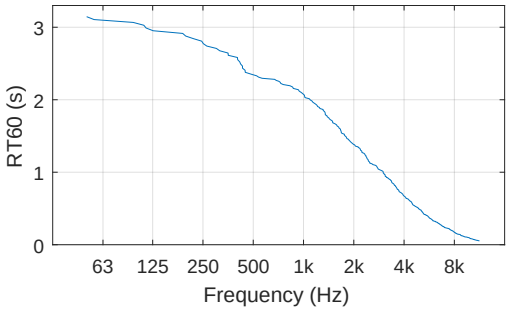
(b) Mode gain.



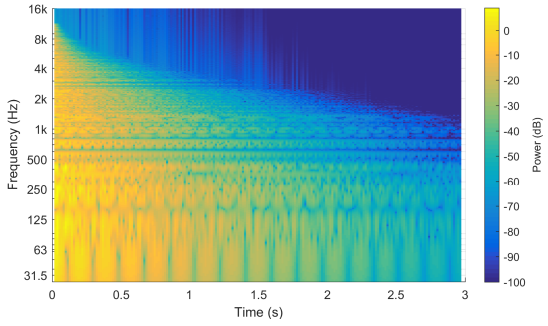
(c) Impulse response.



(d) Impulse, log squared.

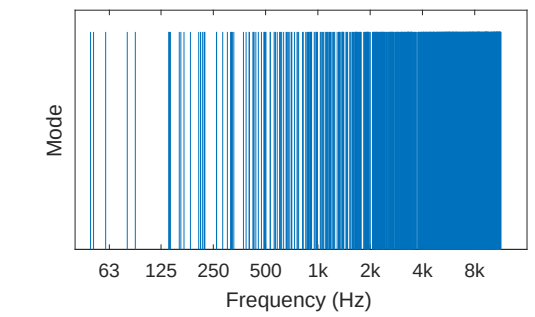


(e) Reverberation time.

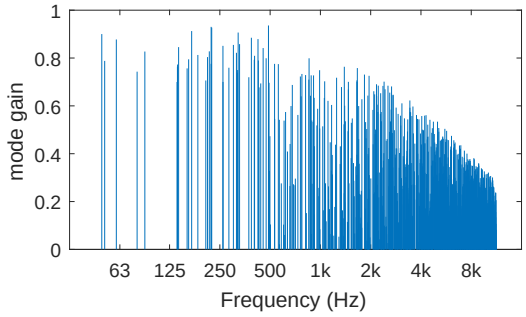


(f) Spectrogram.

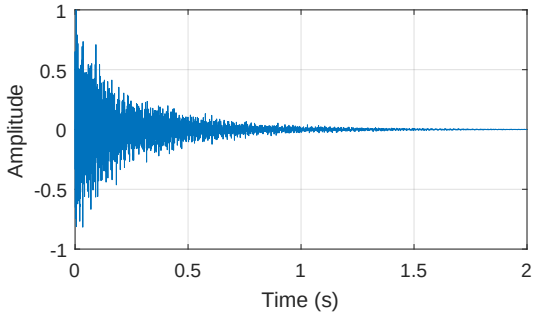
Figure 12: 500 modes.



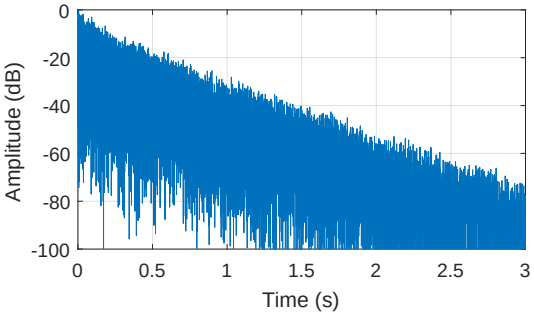
(a) Modal spacing.



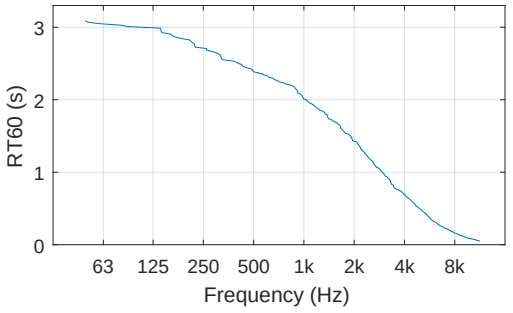
(b) Mode gain.



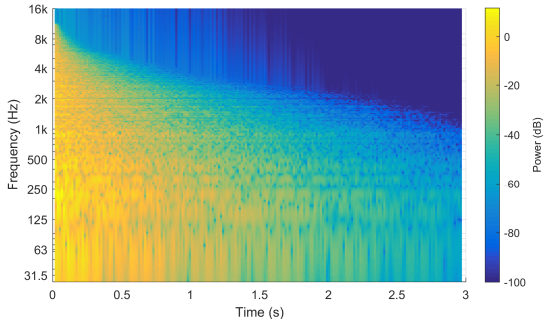
(c) Impulse response.



(d) Impulse, log squared.

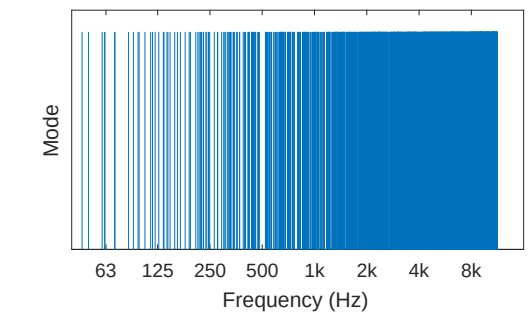


(e) Reverberation time.

