Physical Modeling using Digital Waveguides

Julius O. Smith III

Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University, Stanford, CA 94305
Internet address: jos@ccrma.stanford.edu

Version (with minor corrections) published in:
Computer Music Journal
special issue on Physical Modeling of Musical Instruments, Part I
Volume 16, no. 4, pp. 74–91, Winter, 1992
This revision: 3/16/93

Introduction

Music synthesis based on a physical model promises the highest quality when it comes to imitating natural instruments. Because the artificial instrument can have the same control parameters as the real instrument, expressivity of control is unbounded.

Historically, physical models have led to prohibitively expensive synthesis algorithms, and commercially available synthesizers do not yet appear to make use of them. These days, most synthesizers use either processed digital recordings ("sampling synthesis") or an abstract algorithm such as Frequency Modulation (FM). However, as computers become faster and cheaper, and as algorithms based on physical models become more efficient, we may expect to hear more from them.

Most attempts to synthesize sounds based on a physical model have been based on numerical integration of the *wave equation* (covered in any textbook on acoustics). These methods generally require at least one operation (multiply and/or addition) for each point on a *grid* which permeates the instrument. In principle, the grid spacing must be less than half a wavelength at the highest audio frequency. This is essentially why the computational costs are so high in "brute force" numerical solutions to the wave equation.

More recently developed "digital waveguide" methods follow a different path to the physical model: the wave equation is first solved in a general way to obtain *traveling waves* in the medium interior. The traveling waves are explicitly simulated in the waveguide model, in contrast to computing a physical variable. (The traveling waves must be summed together to produce a physical output.) In the lossless case, a traveling wave between two points in the medium can be simulated using nothing but a *digital delay line.* In the general linear case, in which there are frequency-dependent losses and dispersion, the *commutativity* of linear time-invariant systems allows the losses and dispersion to be *lumped* at discrete points such that most of the simulation still consists multiply-free delay lines. This is essentially why computational costs are so low in "waveguide synthesis" algorithms.

Computer-music programmers know very well that a delay line can be implemented by a single fetch, store, and pointer update for each sample of output. If the delay line is, say, 500 samples long, (corresponding to a pitch of 44100/500 = 88 Hz in a CD-quality string or bore model), computational requirements relative to "brute force" numerical integration on the grid are reduced by *three orders of magnitude.* As a result, for very simple physical models, several CD-quality voices can be sustained in real time on a single DSP chip costing only a few dollars.

This paper develops waveguide synthesis beginning with the wave equation for vibrating strings. Transverse waves on a string are taken as the primary example due to the relative clarity of the underlying physics, but the formulation for string simulation is unified with that of the acoustic tube. The technique of *lumping* losses at discrete points in the waveguide, replacing more expensive distributed losses, is described.

The Ideal Vibrating String

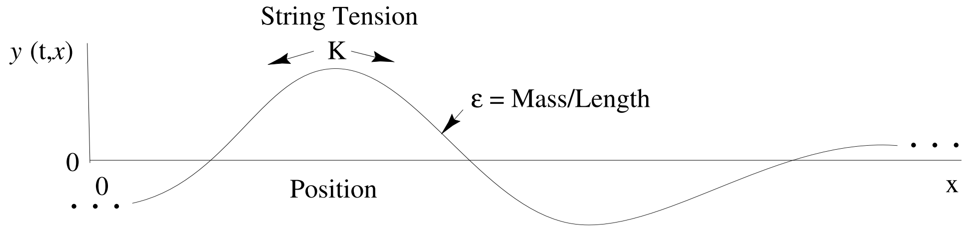


Figure 1. The ideal vibrating string.

The *wave equation* for the ideal (lossless, linear, flexible) vibrating string, depicted in Fig. 1, is given by

$$Ky'' = \epsilon \ddot{y}$$

where

$$\begin{array}{rlrl} K & \triangleq \text{string tension} & y & \triangleq y(t,x) \\ \epsilon & \triangleq \text{linear mass density} & \dot{y} & \triangleq \dfrac{\partial}{\partial t} y(t,x) \\ y & \triangleq \text{string displacement} & y' & \triangleq \dfrac{\partial}{\partial x} y(t,x) \end{array}$$

where "$\triangleq$" means "is defined as." The wave equation is fully derived in (Morse 1936) and in most elementary textbooks on acoustics. It can be interpreted as a statement of Newton's law, *"force = mass × acceleration,"* on a microscopic scale. Since we are concerned with transverse vibrations on the string, the relevant restoring force (per unit length) is given by the string tension times the curvature of the string (Ky''); the restoring force is balanced at all times by the inertial force per unit length of the string which is equal to mass density times transverse acceleration ($\epsilon\ddot{y}$).

The same wave equation applies to any perfectly elastic medium which is displaced along one dimension. For example, the air column of a clarinet or organ pipe can be modeled using the one-dimensional wave equation, substituting air pressure deviation for string displacement, and longitudinal volume velocity of air in the bore for transverse velocity on the string. We refer to the general class of such media as *one-dimensional waveguides.* Extensions to two and three dimensions (and more, for the mathematically curious), are also possible.

Traveling-Wave Solution

It can be readily checked that the wave equation is solved by any string shape which travels to the left or right with speed $c = \sqrt{K/\epsilon}$. If we denote right-going traveling waves by $y_r(x - ct)$ and left-going traveling waves by $y_l(x + ct)$, where y_r and y_l are arbitrary twice-differentiable functions, then the general class of solutions to the lossless, one-dimensional, second-order wave equation can be expressed as

$$y(x,t) = y_r(x - ct) + y_l(x + ct)$$

Note that $\ddot{y}_r = c^2 y_r''$ and $\ddot{y}_l = c^2 y_l''$ which shows that the wave equation is obeyed regardless of the traveling wave shapes y_r and y_l. (But note that the derivation of the wave equation itself assumes the string slope is much less than 1 at all times and positions.) The traveling-wave solution of the wave equation was first published by d'Alembert in 1747.

An example of the appearance of the traveling wave components shortly after plucking an infinitely long string at three points is shown in Fig. 2.

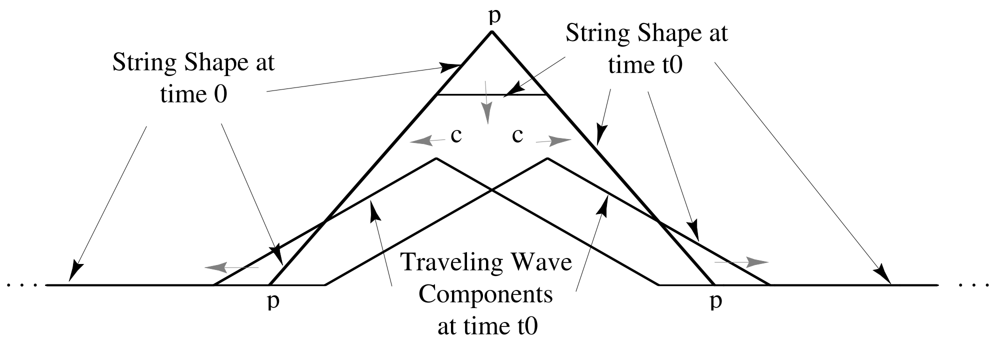


Figure 2. An infinitely long string, "plucked" simultaneously at *three* points, labeled "p" in the figure, so as to produce an initial triangular displacement. The initial displacement is modeled as the sum of two identical triangular pulses which are exactly on top of each other at time 0. At time t_0 shortly after time 0, the traveling waves have begun to separate, and their sum gives the physical string displacement at time t_0 which is also shown. Note that only three short string segments are in motion at that time: the flat top segment which is heading to zero where it will halt forever, and two short pieces on the left and right which are the leading edges of the left- and right-going traveling waves. The string is not moving where the traveling waves overlap at the same slope. When the traveling waves fully separate, the string will be at rest everywhere but for two half-amplitude triangular pulses heading off to plus and minus infinity at speed c.

Sampling the Traveling Waves

To carry the traveling-wave solution into the "digital domain," it is necessary to *sample* the traveling-wave amplitudes at intervals of T seconds, corresponding to a sampling rate $f_s \triangleq 1/T$ samples per second. For CD-quality audio, we have $f_s = 44.1$ kHz. The natural choice of *spatial sampling interval* X is the distance sound propagates in one temporal sampling interval T, or $X \triangleq cT$ meters. In air, assuming the speed of sound to be 331 meters per second, we have $X = 331/44100 = 7.5$ millimeters for the spatial sampling interval, or a spatial sampling rate of

133 samples per meter. In a traveling-wave simulation, the whole wave moves left or right one spatial sample each time sample; hence, simulation only requires digital delay lines.

Formally, sampling is carried out by the change of variables

$$\begin{aligned} x \rightarrow x_m &= mX \\ t \rightarrow t_n &= nT \end{aligned}$$

Substituting into the traveling-wave solution of the wave equation gives

$$\begin{aligned} y(t_n, x_m) &= y_r(t_n - x_m/c) + y_l(t_n + x_m/c) \\ &= y_r(nT - mX/c) + y_l(nT + mX/c) \\ &= y_r\left[(n-m)T\right] + y_l\left[(n+m)T\right] \end{aligned}$$

Since T multiplies all arguments, let's suppress it by defining

$$y^+(n) \triangleq y_r(nT) \qquad\qquad y^-(n) \triangleq y_l(nT)$$

This new notation also introduces a "+" superscript to denote a traveling-wave component propagating to the right, and a "−" superscript to denote propagation to the left.

