Plucked String Digital Waveguide Model

REALSIMPLE Project*
Edgar J. Berdahl and Julius O. Smith III
Center for Computer Research in Music and Acoustics (CCRMA), and the
Department of Electrical Engineering
Stanford University
Stanford, CA

Abstract

This laboratory assignment should teach the student how to model the physics of a plucked vibrating string using a digital waveguide. Along the way, examples and animations provide the student with intuition into advanced topics such as sampling and filtering. These not only help the student understand the basic theory and intuition behind digital waveguide modeling, but they also motivate him or her to study further. The prerequisite assignments are the monochord laboratory assignment,[1] the weighted monochord laboratory assignment,[2] the harmonics laboratory assignment,[3] and the traveling waves laboratory assignment.[4]

Contents

*Work supported by the Wallenberg Global Learning Network
[1]http://ccrma.stanford.edu/realsimple/lab_inst/
[2]http://ccrma.stanford.edu/realsimple/weighted_mono/
[3]http://ccrma.stanford.edu/realsimple/harmonics/
[4]http://ccrma.stanford.edu/realsimple/travelingwaves/

1 Introduction

This laboratory assignment should teach the student how to model the physics of a plucked vibrating string using a digital waveguide. First, physical models and their applications are explained. Next, the vibrating string is introduced as an example of a finite-length waveguide with inverting reflections. The traveling wave decomposition of the vibrating string is sampled to derive the digital waveguide model. A lowpass filter is added to the model's feedback loop to increase modeling accuracy.

The student then "plays" the digital waveguide string simulator in `pd`. At first the student is free to adjust conveniently-formulated model parameters. Then, using a second, more elementary `pd` patch, the student adjusts the digital waveguide's parameters directly. This final step ensures that the student understands the operation of the digital waveguide model.

2 Summary Of Objectives

- Explain what *physical models* are and why they are useful.
- Review the behavior of *traveling waves* in a finite-length waveguide with inverting reflections.
- Provide intuition into how *sampling* works.
- Explain how to construct a *digital waveguide vibrating string* simulator.
- Demonstrate the operation of *lowpass filters* by example.
- Explore the behavior of a digital waveguide vibrating string simulator in `pd` by *playing* and *adjusting* the virtual string.
- *Manually calculate parameters* for the digital waveguide to reinforce understanding.

3 Physical Modeling

A physical model is a model where the physics of some thing are used to represent the thing. The thing is usually either a static object, such as a bolt or house, or a system, such as an automobile part or a musical instrument. A physical model has many uses because the physical description is specific:

- The description is so specific that the physics may be simulated by a computer.
- The simulated behavior and actual measured behavior may be compared to determine how accurate the model is. This is known as *model validation* and aids in verifying understanding of the thing's physics.
- Simpler models yield insight into the modeled thing because they may be easily understood.
- More detailed models may be harder to understand, but they should more accurately model the thing's characteristics.

Once the model has been validated, additional insight into the modeled thing may be obtained by asking *"What if?"* questions. For example, consider the following:

- How would the modeled thing behave if it were made out of tin instead of steel?
- How would the modeled thing behave if gravity were $1m/s^2$ instead of $9.8m/s^2$?
- How would a plucked vibrating string look if time were slowed down by a factor of 100? (This would allow one to `see waves propagate with the human eye`[5].)
- How would a plucked vibrating string sound if the `characteristics of one of the string ends were modified`[6]?
- What would a musical instrument model sound like if it were programmed to `play a song`[7]?
- Consider a system with a sensor and an actuator (motor). What would happen if the motor were programmed to `perform an action as a function of the sensor signal`[8]?

4 Traveling Waves In A Vibrating String

Vibrating strings in many instruments, such as guitars and pianos, behave physically approximately like finite-length one-dimensional waveguides with inverting reflections, so we will first study the physics of this idealized case. As you will recall from the traveling waves laboratory assignment[9], the traveling-wave solution for an infinitely-long one-dimensional waveguide is

$$y(t, x) = y_l(t, x) + y_r(t, x) \tag{1}$$

where $y_r(t, x)$ is the right-going traveling wave, $y_l(t, x)$ is the left-going traveling wave, and their sum is $y(t, x)$. For example, if the wave variable is displacement, then $y_l(t, x)$, $y_r(t, x)$, and $y(t, x)$ are displacements and are thus measured in meters.