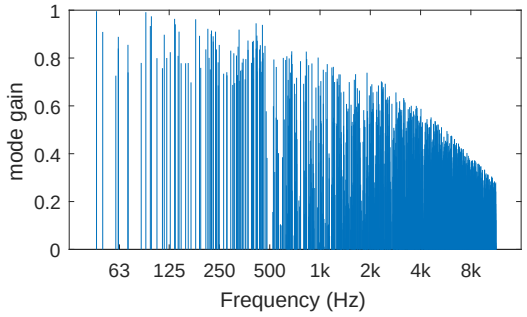


(f) Spectrogram.

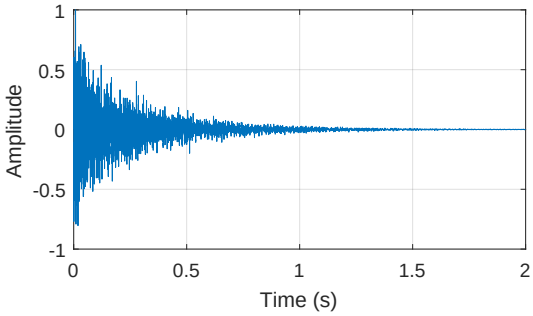
Figure 13: 1000 modes.



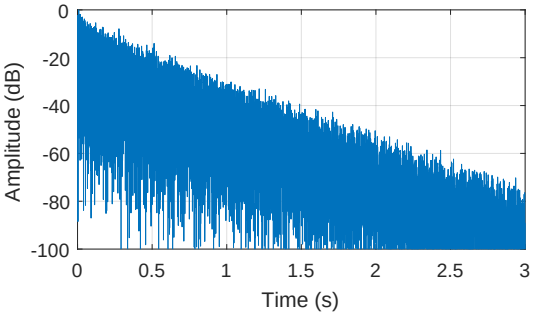
(a) Modal spacing.



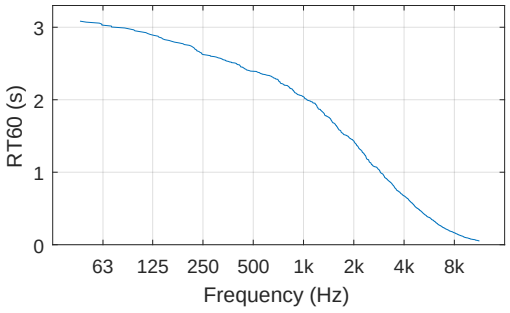
(b) Mode gain.



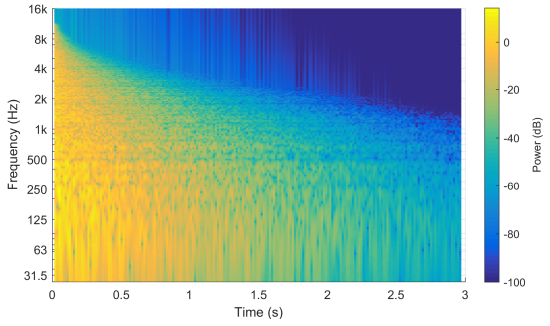
(c) Impulse response.



(d) Impulse, log squared.

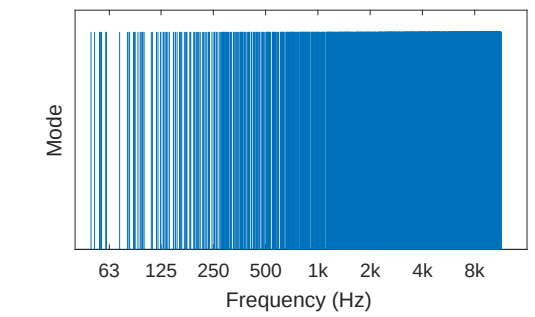


(e) Reverberation time.

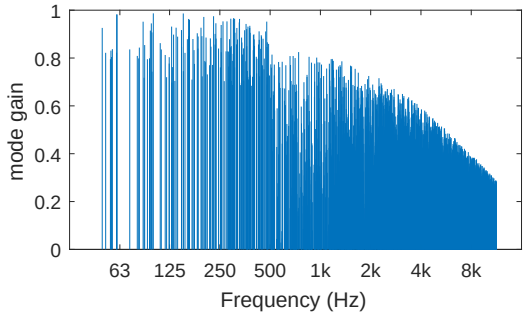


(f) Spectrogram.

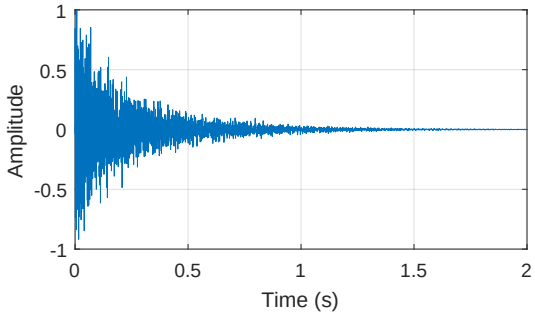
Figure 14: 2000 modes.



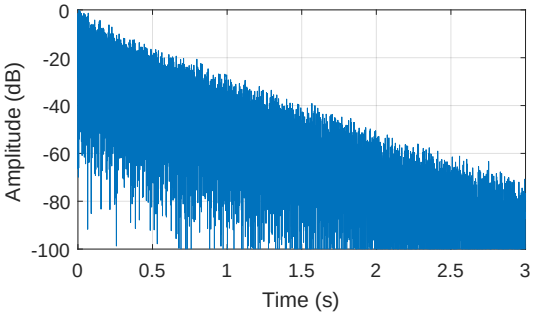
(a) Modal spacing.



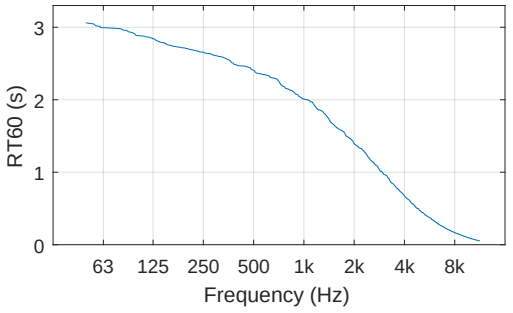
(b) Mode gain.



