SYMPATHETIC STRING VIBRATION AS A REVERBERATION EFFECT

Ross Penniman,
Music Engineering Department
University of Miami
Coral Gables, FL, USA
rpenniman@umiami.edu

ABSTRACT

This paper explores using models of sympathetic string vibrations as a way of producing a reverb effect. It is based on many existing works of modeling strings using digital wave-guides. Components of the model are reviewed as well as techniques anticipated for extending the model. A MATLAB application is presented with a graphical interface for ease of use. Applications and future work are discussed.

1. INTRODUCTION

Extensive work has been done to produce digital models of vibrating strings. Until recently, this work has focused on simulating plucked or bowed strings on instruments. In [4] a method of simulating the "sustain pedal effect" is developed. This effect is produced by pressing the damper pedal on the piano, which allows all the strings to vibrate freely. This effect significantly alters the sound of the piano, and can also be used for external sounds when one sings or plays into an open piano with the damper pedal held down. The sustain pedal effect frequently sounds like reverberation, which provided the nexus for this work.

The goal of this project is to adapt existing techniques to produce more diverse reverberation effects. First, the techniques used to model stringed instruments are reviewed. Second, a MATLAB application is presented which provides a simple interface to using the algorithm. Finally, applications of the algorithm are examined along with anticipated future work.

2. STRING MODEL COMPONENTS

The algorithm is based on the Karplus-Strong algorithm [1] which was later refined by many others including [2] and [3]. There are essentially four parts to the filter. First is a loss filter (H_L) to simulate losses in the string as well as losses as the wave reflects at either end of the string. Second is an all-pass filter (H_D) which is designed to have a group delay which simulates the dispersion characteristics of the string. Third is a fractional delay filter (H_F) to fine-tune the total delay time of the model. Finally is a simple delay line (z^{-M}) which determines the rough pitch of the string model.

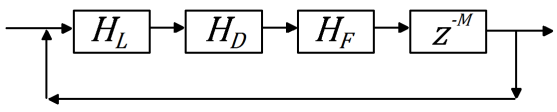

Figure 1: *Flow diagram of single string model.*

2.1. Loss Filter

The loss filter was implemented using a one-pole IIR filter. The technique is described in both [3] and [4]. The transfer function of the filter is given by.

$$H_L(z) = g\frac{1+a}{1+az^{-1}} \qquad (1)$$

$$-1 < a < 0$$

The filter effects greater losses in high frequencies as the pole moves closer to the unit circle. The g term is static gain which simulates damping on the string (requiring $g < 1$). Values for a ranging from 0 to -0.5 were used in this project. A polynomial function to calculate the group delay was found empirically and an accuracy of ± 0.05 samples was achieved.

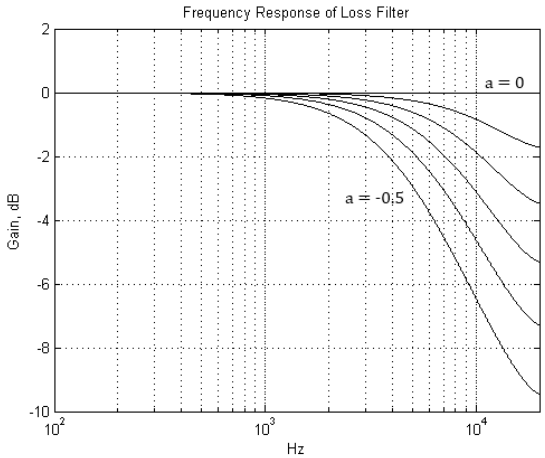

Figure 2: *Frequency response of loss filter.*

2.2. Dispersion

A string which is perfectly flexible is considered a non-dispersive medium, which means that waves travel at the same velocity along the string, regardless of frequency. A string which is "stiff," however, is a dispersive medium. In this case, high frequencies will actually travel faster along the string than low frequencies. The effect is most noticeable in metal strings and has two main implications. First is that an impulse introduced into the string will get "smeared" with respect to time as the high frequencies out-run the low frequencies (see Fig. 3). The second

is that the whole number ratios between the partials of the string will get distorted, and the partials will become inharmonic, sounding more and more like a bell. The distortions in the frequencies of the partials are given by:

$$f_k = k f_0 \sqrt{1 + Bk^2} \tag{2}$$

where f_0 is the fundamental frequency, k is the partial number, and B is the inharmonicity coefficient (derived from properties of the string).