A vibrating string that is rigidly-terminated at $x = 0$m and $x = L$m, will be subject to boundary conditions that cause traveling waves to be reflected with a sign inversion. This kind of reflection is termed an inverting reflection.

$$y_r(t, 0) = -y_l(t, 0) \tag{2}$$

and

$$y_l(t, L) = -y_r(t, L) \tag{3}$$

Figure 1 shows how the vibrating string behaves when it is initialized at $t = 0$ according to a triangular pluck for $L = 1$m. The top and middle frames show the right-going and left-going traveling waves, respectively. Notice that for each of these two upper frames, the wave traveling off

[5] http://cnx.org/content/m13513/latest/
[6] http://cnx.org/content/m13513/latest/
[7] http://ccrma.stanford.edu/~jos/pasp/Sound_Examples.html
[8] http://ccrma.stanford.edu/realsimple/pidcontrol
[9] http://ccrma.stanford.edu/realsimple/travelingwaves/

of the right side is inverted and arrives on the left side of the other according to (2) and (3). The traveling wave-based behavior is quite simple.

The lowest frame shows the vibrating string's net displacement. This is what you actually see when you watch a string vibrate, although it happens so fast in real life that it looks like a blur. Due to (1), the lowest frame of the animation is the sum of the upper two frames. You can see that its behavior appears to be more complex.

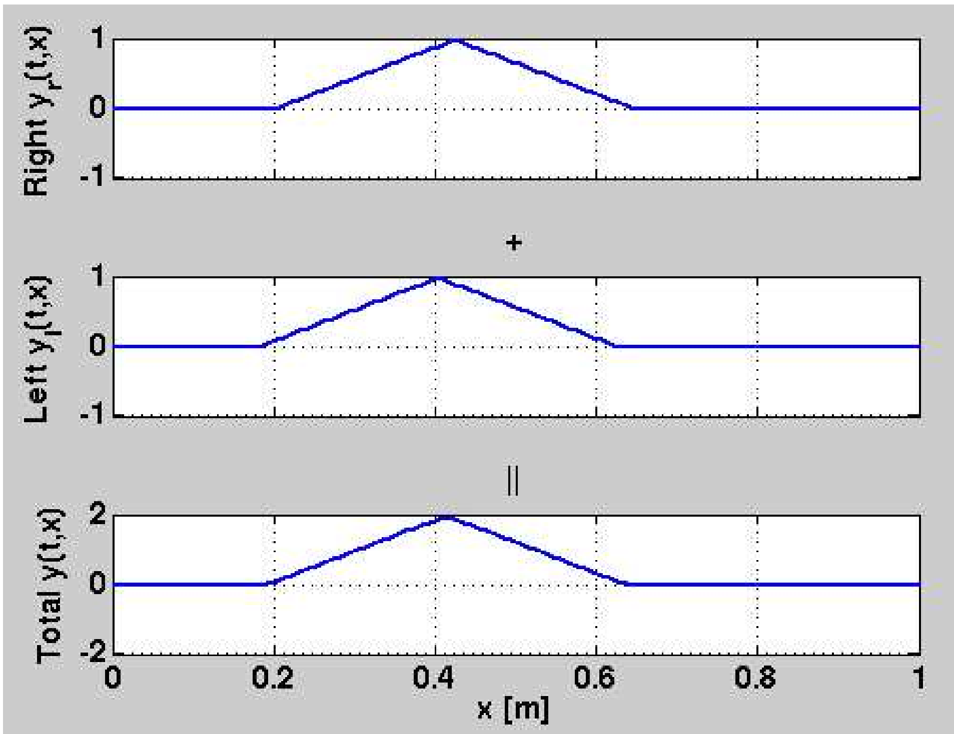

Figure 1: Traveling waves in a vibrating string

This is the first frame from an animation. See the REALSIMPLE website for the animated GIF.[10]

Note that here the waves are represented as functions of time t, where t is a real number. Even over a finite window of time, there are an infinite number of points at which these functions may be evaluated. For example, over a one second window (i.e. for $0 \leq t \leq 1$) at the specific point x_0 along the waveguide, the function $y(t, x_0)$ may be evaluated at an infinite number of points. Try to imagine how one would go about storing the function y in a computer. It seems like the computer would require an infinite amount of memory!

