Extensions of the Karplus-Strong Plucked-String Algorithm
Author(s): David A. Jaffe and Julius O. Smith
Source: *Computer Music Journal*, Vol. 7, No. 2 (Summer, 1983), pp. 56-69
Published by: The MIT Press
Stable URL: https://www.jstor.org/stable/3680063
Accessed: 06-04-2020 09:39 UTC

David A. Jaffe and Julius O. Smith
Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University
Stanford, California 94305

Extensions of the Karplus-Strong Plucked-String Algorithm

Introduction

In 1960, an efficient computational model for vibrating strings, based on physical resonating, was proposed by McIntyre and Woodhouse (1960). This model plays a crucial role in their recent work on bowed strings (McIntyre, Schumacher, and Woodhouse 1981; 1983), and methods for calibrating the model to recorded data have been developed (Smith 1983).

Independently, in 1978, Alex Strong devised an efficient special case of the McIntyre-Woodhouse string model that produces remarkably rich and realistic timbres despite its simplicity (Karplus and Strong 1983). Since then, Strong and Kevin Karplus have explored several variations and refinements of the algorithm, with an emphasis on small-system implementations. We have found that the Karplus-Strong algorithm can be used with equally impressive results on fast, high-power equipment. The availability of multiplies, for example, allows several modifications and extensions that increase its usefulness and flexibility. These extensions are described in this paper. The developments were motivated by musical needs that arose during the composition of *May All Your Children Be Acrobats* (1981) for computer-generated tape, eight guitars, and voice and *Silicon Valley Breakdown* (1982) for four-channel, computer-generated tape, both written by David Jaffe. Our theoretical approach and the extensions based on it have also been applied to the McIntyre-Woodhouse algorithm (Smith 1983).

David A. Jaffe is also affiliated with the Music Department at Stanford University, and Julius O. Smith is also affiliated with the Electrical Engineering Department there.

Computer Music Journal, Vol. 7, No. 2,
Summer 1983, 0148-9267/83/020056-14 $04.00/0,

The String-Simulation Algorithm

The Karplus-Strong plucked-string algorithm is presented in this issue of *Computer Music Journal*. From our point of view, the algorithm consists of a high-order *digital filter*, which represents the string; and a short *noise burst*, which represents the "pluck."[1] The digital filter is given by the difference equation

$$y_n = x_n + \frac{y_{n-N} + y_{n-(N+1)}}{2}, \qquad (1)$$

where x_n is the input signal amplitude at sample n, y_n is the output amplitude at sample n, and N is the (approximate) desired pitch period of the note in samples. The noise burst is defined by

$$x_n = \begin{cases} Au_n, & n = 0, 1, 2, \ldots, N-1 \\ 0, & n \geq N, \end{cases}$$

where A is the desired amplitude, and $u_n \in [-1,1]$ is the output of a random-number generator. The output y_n is taken beginning at time $n = N$ in our implementation.

Analysis of the String Simulator

Before proceeding to practical extensions of the algorithm, we will describe the theory on which many of them are based. Various concepts from digital filter theory are employed. For a tutorial introduction to digital filter theory, see the works by Smith (1982b) and Steiglitz (1974).

The input-output relation of Eq. (1) may be ex-

1. In some situations, the sound more closely resembles a string struck with a hammer or mallet than one plucked with a pick, but we will always use the term *pluck* when referring to the excitation.

pressed differently by means of *delay-operator* notation. We define the unit-sample delay operator d by the relation

$$d^k x_n \triangleq x_{n-k},$$

where x_n is an arbitrary signal, and k is an integer. (The symbol $\triangleq$ means "is defined as.") Thus, multiplying a signal by d^k delays the signal in time by k samples. In these terms, Eq. (1) becomes

$$y_n = x_n + \frac{d^N y_n + d^{N+1} y_n}{2}$$

$$= x_n + d^N \frac{1 + d}{2} y_n.$$

Solving for y_n yields

$$y_n = \frac{x_n}{1 - \frac{1+d}{2} d^N}. \qquad (2)$$

We can convert linear delay-operator equations immediately to z-transform equations by replacing each time signal with its z-transform, and replacing d with z^{-1}. It is customary to denote a time signal in lowercase letters (e.g., x_n) and the corresponding z-transform in uppercase letters (e.g., $X(z)$). The *transfer function* of a (linear, time-invariant) digital filter is the z-transform of the output signal divided by the z-transform of the input. The transfer function of the string simulator is then found to be

$$H(z) \triangleq \frac{Y(z)}{X(z)} = \frac{1}{1 - \frac{1+z^{-1}}{2} z^{-N}}$$

$$= \frac{1}{1 - H_a(z) H_b(z)},$$

where

$$H_a(z) \triangleq \frac{1 + z^{-1}}{2}$$

$$H_b(z) \triangleq z^{-N}.$$

This form of the description is shown in Fig. 1. The feedback loop consists of a length N delay line $H_b(z)$ in series with a two-point average $H_a(z)$. Corresponding to this breakdown of the string simulator is the following set of difference equations:

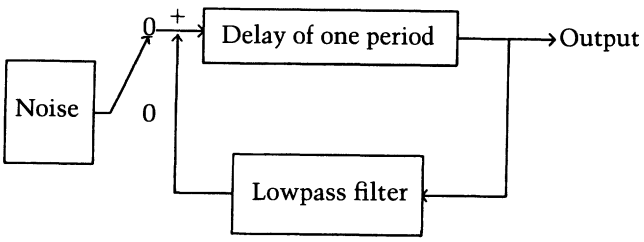


Fig. 1. Block diagram for the basic string simulator.

$$v_n = y_{n-N}$$

$$w_n = \frac{v_n + v_{n-1}}{2}$$

$$y_n = x_n + w_n.$$

The *frequency response* of a digital filter is defined as the transfer function evaluated at $z = e^{j\omega T_s} = \cos(\omega T_s) + j \sin(\omega T_s)$, where T_s is the sampling period in seconds (T_s is the inverse of the sampling rate f_s), $\omega = 2\pi f$ is radian frequency, f is frequency in Hz, and $j = \sqrt{-1}$. The frequency response of the string simulator is then

$$H(e^{j\omega T_s}) = \frac{1}{1 - H_a(e^{j\omega T_s}) H_b(e^{j\omega T_s})},$$

where

$$H_a(e^{j\omega T_s}) = \frac{1 + e^{-j\omega T_s}}{2} = e^{-j\omega T_s/2} \cos(\omega T_s/2)$$

$$= e^{-\pi f T_s} \cos(\pi f T_s)$$

$$H_b(e^{j\omega T_s}) = e^{-j\omega N T_s} = e^{-j2\pi f N T_s}.$$

In this paper it is necessary to consider the amplitude response and phase delay of the feedback filters separately. The *amplitude response* is defined as the magnitude of the frequency response, and it gives the *gain* of the filter as a function of frequency. The *phase delay* is defined as minus the complex angle of the frequency response divided by radian frequency, and it gives the *time delay* (in seconds) experienced by a sinusoid at each frequency.

