Audio Engineering Society

Convention Paper

Presented at the 133rd Convention
2012 October 26–29 San Francisco, CA, USA

Generating Matrix Coefficients for Feedback Delay Networks Using Genetic Algorithm

Michael Chemistruck[1], Kyle Marcolini[2], and Will Pirkle[3]

[1,2] Student, University of Miami, Music Engineering, Coral Gables, FL, 33146, USA
m.chemistruck@umiami.edu, k.marcolini@umiami.edu

[3] Professor, University of Miami, Music Engineering, Coral Gables, FL, 33146, USA
wpirkle@miami.edu

ABSTRACT

The following paper analyzes the use of the Genetic Algorithm (GA) in conjunction with a length-4 feedback delay network for audio reverberation applications. While it is possible to manually assign coefficient values to the feedback network, our goal was to automate the generation of these coefficients to help produce a reverb with characteristics as similar to those of a real room reverberation as possible. We designed a GA to be used in a delay-based reverb that would be more desirable in the use of real-time applications than the more computationally expensive convolution reverb.

1. INTRODUCTION

Optimizing digital reverberators for sound, realism, and computational efficiency in audio applications has been a withstanding problem for decades. The current, most accurate standard convolves an impulse response of a room or a space with an audio signal. The result will be the same audio signal with the spatial characteristics of the room; however, this technique can be very CPU-intensive, especially when applied to multiple audio files simultaneously in real time. Although computers are getting more powerful, convolution is still not practical for large-scale, consumer-grade production scenarios.

The ultimate goal is to create a reverberator solely based off a feedback delay network (FDN) reverberation model. This is much less CPU-intensive and is plausible for real-time use with consumer-grade equipment. By tuning the FDN to generate an impulse response most complementary to the original, the resulting reverb is guaranteed to be similar to the actual room's response.

1.1. Basis for Reverb Algorithm

Generalized feedback reverb algorithms have been around for several decades. Michael Gerzon, a mathematician from Oxford University, proposed an idea about energy preservation of matrices. The foundation behind his research is that an NxN matrix, G, is energy-preserved and unitary if it is multiplied by its transpose and the result is the identity matrix, shown below.

$$GG^T = G^T G = I_N = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \quad (1)$$

The algorithm used for this study is the length-4 generalized feedback delay network, introduced by Stautner and Puckette in 1982. Using Gerzon's research in feedback networks, they developed the length-N generalized reverb algorithm based off a unitary matrix.

Stautner and Puckette show that by splitting a signal into N lines, then sending it through a system of N delays and feeding it back using a set of gain coefficients from a unitary matrix, it is possible to generate an energy-preserved output. This proposed system could simulate the effect of reflections in a room and be used for reverberation as an alternative to convolution.

The diagram in Figure 1 shows the proposed layout for the system, with the input, $x(n)$, the output, $y(n)$, the delays, z^{-DN}, and b_N and c_N as scaling factors. The matrix shown above the flowchart is the unitary matrix, proposed by Stautner and Puckette, consisting of the feedback scaling coefficients.

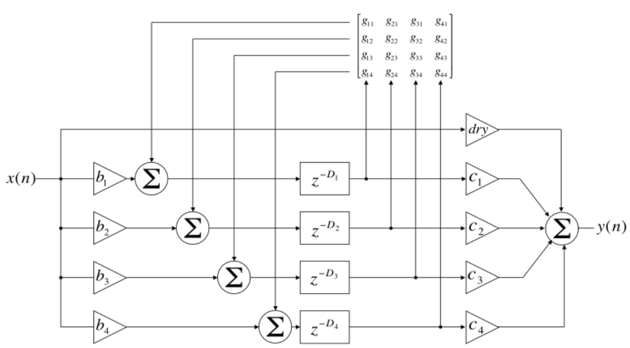

Fig. 1. Length-4 Feedback Delay Network

The basic, length-4 system proposed in 1982 is assigned a unitary feedback matrix, G, shown in (2a) and (2b). The coefficients are used to produce a reverb, but the resulting sound is unnatural sound and does not allow for customization.

$$G = \begin{bmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \bullet g, \quad (2a)$$

where

$$|g| < \frac{1}{\sqrt{2}}. \quad (2b)$$

Initial trials demonstrated that changing the value of g does indeed change reverb characteristics, but not enough to simulate actual room responses. Replacing the matrix in (2a) with a different unitary matrix creates an entirely new reverb with drastically different characteristics. With such a wide range of characteristics available, it seems possible that this method could produce a reverb to match any room of only the proper coefficient matrix was used.