In the case of the digital wave-guide, this is simulated by creating a larger group delay at low frequencies than there is at high-frequencies. As the signal travels around the loop, the high frequencies will go around more times in one second than the low frequencies.

The technique that I used is given in [5]. The idea is to use an arbitrary number of first-order all-pass filters in cascade. The filters all use the same coefficient, which is chosen according to an algorithm described in the paper. The authors show that the resulting group delay better approximates the dispersion of a string as more filters are used in cascade. In my implementation I chose to use 10 filters in cascade to get good results while still remaining computationally efficient. A very accurate method of simulation is given by [6], however this method is significantly more complex to implement.

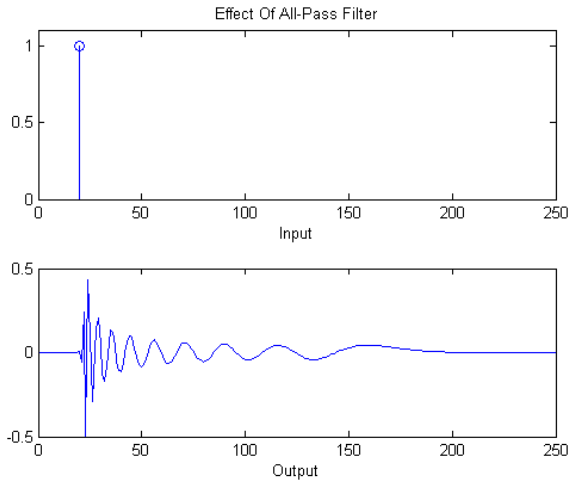

Figure 3: *Time domain smearing of dispersion all pass filter.*

One goal of my implementation was to maintain tight control over the tuning of each string model, and for this I needed exact measurements of the group delay at the fundamental frequency of each string. An equation given by [6] provides a simple method of finding the group delay of a first order filter, which is simply multiplied by the total number of filters in cascade to arrive at the total group delay of the sub-system. The group delay G in samples is given by:

$$G = \frac{1 - \alpha^2}{1 + \alpha^2 + 2\alpha \cos(\omega - \pi)} \tag{3}$$

where α is the coefficient of the first order all-pass filter and is always negative.

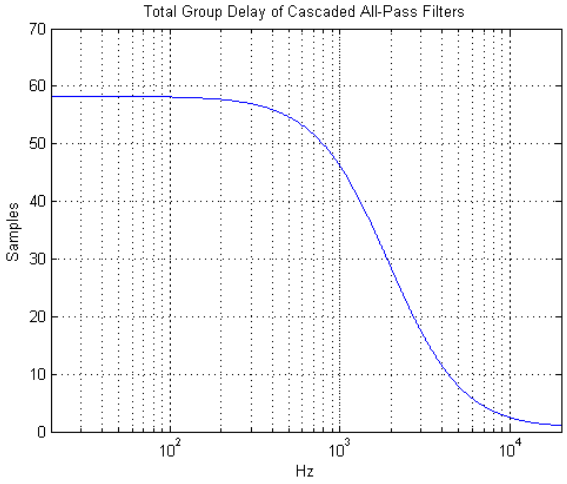

Figure 4: *Total group delay of cascaded all-pass filters.*

2.3. Fractional Delay

The ideal total delay D_i of the system for frequency f_0 is given by the equation:

$$D_i = \frac{f_s}{f_0} \tag{4}$$

where f_s is the sampling frequency.