The amplitude response of each component filter is given by

$$G_a(f) \triangleq |H_a(e^{j\omega T_s})| = |\cos(\omega T_s/2)| = |\cos(\pi f T_s)|$$

$$G_b(f) \triangleq |H_b(e^{j\omega T_s})| = 1.$$

Thus the delay line H_b is lossless, and the two-point average H_a exhibits a gain that decreases with frequency according to the first quadrant of a cosine. We will assume hereafter that all frequencies are restricted to the Nyquist limit, that is, $|f| \leq f_s/2$. In this range, we have $|\cos(\pi f T_s)| = \cos(\pi f T_s)$.

It is convenient to define phase delay in units consisting of samples rather than seconds. The phase delays of H_a and H_b in samples are given by

$$P_a(f) \triangleq -\frac{\angle H_a(e^{j\omega T_s})}{\omega T_s} = \frac{1}{2},$$

$$P_b(f) \triangleq -\frac{\angle H_b(e^{j\omega T_s})}{\omega T_s} = N.$$

($\angle z$ denotes the complex angle of z). The two-point average has a phase delay equal to half a sample, and the delay line has a phase delay equal to its length.

Since the total loop consists of H_a and H_b in series, the loop gain and effective loop length are

$$\textit{loop gain} = G_a(f)G_b(f) = \cos(\pi f T_s),$$

and

$$\textit{loop length} = P_a(f) + P_b(f) = N + 1/2 \quad (\textit{samples})$$

for each sinusoidal frequency f Hz.

In synthesizing a single plucked-string note, we feed in N samples of white noise at amplitude A and listen to the output immediately afterward. It is equivalent to initialize the delay line H_b with sealed random numbers at time 0 and employ no input signal. Since the two-point average H_a is constantly changing the contents of the loop, the output signal is not periodic. It is close to periodic, however, and we use the term *period* in this loose sense. Each period of the synthetic string sound corresponds to the contents of the delay line at a particular time, and each period equals a somewhat lowpass version of the previous period. More precisely, a running two-point average of the samples comprising one period gives the next period in the output waveform. Since the effective loop length is $N + 1/2$ samples, the period is best defined to be $NT_s + T_s/2$ sec. Experience shows this to correspond well with perceived pitch.

Decay of "Harmonics"

Since the signal is only quasi-periodic, it does not consist of discrete sinusoids. Essentially, we have many narrow "bands" of energy decaying to zero at different rates. When these energy bands are centered at frequencies that are an integer multiple of a lowest frequency, they will be referred to as *harmonics*. When the frequency components are not necessarily uniformly spaced, the term *partial* will be used to emphasize the possibility of inharmonicity. Consider, then, a partial at frequency f Hz circulating in the loop. On each pass through the loop, it suffers an attenuation equal to the loop-amplitude response, $G_a(f)G_b(f) = \cos(\pi f T_s)$; that is,

$$\textit{one period's attenuation} = \cos(\pi f T_s).$$

Since the round-trip time in the loop equals $N + 1/2$ samples, the number of trips through the loop after n samples (nT_s sec) is equal to $n/(N + 1/2) = tf_s/(N + 1/2)$. Thus the *attenuation factor* at time $t = nT_s$ is given by

$$\alpha_f(t) \triangleq [\cos(\pi f T_s)]^{\frac{tf_s}{N+1/2}}. \quad (3)$$

For example, an initial partial amplitude A at time 0 becomes amplitude $A\alpha_f(t)$ at time t seconds, where f is the frequency of the partial.

The *time constant* of an exponential decay is traditionally defined as the time when the amplitude has decayed to $1/e \approx 0.37$ times its initial value. The time constant at frequency f is found by equating Eq. (3) to e^{-t/τ_f} and solving for τ_f, which gives

$$\tau_f = \frac{-t}{\ln \alpha_f(t)} = -\frac{\left(N + \frac{1}{2}\right) T_s}{\ln \cos(\pi f T_s)} \quad (\textit{seconds}). \quad (4)$$

For audio, it is normally more useful to define the time constant of decay as the time it takes to decay −60 db, or to 0.001 times the initial value. In this case, we equate Eq. (3) to 0.001 and solve for t. This value of t is often called t_{60}. Conversion from τ_f to

$t_{60}(f)$ is accomplished by

$$t_{60}(f) = \ln(1000)\tau_f \approx 6.91\tau_f. \quad (5)$$

For example, if a sinusoid at frequency f Hz has amplitude A at time 0, then at time $t_{60}(f)$ it has amplitude $A\alpha_f(t_{60}(f)) = A/1000$, or it is 60 db below its starting level.

The above analysis describes the attenuation due to "propagation" around the loop. It does not, however, incorporate the fact that sinusoids that do not "fit" in the loop are quickly destroyed by self-interference. This situation is analogous to making an actual string vibrate. Any signal may be "fed into" the string, but after the input ceases, the remaining energy quickly assumes a quasi-periodic nature. Thus, even though the loop is initialized with random numbers, after a very short time the primary frequencies present in the loop are those that have an integral number of periods in $N + 1/2$ samples. These frequencies are all multiples of the frequency whose period exactly matches the loop length $N + 1/2$. This lowest frequency provides the fundamental, or pitch frequency, of the note:

$$f_1 \triangleq \frac{1}{\left(N + \frac{1}{2}\right) T_s} = \frac{f_s}{N + \frac{1}{2}}. \quad (6)$$

Setting f to the harmonic series beginning with f_1,

$$f_k = \frac{\omega_k}{2\pi} \triangleq k \frac{f_s}{N + \frac{1}{2}}, \quad k = 1, 2, \ldots, N/2, \quad (7)$$

gives the decay factor at time t for the k^{th} harmonic to be

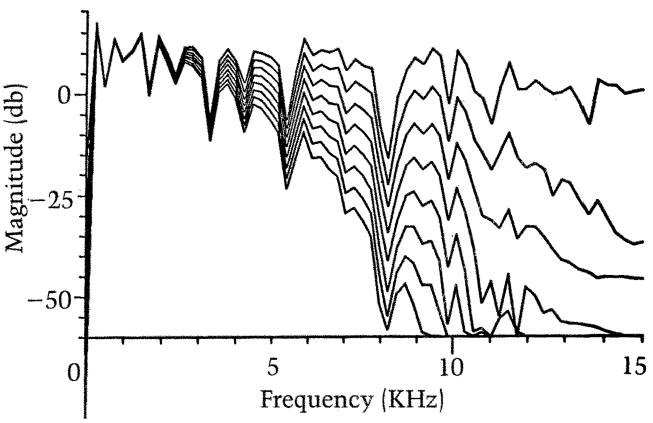
$$\alpha_k(t) = [\cos(\pi f_k T_s)]^{f_1 t}. \quad (8)$$