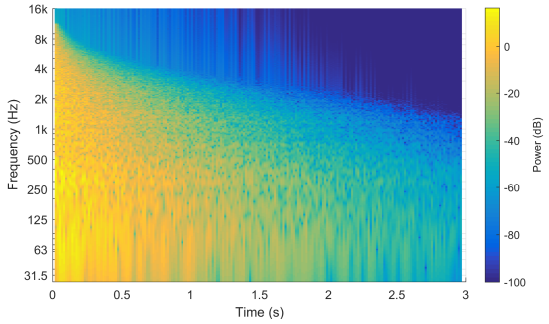
(c) Impulse response.



(d) Impulse, log squared.

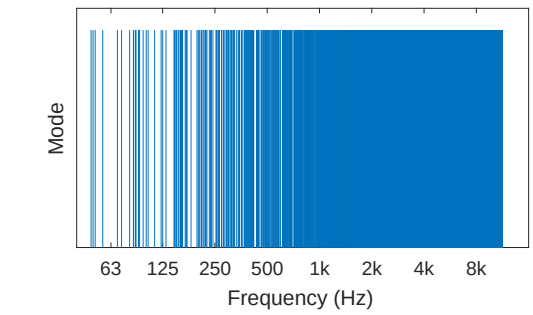


(e) Reverberation time.

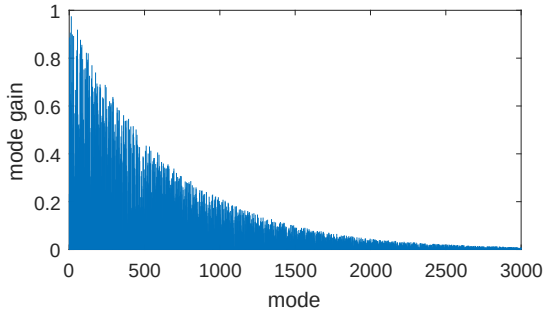


(f) Spectrogram.

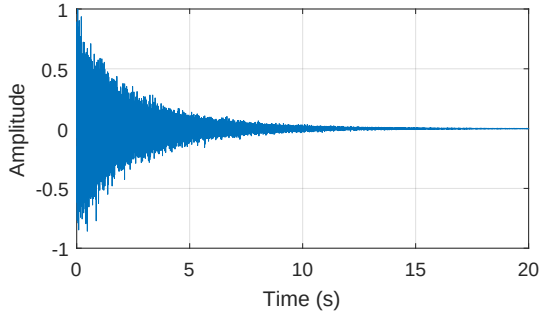
Figure 15: 4000 modes.



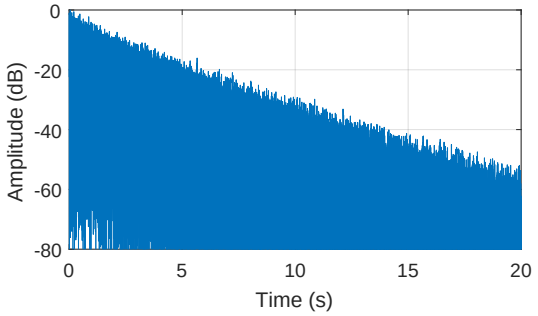
(a) Modal spacing.



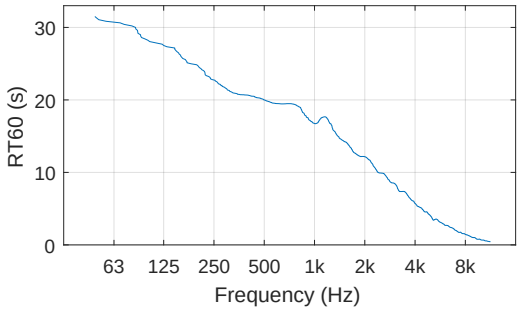
(b) Mode gain.



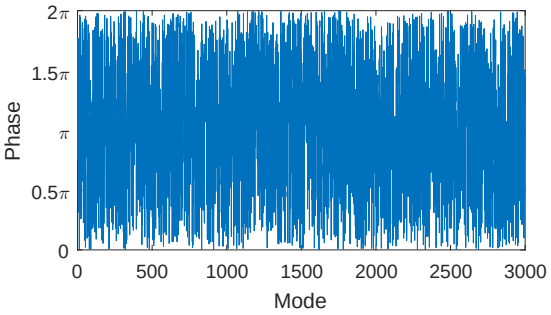
(c) Impulse response.



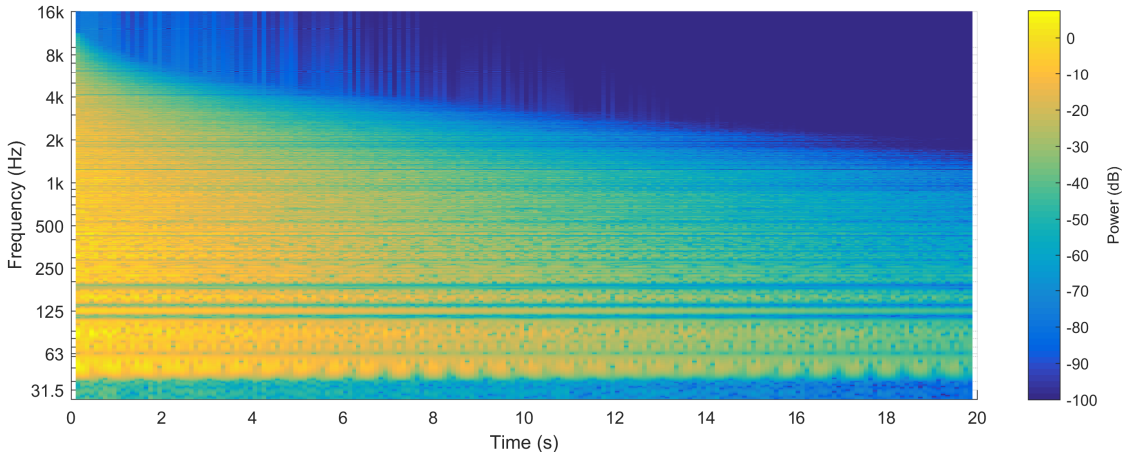
(d) Impulse, log squared.