The term $y_r\left[(n-m)T\right] = y^+(n-m)$ can be thought of as the output of an m-sample delay line whose input is $y^+(n)$. In general, subtracting a positive number m from a time argument n corresponds to *delaying* the waveform by m samples. Since y^+ is the right-going component, we draw its delay line with input $y^+(n)$ on the left and its output $y^+(n-m)$ on the right. This can be seen as the upper "rail" in Fig. 3.

Similarly, the term $y_l\left[(n+m)T\right] \triangleq y^-(n+m)$ can be thought of as the *input* to an m-sample delay line whose *output* is $y^-(n)$. (Adding m to the time argument n produces an m-sample waveform *advance*.) Since y^- is the left-going component, it makes sense to draw the delay line with its input $y^-(n+m)$ on the right and its output $y^-(n)$ on the left. This can be seen as the lower "rail" in Fig. 3. Note that the position along the string, $x_m = mX = mcT$ meters, is laid out from left to right in the diagram, giving a physical interpretation to the horizontal direction in the diagram. Finally, the left- and right-going traveling waves must be summed to produce a physical output according to the formula

$$y(t_n, x_m) = y^+(n-m) + y^-(n+m)$$

We may compute the physical string displacement at any spatial sampling point x_m by simply adding the upper and lower rails together at position m along the delay-line pair. In Fig. 3, "transverse displacement outputs" have been arbitrarily placed at $x = 0$ and $x = 3X$.

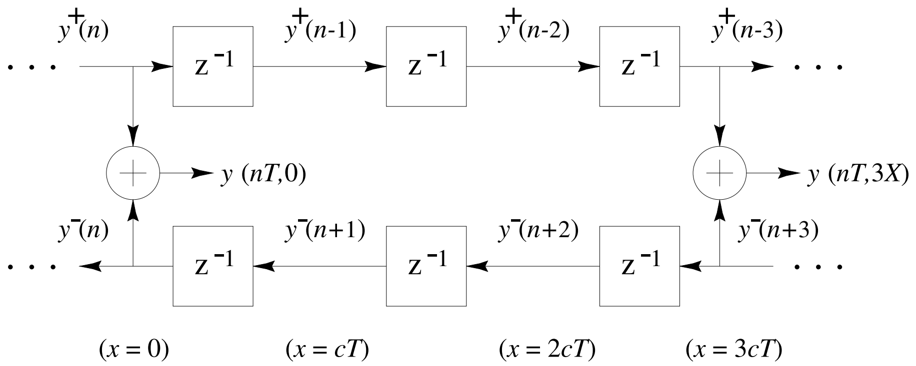


Figure 3. Digital simulation of the ideal, lossless waveguide with observation points at $x = 0$ and $x = 3X = 3cT$. The symbol "z^{-1}" denotes a one-sample delay.

Appendix A contains a C program which illustrates explicitly the implementation of a diagram such as the above in the context of a plucked string simulation.

A more compact simulation diagram which stands for either sampled or continuous simulation is shown in Fig. 4. The figure emphasizes that the ideal, lossless waveguide is simulated by a *bidirectional delay line.*

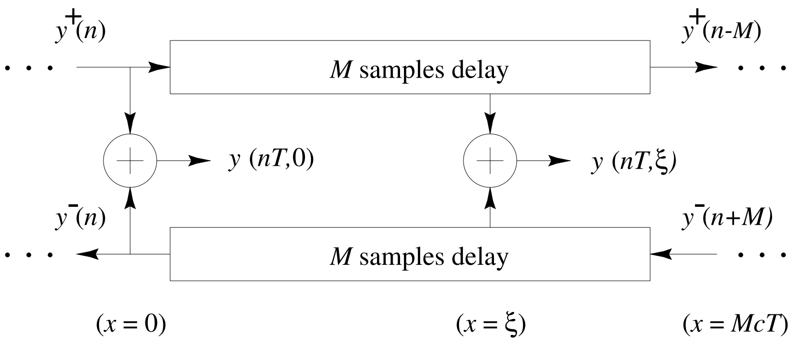


Figure 4. Simplified picture of ideal waveguide simulation.

Any ideal one-dimensional waveguide can be simulated in this way. It is important to note that the simulation is *exact* at the sampling instants, to within the numerical precision of the samples themselves, provided that the waveshapes traveling along the string are initially *bandlimited* to less than half the sampling frequency. In other words, the highest frequencies present in the signals $y_r(t)$ and $y_l(t)$ may not exceed half the temporal sampling frequency $f_s \triangleq 1/T$; equivalently, the highest *spatial* frequencies in the shapes $y_r(x/c)$ and $y_l(x/c)$ may not exceed half the spatial sampling frequency $\nu_s \triangleq 1/X$.

Bandlimited *spatial* interpolation may be used to construct a displacement output for an arbitrary x not a multiple of cT, as suggested by the output drawn in Fig. 4. Similarly, bandlimited interpolation across time serves to evaluate the waveform at an arbitrary time not an integer multiple of T.

Ideally, bandlimited interpolation is carried out by convolving a continuous "sinc function" $\text{sinc}(x) \triangleq \sin(\pi x)/\pi x$ with the signal samples. Specifically, convolving a sampled signal $x(t_n)$ with $\text{sinc}[(t_n - t_0)/T)$ "evaluates" the signal at an arbitrary continuous time t_0. The sinc function is the impulse response of the ideal lowpass filter which cuts off at half the sampling rate.

In practice, the interpolating sinc function must be *windowed* to a finite duration. This means the associated lowpass filter must be granted a "transition band" in which its frequency response is allowed to "roll off" to zero at half the sampling rate. The interpolation quality in the "pass band" can always be made perfect to within the resolution of human hearing by choosing a sufficiently large product of window-length times transition-bandwidth. Given "audibly perfect" quality in the pass band, increasing the transition bandwidth reduces the computational expense of the interpolation. This is one reason why *oversampling* at rates higher than twice the highest audio frequency is helpful. This topic is described further in (Smith and Gossett 1984).

Alternative Wave Variables

We have thus far considered discrete-time simulation of traveling *displacement* waves $y^{\pm}$ in the ideal string. It is equally valid to choose traveling *velocity* $v^{\pm} \triangleq \dot{y}^{\pm}$, *acceleration* $a^{\pm} \triangleq \ddot{y}^{\pm}$, or *slope* waves $y''^{\pm}$, or perhaps some other derivative or integral of displacement with respect to time and/or position.

Conversion between various time derivatives can be carried out by means *integrators* or *differentiators,* as depicted in Fig. 5. Since integration and differentiation are *linear* operators, and since the traveling wave arguments are in units of time, the same conversion formulas hold for the traveling wave *components* $y^{\pm}, v^{\pm}, a^{\pm}$.

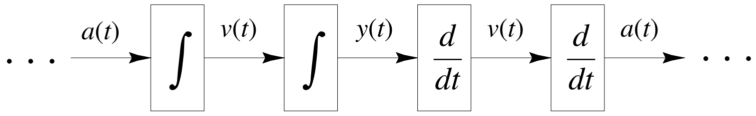
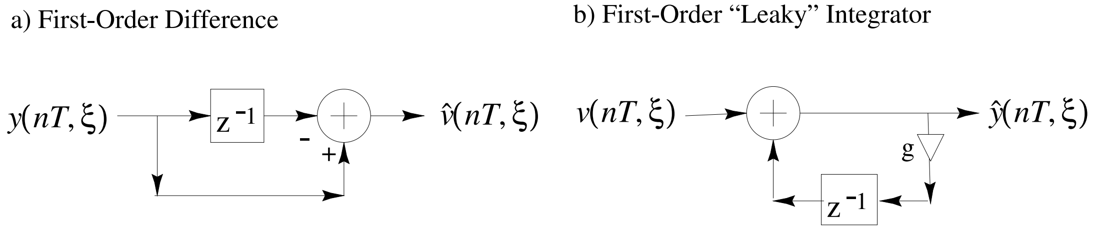


Figure 5. Conversions between various time derivatives of displacement: $y =$ displacement, $v = \dot{y} =$ velocity, $a = \ddot{y} =$ acceleration.

In discrete time, integration and differentiation can be accomplished using digital filters (Rabiner and Gold 1975). Commonly used approximations are shown in Fig. 6.


a) First-Order Difference

b) First-Order “Leaky” Integrator

Figure 6. Simple approximate conversions between time derivatives in the discrete-time case: a) The first-order difference. b) The first-order “leaky” integrator with loss factor g (slightly less than 1) used to avoid infinite DC build-up.

These first-order approximations are accurate (though scaled by T) at low frequencies relative to half the sampling rate, but they are not “best” approximations in any sense other than being most like the definitions of integration and differentiation in continuous time. Much better approximations can be obtained by approaching the problem from a *digital filter design* viewpoint (Loy 1988; Parks and Burrus 1987; Rabiner and Gold 1975; Smith 1985a). Arbitrarily better approximations are possible using higher order digital filters. In principle, a *digital differentiator* is a filter whose frequency response $H(e^{j\omega T})$ optimally approximates $j\omega$ for ω between $-\pi/T$ and π/T. Similarly, a digital integrator must match $1/j\omega$ along the unit circle in the z plane. The reason an exact match is not possible is that the ideal frequency responses $j\omega$ and $1/j\omega$, when wrapped along the unit circle in the z plane, (which is the frequency axis for discrete time systems), are not “simple” functions any more. As a result, there is no filter with a rational transfer function (i.e., finite order) that can match the desired frequency response exactly.