Similarly, the time contrast per harmonic is given by

$$\tau_k \triangleq \frac{-t}{\ln \alpha_k(t)} = -\frac{1}{f_1 \ln \cos(\pi f_k T_s)} \quad (\textit{seconds}). \quad (9)$$

Figure 2 shows the spectral evaluation during the first 16 periods of a note having a period of 128 samples. A 128 length Fast Fourier Transform (FFT) was computed every other period. Each curve in the figure is interpreted as the envelope of the harmonic amplitudes, since a straight line is drawn from one harmonic amplitude to the next.

Fig. 2. Spectral evolution during the first 16 periods.



In certain extensions to the algorithm, H_a is other than a two-point average. In such a case, the attenuation factor of the k^{th} harmonic after t seconds is approximately

$$\alpha_k(t) = G_a(f_k)^{\frac{tf_s}{N + P_a(f_k)}}, \quad (10)$$

where we require $G_a(f) \leq 1$ for stability. The phase delay $P_a(f_k)$ of H_a may be used to create inharmonic spectra (Smith 1983). The spectrum is harmonic only when $P_a(f_k)$ is the same at all harmonic frequencies f_k.

Similarly, when H_a is more general, the time constant of decay for each harmonic becomes

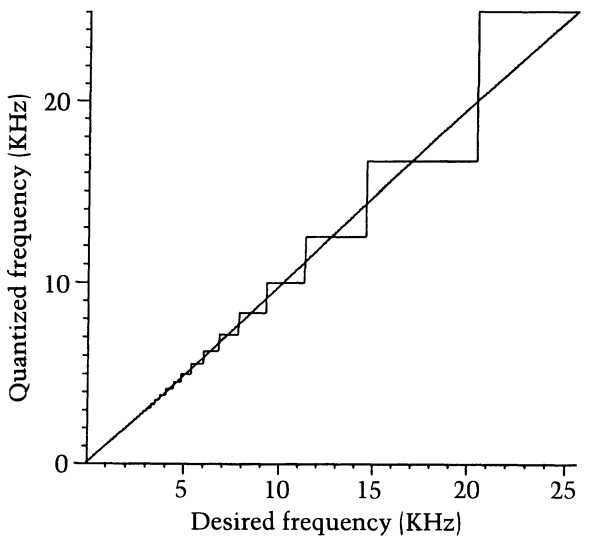
$$\tau_k = -\frac{N + P_a(f_k)}{f_s \ln G_a(2\pi f_k T_s)} \quad (\textit{seconds}). \quad (11)$$

Having provided an analytic vocabulary, we now proceed to a detailed examination of our additions to the Karplus-Strong algorithm.

Tuning

The fact that the delay-line length N must be an integer causes tuning problems. Since the fundamental frequency is $f_1 = f_s/(N + 1/2)$, the allowed pitches are quantized, especially at high frequency. For large values of N (low pitches), the difference

Fig. 3. Desired pitch versus resulting pitch for a 50-KHz sampling rate.



between the pitch at N and $N + 1$ is very slight. However, for high pitches, N and $N + 1$ yield very different pitches and tuning becomes crude. Figure 3 shows the distortion in frequency for the sampling rate 50 KHz. For lower sampling rates, the curve is identical in form, and the distortion occurs at proportionately lower frequencies.

The key to a solution for this problem lies in the expression for loop length in terms of phase delay. The fundamental frequency is given by

$$f_1 = \frac{f_s}{N + P_a(f_1)},$$

where $P_a(f) = 1/2$ when the two-point average is used for H_a. To make up the difference between f_1 and the desired frequency, we need to introduce into the feedback loop a filter that can contribute a small delay without altering the loop gain. The filter we introduce has the difference equation

$$y_n = Cx_n + x_{n-1} - C_{y_{n-1}} \tag{12}$$

and transfer function

$$H_c(z) \triangleq \frac{C + z^{-1}}{1 + Cz^{-1}},$$

where C is the only coefficient to be set. For stability, we must have $|C| < 1$. It can be shown that when the input x_n is bounded by 1, the output is bounded by $2|C| + 1$. The transfer function of the whole string is now

$$H(z) \triangleq \frac{1}{1 - H_a(z)H_b(z)H_c(z)}.$$

The filter H_c is a first-order *allpass* filter, and as such it has a constant amplitude response. Indeed, the amplitude response is simply

$$G_c(f) \triangleq |H_c(e^{j\omega T_s})| = \frac{|C + e^{-j\omega T_s}|}{|1 + Ce^{-j\omega T_s}|} = 1.$$

The use of an allpass filter ensures that no modification of the decay rate will take place. The loop gain is $G_a(f)G_b(f)G_c(f) = \cos(\pi f T_s)$ as before.

We will select the phase delay of H_c so as to tune f_1 to the precise desired frequency. This requires only the ability to select phase delays between 0 and T_s sec, or one sample's worth.

The phase delay of the first-order allpass H_c is given by

$$\begin{aligned} P_c(f) &\triangleq -\frac{\angle H_c(e^{j\omega T_s})}{\omega T_s} \\ &= \frac{-1}{\omega T_s} \angle \frac{C + e^{-j\omega T_s}}{1 + Ce^{-j\omega T_s}} \\ &= \frac{\angle(1 + Ce^{-j\omega T_s})}{\omega T_s} - \frac{\angle(C + e^{-j\omega T_s})}{\omega T_s} \\ &= \frac{1}{\omega T_s} \tan^{-1}\left(\frac{-C \sin(\omega T_s)}{1 + C\cos(\omega T_s)}\right) \\ &\quad - \frac{1}{\omega T_s} \tan^{-1}\left(\frac{-\sin(\omega T_s)}{C + \cos(\omega T_s)}\right). \end{aligned} \tag{13}$$

When the arguments to the arc tangent above have magnitude less than unity, we can use the power-series expansion (Abramowitz and Stegun 1966),

