Synthetic Guitar

Ivan Selesnick

August 4, 2019

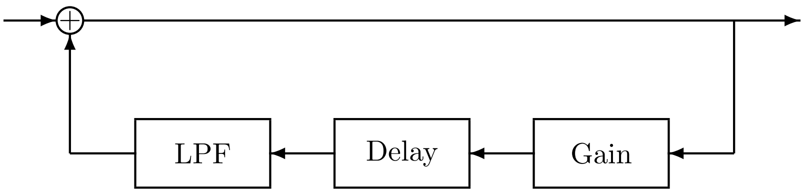
There are several methods for electronically producing the sound of musical instruments. One method is to record the sound, sample it, and save the samples. This method is computationally simple and produces naturalistic sounds, however, it requires a lot of memory and lacks flexibility: For each note, all the samples must be stored in memory. In addition, it is not possible to tune the notes.

A second method to simulate the sound of a musical instrument is to model the sound as a sum of cosine signals. This method requires very little memory — only the frequency and amplitude of each cosine signal. However, this method usually produces somewhat artificial sounds.

A third method is to mathematically model the physics of the instrument using an appropriately designed system. Methods based on physical modeling can sound realistic, need very little memory, are tunable, and can be efficiently implemented. In this lab, we investigate a system for simulating the sound of a plucked string.

K. Karplus and A. Strong described a very simple system for generating a signal which sounds like a plucked string. This system can be used to synthesize the sound of a guitar. They published their method in the paper “Digital Synthesis of Plucked-String and Drum Timbres” in the *Computer Music Journal* in 1983.

The Karplus-Strong system uses this system.



This system contains three components in a feedback loop: a lowpass filter (LPF), a delay, and a gain. We will use the simplest lowpass filter and the simplest delay.

1. The simplest digital lowpass filter is the two-point moving average filter. The transfer function is
$$H(z) = 0.5 + 0.5\,z^{-1}. \tag{1}$$

2. The simplest delay is an integer delay by N samples. The transfer function is
$$D(z) = z^{-N}. \tag{2}$$

3. The transfer function for the gain is
$$G(z) = K. \tag{3}$$
For the system to be stable, it is required that $K < 1$.

Questions

1. What is the transfer function of the Karplus-Strong system pictured above? How many poles does it have?

2. What is the difference equation that implements this system?

3. To implement the system in Matlab with the command

```
y = filter(b,a,x);
```

   What should `a` and `b` be?

To simulate the sound of a plucked string, the input to the system should be an $N + 2$-point random signal. In Matlab, the following command can be used to make the input signal.