Spatial Derivatives

In addition to time derivatives, we may apply any number of *spatial derivatives* to obtain yet more wave variables to choose from. The first spatial derivative of string displacement yields *slope waves:*

$$y'(t,x) \stackrel{\Delta}{=} \frac{\partial}{\partial x} y(t,x) = y_r'(t-x/c) + y_l'(t+x/c) = -\frac{1}{c}\dot{y}_r(t-x/c) + \frac{1}{c}\dot{y}_l(t+x/c)$$

or, in discrete time,

$$\begin{aligned} y'(t_n,x_m) &\stackrel{\Delta}{=} y'(nT,mX) = y_r'\left[(n-m)T\right] + y_l'\left[(n+m)T\right] \stackrel{\Delta}{=} y'^{+}(n-m) + y'^{-}(n+m) \\ &= -\frac{1}{c}\dot{y}^{+}(n-m) + \frac{1}{c}\dot{y}^{-}(n+m) \stackrel{\Delta}{=} -\frac{1}{c}v^{+}(n-m) + \frac{1}{c}v^{-}(n+m) \\ &= \frac{1}{c}\left[v^{-}(n+m) - v^{+}(n-m)\right] \end{aligned}$$

From this we conclude that $v^{-} = cy'^{-}$ and $v^{+} = -cy'^{+}$. That is, traveling slope waves can be computed from traveling velocity waves by dividing by c and negating in the right-going case. Physical string slope can thus be computed from a velocity-wave simulation by *subtracting* the upper rail from the lower rail and dividing by c.

By the wave equation, *curvature waves,* $y'' = \ddot{y}/c^2$, are essentially identical to acceleration waves.

In the field of *acoustics,* the state of a vibrating string at any instant of time t_0 is normally specified by the displacement $y(t_0,x)$ and velocity $\dot{y}(t_0,x)$ for all x. Since displacement is the *sum* of the traveling displacement waves and velocity is proportional to the *difference* of the traveling displacement waves, one state description can be readily obtained from the other.

In summary, all traveling-wave variables can be computed from any one, as long as both the left- and right-going component waves are available. Alternatively, any *two* linearly independent *physical* variables, such as displacement and velocity, can be used to compute all other wave variables. Wave variable conversions requiring differentiation or integration are relatively expensive since a large-order digital filter is necessary to do it right. Slope waves can be computed from velocity waves by a simple scaling, and vice versa, and curvature waves are the same as acceleration waves to within a scale factor.

In the absence of other factors dictating a choice, *velocity waves* are a good overall choice because (1) it is numerically easier to perform digital integration to get displacement than it is to differentiate displacement to get velocity, (2) slope waves are immediately computable from velocity waves. Why are slope waves important? This is the subject of the next section.

Force Waves

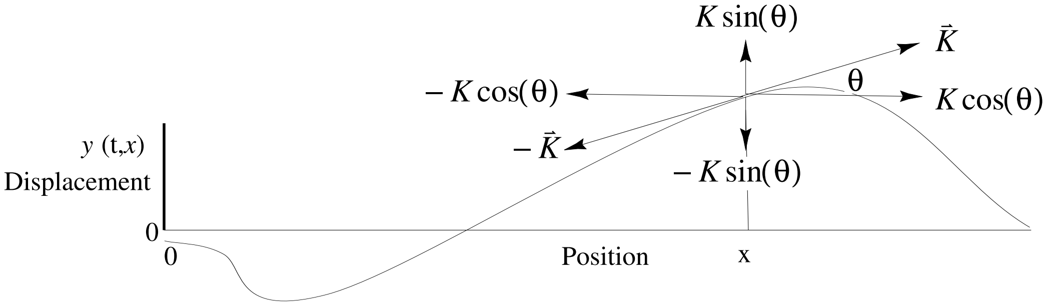


Figure 7. Transverse force propagation in the ideal string.

Referring to Fig. 7, at an arbitrary point x along the string, the vertical force applied at time t *to* the portion of string to the left of position x *by* the portion of string to the right of position x is given by

$$f_l(t,x) = K\sin(\theta) \approx K\tan(\theta) = Ky'(t,x)$$

assuming $|y'(t,x)| \ll 1$, as is assumed in the derivation of the wave equation. Similarly, the force applied *by* the portion to the left of position x *to* the portion to the right is given by

$$f_r(t,x) = -K\sin(\theta) \approx -Ky'(t,x)$$

These forces must cancel since a nonzero net force on a massless point would produce infinite acceleration.

Vertical force waves propagate along the string like any other transverse wave variable (since they are just slope waves multiplied by tension K). We may choose either f_l or f_r as the string force wave variable, one being the negative of the other. It turns out that to obtain a unification of vibrating strings and air columns, we have to pick f_r, the one that *acts to the right.* This makes sense intuitively when one considers longitudinal pressure waves in an acoustic tube: a compression wave traveling to the right in the tube pushes the air in front of it and thus acts to the right. Thus, we define the *force wave variable* to be

$$f(t,x) \stackrel{\Delta}{=} f_r(t,c) = -Ky'(t,x)$$

Substituting from above, we have

$$f(t,x) = \frac{K}{c}\left[\dot{y}_r(t-x/c) - \dot{y}_l(t+x/c)\right]$$

Note that $K/c \stackrel{\Delta}{=} K/\sqrt{K/\epsilon} = \sqrt{K\epsilon}$. This is a fundamental quantity known as the *wave impedance* of the string (also called the *characteristic impedance*), denoted as

$$R \stackrel{\Delta}{=} \sqrt{K\epsilon}$$

The wave impedance can be seen as the geometric mean of the two resistances to displacement: tension (spring force) and mass (inertial force). Note that $R = K/c = \epsilon c$ since $c = \sqrt{K/\epsilon}$.

The digitized, traveling, force-wave components become

$$\begin{aligned} f^+(n) &= Rv^+(n) \\ f^-(n) &= -Rv^-(n) \end{aligned}$$

which gives us that the right-going force wave equals the wave impedance times the right-going velocity wave, and the left-going force wave equals *minus* the wave impedance times the left-going velocity wave. Thus, in a traveling wave, force is always *in phase* with velocity, (considering the minus sign in the left-going case to be associated with the direction of travel rather than a 180° phase shift between force and velocity). Note that if the left-going force wave were defined as the string force acting to the left, the minus sign would disappear.

In the case of the *acoustic tube* (Morse 1936; Markle and Gray 1976), we have the analogous relations

$$\begin{aligned} p^+(n) &= Ru^+(n) \\ p^-(n) &= -Ru^-(n) \end{aligned}$$

where $p^+(n)$ is the right-going traveling *longitudinal pressure wave*, $p^-(n)$ is the left-going pressure wave, and $u^{\pm}(n)$ are the left and right-going *volume velocity waves*. In the acoustic tube context, the wave impedance is given by

$$R = \frac{\rho c}{A}$$

where ρ is the mass per unit volume of air, c is sound speed, and A is the cross-sectional area of the tube. Note that if we had chosen *particle velocity* rather than volume velocity, the wave impedance would have been $R = \rho c$ instead, the wave impedance in open air.

Power Waves

Basic courses in physics teach us that *power* is *work per unit time,* and *work* is a measure of *energy* which is typically defined as *force times distance.* Therefore, power is in physical units of force times distance per unit time, or force times velocity. It therefore should come as no surprise that *traveling power waves* are defined as

$$\begin{aligned} \mathcal{P}^+(n) &\stackrel{\Delta}{=} f^+(n)v^+(n) \\ \mathcal{P}^-(n) &\stackrel{\Delta}{=} -f^-(n)v^-(n) \end{aligned}$$

Note that $\mathcal{P}^+(n) = f^+(n)v^+(n) = R[v^+(n)]^2 = [f^+(n)]^2/R$, and $\mathcal{P}^-(n) = -f^-(n)v^-(n) = R[v^-(n)]^2 = [f^-(n)]^2/R$. Thus both the left- and right-going components are *nonnegative.* The sum of the traveling powers at a point thus gives the total power at that point on the string:

$$\mathcal{P}(t_n, x_m) \stackrel{\Delta}{=} \mathcal{P}^+(n-m) + \mathcal{P}^-(n+m)$$

If we had left out the minus sign in the definition of left-going power waves, the sum would instead be a *net* power flow.

Power waves are important for several reasons. They correspond to the actual ability of the wave to do *work* on the outside world, such as on a violin bridge at the end of the string. Because *energy is conserved* in closed systems, power waves sometimes give a simpler, more fundamental view of wave phenomena, such as in the case of conical acoustic tubes.

Energy Density Waves

The vibrational energy per unit length along the string, or *wave energy density* (Morse 1936) is given by the sum of potential and kinetic energy densities:

$$W(t,x) \triangleq \frac{1}{2}Ky'^2(t,x) + \frac{1}{2}\epsilon\dot{y}^2(t,x)$$

Sampling across time and space, and substituting traveling wave components, one can show in a few lines of algebra that the *sampled wave energy density* is given by

$$W(t_n,x_m) \triangleq W^+(n-m) + W^-(n+m)$$

where

$$\begin{aligned} W^+(n) &\triangleq \frac{\mathcal{P}^+(n)}{c} \triangleq \frac{f^+(n)v^+(n)}{c} = \epsilon\left[v^+(n)\right]^2 = \frac{\left[f^+(n)\right]^2}{K} \\ W^-(n) &\triangleq \frac{\mathcal{P}^-(n)}{c} \triangleq -\frac{f^-(n)v^-(n)}{c} = \epsilon\left[v^-(n)\right]^2 = \frac{\left[f^-(n)\right]^2}{K} \end{aligned}$$

Thus, traveling power waves (energy per unit time) can be converted to energy density waves (energy per unit length) by simply dividing by c, the speed of propagation. Quite naturally, the *total wave energy* in the string is given by the integral along the string of the energy density:

$$\mathcal{E}(t) \triangleq \int_{x=-\infty}^{\infty} W(t,x)dx \approx \sum_{m=-\infty}^{\infty} W(t,x_m)X$$

In practice, of course, the string length is finite, and the limits of integration are from the x coordinate of the left endpoint to that of the right endpoint, e.g., 0 to L.