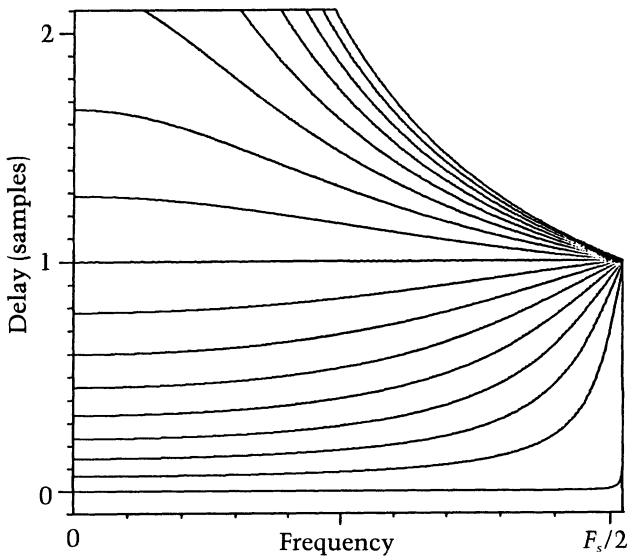
$$\tan^{-1}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots, \quad |x| < 1.$$

Thus we can approximate the *low-frequency* phase delay by

$$\begin{aligned} P_c(f) &\approx \frac{\sin(\omega T_s)}{\omega T_s(C + \cos(\omega T_s))} - \frac{C \sin(\omega T_s)}{\omega T_s(1 + C\cos(\omega T_s))} \\ &\approx \frac{1}{C + 1} - \frac{C}{1 + C} = \frac{1 - C}{1 + C}. \end{aligned} \tag{14}$$

A plot of the exact phase delay is given in Fig. 4 for

Fig. 4. Phase delay for the fine-tuning allpass filter $H_c(z) = (C + z^{-1})/(1 + C z^{-1})$.



17 values of C equally spaced between -0.999 and 0.999. Note that delays between 0 and 1 sample can be provided somewhat uniformly across the frequency axis. A delay of 0 samples corresponds to $C = 1$, where the pole and zero of $H_c(z)$ cancel to give $H_c(z) \equiv 1$. However, pole-zero cancellation on the unit circle is not a good thing in practice, since round-off errors may yield an unstable filter. Therefore, it is preferable to shift the range of one-sample delay control to the region $\varepsilon \leq P_c \leq (1 + \varepsilon)$ for some small nonnegative ε $(0 < \varepsilon \ll 1)$. It is best not to shift very far, since the phase-delay curves are less flat in the region beyond one sample's delay.

Note that the delay curves below the one-sample level in Fig. 4 correspond to slightly flattened upper partials, while the delay curves above the one-sample level correspond to slightly sharpened upper partials. The timbre change due to slight systematic shifting of the upper partials, of an amount less than one sample period, was found to be hardly noticeable. It may even be desirable as a source of subtle timbral variation. In any case, it is important to get the perceived pitch right.

To tune the instrument precisely to a desired fundamental frequency f_1, let P_1 equal f_s/f_1, the real value for the period of the first partial, in samples, which would give perfect tuning. Then we desire $N + P_a(f_1) + P_c(f_1) = P_1$. The integer buffer length N and the delay $P_c(f_1)$ required from the allpass filter become

$$\begin{aligned} N &\triangleq \text{Floor}(P_1 - P_a(f_1) - \varepsilon) \\ P_c(f_1) &\triangleq P_1 - N - P_a(f_1), \end{aligned} \tag{15}$$

where $\varepsilon > 0$ is the offset that shifts $P_c(f_1)$ into the range $[\varepsilon, 1+\varepsilon]$, and $P_a(f_1)$ is the delay in samples due to the filter H_a. In the simple case where H_a is a two-point average, $P_a(f_1) = 1/2$.

We next solve for the filter coefficient in Eq. (13) as a function of $P_c(f_1)$. Taking the tangent of both sides, and using an identity for the tangent of a difference leads to the quadratic equation in C,

$$C^2 \sin(\omega_1 T_s P_c(f_1) + \omega_1 T_s) + 2C \sin(\omega_1 T_s P_c(f_1)) + \sin(\omega_1 T_s P_c(f_1) - \omega_1 T_s) = 0,$$

where $\omega_1 \triangleq 2\pi f_1$. The solution is found, after some manipulation (Mont-Reynaud 1982), to be

$$C = \frac{-\sin(\omega_1 T_s P_c(f_1)) \pm \sin(\omega_1 T_s)}{\sin(\omega_1 T_s P_c(f_1) + \omega_1 T_s)}.$$

We have introduced an extra root by producing a quadratic equation. The previous approximation Eq. (14) indicates that the + sign should be taken. Therefore, the final solution is

$$\begin{aligned} C &= \frac{\sin(\omega_1 T_s) - \sin(\omega_1 T_s P_c(f_1))}{\sin(\omega_1 T_s P_c(f_1) + \omega_1 T_s)} \\ &= \frac{\sin\left(\dfrac{\omega_1 T_s - \omega_1 T_s P_c(f_1)}{2}\right)}{\sin\left(\dfrac{\omega_1 T_s + \omega_1 T_s P_c(f_1)}{2}\right)}, \end{aligned} \tag{16}$$

which can be approximated, at low frequencies, by

$$C \approx \frac{1 - P_c(f_1)}{1 + P_c(f_1)}. \tag{17}$$

Although this technique provides a precise fundamental frequency, it does not guarantee an in-tune *percept*, since the perceived pitch does not always coincide with the fundamental frequency. An additional mapping onto a perceptual tuning dimension may be needed for the very high notes. In our case, it was found that tuning the octaves slightly stretched, as is done in piano tuning, gives a more satisfying in-tune percept.

Decay-time Alteration

The basic algorithm naturally results in a shorter decay time for high pitches than for low pitches, reflecting the behavior of real strings. This is due to two effects. First, higher frequencies are more attenuated by the two-point average H_a; second, higher pitch means more trips through the attenuating loop in a given time. This may be seen in Eq. (3).

Unfortunately, the range of decay times between the high and low pitches is too extreme. The high-pitched notes die away so fast that only a click is perceived, while the low-pitched notes last for an unnaturally long time. In addition, the decay time of real strings varies with many factors such as tension, length, thickness, and material. Consequently, we have found it useful to add a means for altering the note duration. The ability to control decay time is essential for a realistic simulation as well as for musical flexibility.

On systems that have separate control of the sampling rate of each voice, sampling-rate change can be used to control decay time. On other systems, however, it is necessary to use other methods to alter decay time.

Decay Shortening

To shorten the decay time, a loss factor ρ can be introduced in the feedback loop. With the loss factor, the difference Eq. (1) for the string becomes

$$y_n = x_n + \rho \frac{y_{n-N} + y_{n-(N+1)}}{2}. \tag{18}$$

The amplitude envelope of a sinusoid at frequency f, previously given by Eq. (8), is now proportional to

$$\alpha_f(t, \rho) = |\rho \cos(\pi f T_s)|^{f_1 t} = |\rho|^{f_1 t} \alpha_f(t).$$

Thus all partials are affected equally: the relative decay rates are unchanged.