1.2. Additions to the algorithm

Using only the delay network produces a respectable-sounding reverb, but a very pronounced tinny, digital characteristic is prevalent. To help abate this, second-order low pass filters (LPFs) are introduced into each delay line to help attenuate excess high frequency content, which produces a more natural sounding reverb. Figure 2 shows the updated network.

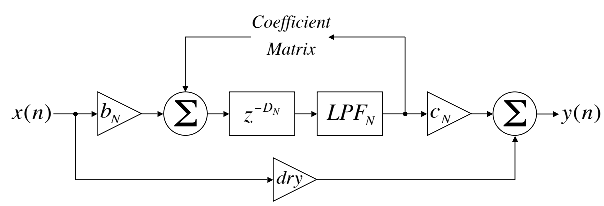

Fig 2. Single, generalized delay line in multi-length FDN with added Low Pass Filter in signal path

Still, the algorithm does not perform well enough to compete with convolution. This could be attributed to using only four relatively short delay lines. While that is enough to simulate the early reflections in a room, it does nothing to simulate *diffusion*, or the wash of later reflections. Another delay network is required after the initial reverberation process to address this. The diffusion algorithm is composed of N different-length delay lines, all less than 300 samples, as shown in Figure 3.

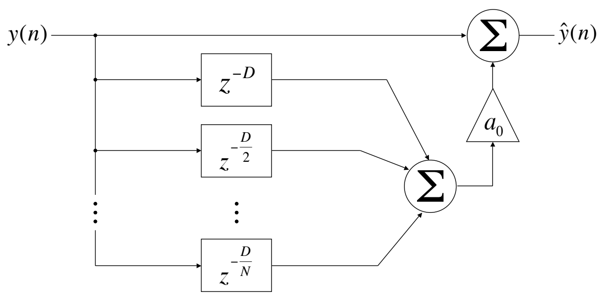

Fig. 3. Generalized, length-N diffusion network. The input, $y(n)$, is the output from FDN, seen in Fig. 2

This addition creates a reverb that has the potential to sound very real and natural. The biggest factor was still determining the proper coefficient matrix. For this study, the Genetic Algorithm is used to find the best coefficient matrix for each room model to eliminate the need for manually auditioning hundreds of different coefficient matrices.

2. THE GENETIC ALGORITHM

The Genetic Algorithm (GA) is an optimization tool used in evolutionary computing. As a search algorithm, the GA is often used in applications where a solution is known to exist, but deriving the solution mathematically is incredibly difficult. It is a guess-and-check algorithm, revolving around the idea of "survival of the fittest."

The GA starts with an initial set of examples (*population*), where each example is called an *individual*, and the individuals are evaluated using a *fitness function*. The fitness function provides a way of quantifying which individuals produce results closest to the desired outcome. After every individual in the population has been evaluated, the population is split into two parts: the lower performing part is discarded while the higher performing part is used to refill the population using a *mating algorithm*.

The mating algorithm picks two individuals (*parents*), and combines them in some way to produce a new individual (*child*). Parents are randomly picked from the top performers to produce children until the population is back to its original size. The genetic algorithm produces many cycles (*generations*) with the idea that the individuals should produce increasingly better results with each additional generation.

3. IMPLEMENTING THE GENETIC ALGORITHM

There are limitless possibilities of random equal-power unitary matrices, but only a handful of those will be useful in accurately modeling the response of a room. The GA is used to pick both the proper feedback coefficient matrix and LPF cutoff frequency for each delay line.