The Lossy One-Dimensional Wave Equation

In any real vibrating string, there are energy losses due to yielding terminations, drag by the surrounding air, and internal friction within the string. While losses in solids generally vary in a complicated way with frequency, they can usually be well approximated by a small number of odd-order terms added to the wave equation. In the simplest case, force is directly proportional to transverse string velocity, independent of frequency. If this proportionality constant is μ, we obtain the modified wave equation

$$Ky'' = \epsilon\ddot{y} + \mu\dot{y}$$

Thus, the wave equation has been extended by a "first-order" term, i.e., a term proportional to the first derivative of y with respect to time. More realistic loss approximations would append terms proportional to $\partial^3 y(t,x)/\partial t^3$, $\partial^5 y(t,x)/\partial t^5$, and so on, giving frequency-dependent losses.

It can be checked that for small displacements, the following modified traveling wave solution satisfies the lossy wave equation:

$$y(t,x) = e^{-(\mu/2\epsilon)x/c} y_r\left(t - x/c\right) + e^{(\mu/2\epsilon)x/c} y_l\left(t + x/c\right)$$

The left-going and right-going traveling-wave components decay *exponentially* in their respective directions of travel.

Sampling these exponentially decaying traveling waves at intervals of T seconds (or $X = cT$ meters) gives

$$y(t_n, x_m) = g^{-m} y^+(n-m) + g^m y^-(n+m)$$

where $g \overset{\Delta}{=} e^{-\mu T/2\epsilon}$.

The digital simulation diagram for the lossy waveguide is shown in Fig. 8.

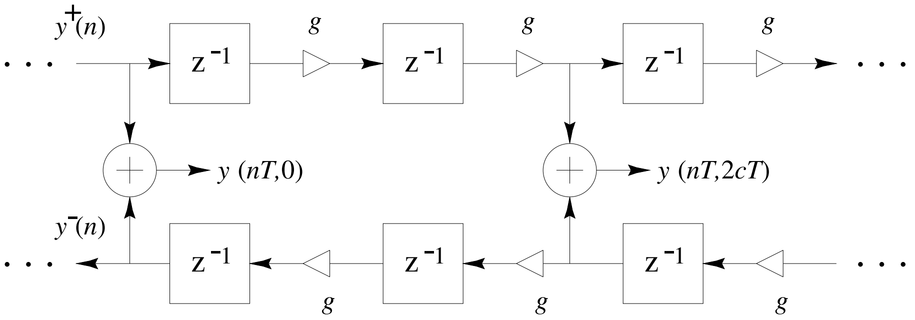


Figure 8. Discrete simulation of the ideal, lossy waveguide. The loss factor $g = e^{-\mu T/2\epsilon}$ *summarizes* the distributed loss incurred in one sampling period.

Again note that the simulation of the decaying traveling-wave solution is *exact* at the sampling positions and instants, even though losses are admitted in the wave equation. Note also that the losses which are *distributed* in the continuous solution have been consolidated, or *lumped,* at discrete intervals of cT meters in the simulation. *The lumping of distributed losses does not introduce an approximation error at the sampling points.* Furthermore, bandlimited interpolation can yield arbitrarily accurate reconstruction between samples. The only restriction is again that all initial conditions and excitations be bandlimited to half the sampling rate.

Consolidating Losses on a Larger Scale

In many applications, it is possible to realize vast computational savings in waveguide simulation by *commuting losses out of unobserved and undriven sections of the medium and consolidating them at a minimum number of points.* Because the digital simulation is linear and time invariant (given constant medium parameters K, ϵ, μ), and because linear, time-invariant elements commute, the diagram in Fig. 9 is exactly equivalent (to within numerical precision) to the previous diagram in Fig. 8.

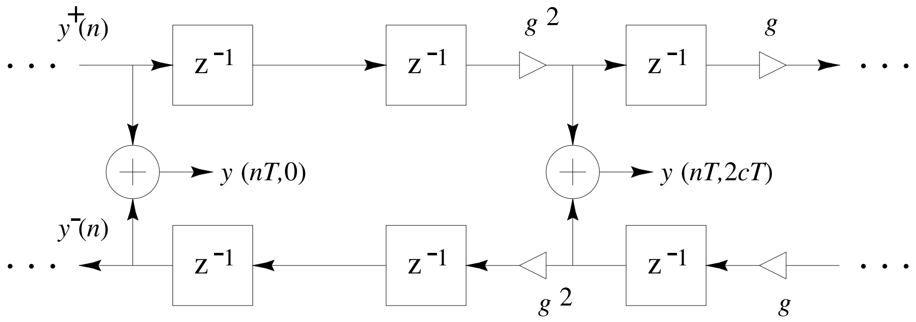


Figure 9. Discrete simulation of the ideal, lossy waveguide. Each per-sample loss factor g is "pushed through" delay elements and combined with other loss factors until an input or output is encountered which inhibits further migration. If further consolidation is possible on the other side of a branching node, a loss factor can be pushed *through* the node by pushing a copy into each departing branch. If there are other *inputs* to the node, the *inverse* of the loss factor must appear on each of them. Similar remarks apply to pushing backwards through a node.

Frequency-Dependent Losses

In nearly all natural wave phenomena, losses increase with frequency. The largest losses in a real stringed instrument occur at the bridge, particularly at frequencies which couple to body resonances. There are also losses due to air drag and internal bulk losses in the string which increase monotonically with frequency. Similarly, air absorption increases with frequency, providing an increase in propagation loss for sound waves in air (Morse and Ingard 1968).

The solution to the lossy wave equation can be generalized to the frequency-dependent case. Instead of factors g distributed throughout the diagram, the factors are now $G(\omega)$. These loss factors, implemented as digital filters, may also be consolidated at a minimum number of points in the waveguide without introducing an approximation error.

In an efficient digital simulation, each lumped loss factor $G(\omega)$ is to be approximated by a rational frequency response $\hat{G}(e^{j\omega T})$. In general, the coefficients of the optimal rational loss filter are obtained by minimizing $||\,G(\omega)-\hat{G}(e^{j\omega T})\,||$ with respect to the filter coefficients or the poles and zeros of the filter. To avoid introducing frequency-dependent delay, the loss filter should be a *zero-phase, finite-impulse-response* (FIR) filter (Rabiner and Gold 1975). Restriction to zero phase requires the impulse response $\hat{g}(n)$ to be finite in length (i.e., an FIR filter) and it must be symmetric about time zero, i.e., $\hat{g}(-n)=\hat{g}(n)$. For real-time implementations, the zero-phase FIR filter can be converted into a causal *linear phase* filter by "stealing" delay from the loop delay lines in an amount equal to half the impulse response duration.

The Dispersive One-Dimensional Wave Equation

Stiffness in a vibrating string introduces a restoring force proportional to the fourth derivative of the string displacement (Morse 1936; Cremer 1984):

$$\epsilon\ddot{y} = Ky'' - \kappa y''''$$

where, for a cylindrical string of radius a and Young's modulus Q, the moment constant κ is equal to $\kappa = Q\pi a^4/4$.

At very low frequencies, or for very small κ, we return to the non-stiff case. At very high frequencies, or for very large κ, we approach the *ideal bar* in which stiffness is the only restoring force. At intermediate frequencies, between the ideal string and bar, the stiffness contribution can be treated as a correction term (Cremer 1984). This is the region of most practical interest because it is the principal operating region for strings, such as piano strings, whose stiffness has audible consequences (an inharmonic, stretched overtone series). The first-order effect of stiffness is to increase the wave propagation speed with frequency:

$$c(\omega) \triangleq c_0 \left(1 + \frac{\kappa\omega^2}{2Kc_0^2}\right)$$

where c_0 is the wave travel speed in the absence of stiffness. Since sound speed depends on frequency, traveling waveshapes will "disperse" as they propagate along the string. That is, a traveling wave is no longer a static shape moving with speed c and expressible as a function of $t \pm x/c$. In a stiff string, the high frequencies propagate *faster* than the low-frequency components. As a result, a traveling velocity step, such as would be caused be a hammer strike, "unravels" into a smoother velocity step with high-frequency "ripples" running out ahead.

In a digital simulation, a frequency-dependent speed of propagation can be implemented using *allpass filters* which have a non-uniform delay versus frequency.

Note that every linear, time-invariant filter can be expressed as a zero-phase filter in cascade with an allpass filter. The zero-phase part implements frequency-dependent gain (damping in a digital waveguide), and the allpass part gives frequency-dependent delay, which in a digital waveguide yields dispersion. Every linear wave equation with constant coefficients, regardless of its order, corresponds to a waveguide which can be modeled as a pure delay and a linear, time-invariant filter which simulate propagation over a given distance.

Rigid Terminations

A *rigid termination* is the simplest case of a string termination. It imposes the constraint that the string cannot move at all at the termination. If we terminate a length L ideal string at $x = 0$ and $x = L$, we then have the "boundary conditions"

$$y(t, 0) \equiv 0 \qquad y(t, L) \equiv 0$$

where "$\equiv$" means "identically equal to," i.e., equal for all t.