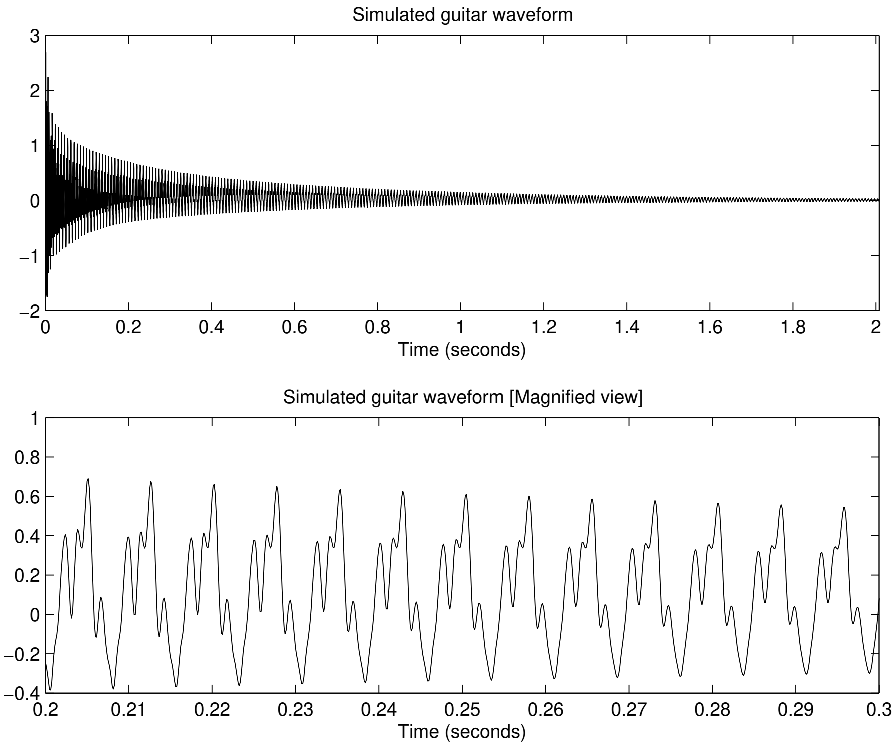
```
x = [randn(1, N) zeros(1, L)];
```

Then when we use `y = filter(b, a, x)` we will get a signal `y` that is $N + L$ points long.

We can then listen to the result with the command `soundsc(y, Fs)`. Where `Fs` is the sampling frequency. For this lab use `Fs = 8000`.

Question: With `Fs = 8000` what should `N+L` be if we want the output signal to be 1 second in duration?

When $N = 100$, $K = 0.98$, the system produces the following output signal. Because the input signal is random for the first N points, the output signal will be a bit different each time this method is used.



In Matlab: Write a Matlab program to simulate the sound of a plucked-string using the Karplus-Strong system above. Use the values $N = 100$, $K = 0.98$. Choose `L` so that the output signal is one second in duration. Use the `filter` command. Listen to the output signal using the `soundsc` command. Does it sound like a plucked string?

Zoom in on the signal (magnify a small portion of it). Does it look almost periodic?

Run your program several times. Does it sound exactly the same each time?

In Matlab: Plot the frequency response of the system using the `freqz` command. To ensure that the plot captures the details of the frequency plot, you can use `freqz` with a third input argument, for example:

```
[H, om] = freqz(b, a, 2^16);
f = om/pi * Fs/2;
plot(f, abs(H))
xlabel(’Frequency (Hz)’)
title(’Frequency response of Karplus-Strong system’)
```

This will give a denser frequency axis grid.

Question: From the frequency response, how could you have predicted that the output signal of the Karplus-Strong system is nearly periodic? (Note: the spectrum of a periodic signal is a line spectrum.)

We can tune the sound by changing the system parameters.

In Matlab: Run the system again, using different values of K. Plot the output signal, listen to the signal, and plot the frequency response. When you change K what effect does it have on the output signal and the frequency response?

In Matlab: Run the system again, using different values of N. Plot the output signal, listen to the signal, and plot the frequency response. When you change N what effect does it have on the output signal and the frequency response?

Note: The fundamental frequency F_o of the sound produced by this system will depend on the sampling frequency F_s, the delay N, and the delay incurred by the lowpass filter. It is

$$F_o = \frac{F_s}{N + d}$$

where d is the delay of the lowpass filter. The two-point averager has a delay of $d = 0.5$.

In Matlab: In the above description, we are driving the system with a burst of noise (that means, the input to the system is a short random signal). Suppose the system is driven by only an impulse. Run the system with the input `x = [1 zeros(1,L)]`. (The output will be just the impulse response of the system.) How does this effect the sound quality of the output signal? Does it sound more or less natural? Take a close look at the output signal, what do you notice?

In Matlab: Suppose the lowpass filter is taken out of the Karplus-Strong system. If the feedback loop of this system contains only the gain K and the delay z^{-N} then it is called a *comb* filter. For the comb filter, what is the difference equation and transfer function? Run the system without the lowpass filter component. Plot the output signal, listen to the signal, and plot the frequency response. How does removing the lowpass filter effect the sound and the frequency response of the system? Why is this system called a comb filter?

Notes:

1. The frequency of the sound produced by the Karplus-Strong system can not be continuously varied. The delay N must be an integer, so only a discrete set of fundamental frequencies F_o can be simulated with this model. To overcome this limitation we can introduce a *fractional delay* in the feedback loop.

2. If you allow the delay in the feedback loop to be a *time-varying* delay, then the frequency of the sound can vary over the duration of the note. Time-varying systems add flexibility.