

Electric Guitar Synthesis using the Karplus-Strong Algorithm

Greg Eustace

Abstract— A modified Karplus-Strong algorithm is implemented here for electric guitar synthesis. This derives from the work of Sullivan in his paper *Extending the Karplus-Strong algorithm to synthesize electric guitar timbers with distortion and feedback* (1990). Modifications to the basic algorithm include linear interpolation for exact tuning, two low pass filters for timbre modification, a DC-blocking filter, coupled strings, distortion and feedback.

Key Words— Karplus-Strong, physical modeling, guitar synthesis.

I. INTRODUCTION

The aim of this project was to implement a modified Karplus-Strong algorithm for electric guitar synthesis in MATLAB. This derives from the work of Sullivan (1990) in his paper *Extending the Karplus-Strong algorithm to synthesize electric guitar timbers with distortion and feedback*. Sullivan suggests a number of modifications including linear interpolation for exact tuning, two low pass filters for timbre modification, a DC-blocking filter, coupled strings, distortion and feedback. Sound examples are included on this CD in 44.1 KHz and 16 bit except where otherwise specified.

II. IMPLEMENTATION

A. The Karplus-Strong Algorithm

The Karplus-Strong algorithm can be used for synthesis of plucked-string and drum timbres. The simplicity of the algorithm contributes to its attractiveness. The process begins by filling a delay line with random numbers. A low pass filter is then attached to the output which is fed back to the delay line. The resulting frequency dependant decay is reminiscent of a plucked string. This implementation couples up to three strings to form chords.

“ks01.wav” is a sound example of the unmodified Karplus-Strong algorithm, at note C3 ($f_0 = 261.64$). “ks02.wav” is

another example using 3 strings to form a major chord built on the note C3.

B. Low-pass filter design

The prescribed 3-point low pass filter is given as:

$$y(n) = a_0 * x(n) + a_1 * x(n-1) + a_0 * x(n-2), \text{ where } a_1 \geq 2a_0 \geq 0.$$

This filter provides independent control over decay rates at two frequencies, chosen to be the fundamental and the Nyquist, with the constraint that the amplitude of the former may not exceed that of the latter. It also provides additional decay, has a monotonically decreasing response and is linear phase (Sullivan 1990). This modification has not been implemented here.

Sullivan suggests that an additional low pass filter be applied to the initial random numbers in order to control the relative initial amplitudes of the harmonics without affecting the fundamental. This filter is given as:

$$y(n) = a * (x(n) + x(n-1) + x(n-2)), \text{ where } a = |2\cos(w_0) + 1| - 1.$$

This filter can be applied repeatedly in order to increase the attenuation of high frequencies, giving the impression of a softly plucked string (Sullivan 1990).

C. DC-offset

Problems relating to DC-offset, and low-frequency energy in general, persisted throughout the project. Data overflow and loss of precision can result if this aspect is not accounted for (Sullivan 1990). A DC-blocking filter is given as:

$$y(n) = a_0 * x(n) + a_1 * x(n-1) + b_1 * y(n-1),$$

where $a_0 = 1 / (1 + w_{co}/2)$, $a_1 = -a_0$, $b_1 = a_0(1 - w_{co}/2)$, and w_{co} is a cut-off frequency given as $2\pi f_0/10$ (Sullivan 1990). This filter has zero-response at 0 Hz and no effect on frequencies at or above the fundamental (Sullivan 1990). Sullivan suggests applying this filter to the contents of the delay line as well as initially subtracting its average value. The filter is also applied inside the feedback loop. In addition, the average value of its output is subtracted. In some cases a DC-offset remains.

Manuscript received April 30, 2006.

The author is an M.A. student with the Music Technology Area, Schulich School of Music, McGill University, 555 Sherbrooke Street West, Montreal, Quebec, Canada H3A 1E3S.

Therefore, the user has the option to apply the DC-blocking filter again before the final output. However this has been observed to modify the dynamic profile of the output in an unnatural way. In addition a linear fade out is applied over the last 50 ms to insure that a click does not occur due to lingering DC-offset. Furthermore, it is necessary to truncate the data after it falls below a specified amplitude threshold.

D. Exact tuning

In the absence of interpolation the pitch of the output is limited to multiples of the sampling period. This renders continuous pitch effects like glissandi impossible. A two point interpolation filter is given by:

$$y(n) = c_0 * x(n) + c_1 * x(n - 1),$$

where $c_1 = 1 - c_0$ and $0 \leq c_0 \leq 1$.

An ascending linear glissando is implemented by decreasing c_0 until it reaches a value of 0 at which point the delay line length is decreased and c_0 is reset to 1. Descending glissandi may be produced, by increasing c_0 and decreasing the delay line length. Also a hammer-on effect (or slur) can be achieved by abruptly changing the delay line length. However, this seems to cause the second note to decay much faster than it normally would.

The interpolation filter was convolved with the low-pass filter to improve computational efficiency when using glissandi, as filter coefficients must be recalculated on a per sample basis in this case.

The sound example “ks03.wav” displays a glissando from notes C3 (261.64 Hz) to B3 (246.96 Hz).