After subtracting the known group delay from the all-pass filters and the loss filter, we have some amount of delay that must be implemented to achieve the desired final pitch. The whole number part of this value can be implemented by a simple delay line. The fractional part of the delay can be implemented by a number of methods. The two most popular methods are the use of an all-pass filter and a Lagrange interpolator [7]. The all-pass filter is designed using a method developed by Thiran to provide maximally flat group delay near 0 Hz. The downside of this method, however, is large group delay values at higher frequencies. The other popular method, Lagrange interpolation, essentially designs a FIR filter to fit a polynomial to the data points. In a first order implementation, the Lagrange method is simply linear interpolation. The advantage is a more consistent group delay across the entire spectrum, while the disadvantage is a roll-off in the high frequency response, especially for low order filters. In this project a second order Lagrange interpolator was chosen to be the best compromise.

3. TUNING

In selecting what pitches to tune the strings to, there are many options to choose from. If the goal is to be a simulation of freely vibrating strings on a piano, then the most appropriate choice is the notes of the chromatic scale tuned to equal temperament. If the idea is to have strings that vibrate in sympathy with the partials of a note, then it makes more sense to use just intonation for the strings. There are other tuning systems, such as Pythagorean and Meantone tuning which can be implemented as well, depending on what the application is. Tuning systems and their relative strengths and weaknesses are described in detail by [8].

4. BODY MODELING

Simulating the body of an instrument is a similar problem to simulating the acoustic response of a given room. In both cases the system can be considered linear and time-invariant (LTI). The simplest and most accurate approach is to take an impulse response of the system and then convolve it with the input signal. This approach has three main drawbacks. First is that you need access to the instrument (or room) itself in order to measure the impulse response. Second is that there is no way to adjust the parameters of the system, such as its size or frequency response. Third is that convolution is a very computationally expensive process.

A number of strategies to mitigate these problems in existing plucked-string simulations have been proposed ([3], [9] and [10]). The former has pointed out that since both the waveguide and the body model are LTI systems, the order in which they process the signal does not matter. They suggest pre-processing the excitation signal with the impulse response of the body before it is fed into the waveguide. This approach uses more memory for the excitation signal, but saves enormously on the required computation power. Unfortunately this approach cannot be applied in the case of a reverberation effect, as the input (excitation) signal is unknown.

In [9], the authors experiment with several different approaches, all with the goal of decreasing the computational complexity and/or gaining some parametric control over the system. The initial discussion is on the use of banks of FIR or IIR filters which duplicate the impulse response of the system. These filter banks can be implemented with a warped frequency scale to maintain the perceptual effect while reducing the overall complexity. A second approach is to factor out the lowest resonances of the body model and implement them as IIR filters while implementing the remaining response using convolution. The advantage to this approach is that the impulse response is significantly shortened by factoring out the prominent low frequency components, saving in computation. It also allows for some parametric control of the system.

A more recent approach, given by [10], is essentially using an impulse response, but employs methods to reduce the computational load. Their method replaces the low frequency components of the impulse response with Kautz filters, and replaces the high frequency components with a feedback delay network reverberation algorithm. This approach also has the advantage in that parameters can be modified in real time, however the parameters are indirectly related to the properties of the instrument being modelled.

Unfortunately, due to the complexity of body modeling and the time constraint to finish this project, it was not possible to include body modeling as part of the simulation.

5. MATLAB PROGRAM: STREZO

I based my implementation on the techniques described by [3]. As mentioned, I used a set of 10 cascaded first order all pass filters designed using the method described by [5]. Fractional delay was achieved with a second order Lagrange interpolator and the loss filter was implemented with a one-pole IIR filter. I had originally hoped to implement this algorithm as a real-time VST plug-in, however time constraints did not allow that. In order to provide a convenient interface for the user, I developed a Matlab GUI that allows the user to easily select parameters and process a variety of sound sources. The name "Strezo" is derived from "string resonator."

