Karplus-Strong Plucked String Algorithm with Improved Pitch Accuracy*

Ed Doering

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

Abstract

The basic Karplus-Strong plucked string algorithm must be modified with a continuously adjustable loop delay to produce an arbitrary pitch with high accuracy. An all-pass filter provides a continuously-adjustable fractional delay, and is an ideal device to insert into the closed loop. The delay characteristics of both the lowpass and all-pass filters are explored, and the modified digital filter coefficients are derived. The filter is then implemented as a LabVIEW "virtual musical instrument" (VMI) to be played from a MIDI file using "MIDI JamSession."

	This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide[1] module for tutorials and documentation that will help you:
	• A Jump to LabVIEW to Audio Signal Processing
	• Get started with LabVIEW
	• Obtain a fully-functional evaluation edition of LabVIEW

Table 1

1 Introduction

In the project activity of the prerequisite module Karplus-Strong Plucked String Algorithm[2], you undoubtedly noticed that the pitch accuracy of the basic Karplus-Strong algorithm needs improvement. For example, listen to the short MIDI test sequence ksdemo.mid[3] rendered to audio with the basic algorithm using a sampling frequency of 20 kHz: ksdemo_20kHz.wav[4] . The individual notes sound reasonable, but when the notes are played simultaneously as a chord, the pitch inaccuracy becomes noticeable. The accuracy gets worse

*Version 1.2: Mar 18, 2008 9:46 am -0500

†http://creativecommons.org/licenses/by/2.0/

[1]"NI LabVIEW Getting Started FAQ" <http://cnx.org/content/m15428/latest/>

[2]"Karplus-Strong Plucked String Algorithm" <http://cnx.org/content/m15489/latest/>

[3]http://cnx.org/content/m15490/latest/ksdemo.mid

[4]http://cnx.org/content/m15490/latest/ksdemo_20kHz.wav

at lower sampling frequencies such as 10 kHz: ksdemo_10kHz.wav[5] ; increasing the sampling frequency improves the accuracy, as at 44.1 kHz: ksdemo_44kHz.wav[6] , however, a discerning ear can still detect some problems.

The pitch of the plucked string tone is determined by the loop time, which must be made continuously variable in order to precisely control the pitch. In the basic algorithm, the length of the delay line determines the loop time, and the delay line can only be varied in integer amounts. The **all-pass filter** will be introduced and studied in this module as a means to introduce an adjustable fractional delay into the loop.

As a preview of the results that you can achieve, consider the same MIDI test sequence rendered to audio using the techniques introduced in this section: ks2demo_10kHz.wav[7] and ks2demo_20kHz.wav[8] .

2 Lowpass Filter Delay

In the prerequisite module[9], the loop time was determined to be the product of the delay line length and the sampling interval. The reciprocal of the loop time is the pitch (frequency) of the output signal f_0:

$$f_0 = \frac{f_S}{N} \tag{1}$$

where f_S is the sampling frequency in Hz and N is the length of the delay line in samples. Because the delay line length must be an integer number of samples, the pitch cannot be set arbitrarily.

Try the following exercises to explore this concept in more detail.

Exercise 1 ***(Solution on p. 7.)***

The sampling frequency is 40.00 kHz, and the length of the delay line is 40 samples. What is the pitch of the output signal? If the delay line length is decreased by one sample, what is the new pitch?

Exercise 2 ***(Solution on p. 7.)***

The sampling frequency is 10.00 kHz, and the length of the delay line is 10 samples. What is the pitch of the output signal? If the delay line length is decreased by one sample, what is the new pitch?

For each of the two exercises, the first pitch is exactly the same, i.e., 1000 Hz. However, the change in pitch caused by decreasing the delay line by only one sample is substantial (1026 Hz compared to 1111 Hz). But how perceptible is this difference? The module Musical Intervals and the Equal-Tempered Scale[10] includes a LabVIEW interactive front panel that displays the frequency of each key on a standard 88-key piano. Pitch C6 is 1046 Hz while pitch C♯6 (a half-step higher) is 1109 Hz. These values are similar to the change from 1000 Hz to 1111 Hz caused by altering the delay line length by only one sample, so the change is certainly very audible. The abrupt "jump" in frequency becomes less pronounced at lower pitches where the delay line length is longer.

Flexibility to adjust the overall loop time in a continuous fashion is required to improve pitch accuracy. Moreover, any sources of delay in the loop must be accurately known. So far the delay of the low pass filter has been taken as zero, but in fact the low pass filter introduces a delay of its own.

The Figure 1 screencast video describes first how to calculate the delay of an arbitrary digital filter with transfer function $H(z)$.