E. Distortion

A distortion transfer function that provides soft clipping, as is characteristic of tube amplifiers, is given as:

$$y(x) = \begin{cases} 2/3, & x = 1 \\ x - x^3/3, & -1 < x < 1 \\ -2/3, & x = -1 \end{cases}$$

This was implemented directly rather than by table look up which doesn't seem to be beneficial. The degree of distortion is controlled by adjusting a pre-distortion gain factor (PDG) (Sullivan 1990).

A sound example, “ks04.wav” is given with distortion (PDG = 9).

F. Feedback

When the amplifier excites a string an oscillation grows at the fundamental frequency or one its harmonics to produce feedback. The result can be useful for achieving sustained notes and unique timbre effects (Sullivan 1990).

A feedback gain factor (FBG) controls the feedback gain directly. As well, distortion was observed to limit the growth of feedback. The amount of feedback and distortion are also controlled by mixing the dry and wet signals, which can be specified as a percentage.

The speaker-string distance is specified as a frequency corresponding to the length of the feedback loop. This frequency influences the feedback gain as well as the favored harmonics (Sullivan 1990). Specifying a feedback frequency (FBF) which is not harmonically related to the fundamental causes the resulting tone to have a richer spectrum and also helps to avoid amplitude overload. The user must also specify the number of feedback loops (L). In reality, the number of feedback loops is determined by the length of the note, but control over this parameter provides for some interesting timbre effects.

Sound example “ks05.wav” uses the parameters: PDG = 6, L = 10, FBG = 1 & FBF = E3). Sound example “ks06.wav” uses the parameters: PDG = 6, L = 10, FBG = 1 & FBF = G3. Sound example “ks07.wav” uses the parameters: PDG = 6, L = 10, FBG = 15 & FBF = E3. Sound example “ks08.wav” uses the parameters: PDG = 6, L = 100, FBG = 2 & FBF = E3.

G. Reverberation

Reverberation was added using the JCrever algorithm developed by John Chowning and based on the work of Schroeder (Schroeder 1961). The algorithm consists of three cascaded allpass filters and four parallel comb filters as well as a decorrelation delay between stereo channels. The reverb wet/dry amount can be specified as percentage.

A sound example “ks09.wav:” is given with distortion and feedback and reverb wet/dry amount equal to 60%.

III. RESULTS AND CONCLUSIONS

An implementation of Sullivan's algorithm for electric guitar synthesis was presented. A few problems with the algorithm have yet to be addressed.

The Karplus-Strong algorithm performs poorly for large delay line lengths (i.e. low frequencies) producing an unnatural sounding granular texture. It should be noted that calculating lower notes at lower sampling rates produced more realistic sounding results. The sound example “ks10.wav” is sampled at 22050 Hz, with distortion and feedback at note C2 ($f_0 = 130.82$ Hz).

Linear interpolation preserves harmonic relationships reasonably well, getting worse for higher frequencies and for fractional delay values near 0 or 1. A more sophisticated fractional delay algorithm could be employed. For a review of the literature on fractional delays see (Laakso, et al. 1996).

The main problem with using multiple strings is that higher frequency strings decay much faster than lower ones. Furthermore, distortion products change as the higher strings fall out of favor, producing unnatural sounding discontinuities. The sound example “ks11.wav” is given. Jaffe and Smith

discuss techniques for stretching decay rates and these should be further explored (Jaffe, and Smith 1983).

It would be interesting to perform non-linear glissandi, to more closely model the transition in frequency involved in bending strings. This could be achieved by applying a non-linear (e.g. exponential) function to the c_0 coefficient of the linear interpolator.

An interesting addition to the algorithm would be to change the feedback frequency over the course of the note, so as to simulate movement of the guitar relative to the amplifier. This could be achieved using linear interpolation for example.

Sullivan also suggested that future work should allow for the specification of a plucking point, filtering to coloration of the amplifier and the guitar body, as well as several effects typically associated with the guitar including wah-wah, sustain and phasing.

At the time of this writing the software still requires a few significant updates. In order to be a true physical model the distortion and feedback must be applied as the string is synthesized, rather than preceding it. Also there is a problem with the basic string algorithm which produces an audible ringing.

ACKNOWLEDGMENT

The author would like to thank G. Scavone.

REFERENCES

- [1] Karplus, K., A. Strong. 1983. Digital Synthesis of plucked-string and drum timbres. *Computer Music Journal* 7 (2): 43-55.
- [2] Sullivan, C. 1990. Extending the Karplus-Strong algorithm to synthesize electric guitar timbres using distortion and feedback. *Computer Music Journal* 14 (3): 26-37.
- [3] M. Schroeder, Logan, B. 1961. Colorless artificial reverberation. *IRE Transactions* 9: 209-14.
- [4] Laakso, T., V. Valimaki, M. Karjalainen, and U. Laine. 1996. Splitting the unit delay: Tools for fractional delay filter design. *IEEE Signal Processing Mag.* 13 (1): 30-60.
- [5] Jaffe, D., and J. Smith. 1983. Extensions of the Karplus-Strong plucked-string algorithm. *Computer Music Journal* 7(2): 56-69.