Note that ρ cannot be used to lengthen the decay time, since the amplitude at 0 Hz would increase exponentially. In general, we must have $|\rho| \leq 1$ if the string is to be stable. Thus ρ is used to shorten the low-pitched notes to make them more comparable in duration with notes from a real string. With the loss factor operative, the decay-time constant for the fundamental frequency becomes

$$\tau_1(\rho) = -\frac{1}{f_1 \ln |\rho \cos(\pi f_1 T_s)|}. \tag{19}$$

Decay shortening produces a damped version of the algorithm, analogous to substitution of a soft material for the bridge of a string instrument.

Decay Stretching

To stretch the decay, the feedback average (H_a) can be changed to a two-point weighted average. This reduces the amount of loss at high frequencies. Thus, we replace $H_a(z)$ by

$$H_a(z, S) = (1 - S) + Sz^{-1}, \tag{20}$$

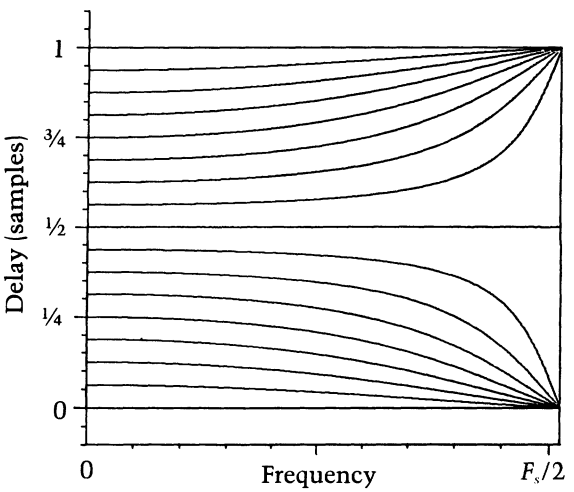
where S, the stretching factor, is between 0 and 1. The gain of this filter is

$$\begin{aligned} G_a(f, S) &= |(1 - S) + Se^{-j\omega T_s}| \\ &= \sqrt{((1 - S) + S\cos \omega T_s)^2 + (S \sin(\omega T_s))^2} \\ &= \sqrt{(1 - S)^2 + S^2 + 2S(1 - S)\cos \omega T_s} \\ &\quad . \end{aligned} \tag{21}$$

With $S = 1/2$, $H_a(z, S)$ reduces to the previous case $H_a(z)$. For stability of the overall string, we must have $0 < S < 1$. If $S = 0$ or 1, the frequency-dependent term disappears, and the gain response is unity for all f; in this case, the initial white-noise burst circulates forever in the loop, producing harmonics that never decay. At intermediate values, $0 < S < 1$, the effective note duration (t_{60}) is finite, and it is minimum for $S = 1/2$. The amplitude trajectory and the decay-time constant for each partial can be obtained by substituting Eq. (21) into Eqs. (10) and (11), respectively.

For the greatest control, both the uniform-loss method and the weighted two-point-average method may be used for decay-time alteration. The resulting decay time is then a function of loss factor ρ and stretch factor S. Karplus and Strong (1983) describe a method of decay stretching that uses no multiplies.

Fig. 5. Phase delay for the decay-stretching, one-zero filter $H_a(z, S) = (1 - S) + S z^{-1}$.



Effect of Decay Stretching on Tuning

Changing S changes the effective loop length as a function of frequency, since it changes the phase delay of the overall loop. We must therefore compute $P_a(f_1)$ for use in Eq. (15) when fine tuning with the allpass filter H_c. The phase delay of the weighted two-point average is given by

$$P_a(f, S) \triangleq -\frac{\angle H_a(e^{j\omega T_s}, S)}{\omega T_s}$$

$$= -\frac{\angle[(1 - S) + Se^{-j\omega T}]}{\omega T_s}$$

$$= -\frac{1}{\omega T_s}\tan^{-1}\left(\frac{-S\sin(\omega T)}{(1 - S) + S\cos(\omega T)}\right), \quad (22)$$

and for low frequencies, relative to the sampling rate, we may use the approximation

$$P_a(f, S) \approx \frac{S\sin(\omega T_s)}{\omega T_s(1 - S) + S\omega T_s\cos(\omega T_s)}$$

$$\approx S, \qquad 0 \leq S \leq 1.$$

For $S = 1/2$, we have the basic string algorithm, and the phase delay of H_a is 1/2, as given by the above approximation. For other values of S the approximation is always precise at $f = 0$. Figure 5 shows the true phase-delay curves of $H_a(z, S)$ as S is stepped uniformly through 17 values from 0 to 1.

Note that for $S \geq 0$ the phase delay is quite flat over most of the frequency axis. Another point of interest is that since $G_a(z, S) = G_a(z, 1 - S)$, we may choose the case that yields the best phase delay curve for the fine-tuning allpass H_c.

Since the tuning calculation needs to be done only once per note, the precise form of Eq. (22) can be used for each new frequency without much computational expense.

Dynamics

The loudness of the signal output by the algorithm is a function of the amplitude of the input noise burst. However, this is an unsatisfactory control in simulating the timbral effect of dynamic level as it occurs in the case of a real string instrument. The effect of varying initial amplitude gives the impression of a change more in the distance between the listener and the apparent source than in dynamics. Since strings plucked hard have more energy in the higher partials than strings plucked lightly (due to nonlinearities becoming important), the dynamic simulation is based on modeling this difference in spectral balance. We therefore change the effective spectral bandwidth of a note to modulate its apparent intensity.

The bandwidth is controlled by means of a one-pole, lowpass filter applied to the initial noise burst (before it is fed into the string). This filter will be referred to as the *dynamics filter*. The difference equation of the dynamics filter is

$$y_n = (1 - R)x_n + Ry_{n-1},$$

and its transfer function is

$$H_d(z) \triangleq \frac{1 - R}{1 - Rz^{-1}}, \quad (23)$$

where R is a real number between 0 and 1, computed as a function of fundamental frequency f_1 and the desired dynamic level L. When a series of notes at pitch f_1 is played while R is moved gradually toward 1, a diminuendo is approximated in terms of both decreasing loudness and spectral bandwidth reduction.

We define the *dynamic level* L as a bandwidth be-

tween 0 and $f_s/2$. If L is small, the spectrum is more lowpass filtered, corresponding to a softer dynamic level. Conversely, large L gives a bright spectrum corresponding to louder notes. It is not sufficient to use a fixed lowpass filter for all pitches, since low-pitched notes would then be louder than high-pitched notes. Rather, for a given dynamic level, R must be changed with pitch to yield a uniform perceived loudness. While this is a difficult problem in general, a good approximation is obtained by varying R so that the amplitude of the fundamental frequency is constant.