3.1. Parameters

Every generation in the GA consists of 200 individuals, each one containing one 4x4 matrix of feedback coefficients and one LPF per delay line. Individuals' contents are used to generate an impulse response the same length as a user-specified impulse response. Each individual also records its own error value as determined by the fitness function.

3.2. Individual Fitness

There are many ways to compare two audio files; the simplest is comparing the average of each sample for the entire signal, but this method does not produce the expected correlation between the sound of the reverb and the total error.

An *audio envelope detector* is used as the comparison function for the study. Envelopes have associated *attack* and *release* times, which define how closely the envelope traces the audio signal, as shown in Figure 4.

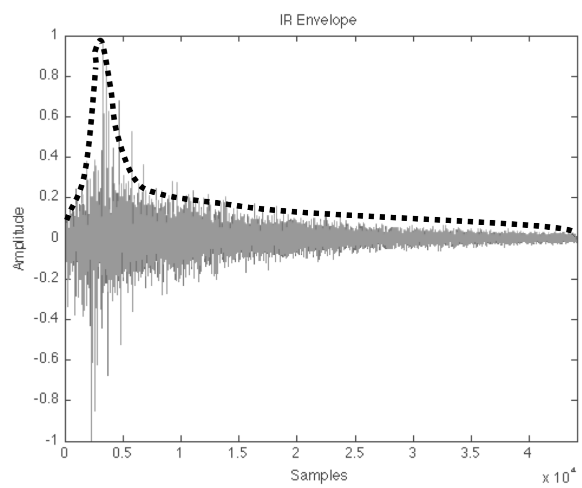

Fig. 4. Envelope traced of the impulse response. ***There is a sharp attack, long sustain, and short release.***

The envelope detector used in this study implements a generic Mean-Square algorithm. The output of the envelope detector for each sample is defined in (3), were $x(n)$ is the square of the input sample, $y(n)$ is the envelope output, $y(n\text{-}1)$ is the previous envelope output sample, a_0 is the attack time, and r_0 is the release time.

$$\begin{aligned} x(n) > y(n-1):\ & y(n) = a_0\left[y(n-1) - x(n)\right] + x(n) \\ x(n) \le y(n-1):\ & y(n) = r_0\left[y(n-1) - x(n)\right] + x(n) \end{aligned} \tag{3}$$

To determine fitness, the envelope of the synthetic impulse response is compared to that of the user-defined impulse. The difference between each sample creates a delta value, ε. Each ε is then summed. The individual with the lowest total delta is deemed best. Fig. 5 shows the error calculation process, described in (4), by finding the mean of the difference in envelopes.

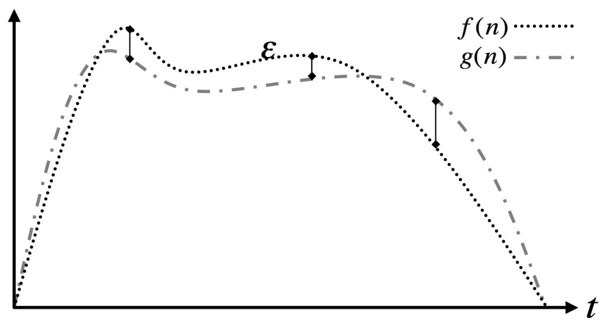

Fig. 5. Error calculated by subtracting envelopes of both signals and scaled by total number of samples.

$$\varepsilon = \frac{1}{NM}\sum_{n=0}^{N-1}\left|f(n)-g(n)\right| \tag{4}$$

3.3. Mating and Mutating

After determining the fitness of each individual, the set is sorted from smallest to largest delta, from which the mating process begins. The set is split into two parts: the top 35% are used as parents while the others are discarded and replaced with the results of the mating process.

To mate, two individuals from the top portion are chosen at random. Each component of their coefficient matrices is averaged to create the child coefficient matrix. Each LPF cutoff frequency is also averaged to create the child LPF cutoff frequencies. This child is added to the set, and the process is repeated until the population is back to its original size. This process is outlined in Figure 6.