Since $y(t, 0) = y_r(t) + y_l(t) = y^+(t/T) + y^-(t/T)$ and $y(t, L) = y_r(t - L/c) + y_l(t + L/c)$, the constraints on the sampled traveling waves become

$$\begin{aligned} y^+(n) &= -y^-(n) \\ y^-(n + N/2) &= -y^+(n - N/2) \end{aligned}$$

where $N \triangleq 2L/X$ is the time in samples to propagate from one end of the string to the other and back, or the total "string loop" delay. The loop delay is also equal to twice the number of spatial samples along the string. A digital simulation diagram for the terminated ideal string is shown in Fig. 10. A "pick-up" is shown at the arbitrary location $x = \xi$.

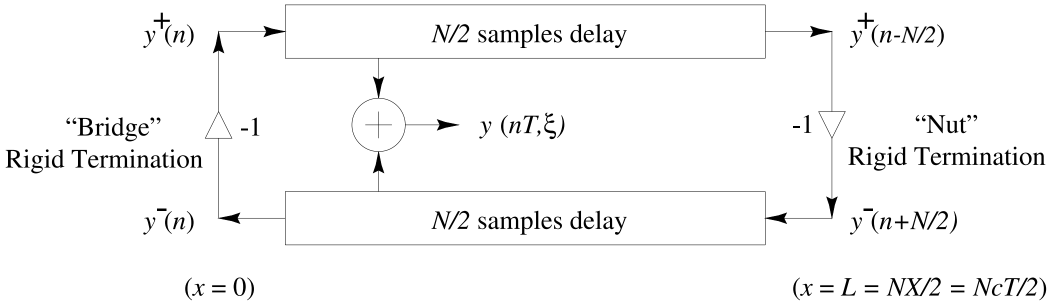


Figure 10. The rigidly terminated ideal string, with a position output indicated at $x = \xi$. Rigid terminations reflect traveling displacement, velocity, or acceleration waves with a sign inversion. Slope or force waves reflect with no sign inversion.

The Ideal Plucked String

The ideal *plucked string* (Morse 1936) is defined as an initial string displacement and a zero initial velocity distribution. In general, the initial displacement along the string $y(0, x)$ and the initial velocity distribution $\dot{y}(0, x)$ fully determine the resulting motion in the absence of further excitation.

An example of the appearance of the traveling wave components and the resulting string shape shortly after plucking a doubly terminated string at a point one fourth along its length is shown in Fig. 11. The negative traveling-wave portions can be thought of as inverted reflections of the incident waves, or as doubly flipped "images" which are coming from the other side of the terminations.

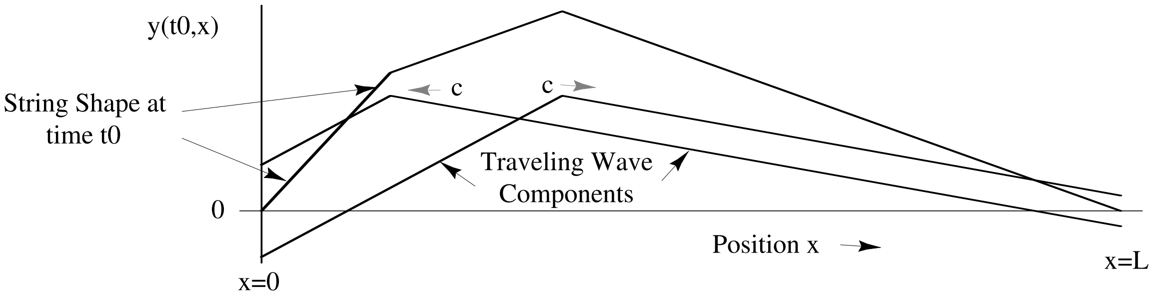


Figure 11. A doubly terminated string, "plucked" at one fourth its length.

An example of an initial "pluck" excitation in a digital waveguide string model is shown in Fig. 12. There is one fine point to note for the discrete-time case: We cannot admit a sharp corner in the string since that would have infinite bandwidth which would alias when sampled. Therefore, for the discrete-time case, we define the ideal pluck to consist of an arbitrary shape as in Fig. 12

lowpass filtered to half the sampling rate (or less). Alternatively, we can simply require the initial displacement shape to be bandlimited to spatial frequencies less than $f_s/2c$. Since all real strings have some degree of stiffness which prevents the formation of perfectly sharp corners, and since real plectra are never in contact with the string at only one point, and since the frequencies we do allow span the full range of human hearing, the bandlimited restriction is not a binding one. So, given proper bandlimiting of the initial shape, it is valid to replace the continuous curve with its samples without changing the information content.

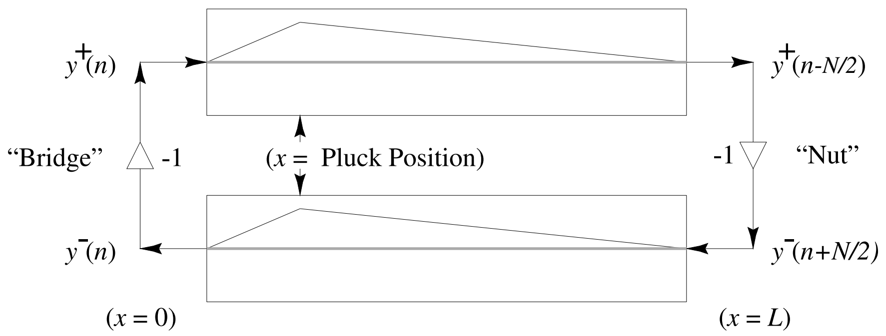


Figure 12. Initial conditions for the ideal plucked string. The initial contents of the sampled, traveling-wave delay lines are in effect *plotted* inside the delay-line boxes. The amplitude of each traveling-wave delay line is half the amplitude of the initial string displacement. The sum of the upper and lower delay lines gives the actual initial string displacement.

Note that acceleration (or curvature) waves are a simple choice for plucked string simulation, since the ideal pluck corresponds to an initial *impulse* in the delay lines at the pluck point. Of course, since we require a bandlimited excitation, the initial acceleration distribution will be replaced by the impulse response of the anti-aliasing filter. If the anti-aliasing filter chosen is the ideal lowpass filter cutting off at $f_s/2$, the initial acceleration $a(0,x) \triangleq \ddot{y}(0,x)$ for the ideal pluck becomes

$$a(0,x) = \frac{A}{X}\text{sinc}\left(\frac{x - x_p}{X}\right)$$

where A is amplitude, x_p is the pick position, and $\text{sinc}[(x-x_p)/X]$ is the ideal, bandlimited impulse, centered at x_p and having a rectangular spatial frequency response extending from $-\pi X$ to πX. Division by X normalizes the area under the initial shape curve. If x_p is chosen to lie exactly on a spatial sample $x_m = mX$, the initial conditions for the ideal plucked string are as shown in Fig. 13 for the case of acceleration or curvature waves. All initial samples are zero except one in each delay line.

More generally, bowed string models involving a string with a *periodic* plucking excitation (Smith 1983) can be calibrated to recorded data by means of *linear predictive coding* (LPC) which has been very successful in speech modeling (Makhoul 1975; Markle and Gray 1976).

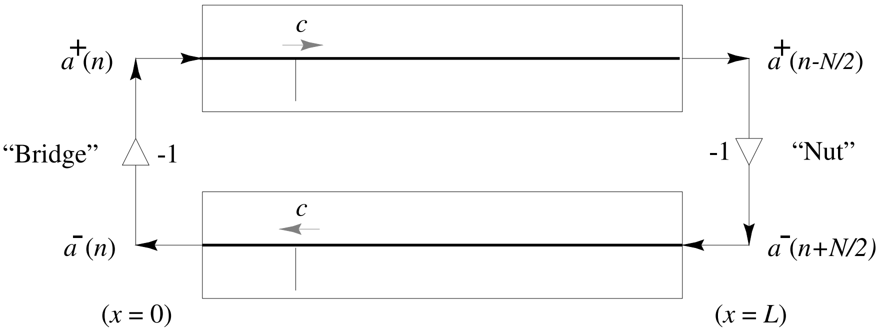


Figure 13. Initial conditions for the ideal plucked string when the wave variables are chosen to be proportional to acceleration or curvature. If the bandlimited ideal pluck position is centered on a spatial sample, there is only a single nonzero sample in each of the initial delay lines.

The Ideal Struck String

The ideal *struck string* (Morse 1936) involves a *zero* initial string displacement but a nonzero initial velocity distribution. In concept, a "hammer strike" transfers an "impulse" of momentum to the string at time 0 along the striking face of the hammer. An example of "struck" initial conditions is shown in Fig. 14 for a striking "hammer" having a rectangular shape.

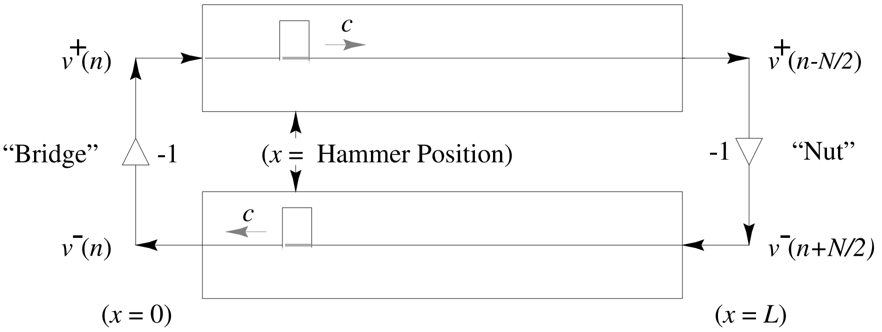


Figure 14. Initial conditions for the ideal struck string in a *velocity wave* simulation. Since $v^{\pm} = \pm f^{\pm}/R = \mp c y'^{\pm}$, the initial velocity distribution can be integrated with respect to x from $x = 0$, divided by c, and negated in the upper rail to obtain equivalent initial displacement waves (Morse 1936).

