Digital Reverberator Design in Max/MSP using Schroeder Algorithm

Ilya Dmitrichenko
DSP Programming Case Study (MU3014C)
London Metropolitan University

January 27, 2012

Contents

List of Figures

1 Introduction

1.1 Project Organization

As a matter of good practice, a source code revision control system had been used in the course of this project. The file format that is used by *Max*, is of plain text JSON (JavaScript Object Notation) type, which is perfectly suited for use with any revision management system as opposed to audio files, which are usually of relatively large size and not very much suitable for use with regular version management systems.

The revision control tool of choice for this project was *Git* [1], moreover, in addition to a great set of benefits of using *Git* itself, the *GitHub* service enables an excellent web integration with a simple user interface. The entire work history of this project can be examined on-line:

`https://github.com/errordeveloper/maxmsp-misc-mu3014c/commits`

There is no need for the reader to be familiar with how to use *Git*, since the archive of the current version of the code can be downloaded from *GitHub*:

`https://github.com/errordeveloper/maxmsp-misc-mu3014c/downloads`

In the project's top-level exists `‘patchers‘` directory, that is where most of the files of interest are located, unless explicitly specified, any of file names mentioned in this report can be found there.

1.2 Approaches to Reverberator Design

Studying the papers which describe various reverberation network topologies, it becomes absolutely clear that the degree of experimentation involved in the design process is usually very high. According to Jon Dattorro's article in the Journal of Audio Engineering Society (1997) [1], where he included a letter from Barry Blesser, who refers to the conversation with Manfred Schroeder in the late 1970s where Schroeder have said: *"We did the electronic reverberation for amusement because we thought it would be fun. Since it took the better part of a day to do 10 seconds of reverberation, we only ran one sample of music. The notion of delay time selections was random in that we just picked a bunch of numbers and there was no mathematical basis. We just wanted to prove it could be done."* Also John Stautner and Miller

[1] More information can be found on *Git* homepage:

`http://git-scm.com/about`

Puckette in their article "Designing Multi-Channel Reverberators" (Computer Music Journal, 1982) [9] point out that *"...no attempt has been made to imitate the ambience of an existing room or concert hall, the methods described herein may lead to such applications when they are combined with a consideration of the perceptual importance of attributes of the soundfield in a real room."*. Although, the later papers by William Gardner [2, 3] suggest formulas to calculate coefficients, the use of reverberation is not purelly to imitate a specific type of room or even an existing room somewhere - it is one of the greates tools that musicians can utilies for their artistic expression.

1.3 Brief Overview of History of Reverb Topologies

Manfred Schroeder had been a pioneer in artificial reverberator design, he worked at Bell Laboratories since mid-1950s and published an number of papers in the following two decades [1]. The original Schroeder's topology [7] consisted of 4 parallel comb filters generating early echoes and 2 all-pass filters all in series [2]. Moorer suggests that Schroeder's design have exhibited fluttering decay and poor echo density [4]. The fluttering (sometimes also called "metallic ringing") can be observed in response to impulsive sounds (for example the snare drum), according to Gardner [2], though it is still certainly good for short reverberation times and moderate reverberation levels [3]. One of the major disadvantages is that it *"...does not provide a frequency dependent reverberation time"*. Moorer [4] has modified the comb filter block by adding two extra comb filters and a one-pole low-pass filter in the feedback loop of each of the comb filters. The low-pass in feedback was intended to dump the higher frequencies similarly to how those are absorbed in air. Moorer's technique has helped to reduce the unnaturally sounding flutter, however it has not eliminated it entirely. There had been a great amount of research in this area, where the two biggest problems had been - eliminate unnatural colouration and achieve the highest echo density. It should be noted that most of the later topologies were designed by modifying the feedback path. For example, Puckette and Stautner [9] have achieved much higher echo density in their multi-channel system by using a rotation matrix in the feedback loop. Miller Puckette also presents this as an example[2] in his book "The Theory and Technique of Electronic Music"[5]. Gardner [3] also highlighted a number of other various feedback loop improvements as well as the work of Jean-Marc Jot in early 1990s, who introduced time-varying feedback correcting algorithm, which deserves a separate case study and the author of this report certainly desires to do so. However, the subject set herein is the earlies reverberator, one which Manfred Schroeder has invented in early 1960s.