5 Sampled Traveling Waves

We may apply sampling theory to solve this problem practically. According to the Nyquist-Shannon sampling theorem, as long the function $y(t, x_0)$ does not contain any energy above the frequency $f_S/2$, we may represent it exactly using

[10] http://ccrma.stanford.edu/realsimple

$$y(nT, x_0) = y(t, x_0) \tag{4}$$

where n is an integer and $T = 1/f_S$ is the *sampling interval*, which is measured in seconds. In a practical sense, this means that any finite-length signal we can measure can be stored using a finite number of points. This is illustrated in Figure 2 where each circle represents a particular value of n. Note that because the waves do not change too fast (i.e. they do not contain any energy above $\frac{f_S}{2}$Hz), the *sampled* or *digitized* vibrating string representation in Figure 2 behaves analogously to the vibrating string in Figure 1 since they are both initialized with the same triangular pluck.

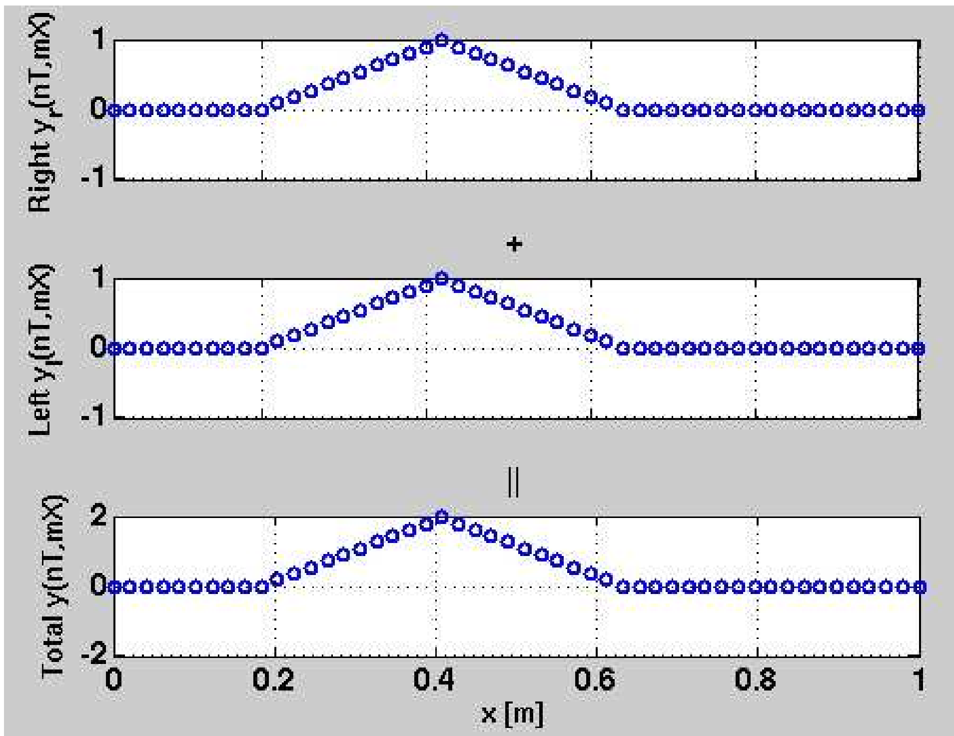

Figure 2: Sampled traveling waves in a vibrating string

This is the first frame from an animation. See the REALSIMPLE website for the animated GIF.[11]

One of the most common sample rates used in audio, which is the sampling rate of compact discs (CDs), is $f_S = 44.1$kHz. According to the Nyquist-Shannon sampling theorem, what is the maximum frequency that audio signals on CDs may represent? Consider how this compares to the upper frequency limit of human hearing, which is about 20kHz.

We have effectively also sampled the wave function with respect to x. We let X be the *spatial sampling interval*, which is the distance that a traveling wave in the waveguide travels during one temporal sampling interval T. Since c measured in m/s is the wave speed in the waveguide, $X = cT$.

[11]http://ccrma.stanford.edu/realsimple

6 Delay Lines And Gain Blocks

From the animations, we see that a digital waveguide model must essentially consist of at least two elements: 1) a delay element that models the time delay seen by traveling waves between the string ends and 2) an element that inverts the sign of the wave as it reflects back from an end. The delay element is known as a delay line, which delays an input signal by a specified number of samples, and is depicted as a wide rectangle in Figure 3. The sign-inverting element may be implemented more generally as a gain block with a gain of -1. In general, a gain block takes a signal as input and outputs a scaled version of the signal. The gain blocks are depicted as triangles in Figure 3, which is a block diagram of the most basic digital waveguide model of a vibrating string.