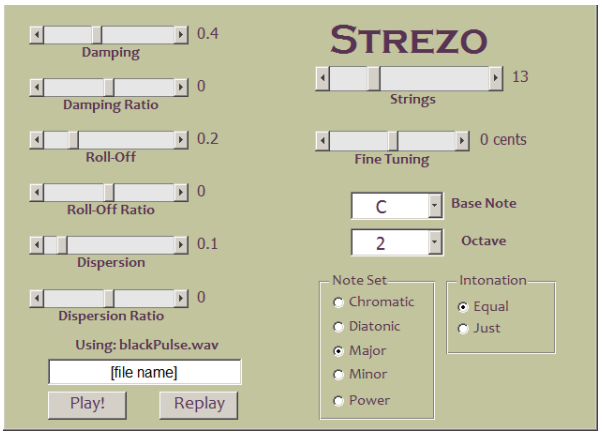

Figure 5: *Screen shot of Strezo.*

5.1. Parameters

5.1.1. Damping: Determines the decay time of each string by changing the fixed gain applied to the signal each time it goes around the digital waveguide. Larger values correspond to a shorter decay time (more damping).

5.1.2. Roll-Off: Determines the frequency dependent decay time by changing the location of the pole in the loss filter. Larger values correspond to faster decay of high frequencies (more muffled sound).

5.1.3. Dispersion: Determines the amount of dispersion that is simulated for each of the strings. Larger values create more inharmonic, bell-like sounds.

5.1.4. Ratio (for each: Damping, Roll-Off, and Dispersion): The ratio control ranges from -1 to 1 and allows the user to scale the parameters according to the relative pitch of each string. Negative ratios give the lower strings a larger parameter value, and positive ratios give the higher strings a larger parameter value. The middle string of the set (or lower-middle) keeps the value chosen by the parent parameter.

5.1.5. Strings: Is the breadth of strings to be simulated, out of which the "Note Set" is chosen. Only the "chromatic" set uses all the strings, the other sets use some subset of the number indicated by "Strings."

5.1.6. Fine Tuning: Allows the user to tune the base note up or down by up to 100 cents (a cent is1/1200th of an octave).

5.1.7. Base Note: Is the lowest note of the set, and the note used as the fundamental when just intonation is selected (choices are the 12 notes of the chromatic scale).

5.1.8. Octave: The octave in which the base note is located (range is 1 – 4).

5.1.9. Note Set: The subset of notes for which string models are created in each octave.

Chromatic: All 12 notes of the chromatic scale
Diatonic: 7 notes of the major scale (Ionian mode)
Major: 3 notes of major triad
Minor: 3 notes of minor triad
Power: Only root and fifth (named for guitar power chords)

5.1.10. Intonation: Either equal temperament or just intonation can be selected.

5.2. Additional Notes

The user can type in the filename of the sound source. The "Play!" button processes the sound source using the selected parameters. The output is 2 seconds longer than the input to allow for decay of the effect. The "Replay" button is a convenience feature to replay the last output without having to redo all of the computation.

Processing time on a modern computer (at the time of writing) is roughly comparable to the length of the output, but is dependent on the parameters selected.

6. APPLICATIONS

While the algorithm presented in Strezo it not intended to be a replacement for general purpose reverberation, it does have some unique properties that could make it useful for musicians and producers. It can significantly alter the timbre of the sound it is processing. It also has the ability to make inharmonic sounds (such as drums) sound more harmonic. One application might be to have the ride cymbal or snare drum resonate at pitches corresponding to the chord progression of the song. It can also be used in more subtle ways to enhance the sound of a piano or guitar that may already have sympathetic vibrations of strings. As a real-time program, the algorithm could be adapted to accept MIDI input from another source to determine the pitches of the string models on the fly.

