Plucked String "Nothing Else Matters" using Karplus-Strong Synthesis

Julian Urrego
Audio Engineering
University of Miami
Coral Gables, USA

***Abstract* – With the advent of physical modeling, the implementation of instrument synthesis has become easier. Ironically, the KS algorithm was discovered as a simple computational technique that seemingly had nothing to do with physics [4]. It was Julius Smith and David Jaffe who first realized the potential of the KS algorithm and its relation to the physics of the plucked string. With the KS filter an ordinary impulse response can be turned into what sounds like a plucked string. This project aims to demonstrate the ease of use of the KS filter and how it can be augmented to play a song. The song in question is Metallica's 1991 *Nothing Else Matters*. With its beautiful introduction displaying the intricate and romantic nature of the guitar, the song is perfect for KS synthesis. This project will use the knowledge gained in Digital Audio I, particularly Lab 6 that described the basics of KS synthesis in Matlab. To create each guitar note in the song, a noise burst must be filtered through a KS filter which uses the combination of a low pass filter and an allpass filter. Each note must then be put in the proper order in an array. Chords can be made by adding notes together. The result will be the emulation of the rhythm and pitch of the introduction to *Nothing Else Matters*. The findings imply that KS filtering is an essential part of digital music. In today's world, string instruments can be synthesized extremely accurately and be used to play any piece.**

Keywords–digital audio, filter, Karplus-Strong, plucked string

I. INTRODUCTION TO PLUCKED-STRING FILTERS

The comb filter is a versatile building block for creating many different sounds. This is exactly the basic filter used in creating a Karplus-Strong filter. As stated by Ken Steiglitz in his book *A Digital Signal Processing Primer* the behavior of a comb filter is very similar to a standing wave [3]. This provides a useful approach, but it doesn't provide the complete answer. Standing waves do not sound like plucked strings because standing waves do not change in frequency. The great thing about plucked strings is that they decay and don't sound forever. Music would be boring if all there was were perfectly periodic signals to be used. Another method for producing plucked-string synthesis is called wavetable-synthesis. The wavetable-synthesis technique is very simple but rather dull musically, since it produces purely periodic tones [1]. As mentioned, plucked strings also don't vibrate forever. Comb filters luckily replicate this behavior but there is still a critical factor missing: the insight that the different frequency components produced by a vibrating string decay at

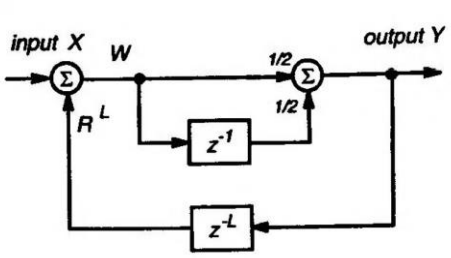
different rates [3]. High frequencies are known to decay at a faster rate than low frequencies. In order to take this effect into account, Karplus and Strong suggested inserting a lowpass filter in the feedback loop of a comb filter so that every time the past output signal returns, its high-frequency components are diminished relative to its low-frequency components [3]. Steiglitz proposes that the following lowpass filter works very well:

$$Y_t = ½ [x_t + x_{t-1}]$$

The magnitude response $|H(\omega)| = |\cos(\omega/2)|$ slopes gently down from unity at the dc frequency to zero at the nyquist frequency. This is useful because we don't want the high frequencies too decay too rapidly. Although this lowpass filter might appear simple, it is all that is needed for a good job in diminishing the high frequencies. Because there is a filter in a feedback loop, the phase response can be affected in a bad way. For the above lowpass filter, the loop delay comes out to be L + ½ and the fundamental frequency is $f_s / (L + ½)$. This delay can be heard by listeners when a large L is used.

II. Implementation of Karplus-Strong Filters

The easiest way to begin implementing a Karplus-Strong filter is to construct a signal flowgraph. The following is a signal flowgraph of the Karplus-Strong filter [3].



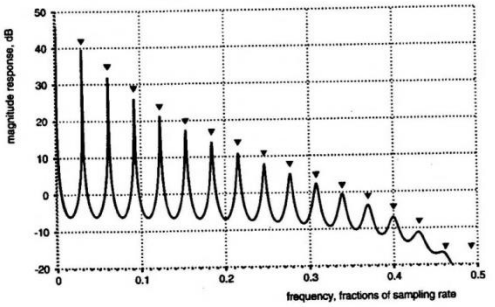
In a general physical model, the loop filter is determined by the cascade of (1) the filtering experienced by a traveling wave in traversing the string twice, and (2) the reflection transfer functions of the two terminations [6]. Now it is possible to determine the difference equation of the output.