[2] It had been a trivial task to implement Puckette's reverb in Max and it can be found in the project repository, stored as `'reverb.matrix-rotation.maxpat'`

2 Implementation

2.1 Building Blocks

As mentioned above, the Schroeder's reverb contained 4 parallel comb filters in series with 2 all-pass filters, these are also often referred to as the *unit reverberators*.

Comb Filter

The comb filter has a few different uses in audio signal processing (e.g. octave doubling [5]) and it seems at first as just a simple digital filter. However, instead of delay line being of 1 or 2 samples in length, it uses a significantly larger length of delay, hence exhibits a number of effects, one of which finds the use in reverberation network designs.

Two variants of comb filter design had been encountered in different sources[3], both appear a little different and it was not very clear at first what exactly the difference is (see figures 1 and 2). It is noticeably difficult to implement an arbitrary algorithm using *Max* and obtain plain text output which could be copied into a report document, hence a small C program was written to simulate impulse response of the two topologies and prove that the filters differ very slightly. Code listing 1 contains just the functions which implement the algorithms of each of the topologies under consideration[4].

With $g = 0.65$ the following output had been obtain for the first 6 samples:

```
comb_var_a = < 0.000000 1.000000 0.650000 0.422500 0.274625 0.178506 >
comb_var_b = < 1.000000 0.650000 0.422500 0.274625 0.178506 0.116029 >
```

Hereby it is clear that two of the comb filter variants differ in regards to the first sample of the impulse response, the variant A is deduced version of B if it have had an additional delay line (as shown in figure 3). The benefit of using the variant A in reverberator design is due to the fact that a number of comb filters will be placed in parallel and will result in an undesired effect as show below in terms of difference equations for two comb filter with the same input $x_A[n]$ connected in parallel.

[3]For example, Miller Puckette in "Theory and Technique of Electronic Music" [5] on page 185 in figure 7.7 gives topology B, though in the CMJ article [9] he clearly shows topology A. While William Gardner [3, 2] and James Moorer also use A, Curtis Roads in "The Computer Music Tutorial"[6] on page 479 in fugure 11.79 shows B.

[4]The entire program can be view on *GitHub*:

`https://github.com/errordeveloper/maxmsp-misc-mu3014c/blob/master/tests/comb_filters.c`

The difference equation for variant B (fig. 2) of the comb filter can be written:

$$y_B[n] = x_B[n] + gy_B[n - m] \tag{1}$$

For any two different values m_1 and m_2, and provided that the input $x_B[n]$ is a common node, as well as output $y_{B_1}[n]$ and $y_{B_2}[n]$ are summed in a common node, then the following equation applies:

$$y_{B_1}[n] = x_B[n] + gy_{B_1}[n - m_1]$$
$$y_{B_2}[n] = x_B[n] + gy_{B_2}[n - m_2]$$
$$y_{B_1}[n] + y_{B_2}[n] = 2x_B[n] + gy_{B_1}[n - m_1] + gy_{B_2}[n - m_2]$$

Hence, the output of two parallel comb filters of topology B will contain double amplitude of $x_B[n]$, which, in other terms, means that using topology B in a reverb will result in undesired contribution to the amount of "dry" signal in the output.

The figure 3 has been drawn based on empirical results of a simulation using the code listed below, however additional proof seems necessary.

```
void  comb_test_topology_a
 (float y[], float x[], float d, float g) {

  printf("comb_var_a = < ");
  for (int i = 0; i < SIZE; i++) {

    y[i] = d;
    d = x[i];
    x[i+1] += d*g;

    printf("%f ", y[i]);
  }
  printf(">\n");
}

void  comb_test_topology_b
 (float y[], float x[], float d, float g) {

   printf("comb_var_b = < ");
   for (int i = 0; i < SIZE; i++) {

     d = y[i] = x[i] + g*d;
     printf("%f ", y[i]);
   }
   printf(">\n");
}
```

Listing 1: Extract of the comb filter test program

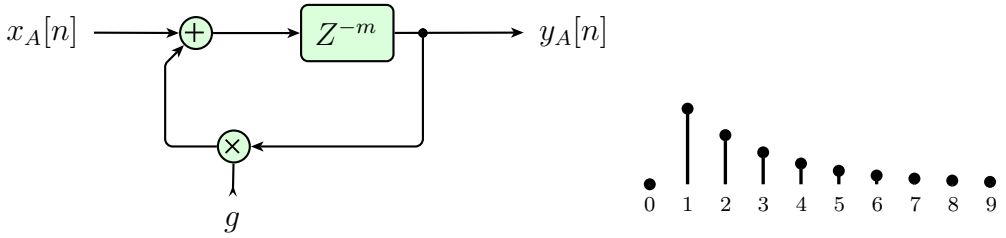

Figure 1: *Variant A of comb filter and it's impulse response* ($m = 1$, $g = 0.65$)