(e) Reverberation time.



(f) Mode gain phase.



(g) Spectrogram.

Figure 16: Long reverberation time simulation.

References

[1] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[2] T. Rossing, P. Wheeler, and F. Moore, *The Science of Sound*. Addison Wesley, 2002.

[3] M. Karjalainen and H. Järveläinen, "More about this reverberation science: Perceptually good late reverberation," in *Proc. Audio Engineering Society Convention 111*, Audio Engineering Society, 2001.

[4] V. Välimäki, J. Parker, L. Savioja, J. O. Smith, and J. Abel, "More than 50 years of artificial reverberation," in *Proc. Audio Engineering Society Conference: 60th International Conference: DREAMS (Dereverberation and Reverberation of Audio, Music, and Speech)*, Jan 2016.

[5] J. S. Abel, S. Coffin, and K. Spratt, "A modal architecture for artificial reverberation with application to room acoustics modeling," in *Proc. Audio Engineering Society Convention 137, Los Angeles, CA, USA*, Audio Engineering Society, Oct 2014.

[6] F. Toole, *Sound Reproduction - Rooms and Loudspeakers*. Focal Press, 1st ed., 2008.

[7] T. Paatero and M. Karjalainen, "New digital filter techniques for room response modeling," in *Audio Engineering Society Conference: 21st International Conference: Architectural Acoustics and Sound Reinforcement*, Audio Engineering Society, 2002.

[8] M. Karjalainen, P. A. A. Esquef, P. Antsalo, A. Mäkivirta, and V. Välimäki, "AR/ARMA analysis and modeling of modes in resonant and reverberant systems," in *Audio Engineering Society Convention 112*, Audio Engineering Society, 2002.

[9] B. Bank, "Direct design of parallel second-order filters for instrument body modeling," in *Proc. International Computer Music Conference (ICMC 2007), Copenhagen, Denmark*, pp. 458–465, 2007.

[10] S. Zambon, H.-M. Lehtonen, and B. Bank, "Simulation of piano sustain-pedal effect by parallel second-order filters," in *DAFx-08 Int. Conference, Espoo*, 2008.

[11] J. S. Abel and K. J. Werner, "Distortion and pitch processing using a modal reverberator architecture," in *Proc. 18th Int. Conference on Digital Audio Effects (DAFx-15), Trondheim, Norway*, Nov 2015.

A MATLAB Code

roomresponse.m

```
% ELEC-E5630 - Acoustics and Audio Technology Seminar
% Juri Lukkarila, 2016

%% Frequency response

% Import measurement data from text file:
filename = 'D:\Dropbox\KOULU\AKU SEMINAR\Latex\figures\PC_stereo_uus.csv';
delimiter = ',';
formatSpec = '%f%f%[^\n\r]';
fileID = fopen(filename,'r');
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,...
    'MultipleDelimsAsOne', true);
fclose(fileID);
freq = dataArray{:, 1};
mag = dataArray{:, 2};

% Clear temporary variables
clearvars filename delimiter formatSpec fileID dataArray ans;

% apply 1/96 octave smoothing
mag = smooth_spectrum(mag,freq,96);

%% Plot frequency response
figure()
semilogx(freq, mag); grid on;
xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
axis([20 22000 30 110])
set(gca,'XTick',[31.5 63 125 250 500 1000 2000 4000 8000 16000])
set(gca,'XTickLabel',{31.5 63 125 250 500 '1k' '2k' '4k' '8k' '16k'})
set(gcf,'PaperUnits','centimeters','PaperPosition',[-1.6 0 24 8],...
        'PaperSize',[20.6 7.8])
print(gcf, 'freqresponse', '-dpdf', '-painters');

%% Impulse response

% read measured impulse response
[imp, Fs] = audioread('PC_stereo_uus.wav',[47200,inf]);
N = length(imp);                % samples
Ts = 1/Fs;                      % sample time
t = 0:Ts:(N-1)*Ts;              % time vector

figure();
plot(t, imp); grid on;
xlabel('Time (s)');
ylabel('Amplitude');
axis([0 0.2 -1 1]);
set(gca,'XTick',0:0.025:0.2);
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf, 'impulse', '-dpdf', '-painters');

%% Log squared

imp_db = 10*log10(imp.^2);

figure();
plot(t, imp_db); grid on;
xlabel('Time (s)');
ylabel('Magnitude (dB)')
axis([0 0.4 -80 0]);
set(gca,'XTick',0:0.05:0.4);
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf, 'impulse_db', '-dpdf', '-painters');

%% Waterfall plot

% spectrogram parameters
window = floor(N/8);
overlap = window/2;
nfft = N;

[s,f,t] = spectrogram(imp,window,overlap,nfft,Fs);

% real values
S = abs(s);

% matrix size
dims = size(s);

% initiliaze matrix for loop
S_norm = zeros(dims(1), dims(2));
S_smooth = zeros(dims(1), dims(2));
```

```
% normalize
s_max = max(max(S));          % max value
for n = 1:dims(2)
    S_norm(:,n) = S(:,n)./s_max;
end

% log scale
S_db = 20*log10(S_norm);

%% 1/48 octave smoothing
for n = 1:dims(2)
    S_smooth(:,n) = smooth_spectrum(S_db(:,n),f,48);
end

%% plot

% custom colormap
map = [0 0 0.3 ; 0 0 0.4 ; 0 0 0.5 ; 0 0 0.6 ;0 0 0.7 ; 0 0 0.8 ;0 0 0.9 ; 0 0 1];

figure()
h = waterfall(t,f,S_smooth); colormap(map);
set(h, 'LineWidth',0.05, 'LineSmoothing','on'); grid on;
view([75 40]); axis([0.1 0.75 20 20000 -82 0]);
xlabel('Time (s)'); ylabel('Frequency (Hz)'); zlabel('Magnitude (dB)')
set(gca,'YTick',[31.5 63 125 250 500 1000 2000 4000 8000 16000])
set(gca,'YTickLabel',{31.5 63 125 250 500 '1k' '2k' '4k' '8k' '16k'})
set(gca,'YMinorTick','off','MinorGridLineStyle','none');
set(gca,'YScale','log'); set(gco, 'Linesmoothing','on');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 24 13.5],...
        'PaperSize',[24 13.5])
print(gcf, 'waterfall', '-dpng', '-r600');
```

