Matlab Implementation of Reverberation Algorithms.

Fernando A. Beltrán[†], José R. Beltrán[†], Nicolas Holzem[††], Adrian Gogu[†††].

[†]*Department of Electronic Engineering and Communications. University of Zaragoza (Spain)*
beltran@posta.unizar.es, http://www.unizar.es
[††]*Faculté des Sciences Appliquées. Université Libre de Bruxelles (Belgium). matlab_reverb@usa.net*
[†††]*Dpt. of Basis of Electronics. Technical University of Cluj-Napoca (Romania). agogu@bel.utcluj.ro*

Abstract

In this paper, we present the implementation of different reverberation algorithms in the Matlab programming environment. This is a useful tool to analyze the algorithms behavior from the signal processing and sounding point of view. With Matlab environment is possible and simple to view the filter characteristics, impulse response, phase response and all the relevant characteristics of the filters. In addition, the possibility of hearing the results is quite simple and fast. We present the main aspects of programming these algorithms using Matlab and the results produced with these techniques.

1. Introduction

Reverberation is a very common and often unnoticeable phenomenon in our lives. The surrounding walls in a concert hall and in the office, the walls of the buildings in the street, every object around us reflects the sound that is propagating through the space. Because these reflections, what we hear is not only the information that leaves from the sound source; some additional components are added to the original sound. The reflections modify the perception of the sound, changing its loudness, timbre and its spatial characteristics [1].

The presence of reverberation is especially interesting in music. This effect adds life and a sense of space. The reverberation is associated with the architecture and acoustic of concert halls, and this factor is critical for good sounding in concert performances. In recorded music, there has been a great effort in simulating this effect through some electro-acoustic devices. Nowadays, digital signal processing techniques are mostly used for these purposes.

From the first Schroeder work (early 1960's) about artificial reverberators based on discrete-time signal processing [2][3], many implementations have been presented in different publications. Some of them are very relevant (Dattorro, Gardner, and Jot) offering different approaches to obtain a reverberation algorithm with some specific characteristics: natural sounding, absence of tonal coloration, high echo density, control of the reverberation time, etc.

In this paper we will consider a typical impulse response, associated with the reverberation effect, where we can distinguish the early echoes and the late reberveration tail.

This paper is divided as follows. Section 2 presents the basis programming techniques and elementary filters used in the reverberation algorithms. In sections 3, 4 and 5 we describe the implementation details of Datorro, Gardner and Jot's algorithms, respectively. The main conclusions are highlighted in section 6.

2. Matlab basis programming and elementary filters

Filtering in Matlab is a very simple task. The built-in function *filter* allows us to filter a signal, *signal_in*, with the transfer function of a filter H(z) described by the numerator *num* and the denominator *den*, to obtain a signal *signal_out*. The syntax is quite simple and we can write:

```
signal_out=filter(num,den,signal_in).
```

This function is proved to be fast and all the variables, as every variable in Matlab, are vectors containing the signals and the coefficients of the numerator and the denominator.

It should be noted that this approach to the processing of a signal is quite different to a DSP oriented one, in which we have a data buffer and pointers to the samples. In the direct DSP

programming, we have also a whole control over the signal path, with the possibility of accessing the data at the proper point.
Our first task is to rewrite the building blocks we need to program a reverberation algorithm in the Matlab syntax using the *filter* function and to describe how to combine these blocks to obtain the whole algorithm.

2.1 The delay

The simplest filter we have is a delay. Its transfer function can be written as:

$$H(z) = z^{-k}. \quad (1)$$

So, the *filter* function of Matlab will need: *den=1*, *num[j]=0, j=1,…,k,* and *num[k+1]=1*.

2.2 The lowpass filter

Figure 1 shows the block diagram of a lowpass filter. It can be seen that the transfer function is:

$$H(z) = \frac{1-g}{1-gz^{-1}} \quad (2)$$

We can write *num=1-g, den[1]=1, den[2]=-g*.
Lowpass filters are used to simulate air absorption. In order to relate the gain g and the cutoff frequency of the filter we need:

$$g = 2 - \cos\left(2\pi \frac{f_c}{f_s}\right) - \sqrt{\left(\cos\left(2\pi \frac{f_c}{f_s}\right) - 2\right)^2 - 1} \quad (3)$$

where f_c and f_s are, respectively the cutoff frequency and the sample rate.