It remains to be shown how R is computed for a given pitch f_1 and dynamic level L. The main steps are as follows. First, a one-pole, lowpass filter is designed having bandwidth L. Second, the gain of this filter at a "middle" frequency is computed. Third, the dynamics filter is computed as a one-pole, lowpass filter having this gain at the desired fundamental f_1. The remainder of this section gives the equations needed for these steps.

The reference frequency f_m is chosen as the logarithmic middle (geometric mean) of the range to be used (a function of the particular musical context and the sampling frequency):

$$f_m = e^{(1/2)[\log(f_u) + \log(f_l)]} = \sqrt{f_u f_l},$$

where f_u is the upper pitch limit ($< f_s/2$), and f_l is the lower pitch limit.

The one-pole lowpass filter having bandwidth L is given by

$$H_L(z) \triangleq \frac{1 - R_L}{1 - R_L z^{-1}},$$

where

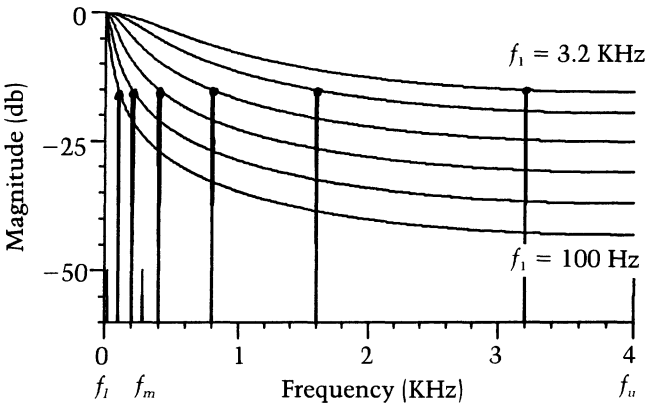
$$R_L \triangleq e^{-\pi L T_s}.$$

The substitution $R_L = e^{-\pi L T_s}$ is a somewhat standard approximate formula for mapping bandwidth to pole radius.

The gain of the lowpass filter H_L at the reference frequency is defined as

$$G_L \triangleq |H_L(e^{j2\pi f_m T_s})| = \frac{1 - R_L}{|1 - R_L e^{-j2\pi f_m T_s}|}.$$

Fig. 6. Frequency response of $H_d(z) = (1 - R)/(1 - R z^{-1})$ computed at dynamic level $L = 100$ for six values of f_1.



Now, for any desired fundamental frequency f_1, R is computed so as to provide gain $G_d(f_1) = G_L$. In other words, all fundamental frequencies are made to have the same amplitude. The value of R is found by solving

$$G_L = \frac{1 - R}{|1 - Re^{-j2\pi f_1 T_s}|}.$$

Squaring both sides of this equation and solving the resulting quadratic polynomial in R yield

$$R = \frac{1 - G_L^2 \cos(2\pi f_1 T_s)}{1 - G_L^2} \pm 2G_L \sin(\pi f_1 T_s) \frac{\sqrt{1 - G_L^2 \cos^2(\pi f_1 T_s)}}{1 - G_L^2}.$$

We use whichever value is < 1 in magnitude to ensure stability.

A family of frequency-response curves for H_d is shown in Fig. 6 for six fundamental frequencies in octave steps from $f_1 = 100$ Hz to $f_1 = 3{,}200$ Hz. The dynamic level in each case is $L = 100$ Hz. A vertical line is drawn to each curve at the fundamental frequency to which it applies. The reference frequency f_m is set to 282.84 Hz (the geometric mean of $f_l = 20$ Hz and $f_u = f_s/2$), and the sampling rate is $f_s = 8{,}000$ Hz.

To add to the effect of simulated dynamics, it is sometimes helpful to do a bit of decay shortening on the low soft notes, using a loss factor ρ as described previously. It is also possible to simulate the spectral characteristics of soft notes by simply

turning the output of the algorithm on late, after some of the high-frequency energy has died away. This latter method has the effect of diminishing the attack noise, corresponding to a milder pluck. For other decay-softening techniques, see the section entitled Varying Attack Characteristics.

Rests and the Ends of Notes

In the case of a real string instrument, the player often plays two notes on the same string, playing the second note before the first has died away. The basic algorithm handles this without a problem for all but very high pitches, since the discontinuity at the beginning of the new note is perceived as a new pluck rather than as a click. (With a real instrument, there is a short time when the pick has muted the vibration for the previous note but has not yet set the string again into vibration, but this brief silence was not found to be necessary for realism in the synthetic model.) A problem arises, however, when there is a rest after a note. If the output of the algorithm is abruptly turned off (replaced by zeros), a discontinuity in the waveform results, causing a click.

Even if the note is allowed to decay for a very long time, a note turned off abruptly may cause a click because the feedback loop has unity gain at 0 Hz. To see this, note that $G_a(0) = \cos(0) = 1$. Thus the final value of the waveform is the mean of the initial noise burst. Since the pseudo-random-number sequence used to initially excite the filter has a mean of zero only in special cases or when an infinite number of samples is taken, the 0 Hz component can be significant. For uniform pseudorandom white noise at amplitude A, the statistical variance is $A^2/3$, which implies a variance in sample mean over N samples equal to $A^2/(3N)$. Thus the standard deviation of the mean in a length of N samples is $A/\sqrt{3N}$. As N decreases, the probability of having a large amount of energy at 0 Hz becomes greater. Karplus and Strong describe a technique they call "dithering" to handle this problem, but this technique was not economical in our context.

Our solution to the discontinuity problem is as follows: the loss factor ρ, as was discussed for decay shortening, is set to a relatively small value in the last few milliseconds of the note. The duration of the decay is dependent on the loss factor chosen. A loss factor close to 1 simulates a string being damped with a soft material such as the flesh of the finger, while a smaller loss factor simulates damping with a hard material, such as a pick. It is useful to compute ρ as a function of a desired t_{60}. Substituting Eq. (19) into Eq. (5) and solving for ρ yields

$$\rho(t_{60}) = \frac{e^{-1/f\tau}}{|\cos(\pi f T_s)|}, \quad \tau \triangleq \frac{t_{60}}{\ln(1000)}.$$

For pitches above about 3 KHz, clicks can appear even at the onset of notes. Onset clicks can also occur when a pianissimo setting of the dynamics filter is used, since the masking effect of the onset noise burst is weakened by the dynamic filter. To alleviate this problem, it may be necessary to multiply the output of the algorithm by an exponential or linear envelope, rather than switching it on abruptly.