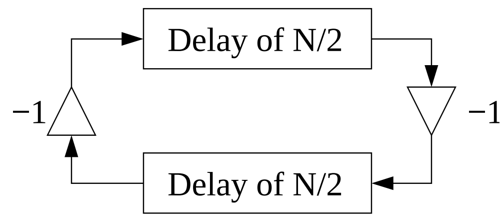

Figure 3: Most basic digital waveguide model of a vibrating string

Note that the delay due to each of the delay lines is $N/2$ samples. This means that the total delay around the feedback loop consisting of the delay lines and gain blocks is N samples, or NT seconds, which is the period of the vibrating string model. The fundamental frequency f_0 of the vibrating string model is the reciprocal of the period, so $f_0 = \frac{1}{NT}$.

This model was used to generate the animation in Figure 2, which never comes to rest. This is okay for the animation since the string's behavior is slowed down so much for visualization purposes. From physical reality, you are aware that vibrating strings eventually come to rest (often after hundreds of periods) until they are excited again via plucking, striking, picking, bowing, etc. This is because traditional physical musical instruments have damping, meaning that various frictional forces eventually suck all of the energy out of the vibrating string.

6.1 Incorporating Damping Into The Model

To incorporate damping into the digital waveguide model, we need to increase one of the gain blocks slightly from -1 to $-g$, where $|g| < 1$. The improved model is depicted in Figure 4.

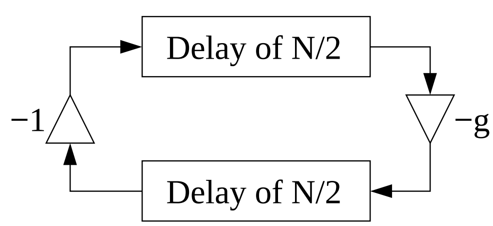

Figure 4: Basic digital waveguide of a vibrating string with damping

Let's now consider the effect of the damping more precisely. Every time a wave travels around the digital waveguide, its amplitude $a(t)$ is scaled by g. After p trips around the waveguide, the amplitude is scaled by g^p. Since each trip lasts NT seconds, the amplitude is scaled by $g^{\frac{\tau}{NT}}$ in τ seconds. If we let τ be the time constant as defined in the monochord laboratory assignment[12], then we have

$$g^{\frac{\tau}{NT}} = e^{-1} \tag{5}$$

and so

$$\tau = \frac{-NT}{ln(g)} \tag{6}$$

Finally, we can say that the amplitude of the waves flowing around the waveguide model $a(t)$ decreases approximately exponentially with time:

$$a(t) = Ae^{-t/\tau} \tag{7}$$

where $a(0) = A$ is the amplitude at time $t = 0$.

6.2 Visualizing The Effects Of *g*

Visualizing the effects of introducing the gain g is slightly trickier because vibrating strings in musical instruments have little damping (i.e. $g \approx 1$ even though $g < 1$). This means that to observe the amplitude decrease, we would need to watch very many periods go by. In order to practically visualize the effects of g, we created a new kind of animation. This animation has one animation frame taken every time delay of N samples, which corresponds to one trip around the digital waveguide and therefore one period[13]. This means that, the difference between two adjacent animation frames corresponds precisely to scaling the contents of the delay lines by g. See Figure 5 for the corresponding visualization of the exponentially-decaying amplitude of the signal flowing around the digital waveguide. In this case, the initial condition is a rectangular pluck rather than a triangular pluck.

This is the first frame from an animation. See the REALSIMPLE website for the animated GIF.[14]

6.3 Incorporating A String Displacement Sensor

Often the sound output of a physical model is associated with only a local measurement made at a point on the simulated instrument. For example, in a simplified sense, the output signal of an electric guitar is the velocity of a small portion of the string directly above the pickup sensor. In our case, we choose to “listen” to the virtual signal defined by the displacement of the string at the position SX meters from the left end of the string. Due to (1), this is the sum of the contents of the delay lines at the respective position. One way to represent this is to split the delay lines and add a summing junction to the block diagram as shown in Figure 6. Note that the total delay around the loop of delay lines is still $S/2 + (N - S)/2 + S/2 + (N - S)/2 = N$ samples.

[12] http://ccrma.stanford.edu/realsimple/lab_inst/

[13] Technically, this is one quasiperiod since the wave is scaled to be slightly smaller after each trip around the waveguide.