axialmodes.m

```
% ELEC-E5630 - Acoustics and Audio Technology Seminar
% Juri Lukkarila, 2016

%% Axial Room Modes

L = 4;          % length
W = 6;          % width
H = 3;          % height

v = 343;        % speed of sound

axial_L = zeros(1,1000);
axial_W = zeros(1,1000);
axial_H = zeros(1,1000);

% calculate modes
for a = 1:length(axial_L)
    axial_L(a) = v/2*sqrt((a/L)^2);
    axial_W(a) = v/2*sqrt((a/W)^2);
    axial_H(a) = v/2*sqrt((a/H)^2);
end

% combine axial modes
axial = sort(round(horzcat(axial_L,axial_W,axial_H),1));

% count unique
uv = unique(axial);
n  = histc(axial,uv);

%% modes
figure(); stem(uv,n, 'Marker', 'none'); set(gca,'XScale','log');
set(gca,'XTick',[31.5 63 125 250 500 1000 2000 4000 8000 16000])
set(gca,'XTickLabel',{31.5 63 125 250 500 '1k' '2k' '4k' '8k' '16k'})
axis([20 24000 0 3.5]);
xlabel('Frequency (Hz)')
ylabel('Modes')

%% mode distribution
figure(); stem(axial(1:200), ones(1,length(axial(1:200))),'Marker', 'none'); set(gca,'XScale','log');
set(gca,'XTick',[31.5 63 125 250 500 1000 2000 4000 8000 16000])
set(gca,'XTickLabel',{31.5 63 125 250 500 '1k' '2k' '4k' '8k' '16k'})
axis([20 3000 0 1.1]);
xlabel('Frequency (Hz)')
ylabel('Mode')
set(gca, 'YTick', []);
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf, 'axialmodes', '-dpdf', '-painters');

%% Octave Band

f = [31.5 63 125 250 500 1000 2000 4000 8000 16000];
f1 = floor(f./sqrt(2));
```

```
f2 = floor(sqrt(2).*f);

f_oct = horzcat(f1',f',f2');

% count modes on octave band
modes_per_octave = zeros(1,10);

for i = 1:10
    modes_per_octave(i) = sum(axial(:)>f1(i) & axial(:)<f2(i));
end

figure(); semilogx(f, modes_per_octave);
set(gca,'XTick',[31.5 63 125 250 500 1000 2000 4000 8000 16000])
set(gca,'XTickLabel',{31.5 63 125 250 500 '1k' '2k' '4k' '8k' '16k'})
set(gca, 'XMinorTick', 'off', 'MinorGridLineStyle', 'none');
set(gca, 'YMinorTick', 'off');
%xlim([31.5 20000]);
axis tight; grid on;
xlabel('Frequency (Hz)')
ylabel('Number of modes')
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf, 'modes_per_octave', '-dpdf', '-painters');
```

