Schroeder Reverberator

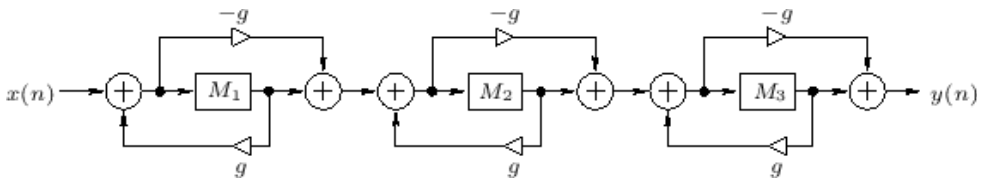
Paul Wittschen
Digital Signal Theory
Final Project Report

Introduction:

For this project, I decided to model and implement a Schroeder reverberator using Matlab. As noted in class, modeling a high quality, digital reverb is quite difficult as there are so many variables and factors to consider, in modeling the behavior of sound and acoustics. Nowadays, reverb is perhaps one of the most commonly used digital effects in the studio and beyond. Having implemented and understood the inner workings of a multitude of common digital effects over this and previous semesters, this common effect has eluded my practical understanding, which is the primary motivation behind this undertaking.

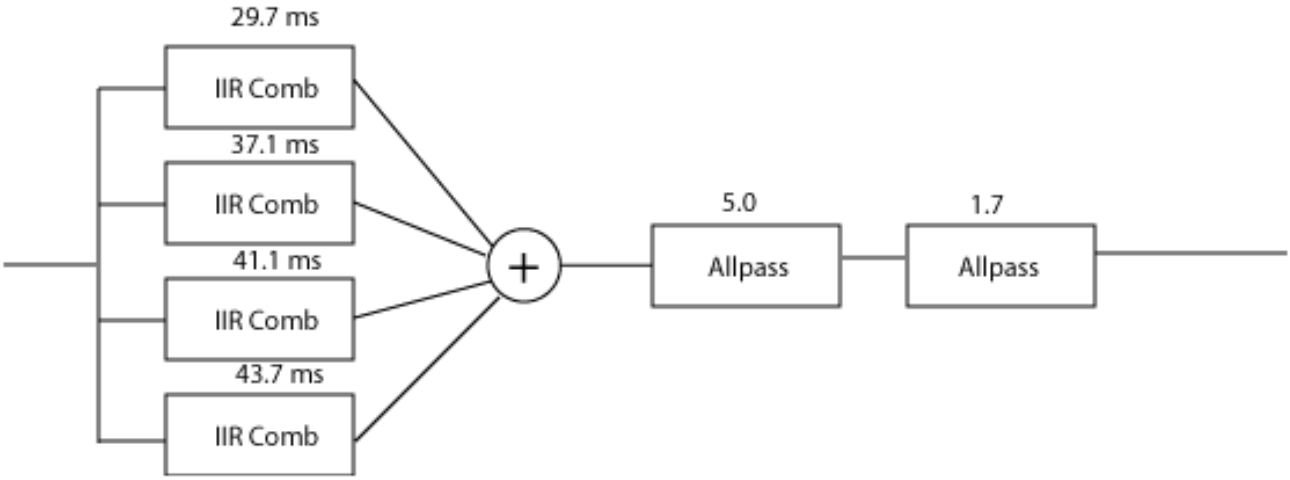
Theoretical Background:

The Schroeder reverberator was developed in the early 1960s by German physicist Manfred Robert Schröder (1926-2009). His first artificial reverb design was described in 1961, in his paper "Colorless Artificial Reverberation", which revealed his use of a series of all-pass filters:



[1]

This design however would be soon improved upon in his more known reverb deign. At the core of the Schroeder reverberator is the use of multiple IIR comb-filters and all-pass filters. The primary idea was to implement the IIR filters for their resulting exponential decay, and the all-pass filters to create a sufficient density of reflections needed for a realistic reverberation (approximately 1000/sec). Thus, the IIR comb-filters would contribute to the reverb length, while the all-pass filters would contribute to the reverb intensity. Schroeder observed the need to use reverb times which were prime numbers. The reasoning behind this was to prevent delay times which were multiples of each other, as these would add together at period points, and in effect, amplify the delayed signal [1]. Instead, a reverb with a flat response is desired. Schroeder's digital reverb first splits the signal into a set of parallel IIR comb-filters. Upon filtering, these signals are then summed back together and passed through a series of all-pass filters, which are not to be put into a parallel configuration, as this would cause phase issues upon output. In Schroeder's first implementation of his reverb design, he used 4 parallel IIR comb-filters, and 2 all-pass filters:



Potential Problems:

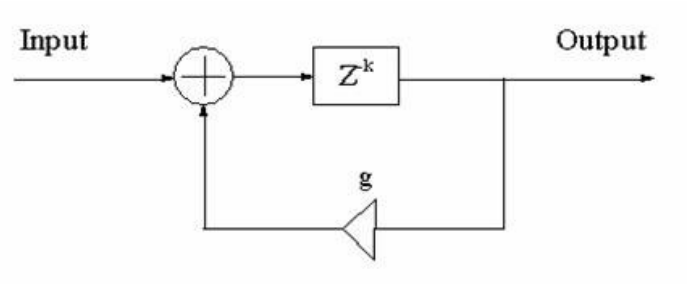
- As briefly noted, this reverb is susceptible to spectral coloration, by overlapping and recurring delays. Again, this can be minimized by setting the delay times on the filters to be prime numbers, or indivisible by one another.

- The produced reverb can be rather periodic in its delay and decay, which is an unnatural feature.

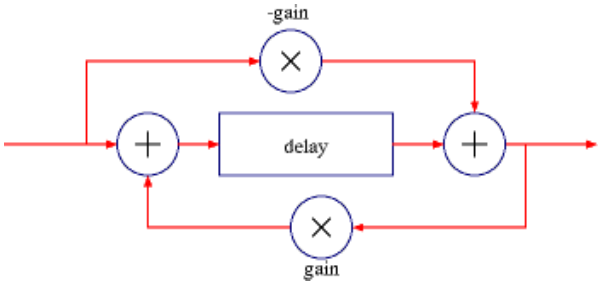
Approach:

In planning the implementation of this digital reverb, I first assessed the various components required. I would require the use of multiple IIR comb-filters and all-pass filters. Additionally, I would need to input proper delay times.

IIR Comb-filter:



All-Pass filter:



Delay Times:

IIR Comb-filter: ~30 – 45 ms

All-Pass filter: ~1.7 – 5 ms

Implementation:

The matlab implementation of this reverb proved to be rather straight forward, having implemented IIR comb-filters and all-pass filters individually. In its design, I wanted to leave some room for manipulation; the user is able to define the number of filters, both IIR and all-pass, the delay times, and the feed forward, feed back, and dry signal coefficients. The matlab function is defined as:

schroeder(x,fs,numIIRs,IIRdelay,numAPs,APdelay,ff,fb,bl)

where x is the input signal, fs is the sampling frequency, numIIRs is the quantity of IIR comb-filters, IIR delay is the delay time, numAPs is the quantity of all-pass filters, Apdelay is the all-pass filter delay time, ff, fb, and bl represent the feed back, feed forward, and dry signal

coefficients.

I implemented the IIR comb-filters using the equation:

$$y(n) = (ff * x(n)) + (fb * y(n\text{-}delay))$$

In order to allow the quantity of filters to be variable, I entered the filtering process within a loop. This outer loop would iterate the number of times as the number of filters. The output of each loop iteration would be placed into an empty vector. Upon completion of loop, these filtered outputs would be added together and scaled (divided) by the number of filtered signals there were. Next, this summed and scaled signal would pass through two all-pass filters, using the equation:

$$y(n) = (bl * x(n)) + (ff * x(n\text{-}delay)) + (fb * y(n\text{-}delay))$$

This filter is also set into a loop for a user defined length, representative of the number of all-pass filters. Following this, the signal has completed its course through the reverb.

In an attempt to address the issues of reverb coloration and proportionality, I implemented a means to vary the reverb times for each filter, between both the L and R channels. For each iteration of a new filter, both all-pass and IIR, the delay times for the L and R channels are randomized between the ideal values of 30-45ms. Otherwise, the user may define the delay times for each filter by entering a vector of delay values for each channel, for each filter.