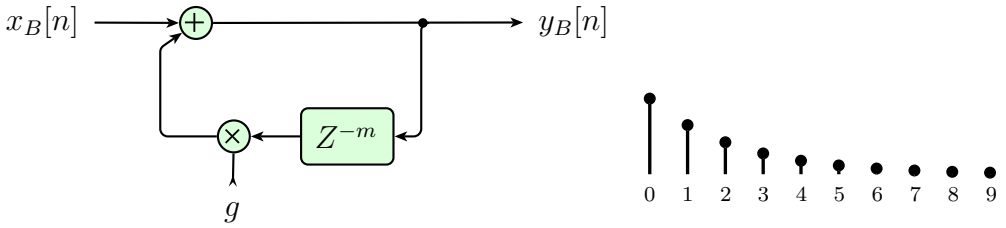

Figure 2: *Variant B of comb filter and it's impulse response* ($m = 1$, $g = 0.65$)

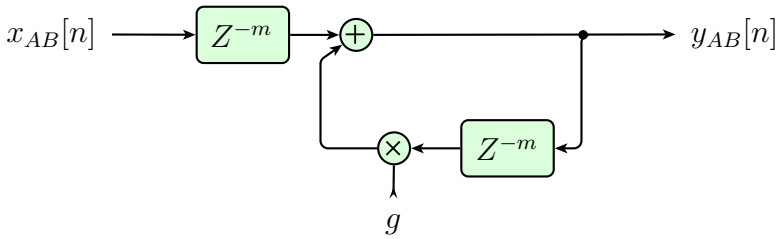

Figure 3: *Variant B modified to produce direct form of A*

Gardner [3] gives the following transfer function for the comb filter (variant A):

$$H_A(z) = \frac{z^{-m}}{1 - gz^{-m}} \tag{2}$$

Given the transfer function (2), the difference equation can be written in Z-domain:

$$\begin{aligned} Y_A(z)(1 - gz^{-m}) &= X_A(z)(z^{-m}) & (3) \\ Y_A(z) &= X_A(z)(z^{-m}) + Y_A(z)(gz^{-m}) & (4) \end{aligned}$$

Hence, the time domain difference equation can be derived:

$$y_A[n] = x_A[n - m] + y_A[n - m] \tag{5}$$

The equation (5) clearly represents the same system which had been defined by observation of previous simulation using the C program above. The comb filter has it's name because the magnitude response graph, which is of comb shape. That is why Schroeder's reverb has the "metallic" sound, it is due to additional harmonics introduced by the comb filters. It is also important to note that the comb filter is an IIR system and it has multiple poles, which are spread harmonically at distance proportional to m modulus proportional to g. Further analysis of frequency response and it's impact on the audible qualities of Schroeder's reverb are outside the scope of this report.

All-pass Filter

As defined by Julius Smith in "Introduction to Digital Filters: With Audio Applications" [8], *"A linear, time-invariant filter ... is said to be lossless if it preserves signal energy for every input signal"*, which generally implies

$$H(e^{j\omega})| = 1 \ \forall\omega \tag{6}$$

Therefore, by definition, all-pass filter may have any topology provided that it's magnitude response is lossless and only it's phase response would be of rather more complex nature, though no definition had been made to what the phase response should be. Effectively, all-pass filter still does affect the audible signal, although only an experienced listener could differ the output of an all-pass system from a "dry" input. The task herein is to implement Schroeder's reverberator and it's building blocks in *Max*, hence only limited number of statements will be made. That is said considering how diverse is the field of all-pass filter design, since no formal network topology definition exists.