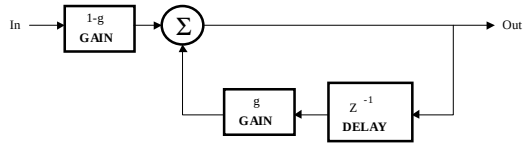

Figure 1. Lowpass filter block diagram

2.3 The allpass filter

In figure 2 we can see the block diagram of an allpass filter. The transfer function of this useful filter in the reverberation algorithms can be written as:

$$H(z) = \frac{g + z^{-k}}{1 + gz^{-k}} \quad (4)$$

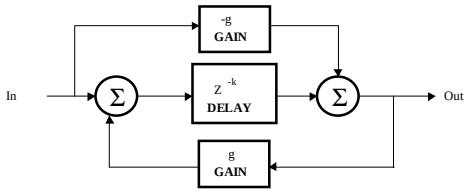

Figure 2. Allpass filter block diagram.

This expression implies that: *num[1]=g*; *num[j]=0, j=2,…,k*; *num[k+1]=1*; *den[1]=1*; *den[j]=0, j=2,…,k*; *den[k+1]=g*.
With the above expressions, we can easily plot the characteristics of the different filters, like poles and zeros, impulse response, frequency response, and phase frequency response, with the corresponding Matlab functions.
In some cases is necessary to extract inner signals from allpass filters. From figure 3 it can be seen that:

$$V(z) = z^{-k}U(z), U(z) = X(z) - gV(z), S(z) = z^{-l}U(z), \quad (5)$$

where S(z) is the desired signal. It can be shown that its transfer function is:

$$S(z) = \frac{z^{-l}X(z)}{1 + gz^{-k}}. \quad (6)$$

The numerator of a delay and the denominator of an allpass filter form the last equation.

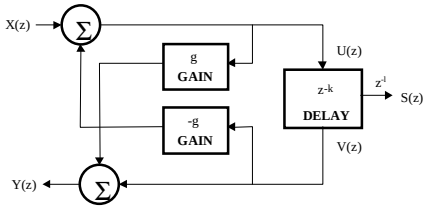

Figure 3. Extracting the signal from inner buffer

2.4 Connecting two filters in series

In order to combine two filters in series we have two possibilities. The first one is to compute the output of the first filter, and input this intermediate output into the second filter. The second one is to calculate a global transfer function for the new structure and make a call to the function *filter* with that function. The whole transfer function could be calculated or computed in Matlab. In fact, the transfer function of two or more filters in series is the product of the individual transfer functions. Therefore, we can use the Matlab function *conv* to compute the polynomial product of the numerators and the denominators of the individual transfer functions. When the number of filters is greater than four or five, the filter length is so long that this approach is not recommendable. This is due to the increasing in computing time with respect to the time spent filtering the signal with individual filters.

2.5 Nesting allpass filters

In figure 4 we can see the structure of a nested system. These systems are used mainly in Gadner's networks. F(z) is a general transfer function in which we have either an allpass filter alone, or an allpass filter with some delay.
Anyway, we can construct the transfer function:

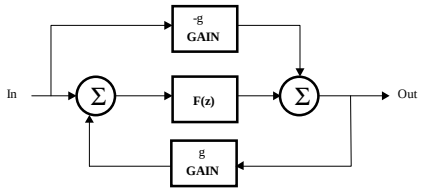

Figure 4. General structure of a nested system.

$$F(z) = \frac{F_n(z)}{F_d(z)}, \qquad H(z) = \frac{-gF_d(z) + F_n(z)}{F_d(z) - gF_n(z)} \tag{7}$$

The main advantage of these kinds of filters in place of a serial connection is that there is no repetition of the same echo pattern. Instead, the pattern size increases, so echo density increases too, as in real rooms.

3. Dattorro's reverberation network

Dattorro published this topology in 1997, among other digital audio effects [4], reporting the following features:

- The given topology is applicable to a broad range of signal sources.
- Five knobs control particular aspects of the reverberated sound. However, not all of these knobs are directly related to physical characteristics of reverberation.