Glissandi and Slurs

A slur can be simulated by changing the order of the filter without reexciting it; that is, by changing the delay buffer length N. The result is analogous to a performer's refretting a string without replucking it (what guitarists call the "hammer-on" and "pull-off" technique). A rapid alternation of ascending and descending pitch changes gives a good left-hand-trill effect.

If the buffer length is gradually changed over time, a crude glissando results. For low pitches and high sampling rates, it can be quite smooth, but for higher pitches and lower sampling rates, the pitch quantization resulting from the integer-delay lengths becomes noticeable. In the upper range, the effect is similar to a glissando on a fretted instrument, where the pitch changes in discrete steps, and it can be musically useful. Of course, the synthetic quantization is not in semitone intervals.

A perfectly smooth glissando can be created by ramping C, the tuning coefficient, during the time between buffer-length changes. This technique can also be used to create vibrato.

Sympathetic String Simulation

In a real string instrument, sympathetic vibrations of other open strings, as well as resonances in the instrument body, give each pitch in the range of the instrument an individual character and thus give the instrument as a whole a distinctive identifiable character. In comparison, the basic form of the Karplus-Strong algorithm, like many instrument-simulation algorithms, has an excessive homogeneity of character throughout its range. One way to remedy this situation is to feed the output of the string into a body resonator. This technique has produced impressive results, but its discussion is beyond the scope of this paper (see Smith 1983). Another way to combat unnatural uniformity is to create the effect of an instrument with sympathetic strings, using a modified version of the basic algorithm.

Just as a sympathetic string is set into motion by the vibration of another string, the illusion of a sympathetically vibrating string can be created by exciting one copy of the string simulator by a small percentage of the output from another (plucked) string, tuned to a different pitch.

In the discussion that follows, the algorithm that is excited with the noise burst is referred to as the *plucked string* and the algorithm that is excited only by the plucked string is referred to as the *sympathetic string*. All partials of the plucked string that do not coincide with those of the sympathetic string will be highly attenuated. Thus the sympathetic string acts as a bank of very narrow bandpass filters with center frequencies at the partial frequencies of the sympathetic string. The partials of the plucked string that will strongly resonate are those for which

$$f_j = f'_k,$$

where f'_k is the frequency of the k^{th} partial of the sympathetically resonating string, and f_j is the frequency of the j^{th} partial of the plucked string.

A problem can arise after several successive noise bursts have excited the plucked string. The repeated reintroduction of energy into the sympathetic string may cause it to overflow. Therefore, it is essential that a loss factor ρ, such as was introduced with reference to decay shortening, be used to provide energy dissipation.

The effect of several sympathetic strings can be created simply by a bank of parallel sympathetic strings, as defined above, each tuned to a different frequency. The resulting overall string-transfer function (omitting fine tuning and decay alteration for clarity) is then

$$H(z) = \frac{1}{1 - \frac{1+z^{-1}}{2} z^{-N}} \left((1-\gamma) + \gamma \sum_{i=1}^{M} \frac{1}{1 - \rho_i \frac{1+z^{-1}}{2} z^{-N_i}} \right),$$

where γ is the fractional part of the plucked-string signal sent to the sympathetic strings, M is the number of sympathetic strings, ρ_i is the loss factor for the i^{th} sympathetic string, $N_i = f_s/f_i$ where f_i is the fundamental frequency of the i^{th} sympathetic string, and f_s/N is the pitch of the plucked string and hence of the played note.

The sympathetic-string version of the algorithm is also helpful in creating a stereo or quadraphonic image. By distributing around the room the outputs of several banks of differently tuned sympathetic strings, all fed with the same plucked string, the effect of being inside a huge guitar can be created.

Attractive musical results have been created by replacing the plucked string with another computer instrument, so that the bank of sympathetic strings is used as a "reverberator." One can achieve the effect, for example, of a clarinet being played into an open grand piano with the pedal down.

Simulation of a Moving Pick

An effective means of simulating pick position is to introduce zeros uniformly distributed over the spectrum of the noise burst. This can quite accurately simulate the effect of plucking a string at varying distances from the bridge. The noise burst is filtered with a comb filter, H_e, having the difference

equation

$$y_n = x_n - x_{n-\mu N},$$

where μ is the fraction of the string between the bridge and pluck point. When $\mu = 1/2$, the even harmonics are removed, and the effect is that of plucking a string at its midpoint. Similarly, when $\mu = 1/10$, every tenth harmonic is suppressed, and the effect is like plucking a tenth of the way up the string. With $\mu = 1/N$, the filter approximates a differentiator, creating a sharp *sul ponticello* sound. For the theory behind the simulation of pick location, see Smith's paper (1982a).

Varying the Character and Number of Attacks

Since the attack is very important in perception of timbre, it is advantageous to be able to alter its character. To give a noticeably more noisy attack, approximating the sound of a snap or "Bartók pizzicato," the duration of the noise burst x_n can simply be increased from $t_x = NT_s$ to some $t_x > NT_s$. Similarly, the attack can be subdued by making $0 < t_x < NT_s$, though for very small t_x the pluck illusion (as well as loudness) fades.

A variety of other methods can soften the attack. The string can be excited with a rich harmonic spectrum rather than a noise spectrum, or with some mixture of the two, with the sum of their amplitudes not exceeding 1. Another possibility is to lowpass filter the noise burst. Yet another way to soften the attack is to turn on the output of the algorithm late, after some of the high-frequency energy has been filtered out.

A realistic simulation of the up-and-down picking pattern characteristic of a mandolin tremolo has been created by using a one-pole, lowpass filter, H_f, to mellow the "up" picks while using the standard unfiltered noise burst for the "down" picks.

A crude simulation of instruments having multiple strings tuned in unison, such as the mandolin or bazooki, can be created by simply exciting the string with two successive noise bursts separated by a short amount of time, on the order of .05 sec. While multiple attacks can be achieved in this manner, the steady state fuses into a single note. In a real mandolin, the strings are never perfectly tuned, and the beating effect of the slightly mistuned strings is a strong recognition cue. A better mandolin simulation simply uses two parallel forms of the algorithm, differing in pitch by a few cents and excited at slightly different times.

Use of Other Filters in Feedback Loop

The use of filters other than a one-zero for H_a will give a different decay characteristic and, in turn, a different timbre. However, care must be taken that the amplitude response G_a does not reach unity near any partial frequency. Energy at any frequency f_k for which $G_a(f_k) = 1$ will never decay, and if $G_a(f_k) > 1$, the amplitude will grow exponentially until overflow.