modalreverb.m

```
%% ELEC-E5630 - Acoustics and Audio Technology Seminar
% Juri Lukkarila, 2016

%% Modal Filter Reverb: number of modes

close all; clearvars; clc

rng(0,'twister');                                   % random seed
fs = 44100;                                         % sample rate
len = 3;                                            % length of input (s)
modes = [125 250 500 1000 2000 4000];               % number of modes

% octave band
modes_per_octave = [1 2 4 8 16 32 64 128];
f_oct = [63 125 250 500 1000 2000 4000 8000];       % middle freqs
f1 = floor(f_oct./sqrt(2));                         % lower freq limit
f2 = floor(sqrt(2).*f_oct);                         % upper freq limit

% plot modes per octave
figure(); loglog(f_oct, modes_per_octave, '.:', 'MarkerSize',14);
axis([31.5 16000 1 128]); grid on;
xlabel('Frequency (Hz)'); ylabel('Number of modes')
set(gca,'XTick',[63 125 250 500 1000 2000 4000 8000])
set(gca,'XTickLabel',{63 125 250 500 '1k' '2k' '4k' '8k'})
set(gca,'YTick',[1 2 4 8 16 32 64 128])
set(gca,'YTickLabel',{'x' '2x' '4x' '8x' '16x' '32x' '64x' '128x'})
set(gca,'XMinorTick', 'off','YMinorTick','off','MinorGridLineStyle','none');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,'MFR_modes_octave', '-dpdf', '-painters');

%% Loop through different number of modes
for l = 1:6

% number of modes
M = modes(l);

% scale number of modes per octave
if l == 1
    modes_per_octave = [1 1 2 4 8 16 32 64];
elseif l == 2
    modes_per_octave = [1 2 4 8 16 32 64 128];
elseif l >= 3
    modes_per_octave = modes_per_octave.*2;
end

fprintf('Modes = %d, dist. %d %d %d %d %d %d %d %d, sum = %d\n',...
    M, modes_per_octave, sum(modes_per_octave))

% random frequencies inside each octave band
f = zeros(1,M);
index = 1;
for i = 1:8
    for k = index:(index + modes_per_octave(i)-1)
        if k > M
            break
        end
        f(k) =  (f2(i)-f1(i)).*rand + f1(i);    % freq. betweeen band limits
    end
    index = k + 1;
end
```

```
% frequency vector
f = sort(round(f,1));
w = 2*pi.*f./fs;

% plot modal spacing
figure(); stem(f, ones(1,length(f)),'Marker','None', 'LineWidth',0.05);
axis([40 16000 0 1.1]);
xlabel('Frequency (Hz)'); ylabel('Mode')
set(gca,'XScale','log','YTick', [])
set(gca,'XTick',[63 125 250 500 1000 2000 4000 8000])
set(gca,'XTickLabel',{63 125 250 500 '1k' '2k' '4k' '8k'})
set(gca,'XMinorTick', 'off','YMinorTick','off','MinorGridLineStyle','none');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,strcat('MFR_modal_spacing_', int2str(M)), '-dpdf', '-painters');

%% RT

r1 = 3;     % reverberation time, lowest freq (s)

r_begin = [3 6 12 14 18 22];
smooth_factor = [0.2 0.2 0.1 0.05 0.04 0.02];

% RT curve with some ramdomness
t = linspace(0,2,M);        % time vector for exponential function.
RT = r1.*(0.1.*rand(1,M) + 0.9).*exp(-2*t);
RT(1:r_begin(1)) = r1;              % force first values to chosen number
RT = smooth(RT,smooth_factor(1),'loess');

% plot
figure(); semilogx(f,RT); grid on;
xlabel('Frequency (Hz)'); ylabel('RT60 (s)');
axis([31.5 16000 0 1.1*r1]);
set(gca,'XTick',[63 125 250 500 1000 2000 4000 8000])
set(gca,'XTickLabel',{63 125 250 500 '1k' '2k' '4k' '8k'})
set(gca,'XMinorTick', 'off','YMinorTick','off','MinorGridLineStyle','none');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,strcat('MFR_RT60_', int2str(M)), '-dpdf', '-painters');

%% mode damping
a = log(1000)./(RT.*fs);

low = round(M/25);
high = M - low;

% mode gain
gp_low = 0.3.*rand(1,low)+0.7;                              % 0.3 - 1
gp_high = 0.89.*rand(1,high)+0.01;                          % 0.01 - 0.9
gp = horzcat(gp_low, gp_high).*nthroot(logspace(0,-1,M),2);

figure(); stem(f,gp,'Marker','None', 'LineWidth',0.05);
ylabel('mode gain'); xlabel('Frequency (Hz)'); axis([31.5 16000 0 1]);
set(gca,'XTick',[63 125 250 500 1000 2000 4000 8000], 'XScale','log')
set(gca,'XTickLabel',{63 125 250 500 '1k' '2k' '4k' '8k'})
set(gca,'XMinorTick', 'off');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,strcat('MFR_gain_', int2str(M)), '-dpdf', '-painters');