Looking at Gardner's work [3, 2], again there appear to be two slightly different implementation of all-pass filter, both of which are shown in figures 4 and 6. Both

of these were also encountered in other literature, including Roads [6] and Puckette [5]. To compare these, it should be sufficient to write down the difference equation for each of the variants, using A and B as subscripts for clarity.

$$y_A[n] = -gx_A[n] + (1 - g^2)x_A[n - m] + gy_A[n - m] \tag{7}$$

$$y_B[n] = -gx_B[n] + x_B[n - m] + gy_B[n - m] \tag{8}$$

A very similar C program[5] (lis. 2) has been written to simulate the behaviour for two all-pass filter implementations (fig. 4 & 6). Unlike the program in listing 1, this was rather much simpler to implement since the difference equation was known already[6]. It helped, however, to obtain the impulse response for each of the variants which is show below for the first 6 sample with $g = 0.5$ and $m = 1$. It should be also noted that the value of m will affect the frequency response of the filter and it's impulse response, nevertheless setting $m = 1$ in this simulation does give a good picture where the difference between two topologies can be observer. Namely, the data shown below confirms that the extra multiply node on the diagram only affect the decay curve of the impulse response.

```
allpass_var_a = < -0.500000 0.500000 0.250000 0.125000 0.062500 0.031250 >
allpass_var_b = < -0.500000 0.750000 0.375000 0.187500 0.093750 0.046875 >
```

As it has been said already, the objection of this report is rather more practical, hence there is no detailed derivation of $H(z)$ and proof of that equation 6 is true for the above all-pass filters, in fact that would be rather redundant.

[5]Full source code is also available on *GitHub*:

`https://github.com/errordeveloper/maxmsp-misc-mu3014c/blob/master/tests/allpass_filters.c`

[6]It was somewhat more difficult to write the equation for the comb filters, since the diagrams look very different to a common n-pole/n-zero filter.

```
void  allpass_test_topology_a
 (float y[], float x[], float d, float g) {

  printf("allpass_var_a = < ");
  for (int i = 0; i < SIZE; i++) {

    if (i == 0) {
      y[i] = -g*x[i];
    } else {
      y[i] = -g*x[i]+(1-g*g)*x[i-1]+g*y[i-1];
    }
    printf("%f ", y[i]);
  }
  printf(">\n");
}

void  allpass_test_topology_b
 (float y[], float x[], float d, float g) {
   printf("allpass_var_b = < ");
   for (int i = 0; i < SIZE; i++) {

     if (i == 0) {
       y[i] = -g*x[i];
     } else {
       y[i] = -g*x[i]+x[i-1]+g*y[i-1];
     }
     printf("%f ", y[i]);
   }
   printf(">\n");
}
```