[14] http://ccrma.stanford.edu/realsimple

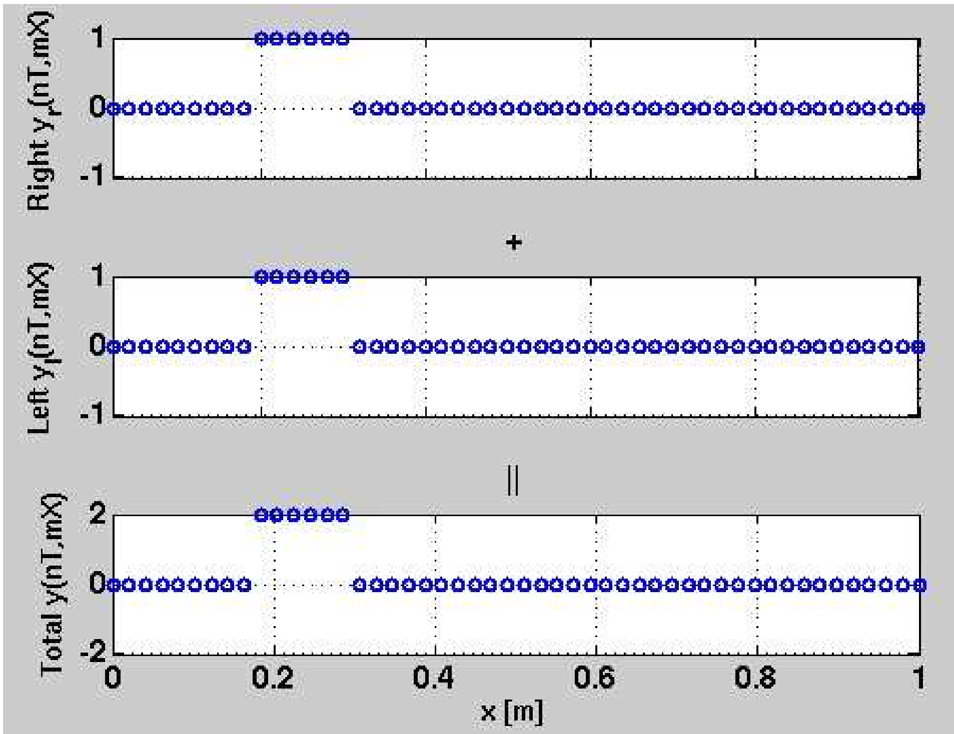

Figure 5: Visualization of a damped vibrating string simulation (one animation frame for every N samples in time)

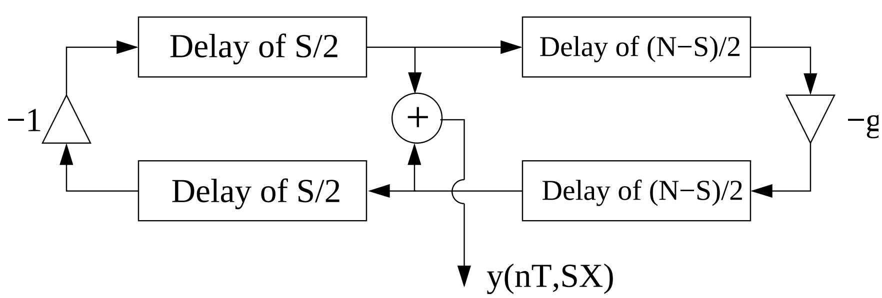

Figure 6: Basic digital waveguide model with a displacement sensor

7 Adding A Lowpass Filter

There is one final weakness of the digital waveguide model that we have developed so far. Energy at higher frequencies in vibrating strings tends to decay more quickly than at lower frequencies. This is true of physical systems in general because any mass vibrating at an infinitely-large frequency would have infinite energy. However, the current model does not implement this decay phenomenon. The simplest way to ensure that high-frequency energy decays quickly is to insert what is called a lowpass filter into the loop. Lowpass filters *pass* more energy at lower frequencies than they do at higher frequencies. In a loose sense, this is equivalent to assigning smaller g's to higher frequencies than to lower frequencies. Figure 7 depicts the finalized basic digital waveguide model that incorporates a lowpass filter.