[5] http://cnx.org/content/m15490/latest/ksdemo_10kHz.wav
[6] http://cnx.org/content/m15490/latest/ksdemo_44kHz.wav
[7] http://cnx.org/content/m15490/latest/ks2demo_10kHz.wav
[8] http://cnx.org/content/m15490/latest/ks2demo_20kHz.wav
[9] "Karplus-Strong Plucked String Algorithm" <http://cnx.org/content/m15489/latest/>
[10] "[mini-project] Musical intervals and the equal-tempered scale" <http://cnx.org/content/m15440/latest/>

Image not finished

Figure 1: [video] Calculating the delay of a filter given H(z)

In general, therefore, the delay is the negated slope of the filter's phase function, and the delay varies with frequency.

Now, consider the specific low pass filter used in the basic Karplus-Strong algorithm. The filter coefficient "g" will be taken as 0.5, making the filter a true two-point averager:

$$H_{\text{LPF}}(z) = \frac{1 + z^{-1}}{2} \tag{2}$$

The Figure 2 screencast video continues the discussion by deriving the delay of the low pass filter of (2). Several techniques for working with complex numbers in LabVIEW are presented and used to visualize the magnitude and phase response of the filter.

Image not finished

Figure 2: [video] Calculating the delay of the low pass filter

Because the delay of the low pass filter is always 1/2, the pitch may be expressed more precisely as

$$f_0 = \frac{f_S}{N + \frac{1}{2}} \tag{3}$$

While this result more accurately calculates the pitch, it does nothing to address the frequency resolution problem.

3 All-Pass Filter Delay

Now, consider the **all-pass filter** (**APF**) as a means to introduce a variable and fractional delay into the loop. The all-pass filter has a unit magnitude response over all frequencies, so it does not "color" the signal passing through. However, the all-pass filter has a phase response that is approximately linear for all but the highest frequencies, so it introduces an approximately constant delay. Even better, the slope of the phase response is continuously variable, making it possible to adjust the delay as needed between 0 and 1 samples.

The all-pass filter transfer function is

$$H_{\text{APF}}(z) = \frac{C + z^{-1}}{1 + Cz^{-1}} \tag{4}$$

where $|C| < 1$ to ensure stability.

The Figure 3 screencast video continues the discussion by exploring the delay of the all-pass filter of (4) as a function of the parameter C.

Image not finished

Figure 3: [video] Calculating the delay of the low pass filter

4 Implementing the Pitch-Accurate Algorithm

Including the all-pass filter in the basic Karplus-Strong algorithm allows the loop time to be set to an arbitrary value, making it possible to sound a tone with any desired pitch.

This section guides you through the necessary steps to augment the basic algorithm with an all-pass filter, including the derivation of necessary equations to calculate the delay line length and the fractional delay. Work through the derivations requested by each of the exercises.

To begin, the pitch of the output signal is the sampling frequency f_S divided by the total loop delay in samples:

$$f_0 = \frac{f_S}{N + \frac{1}{2} + \Delta} \tag{5}$$

where Δ is the fractional delay introduced by the all-pass filter.

Exercise 3 ***(Solution on p. 7.)***

Refer to (5). Derive a pair of equations that can be used to calculate the length of the delay line N and the value of the fractional delay Δ.

The all-pass filter delay can be approximated by (6) (see Moore):

$$\Delta = \frac{1 - C}{1 + C} \tag{6}$$

Exercise 4 ***(Solution on p. 7.)***

Refer to (6). Solve the equation for the all-pass filter coefficient C in terms of the required fractional delay.

The all-pass filter can be inserted at any point in the loop; Figure 4 shows the all-pass filter placed after the low pass filter.