Listing 2: Extract of the all-pass filter test program

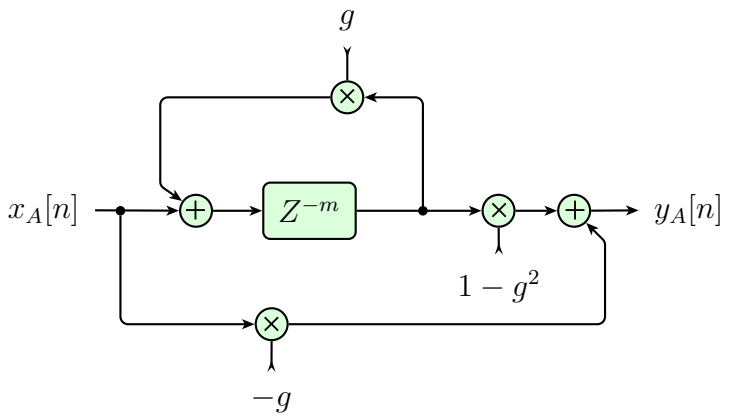

Figure 4: *Variant A of all-pass filter*

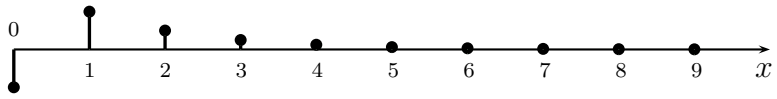

Figure 5: *Impulse response of varian A of all-pass filter*

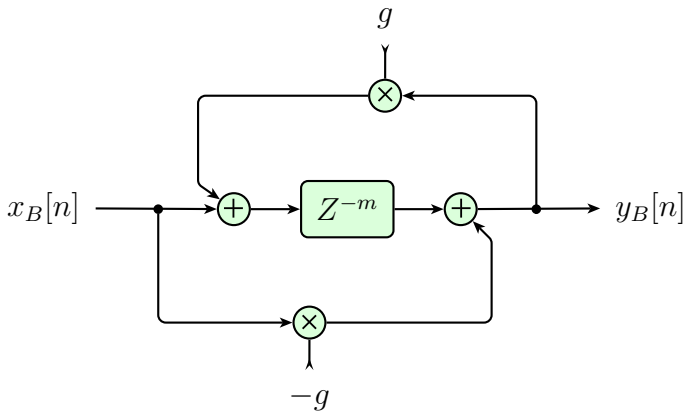

Figure 6: *Variant B of all-pass filter*

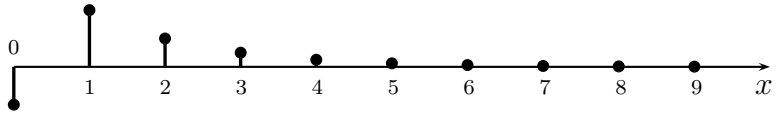

Figure 7: *Impulse response of varian B of all-pass filter*

2.2 Max/MSP Patches

The implementation of Schroeder's reverberator requires two major components - a comb and an all-pass filter. *Max/MSP* provides built-in objects for these two types of filters, however the task of this project had been to implement the reverberator utilising only the most primitive of instances of built-in objects. Figures 8 and 9 demonstrate the unit reverberators which had been implemented and figure 10 show the inner body of the patch filed as `'reverb.schroeder~.maxpat'`.

Once the reverberator has been implemented, the major challenge was to find the appropriate ratio of the gain and delay values. According to Gardner [3], the following formula can be used to find the values of g_i from m_i:

$$g_i = 10^{-3m_i/T_r} \tag{9}$$

where m_i is the delay time in seconds and T_r is the reverberation time. The implementation in *Max* (shown in figure 10) would had been rather unacceptable without this particular relation, because the lister may simply damaged their ears. The sub-patch `'calc.coef'` implements the equation given by Gardner, there are also `'filt.comb-lp~'` and `'filt.allpass-time-delay~'` which are representing the comb filter with low-pass feedback (Moorer's comb) and an all-pass filter as seen in figure 4 . The reason why the low-pass filter had to be placed in the feedback loop of the comb filter, is simply because the sound without it was rather hard to believe being of a reverb unit. It is very clear that a commercially available reverberator today sounds much different from it's "early digital" ancestor circa 1960s, moreover it becomes obvious what sort of "metallic" and "unnatural" sounds were described by all of the researchers cited.

The source code repository of this project (see sec. 1.1) also contains several files which are the results of some of the early experiments. Amongst miscellaneous, there is a implementation of Puckette & Stautner rotation-matrix stereo reverb (filename `'reverb.matrix-rotation.maxpat'`) as well as some more primitive filters (e.g. `'filt.1st-order~.maxpat'` and `'filt.2nd-order~.maxpat'`).

