The Karplus-Strong Algortihm

In 1983, Alex Strong and Kevin Karplus published a simple but effective algorithm for synthesizing the sound of a plucked string.

Pick the *period* N. Then:

1. The first N outputs $y[0], \ldots, y[N-1]$ are random.
2. For $n \geq N$, output $y[n] = (y[n-N] + y[n-(N+1)])/2$. (By convention $y[-1] = 0$.)

If played at the frequency f_s, this sequence sounds like a string being plucked at frequency $f_s/(N+1/2)$

Explanation

The Karplus-Strong algorithm is an example of *digital waveguide synthesis*. An instrument is physically modeled and simulated. In this case, the random samples crudely represents the initial pluck: each part of the string is in a random position moving at a random velocity.

The delay and feedback cause the waveform to repeat itself, oscillating as a string would. If we just had $y[n] = y[n-N]$, we would have a waveform that repeats with frequency f_s/N.

Instead, taking the average of two consecutive samples acts as a one-zero low-pass filter, mimicking dampening effects of a real string as it vibrates. Higher frequency oscillations lose energy quicker than lower frequency oscillations.

The filter $y[n] = (x[n] + x[n-1])/2$ has the transfer function $H(z) = (1 + z^{-1})/2$. When $z = e^{ia}$, this is

$e^{-ia/2}(e^{ia/2} + e^{-ia/2})/2 = e^{-ia/2} cos a/2.$

Thus an input e^{ian} comes out as $e^{ia(n-1/2)}$, explaining why we divide the sampling frequency by $N + 1/2$ to arrive at the frequency of the plucked string.

Extensions

Although the basic algorithm produces surprisingly good results, we can do better.

At higher frequencies, rounding $f_s/(N + 1/2)$ to the nearest integer is too crude. We can correct for the error by introducing an allpass filter in the loop: $y[n] = Cx[n] + x[n-1] - Cy[n-1]$.

At lower frequencies, the sound decays too slowly. We can shorten the decay by introducing a loss factor $\rho < 1$, and set $y[n] = \rho(y[n-N] + y[n-(N+1)])/2$.

At higher frequencies, we have the opposite problem. We can stretch the decay by weighting the average. Pick some $0 < S < 1$ and set $y[n] = ((1-S)y[n-N] + Sy[n-(N+1)])/2$. This changes the phase delay; see Jaffe and Smith for the exact formula (or derive it yourself).

When a real string is plucked harder, the waveform contains more high frequency components. Thus by putting the output through an appropriate low-pass filter we change the loudness of the output. One possible *dynamics filter* is $y[n] = (1-R)x[n] + Ry[n-1]$ for some $0 < R < 1$ that depends on the frequency and desired loudness.

To simulate string muting, we can introduce a loss factor when a note ends.

Slurs can be simulated by using a new value of N on the fly. Similarly, glissandi can be simulated by changing N gradually.

References

- Kevin Karplus, Alex Strong, “Digital Synthesis of Plucked String and Drum Timbres”, Computer Music Journal (MIT Press) **7**(2), 1983.
- David A. Jaffe, Julius O. Smith, “Extensions of the Karplus-Strong Plucked-String Algorithm”, *Computer Music Journal* (MIT Press), **7**(2), 1983.

Contents

[back to top]

Ben Lynn *blynn@cs.stanford.edu*