7. FUTURE WORK

Given the short amount of time available to complete this project there are a number of things that I wanted to include but was not able to. The most significant of these is modeling the body of the instrument. This would provide more realistic simulations of real instruments as well as provide additional means to shape the timbre of the sound.

Another possibility that would likely make the sound more rich and realistic would be to give more attention to modeling the interaction between the strings and the bridge. This might include modifying the loss filter to simulate the mechanical impedance of the bridge, or utilizing the feedback delay network technique (normally used for reverberation) to simulate the ways that strings couple together at the bridge.

Using the method of all-pass filter design described by [6] would obtain very accurate modeling of string dispersion characteristics. This would be a minor improvement, however, as by my own observations and those of [3], dispersion has a relatively insignificant contribution to the overall timbre in most cases.

Utilizing "stretched" tuning (as is frequently done on pianos) would also offer a small improvement. Pianos are typically tuned with intervals slightly larger than what is strictly indicated by equal temperament tuning. The reason for this is that the inharmonicity of the strings caused by dispersion makes the notes sound more in tune when octaves have a ratio slightly more than 2:1.

8. CONCLUSION

A program was developed to use the simulation of string vibrations as a reverberation effect. The algorithm was based on existing work, and while limited in scope, shows promise of effectiveness in more complex implementations. The GUI created for the algorithm provides the user with an easy way to process sound samples and experiment with settings. Significantly more work is needed to realize the full potential of this approach.

9. REFERENCES

[1] Karplus, Kevin, and Alex Strong. "Digital Synthesis of Plucked-String and Drum Timbres." *Computer Music J.,* vol. 7, no. 2, pp. 43-55 (1983).

[2] Jaffe, David A., and Julius O. Smith. "Extensions of the Karplus-Strong Plucked-String Algorithm." *Computer Music J.,* vol. 7, no. 2, pp. 56-69 (1983).

[3] Välimäki, Vesa, Jyri Huopaniemi, Matti Karjalainen, and Zoltan Jánosy. "Physical Modelling of Plucked String Instruments with Application to Real-Time Sound Synthesis." *J. of the Audio Eng. Soc.,* vol. 44, no. 5, pp. 331-353 (1996).

[4] Lehtonen, Heidi-Maria, Henri Penttinen, Jukka Rauhala, and Vesa Välimäki. "Analysis and modeling of piano sustain-pedal effects." *The J. of the Acoustical Soc. of Amer., vol.* 122, no. 3, pp. 1787-1797 (2007).

[5] Rauhala, Jukka, and Vesa Välimäki. "Dispersion modeling in waveguide piano synthesis using tunable allpass filters." in *Proc. Digital Audio Effects (DAFx-06),* Montreal, Quebec, Canada, pp. 71-76 (2006).

[6] Abel, Jonathan S., and Julius O. Smith. "Robust design of very high-order allpass dispersion filters." In *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06),* Montreal, Quebec, Canada, pp. 13-18 (2006).

[7] Laakso, Timo, Vesa Välimäki, Matti Karjalainen, and Unto Laine. "Splitting the unit delay, tools for fractional delay filter design." *IEEE Signal Processing Magazine,* vol. 13, no. 1, pp. 30-60 (1996).

[8] Rossing, Thomas, F. Richard Moore, and Paul Wheeler, "Musical Scales and Temperament," in *The Science of Sound,* 3rd ed. San Francisco, CA: Addison Wesley, 2002, ch. 9, pp. 174-188.

[9] Karjalainen, Matti, and Julius O. Smith. "Body modeling techniques for string instrument synthesis." in *Proc. Int. Computer Music Conf.,* Hong Kong, pp. 232-239 (1996).

[10] Penttinen, Henri, Matti Karjalainen, Tuomas Paatero, and Hanna Järveläinen. "New techniques to model reverberant instrument body responses." in *Proc. Int. Computer Music Conf.,* Havana, Cuba, pp. 182-185 (2001).