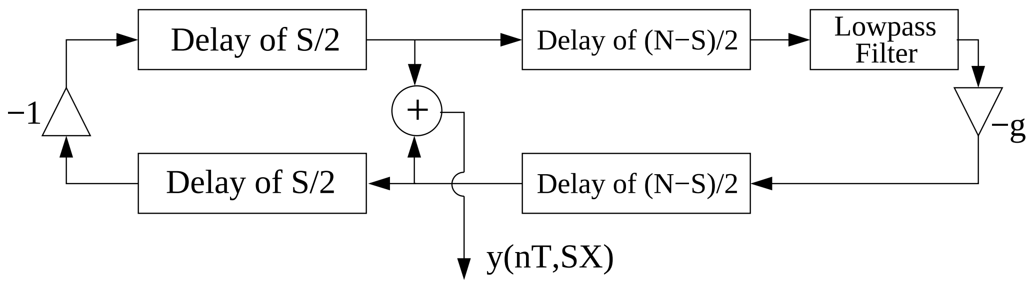

Figure 7: Digital waveguide model with lowpass filter

Sharp edges (or transients) in signals are characterized by energy at high frequencies. Since lowpass filters remove high-frequency energy, they tend to smooth out sharp edges in signals. This is precisely what the lowpass filter in the digital waveguide model does. For instance, if the waveguide is initialized with a rectangular pluck, the rectangular signal not only becomes slightly smaller due to g, but the sharp edges become smoothed out more and more after each trip around the waveguide. The result is visualized in Figure 8 with one animation frame taken every period of N samples.

This is the first frame from an animation. See the REALSIMPLE website for the animated GIF.[15]

This improved digital waveguide physical model more accurately captures the behavior of a vibrating string, and so, as you will experience in the following section, it also produces much more realistic simulated string sounds. Much more information is available on digital waveguide modeling [1].

8 Part 1: Exploration In `pd`

Now that you understand the very basics of digital waveguides, you will have the opportunity to "play" a digital waveguide model of a plucked string in `pd`. This will make it possible to answer some *"What if?"* questions.

1. Download the `pd` patch `3-1.pd`[16], and open it in `pd`.

[15] http://ccrma.stanford.edu/realsimple

[16] http://ccrma.stanford.edu/realsimple/waveguideintro/3-1.pd

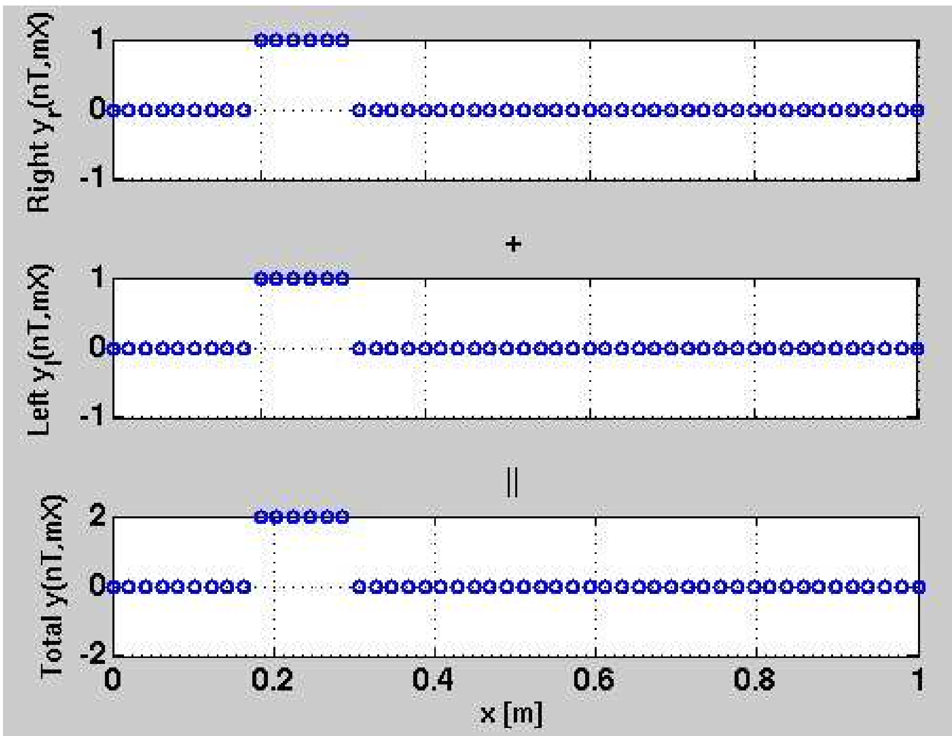

Figure 8: Visualization of a damped vibrating string simulation with lowpass filter (one animation frame per period of N samples in time) where the string is initialized according to a rectangular pluck

2. Ensure that the patch is not in editing mode, and check the "compute audio" box in the main `pd` window.

3. To virtually pluck the string, press any key or click the "PLUCK" button. You may optionally check the box "pluck repeatedly." Adjust the "Master volume" slider on the right until the volume level is comfortable. The patch should now look similar to Figure 9.