During mating, there is always a possibility of *mutation*, or the altering of an individual from its initial state, which creates an entirely unique individual. If a mutation produces better results, it is promoted into the top 35% and used for mating. If not, it is discarded and has no further effect on the population

Every component in the coefficient matrix has a 10% chance of mutation. If a component is flagged for mutation, it is assigned a value between -1 and +1 randomly. This value is used instead of the average of the parents' components. LPF cutoff frequencies have a 5% chance of being mutated. Mutations cause the LPF cutoff frequency to be set to a random value between 5-15kHz.

As with any other implementation of the genetic algorithm, there is always the possibility of *degeneration*. For the purposes of this study, degeneration is defined to have occurred if at least 75% of the population has the same delta. If degeneration is found, every individual is discarded except for the best-ranked one. Then, the set is filled with new, randomly generated individuals using the same algorithm that initially seeded the population.

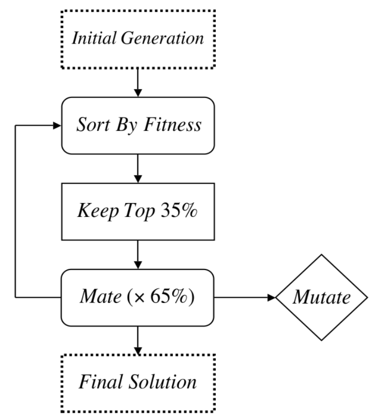

Fig. 6. Process taken in creating the best synthetic reverb using the Genetic Algorithm.

4. USER TRIALS

To verify the accuracy of the 'best' synthetic reverb, a trial was performed, comparing four different reverbs: the recorded sound of a sample played into a room, that same dry sample convolved with the impulse response of a room, and the results of that sample run through two different setups of the GA FDN.

4.1. Generating Reverb Samples

The impulse responses of each acoustical space were recorded and imported into *Impulse Response Utility* in Mac OS X. The measured impulse responses of each acoustic space were loaded into the GA to generate two different synthetic reverbs for the listening test: one after fifty generations and another after two hundred generations. The impulse response of each space was also convolved directly with the drum loop to produce a convolution reverb. All samples were normalized for the most accurate testing situation.

4.2. Listening Test Comparison

In the listening test, users were asked to rate the two GA reverbs against the convolution example. Each user was presented with an audio clip recorded in the space first. The remaining clips were the GA and convolution examples, presented in a random order. The overall results were tallied and can be seen in Figure 7.

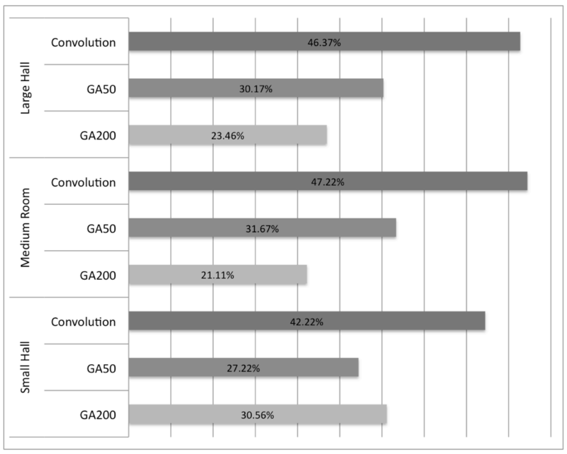

Fig 7. Weighted average (percentage) of how popular each reverb was for all listeners.

5. CONCLUSION

In every acoustical space, the direct convolution of the impulse response of the space with the audio sample proves to be the most accurate sounding reverb by a decent margin. It is interesting to note that the 50-iteration reverb matches two of the three spaces better than the 200-iteration. This suggests that there is a problem somewhere in the generation of the synthetic impulses. If the fitness function were tuned properly, more generations should always lead to a better result. It seems that using envelopes to compare two impulse responses may not be the best metric for evaluating their similarities.

6. DISCUSSION

While the results of using the GA as a coefficient generator for the feedback-delay network create somewhat realistic reverbs, they do not produce reverbs as realistic as direct convolution in this study. Many examples still contain unnatural resonances and have a distinct lack of low frequency content. There are many feature additions to the algorithm and implementation that can possibly improve the overall performance of the system.