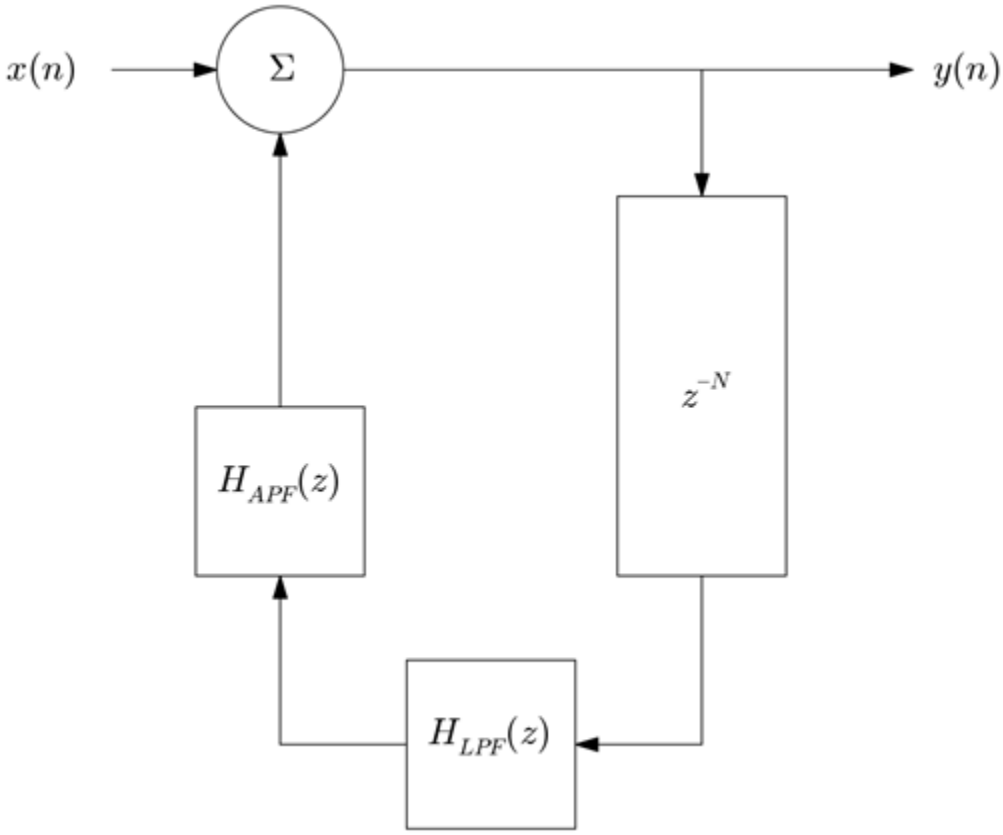

Figure 4: Block diagram of the pitch-accurate Karplus-Strong algorithm

Exercise 5 ***(Solution on p. 7.)***

To simplify the derivation of the overall filter transfer function that relates y(n) to x(n), consider the three feedback elements (delay line, low pass filter, and all-pass filter) to be a single element with transfer function $G(z)$. Derive the transfer function of the combined element.

Exercise 6 ***(Solution on p. 7.)***

Refer to the block diagram of Figure 4. Considering that all three elements are represented by a single feedback element G(z), derive the overall transfer function H(z) for the digital filter in terms of G(z).

Recall the low pass filter transfer function ((7)):

$$H_{LPF}(z) = g\left(1 + z^{-1}\right) \tag{7}$$

The all-pass filter transfer function is described by (4).

Exercise 7 ***(Solution on p. 7.)***

Derive the overall transfer function H(z) in terms of the filter parameters g and C. Write the transfer function in standard form as the ratio of two polynomials in z.

5 Project Activity: Karplus-Strong VMI

As in the prerequisite module[11], convert the pitch-accurate Karplus-Strong algorithm into a **virtual musical instrument** (**VMI**) that can be played by "MIDI Jam Session." If necessary, visit MIDI JamSession[12], download the application VI .zip file, and view the screencast video in that module to learn more about the application and how to create your own virtual musical instrument. Your VMI will accept parameters that specify frequency, amplitude, and duration of a single note, and will produce a corresponding array of audio samples using the Karplus-Strong algorithm described in the previous section.

For best results, select a MIDI music file that contains a solo instrument or perhaps a duet. For example, try "Sonata in A Minor for Cello and Bass Continuo" by Antonio Vivaldi. A MIDI version of the sonata is available at the Classical Guitar MIDI Archives[13] , specifically Vivaldi_Sonata_Cello_Bass.mid[14] .

6 References

- Moore, F.R., "Elements of Computer Music," Prentice-Hall, 1990, ISBN 0-13-252552-6.
- Karplus, K., and A. Strong, "Digital Synthesis of Plucked String and Drum Timbres," Computer Music Journal 7(2): 43-55, 1983.

[11] "Karplus-Strong Plucked String Algorithm" <http://cnx.org/content/m15489/latest/>
[12] "[LabVIEW application] MIDI_JamSession" <http://cnx.org/content/m15053/latest/>
[13] http://www.classicalguitarmidi.com
[14] http://www.classicalguitarmidi.com/subivic/Vivaldi_Sonata_Cello_Bass.mid

Solutions to Exercises in this Module

Solution to Exercise (p. 2)
1000 Hz (40 kHz divided by 40 samples); 1026 Hz
Solution to Exercise (p. 2)
1000 Hz (10 kHz divided by 10 samples); 1111 Hz
Solution to Exercise (p. 4)
$N = \lfloor \frac{f_S}{f_0} - \frac{1}{2} \rfloor$ (the "floor" operator converts the operand to an integer by selecting the largest integer that is less than the operand); $\Delta = \frac{f_S}{f_0} - N$
Solution to Exercise (p. 4)
$C = \frac{1-\Delta}{1+\Delta}$
Solution to Exercise (p. 5)
The elements are in cascade, so the individual transfer functions multiply together: $G(z) = z^{-N} H_{LPF}(z) H_{APF}(z)$
Solution to Exercise (p. 5)
$H(z) = \frac{1}{1-G(z)}$
Solution to Exercise (p. 5)
$H(z) = \frac{1+Cz^{-1}}{1+Cz^{-1}-gCz^{-N}-g(1+C)z^{-(N+1)}-gz^{-(N+2)}}$