% random phases 0...2pi
phase = (2*pi).*rand(1,M);

% complex mode gain
g = gp.*exp(1i.*phase);

%% impulse response
x = zeros(1,len*fs);
x(1) = 1;

N = length(x);              % samples
ts = 1/fs;                  % sample time
xt = 0:ts:(N-1)*ts;         % time vector

ym = zeros(1,M);            % current values
y = zeros(1,N);             % output

% first value, ym(t-1) = 0
for m = 1:M
    ym(m) = g(m).*x(1);
end

y(1) = sum(ym);

% filter loop
for t = 2:N
    ym_prev = ym;       % store previous value
    for m = 1:M
        ym(m) = g(m).*x(t) + exp(1i*w(m)-a(m))*ym_prev(m);
    end
    y(t) = sum(ym);     % store output
end
```

```
%% export audio
miny = min(real(y));
maxy = max(real(y));
if abs(miny) >= maxy
    impulse = real(y)./abs(miny);
else
    impulse = real(y)./abs(maxy);
end
%soundsc(impulse,fs)
audiowrite(strcat('MFR_impulse_', int2str(M),'.wav'), impulse, fs);

%% Impulse Response plot

figure(); plot(xt, impulse); grid on; axis auto; xlim([0 2]);
xlabel('Time (s)'); ylabel('Amplitude');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,strcat('MFR_imp_', int2str(M)), '-dpdf', '-painters');

% log squared
y_abs = impulse.^2;
y_norm = y_abs./max(y_abs);
ylog = 10*log10(y_norm);

figure(); plot(xt, ylog); grid on; ylim([-100 0])
xlabel('Time (s)'); ylabel('Amplitude (dB)');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,strcat('MFR_imp_log_', int2str(M)), '-dpdf', '-painters');

%% spectrogram
window = round(N/100);      % divide to approx. 100 windows
if mod(window, 2) ~= 0      % if odd
    window = window + 1;
end
overlap = window/2;
nfft = N;
figure();
spectrogram(y,window,overlap,nfft,fs,'yaxis','MinThreshold', -100, 'power');
set(gca,'YScale','log'); axis([0 len 0.03 16]); ylabel('Frequency (Hz)');
set(gca,'YTick',[0.035 0.063 0.125 0.25 0.5 1 2 4 8 16]); xlabel('Time (s)');
set(gca,'YTickLabel',{31.5 63 125 250 500 '1k' '2k' '4k' '8k' '16k'})
% change colorbar size and location
c = colorbar; ax = gca; axpos = ax.Position; cpos = c.Position;
cpos(3) = 0.5*cpos(3); c.Position = cpos; ax.Position = axpos;
c.Label.String = 'Power (dB)';
set(gcf,'PaperUnits','centimeters','PaperPosition',[-0.4 0 18 9.9],...
        'PaperSize',[16.4 9.6])
print(gcf,strcat('MFR_spectrogram_', int2str(M)), '-dpdf', '-r300');
%% 3D
% view([100 50]); colorbar('off');
% set(gcf,'PaperUnits','centimeters','PaperPosition',[-1.2 0 21.4 12],...
%         'PaperSize',[19.4 11.6])
% print(gcf,strcat('MFR_spectrogram_3D_', int2str(M)), '-dpdf', '-r300');

%% Vocal sample

[vocal, fs] = audioread('vocal2.wav');

N = length(vocal);

ym = zeros(1,M);
y = zeros(1,N);

% first value, ym(t-1) = 0
for m = 1:M
    ym(m) = g(m).*vocal(1);
end

y(1) = sum(ym);

% filter loop
for t = 2:N
    ym_prev = ym;
    for m = 1:M
        ym(m) = g(m).*vocal(t) + exp(1i*w(m)-a(m))*ym_prev(m);
    end
    y(t) = sum(ym);
end

%% Export audio
miny = min(real(y));
maxy = max(real(y));
if abs(miny) >= maxy
    reverb = real(y)./abs(miny);
else
    reverb = real(y)./abs(maxy);
end
%soundsc(reverb,fs)
audiowrite(strcat('MFR_vocal_reverb_', int2str(M),'.wav'), reverb', fs);
```

```
%% mixed with dry signal
mix = (0.7.*vocal)' + 0.3.*reverb;
%soundsc(real(vocal_mix),fs)
audiowrite(strcat('MFR_vocal_mix_', int2str(M),'.wav'), mix, fs);

%% Drum beat sample

[beat, fs] = audioread('beat.wav');

N = length(beat);

ym = zeros(1,M);
y = zeros(1,N);

% first value, ym(t-1) = 0
for m = 1:M
    ym(m) = g(m).*beat(1);
end

y(1) = sum(ym);

% filter loop
for t = 2:N
    ym_prev = ym;
    for m = 1:M
        ym(m) = g(m).*beat(t) + exp(1i*w(m)-a(m))*ym_prev(m);
    end
    y(t) = sum(ym);
end

%% Export audio
miny = min(real(y));
maxy = max(real(y));
if abs(miny) >= maxy
    reverb = real(y)./abs(miny);
else
    reverb = real(y)./abs(maxy);
end
%soundsc(reverb,fs)
audiowrite(strcat('MFR_beat_reverb_', int2str(M),'.wav'), reverb, fs);

%% mixed with dry signal
mix = (0.7.*beat)' + 0.3.*reverb;
%soundsc(real(vocal_mix),fs)
audiowrite(strcat('MFR_beat_mix_', int2str(M),'.wav'), mix, fs);

end

close all;
```