The Damped Plucked String

Without damping, the ideal plucked string sounds more like a cheap electronic organ because the sound is perfectly periodic and never decays. The Fourier transform of the initial "string loop" contents gives the Fourier *series* for the periodic tone produced, and static spectra are very boring to the ear. Incorporating damping means we use *exponentially decaying traveling waves* instead of non-decaying waves. As discussed previously, it saves computation to *lump* the loss factors which implement damping in the waveguide in order to minimize computational cost and round-off error.

To illustrate how significant the computational savings can be, consider the simulation of a "damped guitar string" model in Fig. 15. For simplicity, the length L string is rigidly terminated on both ends. Let the string be "plucked" by initial conditions so that we need not couple an input mechanism to the string. Also, let the output be simply the signal passing through a particular delay element rather than the more realistic summation of opposite elements in the bidirectional delay line.

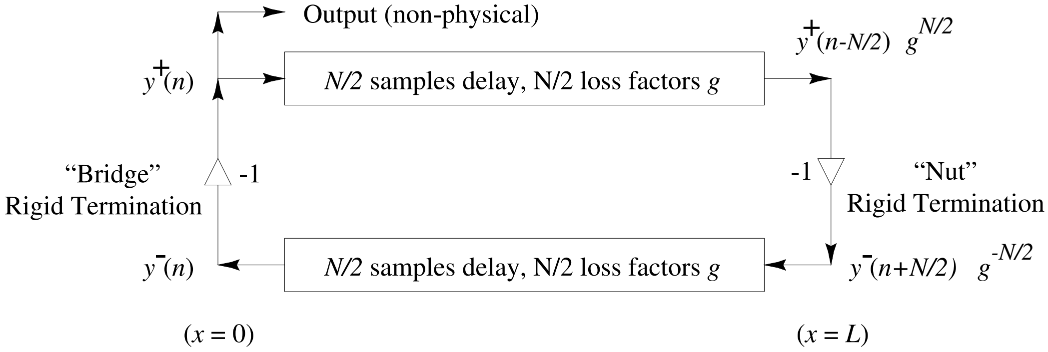


Figure 15. Discrete simulation of the rigidly terminated string with resistive losses. The N loss factors g are embedded between the delay-line elements.

In this string simulator, there is a loop of delay containing $N = 2L/X = f_s/f_1$ samples where f_1 is the desired pitch of the string. Because there is no input/output coupling, we may lump *all* of the losses at a *single point* in the delay loop. Furthermore, the two reflecting terminations (gain factors of -1) may be commuted so as to cancel them. Finally, the right-going delay may be combined with the left-going delay to give a single, length N, delay line. The result of these inaudible simplifications is shown in Fig. 16.

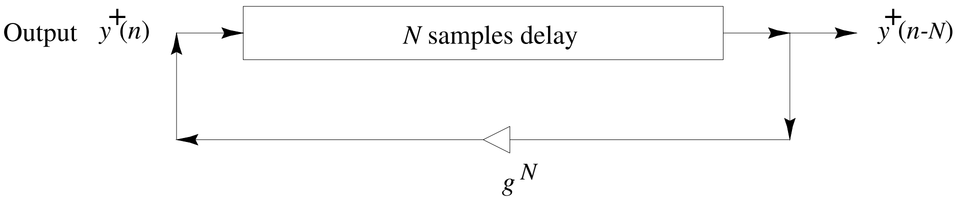


Figure 16. Discrete simulation of the rigidly terminated string with *consolidated* losses (frequency-independent). All N loss factors g have been "pushed" through delay elements and combined at a *single* point.

If the sampling rate is $f_s = 50$ kHz and the desired pitch is $f_1 = 100$ Hz, the loop delay equals $N = 500$ samples. Since delay lines are efficiently implemented as circular buffers, the cost of implementation is normally dominated by the loss factors, each one requiring a multiply every sample, in general. (Losses of the form $1-2^{-k}$, $1-2^{-k}-2^{-l}$, etc., can be efficiently implemented using shifts and adds.) Thus, *the consolidation of loss factors has reduced computational complexity by three orders of magnitude,* i.e., by a factor of 500 in this case. However, *the physical accuracy of the simulation has not been compromised.* In fact, the accuracy is *improved* because the N round-off errors arising from repeated multiplication by g have been replaced by a single round-off error in g^N.

The Plucked String with Frequency-Dependent Damping

As discussed previously, damping should increase at higher frequencies for better realism. This means the loss factors g of the previous section are replaced by digital filters having gains which decrease with frequency and never exceeding 1. These filters commute with delay elements because they are time invariant. Thus, following the reasoning of the previous section, they can be lumped at a single point in the digital waveguide. Let $\hat{G}(z)$ denote the resulting *string loop filter.* We have the constraint $|\hat{G}(e^{j\omega T})| \leq 1$, and making it zero or linear phase will restrict consideration to symmetric FIR filters only.

$\hat{G}(z)$

In the simplest case of a *first-order* lowpass loss filter, $\hat{G}(z) = b_0 + b_1 z^{-1}$, the linear-phase requirement imposes $b_0 = b_1$. Assuming the damping approaches zero at frequency zero implies $b_0 + b_1 = 1$. Thus, two equations in two unknowns uniquely determine the coefficients to be $b_0 = b_1 = 1/2$ which gives a string loop frequency response equal to $\hat{G}(e^{j\omega T}) = \cos(\omega T/2)$, $|\omega| \leq \pi f_s$.

The simulation diagram for the ideal string with the simplest frequency-dependent loss filter is shown in Fig. 17. Readers of the computer music literature will recognize this as the structure of the *Karplus-Strong algorithm* (Karplus and Strong 1983; Jaffe and Smith 1983; Sullivan 1990).

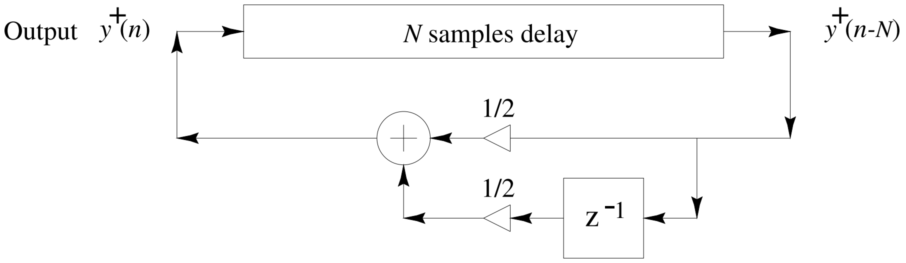


Figure 17. Rigidly terminated string with the simplest frequency-dependent loss filter. All N loss factors (possibly including losses due to yielding terminations) have been consolidated at a single point and replaced by a one-zero filter approximation.

The Karplus-Strong algorithm, per se, is obtained when the initial conditions used to "pluck" the string are obtained by filling the delay line with random numbers, or "white noise." We know the initial *shape* of the string is obtained by *adding* the upper and lower delay lines of Fig. 15, i.e., $y(t_n, x_m) = y^+(n-m) + y^-(n+m)$. It was also noted earlier how the initial *velocity* distribution along the string is determined by the *difference* between the upper and and lower delay lines. Thus, in the Karplus-Strong algorithm, the string is "plucked" by a completely *random initial displacement and initial velocity distribution.* This is a very energetic excitation, and usually in practice a lowpass filter is applied to the white noise to subdue it.

Single-Reed Instruments

A simplified model for a single-reed instrument is shown in Fig. 18.

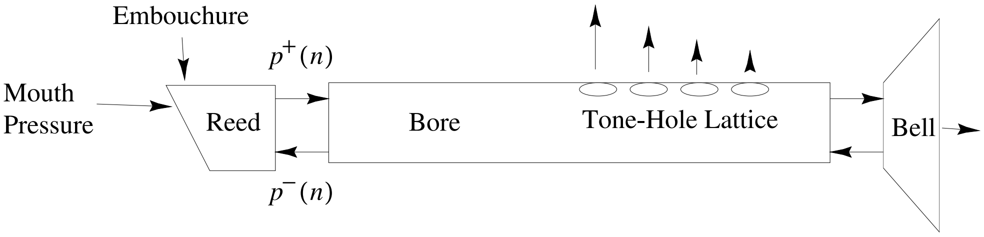


Figure 18. A schematic model for single-reed woodwinds.

If the bore is cylindrical, as in the clarinet, it can be modeled quite simply using a bidirectional delay line. If the bore is conical, such as in a saxophone, it can still be modeled as a bidirectional delay line, but interfacing to it is slightly more complex (Benade 1988; Smith 1991). Because the main control variable for the instrument is air pressure in the mouth at the reed, it is convenient to choose *pressure waves* for the waveguide variables.

To a first order, the bell passes high frequencies and reflects low frequencies, where "high" and "low" are relative to the diameter of the bell. Thus, the bell can be regarded as a simple "cross-over" network, as is used to split signal energy between woofers and tweeters in loudspeakers. Tone holes can be treated similarly. However, much better tone-hole models exist in the literature (Keefe 1982), and they can be adapted to the traveling-wave formulation in a straightforward way.

Since the length of the clarinet bore is only a quarter wavelength of the fundamental, (in the lowest, or "chalumeau" register), and since the bell diameter is much smaller than the bore length, most of the sound energy reflects back into the bore. The low-frequency energy that makes it out of the bore radiates in a fairly omnidirectional pattern. Very high-frequency traveling waves do not "see" the enclosing bell and pass right through it, radiating in a more directional beam. The directionality of the beam is proportional to how many wavelengths fit along the bell diameter; in fact, many wavelengths away from the bell, the radiation pattern is proportional to the two-dimensional spatial Fourier transform of the exit aperture (a disk at the end of the bell) (Morse and Ingard 1968).