There is to consider that this topology only produces the late reverberation tail.

3.1 Dattorro's reverberator structure

In figure 5 we can see Datorro's proposed network to obtain a reverberator. The incoming sound enters to a delay followed by a lowpass filter; then passes through a set of four short cascaded allpass filters. These last four elements form what Dattorro calls the decorrelation stage, and perform a rapid buildup of echo density. Two knobs control this stage: *input_difussion_1* and *input_difussion_2*, which are actually the gains of the two firsts and the two last all pass filters respectively.

The signal then enters the structure called tank, where it recirculates indefinitely, progressively attenuated by taps of value *decay*. The tank is composed of two symmetrical lines, which are cross-coupled. Additionally to the rate of decay, two knobs named *decay_difussion_1* and *decay_difussion_2*, which are the gains of the all pass filters 5-5', and 6-6' respectively, allow to set how the recirculating signal is altered.

The network produces a decorrelated all wet stereo output, formed by the sum of signals extracted from the tank in various places.

3.2 Implementation details

The decorrelation stage implementation is straightforward, as it holds no feedback loop. Each filter successively processes the signal reinjecting the output of one filter to the next one. Tank implementation is somehow more complex because of the recursive structure. In figure 6 we can see a simplified representation of the tank structure. We can obtain the transfer function of the input node *n_in* to the reference node *n_ref*:

$$T(z) = \frac{n\_in}{n\_ref} = \frac{1 + H_2(z)}{1 - H_1(z)H_2(z)} \tag{8}$$

T(z) is the largest transfer function needed to process the input file. Obtaining *n_ref* from *n_in* (in figure 6) is, by far, computationally the most expensive task of the reverberator. Once obtained *n_ref*, the output taps can be obtained by successively calling to the required simple filtering functions described in section 2. It should be noted that the output taps are obtained inside some allpass filters, so eq. (6) must be used. Table (1) shows the output points taken to construct the left and right output channels.

Node	Delay	Sign	Node	Delay	Sign
n_out_4	266	+	n_out_1	353	+
n_out_4	2974	+	n_out_1	3627	+
n_out_5	1913	-	n_out_2	1228	-
n_out_6	1996	+	n_out_3	2673	+
n_out_1	1990	-	n_out_4	2111	-
n_out_2	187	-	n_out_5	335	-
n_out_3	1066	-	n_out_6	121	-

Table 1. Left output (left), right output (right).

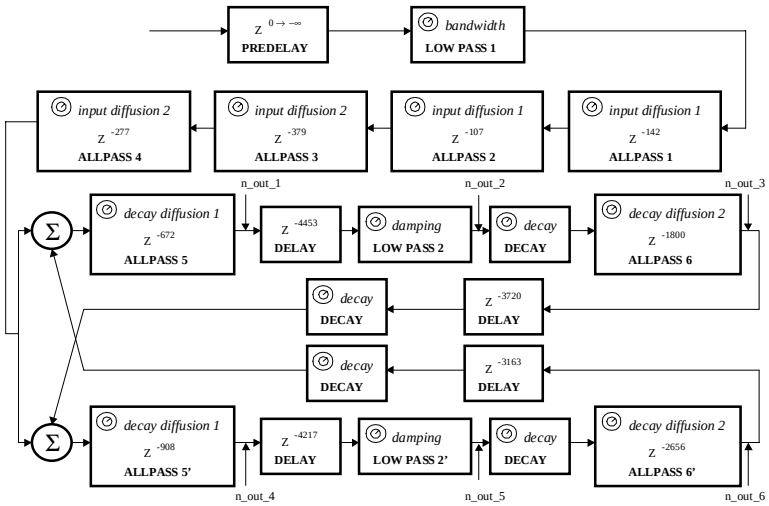

Figure 5. Datorro's reverberator network

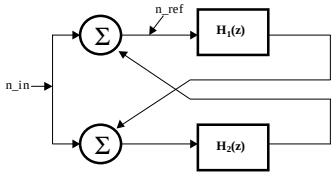

Figure 6. Simplified tank structure.