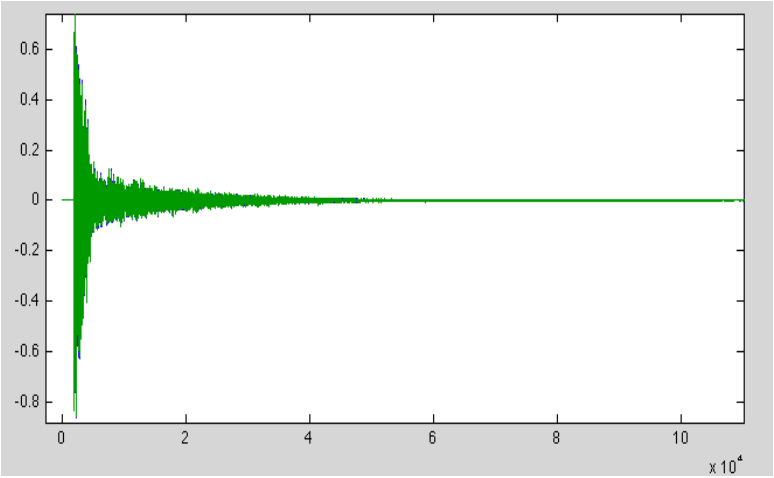
Discussion:

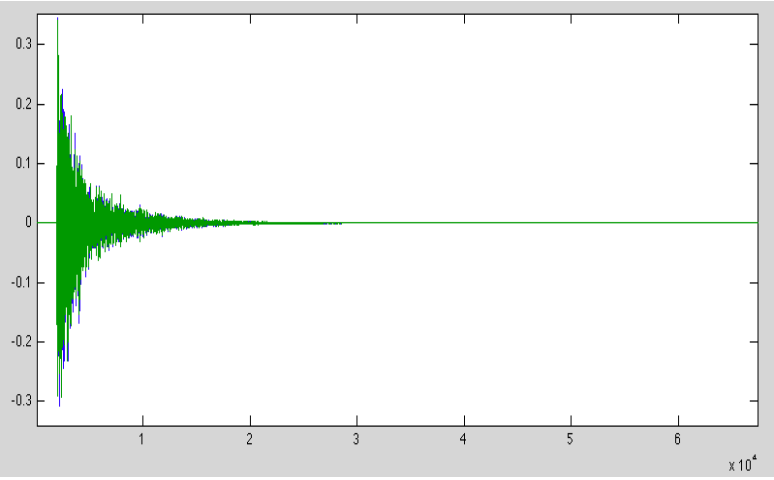
Upon observing the resulting effects of the reverb design, I was pleased to find some processed examples to showcase a somewhat realistic reverb effect. However, many of the examples revealed an obviously artificial component.

Essentially, I found that as the number of filters increased, (both IIR and all-pass), the quality of

the reverb increased (as did the processing time!). On average, I found that I needed to implement at least 32 IIR and 8 all-pass filters to achieve somewhat convincing results which included a smooth delay and decay. For rendered sounds using less than 32-IIR filters, and 8 all-pass filters, the effects of the comb filtering were more apparent and the differences in time between the delays were present. This produced an artificial, 'metallic' sounding reverb.

Here, the waveform of a snare hit is shown, first with 64 IIR / 16 all-pass filters, followed by a processed example with 8 and 4 IIR / 8 all-pass filters.





While it may be difficult to see here, the more filtered 1st example shows a smoother (and longer) decay, while the second, less filtered example shows a rougher decay. This translated into a more natural sounding reverb, versus a less natural, more artificially processed sounding reverb, in which the processes and filtering are noticeable.

Conclusions:

With this implementation of the Schroeder reverberation model, I have demonstrated its basic implementation, using a series of parallel IIR comb filters and a series of all-pass filters to produce a means of artificial yet natural sounding reverb. I have observed that my implementation will produce a more realistic reverb by increasing the number of IIR and all-pass filters. This will produce a more frequent and dense delay of the signal for the initial reverb and following decay, a significant feature of a higher quality reverb.

Future Work:

In an attempt to improve upon this model, I would like to implement a feature by which to produce randomized delay times for the filters which are indivisible with one another, as this is a contributor to spectral coloration in the resulting reverb effect. Additionally, I would like to find other means of optimizing the program to create higher quality reverbs with a lower quantity of IIR and all-pass filters, as efficiency is also an important factor in good reverb design.

References:

[1]https://ccrma.stanford.edu/~jos/pasp/Schroeder_Allpass_Sections.html

[2] http://dspwiki.com/index.php?title=Reverberation