III. Frequency Response

Because it is obscure to find the poles of the plucked-filter like the one above, another method must be followed to examine the frequency response. Taking the Z-transform of the output *w* and *y*, and calculating the transfer function

$$H(z) = ½ [1 + z^{-1}] / (1 - R^L z^{-L} ½ [1 + z^{-1}]).$$

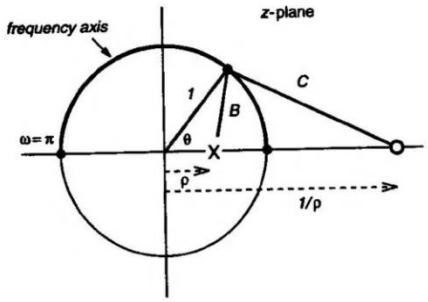
From this transfer function it possible to manually determine the magnitude response by taking the absolute value of $H(\omega)$. The following is the magnitude response of a plucked-string filter with L = 32 and pole Radius R = 0.999 [3].



Steiglitz points out that the resonance peaks increase in width as frequency increases [3]. This is expected as the low-pass filter makes high frequencies decay faster. Another thing to note is the fact that the peaks line up with the expected integer multiples of $f_s/32.5$. Finally we see that the shape of the filter appears to be a combination of a comb filter with a lowpass filter which is exactly what it is. Although this Karplus-Strong filter offers a great sound, it is not the best in terms of control because the pitch is defined as fs/(L + ½). There is no way to specify pitch because a difference of one in L changes the pitch drastically. That is why this project used a more complete Karplus-Strong filter that implemented an allpass filter in addition to a lowpass filter. The next section will talk about the effect of an allpass filter in a Karplus-Strong filter.

IV. ALLPASS FILTERS APPLIED TO KARPLUS-STRONG

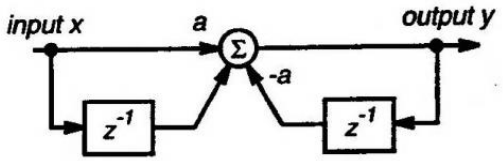
Since there is already a lowpass filter in feedback, it is with care that another filter is designed. The idea is to try to construct a filter that has no effect on the magnitude of phasors, no matter what their frequency [3]. With a pole at some distance ρ from the origin and a zero at 1/ ρ, it is possible to create a magnitude response that is independent of frequency. A filter that passes all frequencies with equal weight is called an allpass filter [3]. To better illustrate the solution, the following is a figure taken from Steiglitz.



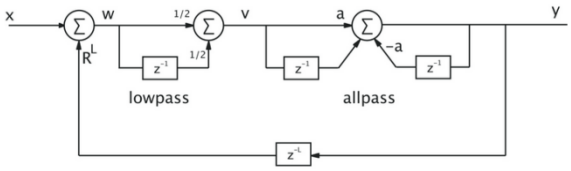
The transfer function of this filter would be:

$$H(z) = z^{-1} + a / (1 + az^{-1})$$

From this point it is possible to construct the signal flowgraph [3].



Now it is possible to put together the lowpass filter in feeback with this allpass filter from above. The result should look like this [4]:



V. PARAMETER DESIGN

Before implementing the above Karplus-Strong filter, a few parameters had to be recognized so that Matlab could figure them out depending on the value of L. Here it is

necessary to introduce δ which is an approximation of the phase delay. The variable a is then defined as:

$$a = 1 - \delta / (1 + \delta).$$

So that δ will always be positive, a must be less than one. Next comes an important equation that is the meat of the function made in Matlab. In order to simulate a plucked string a certain frequency, fo at a certain sampling frequency fs, a useful equation exists:

$$fs / (L + \delta + 0.5) = fo.$$

VI. THE MATLAB PROJECT

A. *Transfer Function*

The first step to accomplishing the goal of playing *Nothing Else Matters* by Metallica using Karplus-Strong synthesized plucked strings is creating a function file in Matlab. This function will be what is used to filter a noise burst through a Karplus-Strong filter. Before getting to code, it is necessary to derive the transfer function of the above signal flowgraph. The best way is to first list the difference equations of the three outpust w, v, and y.

$$w(t) = x(t) + R^L y(t-L)$$

$$v(t) = \tfrac{1}{2} w(t) + \tfrac{1}{2} w(t-1)$$

$$y(t) = av(t) + v(t-1) - ay(t-1)$$

The next step is to take the Z-Transform of each equation.