For example, a one-pole filter with the pole between 0 and 1 gives a tone with the same attack and a more mellow decay than with a one-zero filter; that is, the higher partials decay more rapidly. Placing the pole at $z = Q$ and normalizing the peak-amplitude response of the one-pole filter to unity yield the transfer function

$$H_a(z) = \frac{1 - |Q|}{1 - Qz^{-1}},$$

where $|Q| < 1$ is required for stability of the one-pole filter. The transfer function of the whole string becomes

$$H(z) = \frac{1}{1 - z^{-N}\frac{1-|Q|}{1-Qz^{-1}}}$$

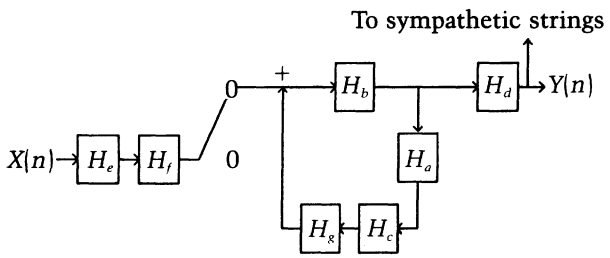
$$= \frac{1 - Qz^{-1}}{1 - Qz^{-1} - (1 - |Q|)z^{-N}}.$$

Thus the difference equation is

$$y_n = x_n - Qx_{n-1} + Qy_{n-1} + (1 - |Q|)y_{n-N}.$$

This version of the algorithm, with $0 \ll Q < 1$ is useful for pitches in the lower half of the range. In the upper range, the notes die away too fast to be of use. This is because the one-pole, lowpass filter, with $Q \gg 0$, filters out the high-frequency energy

Fig. 7. Block diagram containing algorithm extensions discussed in this paper.



in the loop much more drastically than does the one-zero, lowpass filter. A technique similar to that used in dynamics simulation could be used to compensate for this trend.

Simulation of Stiff Strings

The spectral components of the basic algorithm that have significant amplitude are almost perfectly harmonic after the attack noise has been filtered away, corresponding, in the real world, to a perfectly flexible string. But since real strings always have some degree of stiffness, it is desirable to alter the spectrum of the algorithm accordingly. The theory of stiff strings (Morse 1976) indicates that stiffness creates a stretching of the partials according to the approximate formula

$$f_k \approx kf_0\left[1 + \delta + \left(\frac{1 + k^2\pi^2}{8}\right)\delta^2)\right] \triangleq kf_0 s(k),$$

$$k = 1, 2, \ldots, \quad k^2 < \frac{4}{\pi^2\delta^2},$$

where f_0 is near the fundamental frequency, and k is the partial number. The parameter δ has been called the *coefficient of inharmonicity*; if $\delta = 0$, then perfect harmonicity results.

This effect can be created, in principle, by introducing an allpass filter $H_g(z)$ in the string loop (Allen 1982) much as was done for the fine tuning of pitch. The phase delay in samples desired for an allpass filter inserted in the feedback loop of a harmonic string simulator tuned to f_1 is given by solving

$$f_k = \frac{k}{P_0 + P_g(f_k)} = \frac{kf_1 s(k)}{s(1)}$$

$$\Rightarrow \quad P_g(f_k) = \frac{s(1)}{f_1 s(k)} - P_0,$$

where P_0 is the length of the loop in the absence of the allpass H_g (typically $P + 1/2$). Methods for designing allpass filters with prescribed phase delay are reviewed by Smith (1983).

Summary

Figure 7 shows a block diagram of the string simulator with some of our revisions, where H_a is the feedback lowpass filter, H_b is the delay line, H_c is the allpass filter used for tuning, H_d is the lowpass filter used to simulate dynamics, H_e is the comb filter that simulates pick position, H_f is the filter that simulates the difference between "up" and "down" picking, and H_g is the allpass filter used to simulate string stiffness.

The simulator provides a high degree of flexibility that begins to approach that of a skilled player performing on a real musical instrument. Many aspects of a real string instrument have been simulated. Pitch can be precisely specified, and articulation can be finely tuned. An expressive vocabulary is provided by a wide variety of performance nuances, including such "left-hand" techniques as glissandi, slurs, and trills, as well as such "right-hand" techniques as variation in dynamic level, pick position, and attack characteristics. These parameters were found to be sufficient to create shaped musical phrases. Furthermore, parameters such as sustain time, body resonance, string flexibility, bridge and pick hardness, and degree of sympathetic string excitation, which, in the case of real instruments, are usually fixed at the time of instrument construction, are available as performance parameters. It is important to point out that this variety is at no time achieved at the expense of the integrity of the basic sound. Rather, as is the case with a real musical instrument, the diversity exists within the bounds of a clearly defined sound domain.

Conclusion

The algorithm originated by Karplus and Strong and extended by the methods outlined here has proven very useful as a computer instrument. In the process of composing *May All Your Children Be Acrobats* and *Silicon Valley Breakdown*, it was found to be sufficiently flexible to allow for a wide range of musical expression and sufficiently idiosyncratic to maintain a characteristic identity. We expect that new refinements of the algorithm will continue to arise.

References

Abramowitz, M., and I. A. Stegun, eds. 1966. *Handbook of Mathematical Functions*. Washington, D.C.: National Bureau of Standards.

Allen, J. B. 1982. Private communication.

Karplus, K., and A. Strong. 1983. "Digital Synthesis of Plucked-String and Drum Timbres." *Computer Music Journal* 7(2):43–55.

McIntyre, M. E., and J. Woodhouse. 1960. "On the Fundamentals of Bowed String Dynamics." *Acustica* 43(2): 93–108.

McIntyre, M. E., R. T. Schumacher, and J. Woodhouse. 1981. "Aperiodicity in Bowed-String Motion." *Acustica* 49(1):13–32.

McIntyre, M. E., R. T. Schumacher, and J. Woodhouse. In press. "On the Oscillations of Musical Instruments." *Journal of the Acoustical Society of America*.

Mont-Reynaud, B. 1982. Private communication.

Morse, P. M. 1976. *Vibration and Sound*. New York: American Institute of Physics for the Acoustical Society of America. (Originally published in two editions [1936 and 1948].)

Smith, J. O. 1982a. "Synthesis of Bowed Strings." Paper presented at the Acoustical Society of America Conference, Chicago, Illinois. (Reprints available upon request.)

Smith, J. O. 1982b. "Introduction to Digital Filters." Typescript. (Copies available upon request.)

Smith, J. O. 1983. "Techniques for Digital Filter Design and System Identification with Application to the Violin." Ph.D. Diss., Electrical Engineering Department, Stanford University.

Steiglitz, K. 1974. *An Introduction to Discrete Systems*. New York: Wiley.