4. The five sliders on the upper left calibrate the characteristics of the vibrating string model. Will increasing the length of the string increase or decrease the fundamental frequency? (Consider reviewing the weighted monochord laboratory assignment[17] for more background information.)

5. To check your previous answer, you should increase the length of the modeled string by moving the string length slider to the right. You may either listen to the sound of the model change, observe the spectrum change, or observe the number printed in the "fundamental frequency" box change.

6. Next, adjust the decay time constant $T60$ of the model. Review the definition of τ as described in the monochord laboratory assignment[18]. The audio decay time $T60 = 6.91\tau$ is used here instead because a reasonably loud sound will often be inaudible after it has decayed by 60dB. Verify that the sound of the plucked string model with a shorter $T60$ takes less time to decay. Are you able to observe this on the spectrum below?

7. Now consider the brightness slider. If you have already completed the virtual acoustic guitar laboratory[19], you will also know that more precisely, each harmonic has its own decay time constant. The brightness slider adjusts the relative time constants for the decays of the higher harmonics versus the lower harmonics. Move the brightness slider while looking at the spectrum to determine whether a brighter sound corresponds to the higher harmonics decaying faster or decaying more slowly. Make a note to yourself of what the "bright" plucks sound like.

8. The fourth slider labeled "Sensor position" may be used to move the virtual sensor along the string. Adjust the slider so that it is all of the way to the left. This means that the sensor is almost at the end of the string. For one thing, you will notice that the plucks sound more quiet because the amplitude is smaller. Make sure that you understand why this is the case by looking at plots[20] of the lowest-frequency standing waves that arise in strings. You will also notice that the string model with the sensor at the end sounds either especially bright or especially dark (the opposite of bright). Which is it?

9. What sensor locations cause the sensor to measure no energy at the second harmonic? (Hint: You may observe this effect by viewing the spectrum while adjusting the "Sensor position" slider. The "Pluck repeatedly" toggle switch in the upper right may help.)

[17] http://ccrma.stanford.edu/realsimple/weighted_mono/
[18] http://ccrma.stanford.edu/realsimple/lab_inst/
[19] http://ccrma.stanford.edu/realsimple/vguitar
[20] http://cnx.org/content/m13543/latest

10. *Challenge question:* Derive the formula giving all string sensor locations that will ideally cause the sensor to measure no energy at the nth harmonic for arbitrary n. Thinking about the locations of nodes and/or anti-nodes for standing waves may be helpful.