$$W(z) = X(z) + Y(z)\, z^{-L} R^L$$

$$V(z) = \tfrac{1}{2} W(z) + \tfrac{1}{2} W(z)\, z^{-1}$$

$$Y(z) = aV(z) + V(z)\, z^{-1} - aY(z)\, z^{-1}$$

Now we can use algebra to substitute and come up with the all-important transfer function:

$$H(z) = .5a + (.5 + .5a)\, z^{-1} + .5\, z^{-2} / 1 + az^{-1} - .5aR^L z^{-L} - (.5R^L + .5aR^L)\, z^{-(L+1)} - .5R^L z^{-(L+2)}.$$

B. *String Function*

Now with the transfer function in hand, the Matlab function can be created. The function, called String, takes in 3 parameters: the frequency desired to make a plucked string from, the sampling frequency, and the input that will be filtered. The function String ouputs the newly filtered signal as well as the coefficients of the numerator B, and the coefficients of the denominator A. These coefficients are what characterize the filter and are used inside the function to filter. Because the coefficients of the denominator have powers of L, the function has to have a way to be smart and know what order the denominator is. To compensate an array of L-2 zeros are inserted after the 2nd coefficient and before the 3rd coefficient. To calculate the value of L and ensure that delta is positive and less than zero, the String function uses the equation for fo from above and declares L as the truncated value. Delta is then assigned the value that was truncated. This way, delta is always going to be less than one and L is always going to be an integer. To get the value for a, the function just used the value for delta to plug into the equation for a mentioned in section V.

C. *Main Code*

Once the String function was completed, it was only a matter of

implementing it correctly. The sampling frequency used was 44100 Hz. Because the introduction to *Nothing Else Matters* is made up of quarter, eighth, sixteenth, triplets, dotted quarter, and dotted half notes, there had to be six different times initiated. The quarter note was set to 45 milliseconds and the rest of the notes were ratios of the quarter note. For example, an eighth note would be half of 45 milliseconds. Once there were times set, the next step involved creating the noise bursts that would be filtered using the String function. The noise bursts were created by making an array for each length of note. The arrays were composed of a randomly generated sequence of numbers between 0 and 1 and then a trail of zeros that equaled the length of the time respective to the type of note. With the help of guitar tablature for the song, the frequencies for each note was found out and then initialized in the code. Next it was time to create the plucked strings for each frequency. Because some notes were used as various lengths, it was necessary to filter the frequencies more than once using a different noise burst. To create a plucked string, all that needed to be done was use the String function with the frequency of the note, the sampling frequency, and the type of note as the inputs and the filtered signal, B, and A as outputs. B and A are primarily for plotting the magnitude spectrum. To make the chords that are in the song, the sum of the respective notes were taken. The length of the chords depended on the length of the notes summed. When it came to creating the actual song, an array of the notes and chords was constructed. Using the guitar tab as a guide, it was a matter of finding the right note and putting it in the right order. Care had to be taken to arrange the notes in the right order and with the correct lengths. Once the array was completed, it had to be normalized so that there would be no clipping. Finally, the song was ready to be played and saved as a wav file.

VII. CONCLUSION

Throughout the MMI 502 course and labs I gained the knowledge to complete this lab. Using filter design techniques we learned in class, I was able to create a function that served as a Karplus-Strong filter. With that filter, I was able to make individual notes of different times and sequence them in a way that resulted in the playback of *Nothing Else Matters* by Metallica. The playback worked well and was very satisfying and beautiful to listen to.

VIII. REFERENCES

[1] K. Karplus and A. Strong, "Digital Synthesis of Plucked-String and Drum Timbres," *Computer Music Journal,* vol. 7, no. 2, pp. 43-55, 1983.

[3] M. Karjalainen, "Efficiency issues in model-based synthesis of plucked string instruments," *Journal of the Acoustical Society of America,* vol. 100, no. 4, p. 1, 1996.

[3] K. Steiglitz, A Digital Signal Processing Primer, Addison-Wesley Publishing Company, Inc., 1996.

[4] M. Karjalainen, V. Välimäki and T. Tolonen, "Plucked-String Models: From the Karplus-Strong Algorithm to Digital Waveguides and beyond," *Computer Music Journal,* vol. 22,

no. 3, pp. 17-32, 1998.

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

[6] J. O. Smith, "Efficient synthesis of stringed musical instruments.," pp. 64-71, 1993.

[7] J. O. Smith, "J.O. Smith III Comments on Sullivan Karplus-Strong Article," *Computer Music Journal,* vol. 15, no. 2, pp. 10-11, 1991.