The theory of the single reed is decribed in (McIntyre, Schumacher, and Woodhouse 1983). An efficient waveguide synthesis technique based on that theory is described in (Smith 1986). Essentially, if the reed mass is neglected, its effect can be implemented using a single *table lookup* or *segmented polynomial* evaluation per sample whose input is the difference between mouth pressure and incoming bore pressure, and whose output is the reflection coefficient "seen" at the mouthpiece inside the bore. The lookup table (or polynomial) appears as a nonlinear termination of the bore, presenting a signal dependent (and mouth-pressure dependent) reflection coefficient. The player's embouchure controls damping of the reed, reed aperture width, and other parameters, and these can be implemented as parameters on the contents of the lookup table or nonlinear function.

Instruments of the *brass* family have also been convincingly simulated using this general approach (Cook 1991). In the brass model, the "lip reed" is modeled as a second-order mass-spring system whose mass corresponds to the mass of the lips and whose spring constant corresponds to lip tension. Damping is a third control. A waveguide *flute* has also been implemented (Karjalainen et al. 1991).

Bowed Strings

A general block diagram for bowed strings is shown in Fig. 19. The bow divides the string into two sections, so the bow model is a nonlinear two-port, in contrast with the reed which was a one-port terminating the bore at the mouthpiece. In the case of bowed strings, the primary control variable is bow velocity, so *velocity waves* are the natural choice for the delay lines.

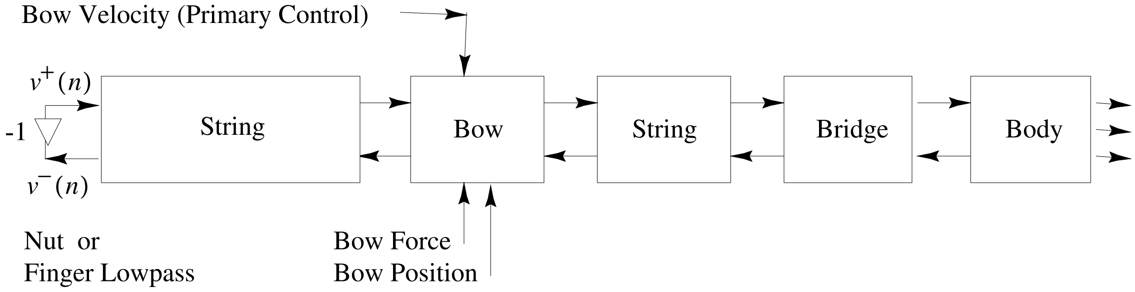


Figure 19. A schematic model for bowed-string instruments.

The theory of bow-string interaction is described in (McIntyre and Woodhouse 1979; McIntyre, Schumacher, and Woodhouse 1983). The basic operation of the bow is to reconcile the bow-string friction curve with the string state and string wave impedance. In a bowed string simulation as in Fig. 19, a velocity input (which is injected equally in the left- and right-going directions) must be found such that the transverse force of the bow against the string is balanced by the reaction force of the moving string. If bow-hair dynamics are neglected, the bow-string interaction can also be simulated using a memoryless table lookup or segmented polynomial (Smith 1986). An overall model for the violin is developed in (Smith 1983).

Conclusions

Starting with the traveling-wave solution to the wave equation and sampling across time and space, we obtained a modeling framework known as the “digital waveguide” approach. Its main feature is computational economy in the context of a true physical model. Successful computational models have been obtained for several musical instruments of the string and wind families, and more are on the way.

While physics-based synthesis can provide extremely high quality and expressivity in a very compact algorithm, new models must be developed for each new kind of instrument, and for many instruments, no sufficiently concise algorithm is known. Sampling synthesis, on the other hand is completely general since it involves only playing back and processing natural recorded sound. However, sampling synthesis demands huge quantities of memory for the highest quality and multidimensional control. It seems reasonable therefore to expect that many musical instrument categories now being implemented via sampling synthesis will ultimately be upgraded to parsimonious, computational models based on musical acoustics. As this evolution proceeds, the traditional instrument quality available from a given area of silicon should increase dramatically.

Appendix A — Programming Example for the Plucked String

```
/* pluck.c - elementary waveguide simulation of plucked strings - JOS 6/6/92 */

/* Note: The word "inline" below can be deleted if your compiler
   does not support it.  It is a nice GNU feature not in the ANSII C spec. */

#import <libc.h>

#define SRATE 44100
#define DOUBLE_TO_SHORT(x) ((int)((x)*32768.0))

typedef struct _DelayLine {
    short *data;
    int length;
    short *pointer;
    short *end;
} DelayLine;

static DelayLine *initDelayLine(int len) {
    DelayLine *dl = (DelayLine *)calloc(len, sizeof(DelayLine));
    dl->length = len;
    if (len > 0)
     dl->data = (short *)calloc(len, len * sizeof(short));
    else
     dl->data = 0;
    dl->pointer = dl->data;
    dl->end = dl->data + len - 1;
    return dl;
}

static void freeDelayLine(DelayLine *dl) {
    if (dl && dl->data)
free(dl->data);
    dl->data = 0;
    free(dl);
}

inline static void setDelayLine(DelayLine *dl, double *values, double scale) {
    int i;
    for (i=0; i<dl->length; i++)
     dl->data[i] = DOUBLE_TO_SHORT(scale * values[i]);
}

/* lg_dl_update(dl, insamp);
 * Places "nut-reflected" sample from upper delay-line into
 * current lower delay-line pointer location (which represents
 * x = 0 position).  The pointer is then incremented (i.e. the
 * wave travels one sample to the left), turning the previous
 * position into an "effective" x = L position for the next
```

```
 * iteration.
 */
static inline void lg_dl_update(DelayLine *dl, short insamp) {
    register short *ptr = dl->pointer;
    *ptr = insamp;
ptr++;
    if (ptr > dl->end)
     ptr = dl->data;
    dl->pointer = ptr;
}

/* rg_dl_update(dl, insamp);
 * Decrements current upper delay-line pointer position (i.e.
 * the wave travels one sample to the right), moving it to the
 * "effective" x = 0 position for the next iteration.  The
 * "bridge-reflected" sample from lower delay-line is then placed
 * into this position.
 */
static inline void rg_dl_update(DelayLine *dl, short insamp) {
    register short *ptr = dl->pointer;
ptr--;
    if (ptr < dl->data)
     ptr = dl->end;
*ptr = insamp;
    dl->pointer = ptr;
}

/* dl_access(dl, position);
 * Returns sample "position" samples into delay-line's past.
 * Position "0" points to the most recently inserted sample.
 */
static inline short dl_access(DelayLine *dl, int position) {
    short *outloc = dl->pointer + position;
    while (outloc < dl->data)
     outloc += dl->length;
    while (outloc > dl->end)
     outloc -= dl->length;
    return *outloc;
}

/*
 *  Right-going delay line:
 *  -->---->---->---
 *  x=0
 *  (pointer)
 *  Left-going delay line:
 *  --<----<----<---
 *  x=0
 *  (pointer)
 */
```

```
/* rg_dl_access(dl, position);
 * Returns spatial sample at location "position", where position zero
 * is equal to the current upper delay-line pointer position (x = 0).
 * In a right-going delay-line, position increases to the right, and
 * delay increases to the right => left = past and right = future.
 */
static inline short rg_dl_access(DelayLine *dl, int position) {
    return dl_access(dl, position);
}

/* lg_dl_access(dl, position);
 * Returns spatial sample at location "position", where position zero
 * is equal to the current lower delay-line pointer position (x = 0).
 * In a left-going delay-line, position increases to the right, and
 * delay DEcreases to the right => left = future and right = past.
 */
static inline short lg_dl_access(DelayLine *dl, int position) {
    return dl_access(dl, position);
}

static DelayLine *upper_rail,*lower_rail;


static inline int initString(double amplitude, double pitch,
                             double pick, double pickup) {
    int i, rail_length = SRATE/pitch/2 + 1;
/*
 * Round pick position to nearest spatial sample.
 * A pick position at x = 0 is not allowed.
 */
    int pickSample = MAX(rail_length * pick, 1);
double upslope = amplitude/pickSample;
    double downslope = amplitude/(rail_length - pickSample - 1);
    double initial_shape[rail_length];

    upper_rail = initDelayLine(rail_length);
    lower_rail = initDelayLine(rail_length);

#ifdef DEBUG
    initial_shape[pickSample] = 1;
#else
    for (i = 0; i < pickSample; i++)
     initial_shape[i] = upslope * i;
    for (i = pickSample; i < rail_length; i++)
     initial_shape[i] = downslope * (rail_length - 1 - i);
#endif

    /*
     * Initial conditions for the ideal plucked string.
     * "Past history" is measured backward from the end of the array.
     */
```

```
    setDelayLine(lower_rail, initial_shape, 0.5);
    setDelayLine(upper_rail, initial_shape, 0.5);

    return pickup * rail_length;
}

static inline void freeString(void) {
    freeDelayLine(upper_rail);
    freeDelayLine(lower_rail);
}

static inline short bridgeReflection(int insamp) {
    static short state = 0; /* filter memory */
    /* Implement a one-pole lowpass with feedback coefficient = 0.5 */
    /* outsamp = 0.5 * outsamp + 0.5 * insamp */
    short outsamp = (state >> 1) + (insamp >> 1);
    state = outsamp;
    return outsamp;
}

static inline short nextStringSample(int pickup_loc) {
    short yp0,ym0,ypM,ymM;
    short outsamp, outsamp1;

    /* Output at pickup location */
    outsamp  = rg_dl_access(upper_rail, pickup_loc);
    outsamp1 = lg_dl_access(lower_rail, pickup_loc);
outsamp += outsamp1;

    ym0 = lg_dl_access(lower_rail, 1);     /* Sample traveling into "bridge" */
    ypM = rg_dl_access(upper_rail, upper_rail->length - 2); /* Sample to "nut" */

    ymM = -ypM;                    /* Inverting reflection at rigid nut */
    yp0 = -bridgeReflection(ym0);  /* Reflection at yielding bridge */

    /* String state update */
    rg_dl_update(upper_rail, yp0); /* Decrement pointer and then update */
    lg_dl_update(lower_rail, ymM); /* Update and then increment pointer */

    return outsamp;
}

/* Utility for writing a mono sound to a sound file on a NeXT machine */
#import <sound/sound.h>
static int writeSound(char *name, short *soundData, int sampleCount) {
    int i, err;
    short *data;
    SNDSoundStruct *sound;
    SNDAlloc(&sound, sampleCount * sizeof(short), SND_FORMAT_LINEAR_16,
             SRATE,1,4);
    data = (short *) ((char *)sound + sound->dataLocation);
```

```
    for (i = 0; i < sampleCount; i++)
     data[i] = soundData[i];
    err = SNDWriteSoundfile(name,sound);
    if(err)
     fprintf(stderr,"*** Could not write sound file %s\n",name);
    else
     printf("File %s written.\n",name);
    return err;
}

static void writeString(void) {
    int i, sampleCount = upper_rail->length;
    short data[sampleCount];

    for (i = 0; i < sampleCount; i++)
     data[i] = rg_dl_access(upper_rail,i);
    writeSound("upper.snd", data, sampleCount);

    for (i = 0; i < sampleCount; i++)
     data[i] = lg_dl_access(lower_rail,i);
    writeSound("lower.snd", data, sampleCount);

    for (i = 0; i < sampleCount; i++)
     data[i] = rg_dl_access(upper_rail,  i) + lg_dl_access(lower_rail, i);
    writeSound("string.snd", data, sampleCount);
}

void main (int argc, char *argv[]) {
    int i, sampleCount;
    short *data;
    double amp, duration, pitch, pick, pickup, writesample;
    int pickupSample;

    if (argc != 8) {
        fprintf(stderr, "Usage: %s amp(<1.0) pitch(Hz) pickPosition(<1.0) "
                "pickupPosition(<1.0) duration(sec) writeSamp out.snd\n",
                argv[0]);
        fprintf(stderr, "example: %s .5 100 .1 .2 1 -1 test.snd\n", argv[0]);
        exit(1);
    }

    sscanf(argv[1],"%lf",&amp);
    sscanf(argv[2],"%lf",&pitch);
    sscanf(argv[3],"%lf",&pick);
    sscanf(argv[4],"%lf",&pickup);
    sscanf(argv[5],"%lf",&duration);
    sscanf(argv[6],"%lf",&writesample);

    sampleCount = duration * SRATE;
    data = (short *) malloc(sampleCount * sizeof(short));
```

```
    pickupSample = initString(amp, pitch, pick, pickup);

    for (i = 0; i < sampleCount; i++) {
        if (i == writesample) {
            printf("Writing string snapshot at sample %d\n",i);
            writeString();
        }
        data[i] = nextStringSample(pickupSample);
   }

writeSound(argv[7], data, sampleCount);
    freeString();
    exit(0);
}
```