4. Gardner's reverberator structure

Gardner proposed in 1992 [5] a set of reverberation algorithms, sharing a common global structure, but with implementation differences depending on the reverberation time. Because these algorithms were intended for the simulation of a virtual acoustic room, they were categorized as small, medium and large room.

4.1 Gardner's reverberator structure

Figure 7 shows the block diagram of the Gardner's reverberator for a large room (the block diagrams for small and medium rooms are quite similar). For natural sounding, the author recommends to use the type small room for reverberation times from 380 to 570ms, medium room for 580ms to 1.29s, and large room for reverberation times larger than 1.30s.

The three reverberators are composed of a single delay line looped on itself through a variable value gain. The reverberation time is adjusted with this parameter. The structure of this algorithm is based, mainly, in allpass filters (simple and nested ones) and delays, followed by a first order low pass filter. The output is obtained by summing signals taken out various points within the delay line. This topology as Datorro's one, only produces the late reverberation tail.

4.2 Implementation details

Gadner's structures are easily implemented following the same philosophy we have presented for Datorro's one. A feedback loop is presented in figure 8. The transfer function can be expressed as:

$$G(z) = \frac{F_d(z)}{F_d(z) - gF_n(z)}, \quad \text{where} \quad F(z) = \frac{F_n(z)}{F_d(z)} \qquad (9)$$

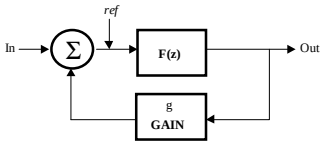

Figure 8. Recirculation loops.

When signal *ref* is obtained, we can build up the output signal processing successively through each filter. Echo density is poorer than the obtained with Datorro reverberator, but computing time is shorter.

5. Jot's reverberation network

In 1992, Jot proposed a reverberator structure [6] with two important properties:

- A reverberator can be designed with arbitrary time and frequency density while simultaneously guaranteeing absence of tonal coloration in the late decay.
- The resulting reverberator can be specified in terms of the desired reverberation time and frequency response envelope.

This reverberator simulates early reflections and late reverberation.

5.1 Jot's reverberator structure

The basic idea for this reverberator is to combine a set of delay lines and attenuation stages with a

unitary matrix A on the feedback loop ($A*A^T=1$) (see figure 9). This idea was applied by Stautner and Puckette to obtain a four-channel reverberator structure, which represents a generalization of Schroeder' parallel comb filter, but offering a higher echo density. The absorptive losses are implemented with a global filter (named FTJ in figure 9).

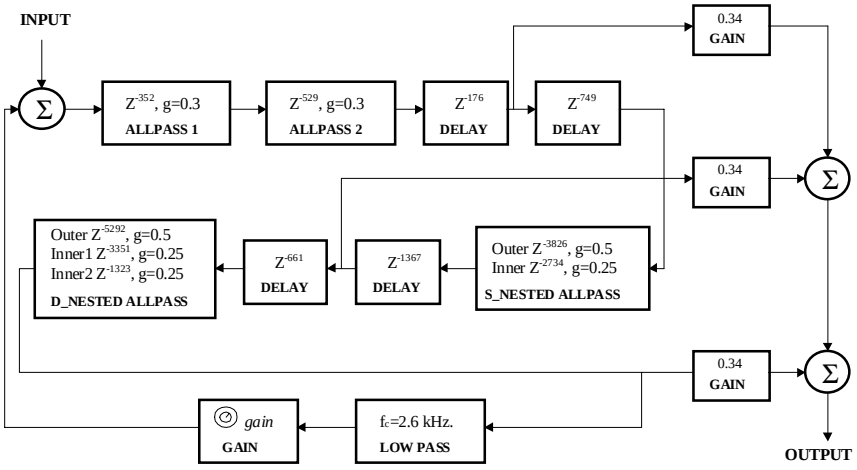

Figure 7. Large room Gardnes's reberverator block diagram

Jot has obtained a better reverberator using a lossless prototype (an energy conserving system whose impulse response is perceptually equivalent to stationary white noise). This structure contains a feedback delay network based on unitary matrix.