modalreverblong.m

```
%% ELEC-E5630 - Acoustics and Audio Technology Seminar
% Juri Lukkarila, 2016

%% Modal Filter Reverb: long reverberation time

close all; clearvars; clc

rng(0,'twister');                                   % random seed
fs = 44100;                                         % sample rate
len = 20;                                           % length of input (s)
M = 3000;                                           % number of modes

% octave band
modes_per_octave = [12 24 48 96 192 384 768 1536];
f_oct = [63 125 250 500 1000 2000 4000 8000];       % middle freqs
f1 = floor(f_oct./sqrt(2));                         % lower freq limit
f2 = floor(sqrt(2).*f_oct);                         % upper freq limit

% frequency vector
f = zeros(1,M);
index = 1;
for i = 1:8
    for k = index:(index + modes_per_octave(i)-1)
        if k > M
            break
        end
        f(k) =  (f2(i)-f1(i)).*rand + f1(i);
    end
    index = k + 1;
end

f = sort(round(f,1));
w = 2*pi.*f./fs;

% plot modal spacing
```

```
figure(); stem(f, ones(1,length(f)),'Marker','None','LineWidth',0.05);
axis([40 16000 0 1.1]);
xlabel('Frequency (Hz)'); ylabel('Mode')
set(gca,'XScale','log','YTick', [])
set(gca,'XTick',[63 125 250 500 1000 2000 4000 8000])
set(gca,'XTickLabel',{63 125 250 500 '1k' '2k' '4k' '8k'})
set(gca,'XMinorTick', 'off','YMinorTick','off','MinorGridLineStyle','none');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,'MFR2_modal_spacing','-dpdf','-painters');

%% RT

r1 = 30;                                % reverberation time, lowest freq (s)

% RT curve with some ramdomness
t = linspace(0,2,M);                    % time vector for exponential function.
RT = r1.*(0.8.*rand(1,M) + 0.4).*exp(-2*t);
RT(1:30) = r1;                          % force first values to chosen number
RT = smooth(RT,0.08,'loess');

% plot
figure(); semilogx(f,RT); grid on;
xlabel('Frequency (Hz)'); ylabel('RT60 (s)');
axis([31.5 16000 0 1.1*r1]);
set(gca,'XTick',[63 125 250 500 1000 2000 4000 8000])
set(gca,'XTickLabel',{63 125 250 500 '1k' '2k' '4k' '8k'})
set(gca,'XMinorTick', 'off','YMinorTick','off','MinorGridLineStyle','none');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,'MFR2_RT60', '-dpdf', '-painters');

%% mode damping
a = log(1000)./(RT.*fs);

% mode gain
gp = (rand(1,M)).*logspace(1,-1,M)./10;
figure(); stem(gp, 'Marker','None', 'LineWidth',0.05);
ylabel('mode gain'); xlabel('mode');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,'MFR2_gain','-dpdf', '-painters');

% phases 0...2pi
phase = 2*pi.*rand(1,M);
figure(); plot(phase); xlabel('Mode'); ylabel('Phase'); ylim([0 2*pi]);
set(gca,'YTick',[0 pi/2 pi 3/2*pi 2*pi])
set(gca,'YTickLabel',{0 '0.5\pi' '\pi' '1.5\pi' '2\pi'})
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,'MFR2_phase','-dpdf', '-painters');

% complex mode gain
g = gp.*exp(1i.*phase);

%% impulse response
x = zeros(1,len*fs);
x(1) = 1;

N = length(x);                  % samples
ts = 1/fs;                      % sample time
xt = 0:ts:(N-1)*ts;             % time vector

ym = zeros(1,M);                % current values
y = zeros(1,N);                 % output

% first value, ym(t-1) = 0
for m = 1:M
    ym(m) = g(m).*x(1);
end

y(1) = sum(ym);

% filter loop
for t = 2:N
    ym_prev = ym;       % store previous value
    for m = 1:M
        ym(m) = g(m).*x(t) + exp(1i*w(m)-a(m))*ym_prev(m);
    end
    y(t) = sum(ym);     % store output
end

%% export audio
miny = min(real(y));
maxy = max(real(y));
if abs(miny) >= maxy
    impulse = real(y)./abs(miny);
else
    impulse = real(y)./abs(maxy);
end
%soundsc(impulse,fs)
audiowrite('MFR2_impulse.wav', impulse, fs);
```

```
%% Impulse Response plot

figure(); plot(xt, impulse); grid on; axis auto; xlim([0 len]);
xlabel('Time (s)'); ylabel('Amplitude');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,'MFR2_imp', '-dpdf', '-painters');

% log squared
y_abs = impulse.^2;
y_norm = y_abs./max(y_abs);
ylog = 10*log10(y_norm);

figure(); plot(xt, ylog); grid on; ylim([-80 0])
xlabel('Time (s)'); ylabel('Amplitude (dB)');
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 10.7 6],...
        'PaperSize',[10.2 5.9])
print(gcf,'MFR2_imp_log','-dpdf', '-painters');

%% spectrogram
window = round(N/100);      % divide to approx. 100 windows
if mod(window, 2) ~= 0      % if odd
    window = window + 1;
end
overlap = window/2;
nfft = N/4;
figure();
spectrogram(y,window,overlap,nfft,fs,'yaxis','MinThreshold', -100, 'power');
set(gca,'YScale','log'); axis([0 len 0.03 16]); ylabel('Frequency (Hz)');
set(gca,'YTick',[0.035 0.063 0.125 0.25 0.5 1 2 4 8 16]); xlabel('Time (s)');
set(gca,'YTickLabel',{31.5 63 125 250 500 '1k' '2k' '4k' '8k' '16k'})
% change colorbar size and location
c = colorbar; ax = gca; axpos = ax.Position; cpos = c.Position;
cpos(3) = 0.5*cpos(3); c.Position = cpos; ax.Position = axpos;
c.Label.String = 'Power (dB)';
set(gcf,'PaperUnits','centimeters','PaperPosition',[-1.6 0 28 9.9],...
        'PaperSize',[23.6 9.6])
print(gcf,'MFR2_spectrogram', '-dpdf', '-r300');
%% 3D
% view([100 50]); colorbar('off');
% set(gcf,'PaperUnits','centimeters','PaperPosition',[-1.2 0 21.4 12],...
%         'PaperSize',[19.4 11.6])
% print(gcf,strcat('MFR_spectrogram_3D_', int2str(M)), '-dpdf', '-r300');

%% Vocal sample

[vocal, fs] = audioread('vocal2.wav');

% pad length
N = length(vocal);
long = zeros(1,len*fs);
long(1:N) = vocal;
N = length(long);

ym = zeros(1,M);
y = zeros(1,N);

% first value, ym(t-1) = 0
for m = 1:M
    ym(m) = g(m).*long(1);
end

y(1) = sum(ym);

% filter loop
for t = 2:N
    ym_prev = ym;
    for m = 1:M
        ym(m) = g(m).*long(t) + exp(1i*w(m)-a(m))*ym_prev(m);
    end
    y(t) = sum(ym);
end

%% Export audio
miny = min(real(y));
maxy = max(real(y));
if abs(miny) >= maxy
    reverb = real(y)./abs(miny);
else
    reverb = real(y)./abs(maxy);
end

audiowrite('MFR2_vocal_reverb.wav', reverb', fs);

mix = 0.7.*long + 0.3.*reverb;
audiowrite('MFR_vocal_mix.wav', mix, fs);

mix2 = 0.6.*long + 0.4.*reverb;
audiowrite('MFR_vocal_mix2.wav', mix2, fs);

%% Drum beat sample
```

```
[beat, fs] = audioread('beat.wav');

N = length(beat);
long = zeros(1,len*fs);
long(1:N) = beat;
N = length(long);

ym = zeros(1,M);
y = zeros(1,N);

% first value, ym(t-1) = 0
for m = 1:M
    ym(m) = g(m).*long(1);
end

y(1) = sum(ym);

% filter loop
for t = 2:N
    ym_prev = ym;
    for m = 1:M
        ym(m) = g(m).*long(t) + exp(1i*w(m)-a(m))*ym_prev(m);
    end
    y(t) = sum(ym);
end

%% Export audio
miny = min(real(y));
maxy = max(real(y));
if abs(miny) >= maxy
    reverb = real(y)./abs(miny);
else
    reverb = real(y)./abs(maxy);
end

audiowrite('MFR2_beat_reverb.wav', reverb, fs);
mix = 0.7.*long + 0.3.*reverb;
audiowrite('MFR2_beat_mix.wav', mix, fs);

mix2 = 0.6.*long + 0.4.*reverb;
audiowrite('MFR2_beat_mix2.wav', mix2, fs);
```