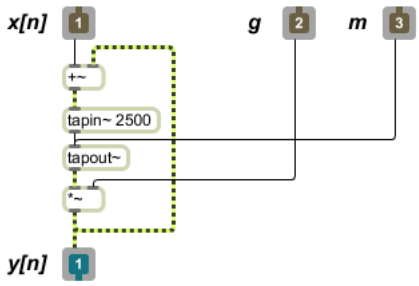

Figure 8: *Comb filter implemented in Max/MSP*

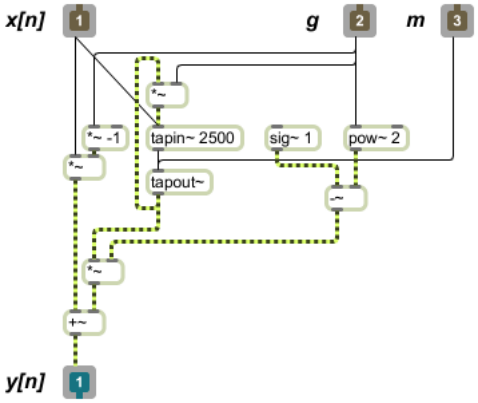

Figure 9: *All-pass filter implemented in Max/MSP*

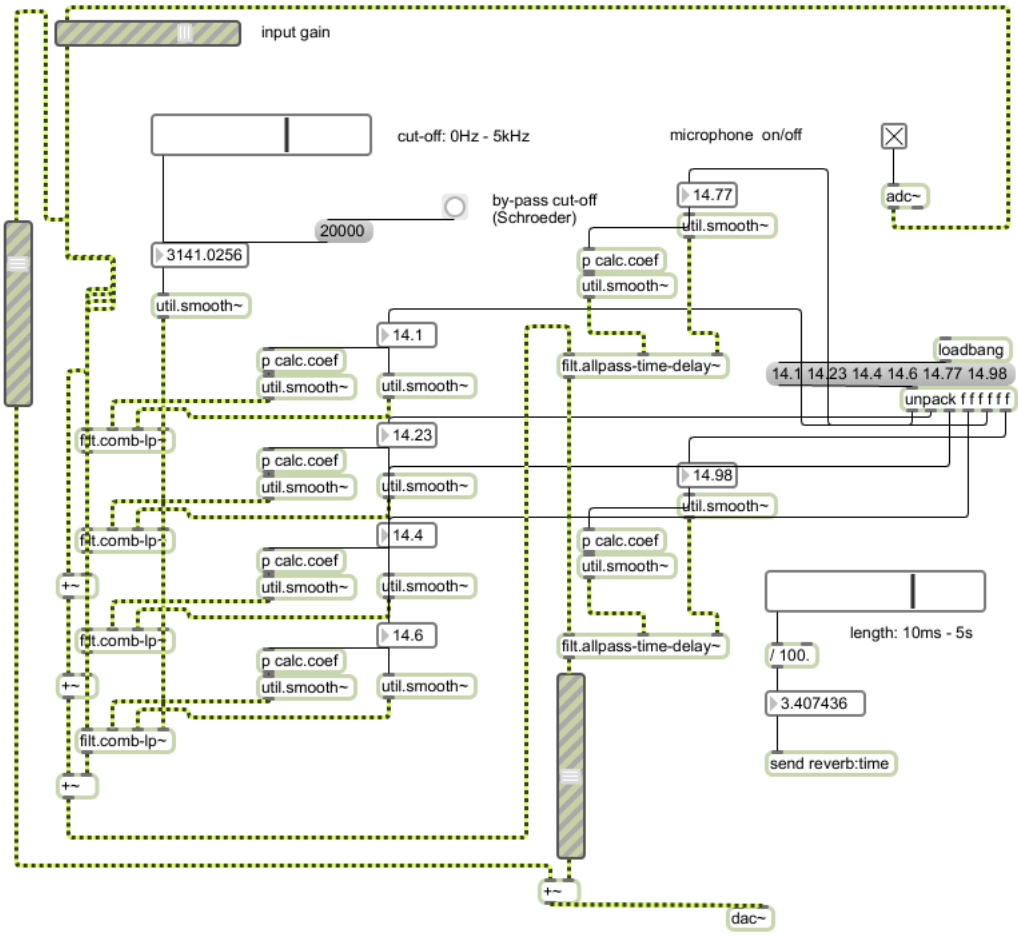

Figure 10: *Internals of the reverberator patch*

Bibliography

[1] Dattorro, J. Effect Design, Part 1: Reverberator and Other Filters. *Journal of Audio Engineering Society vol. 45*, no. 9 (1997), pp. 660–684.

[2] Gardner, W. G. The Virtual Acoustic Room. Master's thesis, MIT Media Laboratory, Perceptual Computing Group, 1992.

[3] Gardner, W. G. Reverberation Algorithms. In *Applications of Digital Signal Processing to Audio and Acoustics*, M. Kahrs and K. Brandenburg, Eds., vol. 437 of *The Kluwer International Series in Engineering and Computer Science*. Springer US, 20 Ames Street, Cambridge, MA 02139, USA, 2002, pp. 85–131. 10.1007/0-306-47042-X_3.

[4] Moorer, J. A. About This Reverberation Business. *Computer Music Journal vol. 3*, no. 2 (1979), pp. 13–28.

[5] Puckette, M. S. *The Theory and Technique of Electronic Music.* World Scientific Publishing Co., 2007.

[6] Roads, C. *The Computer Music Tutorial.* MIT Press, 1996.

[7] Schroeder, M. R. Natural sounding artificial reverberation. *Journal of Audio Engineering Society vol. 10*, no. 3 (1962), pp. 219–223.

[8] Smith, J. O. *Introduction to Digital Filters: With Audio Applications.* W3K Publishing, 2007.

[9] Stautner, J., and Puckette, M. Designing Multi-Channel Reverberators. *Computer Music Journal vol. 6*, no. 1 (1982), pp. 52–65.