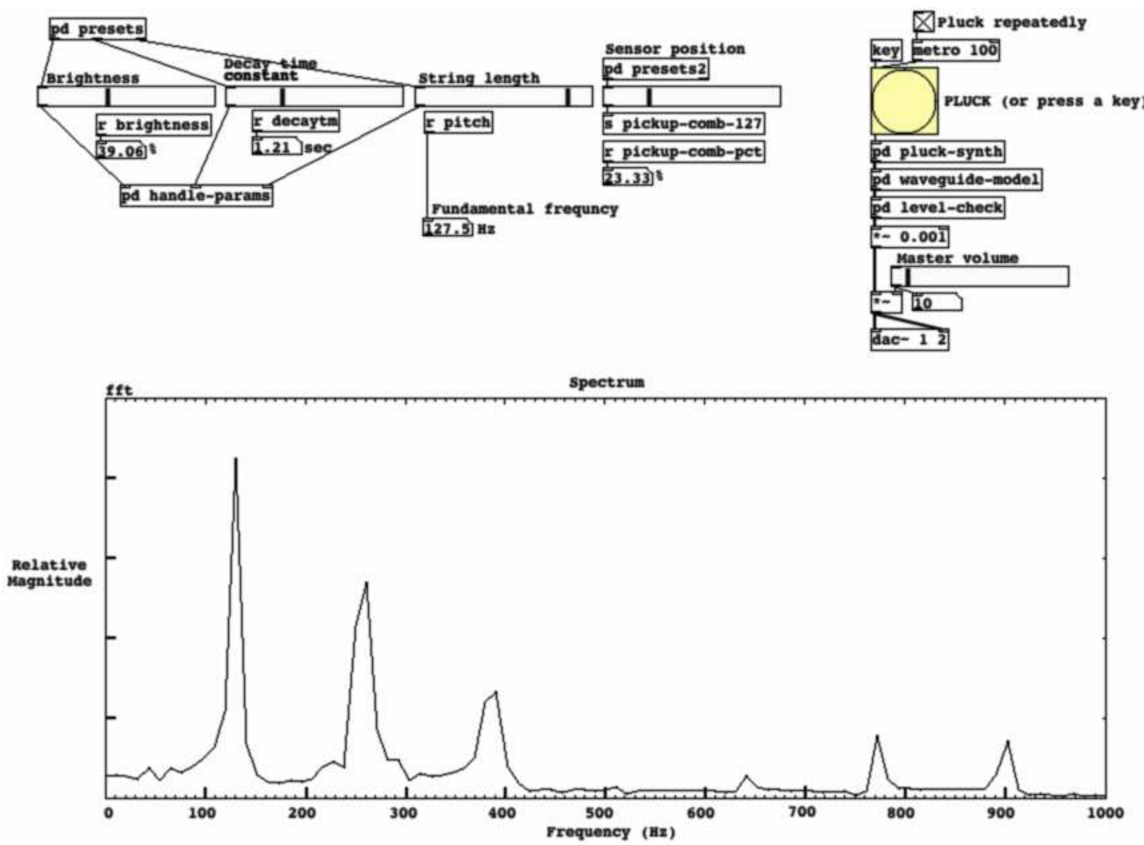

Figure 9: `Pd` patch `3-1.pd`

9 Part 2: Understanding The Underlying Parameters

1. Close `pd` patch `3-1.pd`[21].
2. Download the `pd` patch `3-2.pd`[22], and open it in `pd`.
3. Ensure that the patch is not in editing mode, and check the "compute audio" box in the main `pd` window.
4. This patch is essentially the same as the preceding patch except that you cannot adjust the fundamental frequency and decay time parameters directly. This means that it is harder to obtain vibrating string-like sounds, but by completing this exercise, you will reinforce your understanding of the underlying digital waveguide parameters.

[21] http://ccrma.stanford.edu/realsimple/waveguideintro/3-1.pd
[22] http://ccrma.stanford.edu/realsimple/waveguideintro/3-2.pd

5. To virtually pluck the string, press any key or click the "PLUCK" button. You may optionally check the box "pluck repeatedly." Adjust the "Master volume" slider on the right until the volume level is comfortable. Note that the current sampling rate is displayed on the left.

6. N, the total delay around the loop in samples, is restricted to values between 64 and 2000. 64 corresponds to the smallest delay line size that `pd` typically implements, and the largest value corresponds to a note with very low note. What fundamental frequency corresponds to $N = 64$?

7. What happens to the fundamental frequency if you change the sampling rate f_S? With some sound cards, you may be able to verify your answer by the following:

   - Go to the *Media* menu bar.
   - Select whatever menu item is checked (e.g. portaudio, jack, alsa, oss, or the like).
   - Try entering a new sampling rate. Typical sampling rates that sound cards support are 192kHz, 96kHz, 48kHz, 44.1kHz, 22.05kHz, 11.025kHz, and 8kHz.

8. g, the loop gain, is restricted to $-0.999 \leq g \leq 0.999$. This means that the waveguide is stable and does not have an especially long decay time. Consider values of g such as 0.9, 0.99, and 0.999. What decay time constants $\tau(g = 0.9)$, $\tau(g = 0.99)$, and $\tau(g = 0.999)$ do these loop gains correspond to? Verify that your answer is approximately correct by setting g accordingly in `pd` and listening to how long the decays are.

9. What happens if you make g negative?

10. *Challenge question:* What does negative g mean physically?

Now that you are familiar with the guts of the algorithm, you probably have more respect for the mappings between the sliders and g and N in `3-1.pd`. The mappings for the loop filter are more complicated, and waveguides that more accurately model the physics of musical instruments have even more complex parameter sets [1]. In particular, there is a huge space of parameters, but only a very small part of this space produces sounds reminiscent of the modeled instrument. This brings up one of the fundamental questions in computer music: What are the best mappings between an entire physical parameter space and that portion of the space which is most useful?

References

[1] J. O. Smith, *Physical Audio Signal Processing*, `http://ccrma.stanford.edu/~jos/pasp/`, Aug. 2007, online book.