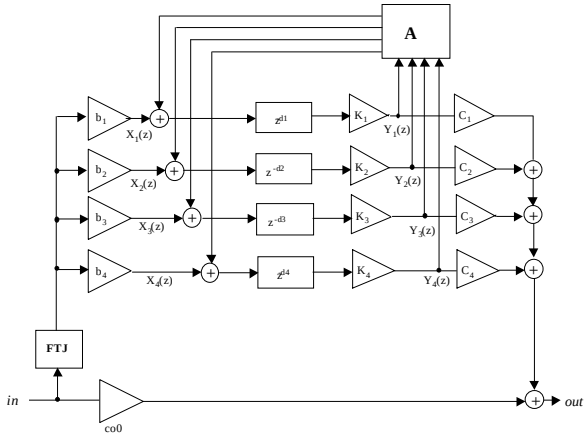

Figure 9. Jot's reverberator structure.

5.2 Implementation details

Jot's structure is, by far, the most difficult to be developed using Matalb programming techniques. The only way to program it is using *for* loops, that must be avoided as much as possible.

We can make the effort to rewrite Jot's output in a compact expression. In this case, we need a column vector of unidimensional Z-transform of the signals $X_i(z)$ and $Y_i(z)$, i=1,…,4 of figure 8.

$$\mathbf{X}(z)=\begin{bmatrix} X_1(z) \\ X_2(z) \\ X_3(z) \\ X_4(z) \end{bmatrix} \text{ and } \mathbf{Y}(z)=\begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \end{bmatrix} \qquad (10)$$

We can define two diagonal matrices for the delays z^{-di} and gains k_i:

$$\mathbf{k}=\begin{bmatrix} k_1 & & & \\ & k_2 & & \\ & & k_3 & \\ & & & k_4 \end{bmatrix} \text{ and } \mathbf{z^{-d}}=\begin{bmatrix} z^{-d_1} & & & \\ & z^{-d_2} & & \\ & & z^{-d_3} & \\ & & & z^{-d_4} \end{bmatrix} \qquad (11)$$

Therefore, the global transfer function of the system will be:

$$\mathbf{G(z)}=\frac{\mathbf{Y(z)}}{\mathbf{X(z)}}=\frac{\mathbf{kz^{-d}}}{\mathbf{I-Akz^{-d}}} \qquad (12)$$

To process the signal in this way we would need a Matlab function *filter* in which the input and output signals and the numerator and the denominator of the transfer function could be matrices. This is not allowed and in the Image Processing Toolbox, the only two-dimensional filters are FIR filters, so the Matlab implementation will be inefficient.

In order to obtain listening results, we have finally implemented this algorithm using a *for* loop, in witch the output in computed sample by sample. In this way we can obtain the four values we need to compute the matricial feedback loop.

6. Conclusions

In this paper we have presented the main aspects of the Matlab implementation of some well-known reverberation algorithms. This tool helps us to understand the behaviour of the different blocks (filters, delays, gains, feedback loops, etc) in which these algorithms are based. Using this tool it is also possible to compare the algorithms complexity and computing time required for them.

With this mathematical environment it is simple to modify different parameters of each algorithm, and to obtain listening results.
As a final note there is to say that Matlab is not well suited for long signals processing, because computing time is too long in typical PC platforms.

References

[1] Gardner, W.G. 1998. Chapter 3. Reverberation Algorithms, in Kahrs, M. and Brandenburg, K. Editors. Applications of Digital Signal Processing to Audio and Acoustics. Kluwer Academic Publishers.

[2] Schroeder. M. R., Logan, B. F. 1961. Colorless Artificial Reverberation. J. Audio Engineering Society. Vol. 9, No. 3.

[3] Schroeder. M. R. 1962. Natural Sounding Artificial Reverberation. J. Audio Engineering Society. Vol. 10, No. 3.

[4] Dattorro, J. 1997. Effect Design. Part 1: Reverberator and Other Filters. Journal of Audio Engineering Society. Vol. 45, No. 9. Pp. 660-684.

[5] Gardner, W.G. 1992. The virtual Acoustic Room. Master Science Thesis at the MIT.

[6] Jot, J.M. 1992. Etude et réalisation d'un spatialisateur de sons par modèles physiques et perceptifs. Ph.D. thesis, Telecom, Paris.