References

A. H. Benade, "Equivalent Circuits for Conical Waveguides," *J. Acoust. Soc. Amer.*, vol. 83, no. 5, pp. 1764–1769, May 1988.

R. Caussé, J. Kergomard, and X. Lurton, "Input impedance of Brass Musical Instruments—Comparison between Experiment and Numerical Models," *J. Acoust. Soc. Amer.*, vol. 75, no. 1, pp. 241–254, Jan. 1984.

P. R. Cook, "Identification of Control Parameters in an Articulatory Vocal Tract Model, with Applications to the Synthesis of Singing," Ph.D. Dissertation, Elec. Eng. Dept., Stanford University, Dec. 1990.

"TBone: An Interactive WaveGuide Brass Instrument Synthesis Workbench for the NeXT Machine," *Proc. 1991 International Computer Music Conference,*, pp. 297–300, Montreal.

L. Cremer, *The Physics of the Violin,* MIT Press, Cambridge MA, 1984.

L. Hiller and P. Ruiz, "Synthesizing Musical Sounds by Solving the Wave Equation for Vibrating Objects," *J. Audio Eng. Soc.,* Part I: vol. 19, no. 6, June 1971; Part II: vol. 19, no. 7, July/Aug. 1971.

A. Hirschberg, "A Quasi-Stationary Model of Air Flow in the Reed Channel of Single-Reed Woodwind Instruments," *Acustica*, vol. 70, pp. 146–154, 1990.

S. Hirschman, P. R. Cook, and J. O. Smith, "Digital Waveguide Modelling and Simulation of Reed Woodwind Instruments," Eng. Dissertation, Elec. Eng. Dept., Stanford University, May 1991.

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

M. Karjalainen, U. K. Laine, T. Laakso, and V. Välimäki, "Transmission-Line Modeling and Real-Time Synthesis of String and Wind Instruments," *Proc. 1991 International Computer Music Conference,*, pp. 293–294, Montreal.

K. Karplus and A. Strong, "Digital Synthesis of Plucked String and Drum Timbres," *Computer Music J.*, vol. 7, no. 2, pp. 43–55, 1983.

D. H. Keefe, "Theory of the Single Woodwind Tone Hole," "Experiments on the Single Woodwind Tone Hole," *J. Acoust. Soc. Amer.*, vol. 72, no. 3, pp. 676–699, Sep. 1982.

N. J. Loy, *An Engineer's Guide to FIR Digital Filters,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1988.

J. Makhoul, "Linear Prediction: A Tutorial Review," *Proc. IEEE*, vol. 63, pp. 561–580, Apr. 1975.

J. D. Markel and A. H. Gray, *Linear Prediction of Speech,* Springer-Verlag, New York, 1976.

M. E. McIntyre and J. Woodhouse, "On the Fundamentals of Bowed String Dynamics," *Acustica*, vol. 43, no. 2, pp. 93–108, Sep. 1979.

M. E. McIntyre, R. T. Schumacher, and J. Woodhouse, "On the Oscillations of Musical Instruments," *J. Acoust. Soc. Amer.*, vol. 74, no. 5, pp. 1325–1345, Nov. 1983.

P. M. Morse, *Vibration and Sound,* published by the American Institute of Physics for the Acoustical Society of America, 1976 (1st ed. 1936, 2nd ed. 1948).

P. M. Morse and K. U. Ingard, *Theoretical Acoustics,* McGraw-Hill, New York, 1968.

T. W. Parks and C. S. Burrus, "Digital Filter Design," John Wiley and Sons, Inc., New York, June 1987.

A. D. Pierce, *Acoustics,* Amer. Inst. Physics for Acoust. Soc. Amer., (516)349-7800 x 481, 1989.

L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1975.

C. Roads and J. Strawn, eds., *Foundations of Computer Music,* MIT Press, Cambridge MA, 1985.

C. Roads, ed., *The Music Machine,* MIT Press, Cambridge MA, 1989.

P. M. Ruiz, "A Technique for Simulating the Vibrations of Strings with a Digital Computer," *M. Music Diss.,* Univ. Ill., Urbana, 1969.

J. O. Smith, "Techniques for Digital Filter Design and System Identification with Application to the Violin," Ph.D. Dissertation, Elec. Eng. Dept., Stanford University, June 1983.

J. O. Smith and P. Gossett, "A Flexible Sampling-Rate Conversion Method," *Proc. IEEE Conf. Acoust. Sp. and Sig. Proc., vol. 2, pp. 19.4.1-19.4.2, San Diego, March 1984.*

J. O. Smith, "Introduction to Digital Filter Theory," In J. Strawn, ed., *Digital Audio Signal Processing: An Anthology.* William Kaufmann, Inc., Los Altos, California, 1985. A shortened version appears in *The Music Machine,* , Roads, C., ed., MIT Press, 1989.

J. O. Smith, "A New Approach to Digital Reverberation using Closed Waveguide Networks," *Proc. 1985 International Computer Music Conference,* Vancouver Canada, Computer Music Association, 1985. Music Dept. Tech. Rep. STAN–M–31, Stanford University, July 1985.

J. O. Smith, "Efficient Simulation of the Reed-Bore and Bow-String Mechanisms," *Proc. 1986 International Computer Music Conference,* The Hague, Netherlands.

J. O. Smith, "Music Applications of Digital Waveguides," (A compendium containing four related papers and presentations.) CCRMA Tech. Rep. STAN–M–67, Stanford University, 1987, (415)723-4971.

J. O. Smith, "Efficient Yet Accurate Models for Strings and Air Columns using Sparse Lumping of Distributed Losses and Dispersion," *Proc. Colloquium on Physical Modeling,*, Grenoble, 1990. CCRMA Tech. Rep. STAN–M–67, Stanford University.

J. O. Smith, "Waveguide Simulation of Non-Cylindrical Acoustic Tubes," *Proc. 1991 International Computer Music Conference,*, pp. 304–307, Montreal.

R. D. Strum and D. E. Kirk, *First Principles of Discrete Systems and Digital Signal Processing,* Addison-Wesley, Reading MA, 1988.

C. R. Sullivan, "Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback," *Computer Music J.*, vol. 14, no. 3, pp. 26–37, Fall 1990.