The first addition will be to alter the way diffusion is handled. Instead of using constant delay times in the diffusion delay network, the genetic algorithm can be used as an adaptive way to generate diffusion times, thus making the input-to-output process a completely GA-controlled system.

This study only uses a single temporal-based method in determining the error between the synthetic impulse and the user-defined impulse. There are no ways currently to compare every individual impulse on a per-sample basis; instead, it is just comparing the shape of each signal in the time domain. Future work will explore the notion of spectral-based comparisons, as well as comparing the actual impulse responses rather than the envelopes of each.

One difficulty encountered is deciding the attack and release times of the envelope detector while determining the fitness of each impulse. Setting these times to zero has the undesirable effect of turning the envelope follower into a Mean-Square averaging algorithm, which is not an accurate measure of similarity. Setting these times too high results in every impulse producing very similar envelopes, making the fitness function rank the individuals more similarly than they actually are. It seems that this value should be directly related to the length of the impulse response rather than kept constant.

This study requires all impulse responses to be manually edited, eliminating any pre-delay, because there is no time-based matching. If two identical impulse responses are slightly out of phase, they will produce a non-zero error. Future work may implement a simple time-matching algorithm in hopes of producing more accurate deltas.

7. ACKNOWLEDGEMENTS

This work is supported by Jordan Whitney and Michael Everman for their original contributions and ideas regarding delay networks and the Genetic Algorithm, as well as Professor Colby Leider, the director of Music Engineering in the Frost School of Music at the University of Miami, Coral Gables, FL.

8. REFERENCES

[1] M. Puckette and J. Stautner, "Designing Multichannel Reverberators," in Computer Music Journal, vol. 6, no. 1, Cambridge: The MIT Press, 1982, pp. 52-65.
[2] M.A. Gerzon, "Unitary (Energy-Preserving) Multichannel Networks With Feedback," in Electronic Letters, vol. 12, no. 11, Oxford: Mathematical Institute, 1976, pp. 278-279.
[3] Gardner, W.G.; , "The Virtual Acoustic Room," Master's Thesis, MIT Media Lab, Aug. 1992.

[4] Valimaki, V.; Parker, J.D.; Savioja, L.; Smith, J.O.; Abel, J.S.; , "Fifty Years of Artificial Reverberation," Audio, Speech, and Language Processing, IEEE Transactions on , vol.20, no.5, pp.1421-1448, July 2012.

[5] Schroeder, M.R.; "Natural Sounding Artificial Reverberation," Audio Engineering Society Convention 13, Oct. 1961.

[6] R. & S.E. Haupt. *Practical Genetic Algorithms*, Wiley-Interscience; 1 edition, 1998.

[7] Cavalcante, R.L.G.; Yamada, I.; Mulgrew, B.; , "An Adaptive Projected Subgradient Approach to Learning in Diffusion Networks," Signal Processing, IEEE Transactions on , vol.57, no.7, pp.2762-2774, July 2009.

[8] Gardner, W. G.; "A realtime multichannel room simulator." J. Acoust. Soc. Am., 92 (A), 2395, Oct. 1992.

[9] Patalia, T.P.; Kulkarni, G.R.; , "Behavioral analysis of genetic algorithm for function optimization," Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference on , vol., no., pp.1-5, 28-29 Dec. 2010.

[10] Bai, Mingsian R.; Bai, Ganyuan; "Optimal Design and Synthesis of Reverberators with a Fuzzy User Interface for Spatial Audio," JAES Volume 53 Issue 9 pp. 812-825, Sept. 2005.

[11] Meng, Q.C.; Feng, T.J.; Chen, Z.; Zhou, C.J.; Bo, J.H.; , "Genetic algorithms encoding study and a sufficient convergence condition of GAs," Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on , vol.1, no., pp.649-652 vol.1, 1999.

[12] Axon, P.E.; Gilford, C.L.S.; Shorter, D.E.L.; , "Artificial reverberation," Proceedings of the IEE - Part B: Radio and Electronic Engineering , vol.102, no.5, pp.624-640, September 1955.