Physical Modeling

Implementierung des erweiterten Karplus-Strong Algorithmus in Native Instruments Reaktor

HAW Hamburg
Department Medientechnik
Sommersemester 2010

Jonas Link

30. November 2010

Inhaltsverzeichnis

1 Vom digitalen Wellenleitermodell zum Karplus-Strong Algorithmus

Der Zusammenhang von Karplus-Strong Algorithmus und der Physik einer realen Gitarrensaite wird in mehreren Publikationen behandelt (siehe Smith [5], Karjalainen et al. [2]).

Im Wesentlichen wird dort die Wellengleichung $Ky'' = \varepsilon\ddot{y}$ (mit $K \triangleq$ *Saitenspannung*, $\varepsilon \triangleq$ *lineare Massendichte* und $y \triangleq$ *Auslenkung*) und ihre Lösungen $y(x,t) = y_r(x+ct) + y_l(x+ct)$ als Ausgangspunkt gewählt. Diese beschreiben, dass sich eine Welle nach Anregung einer Saite nach beiden Seiten hin ausbreitet (wobei y_r die Wellenteile, die nach rechts, y_l, die nach links wandernden, beschreibt).

Digitalisiert ergibt sich das Bild zweier Delay-Lines mit Länge $\frac{N}{2}$, in denen nach Anregung Samples nach rechts bzw. links weiter geschoben werden. Die Verluste innerhalb der Delay-Lines sind vernachlässigbar, an ihren Enden findet je eine Phasendrehung statt, die aufgrund der Symmetrie jedoch weggelassen werden kann. Die einzigen noch übrig bleibenden Verluste sind die frequenzabhängigen Dämpfungen an den Enden der Saite bzw. der Delay-Lines. Diese können, da von einem LTI-System ausgegangen wird, auch in einem Punkt zusammengefasst werden. Auf einer realen Saite überlagern sich die Schwingungen auf der Saite, sobald sie einmal an Brücke oder Steg reflektiert wurden. Im digitalen Wellenleiter müssten die Samples der beiden Delay-Lines addiert werden, jedoch kann auch ein einziges Element der Delay-Line als Output definiert werden, ohne dass sich eine Änderung des Klanges ergibt [5, S. 84].

Resultierend erhalten wir eine einzige Delay-Line der Länge N und das Tiefpass-Filter für die frequenzabhängige Dämpfung. Der Karplus-Strong Algorithmus ist somit eine sehr effiziente Umsetzung eines eindimensionalen, digitalen Wellenleiters.

2 Umsetzung des erweiterten Karplus-Strong Algorithmus (EKS) in Native Instruments *Reaktor*

2.1 Übersicht

Nachdem Kevin Karplus und Alex Strong [3] eher zufällig über ihren Algorithmus gestolpert waren, beschäftigten sich eine Reihe von Wissenschaftlern mit ihrer Entdeckung. Noch in der selben Ausgabe des *Computer Music Journal*, in welchem der Karplus-Strong Algorithmus vorgestellt wurde, ergänzten ihn Jaffe und Smith [1]. Dazu gehörte eine verbesserte Möglichkeit des Stimmens, Anschlagsdynamik und -richtung, einstellbare Pick-Position, einstellbare Saitensteifigkeit, Resonanzen und einige weitere kleine Punkte.

Im Folgenden soll auf diese Erweiterungen eingegangen werden. In der von uns implementierten Version, finden sich die meisten dieser Erweiterungen wieder; weggelassen wurden das Tiefpass-Filter für die Anschlagsrichtung und der Allpass für die Saitensteifigkeit (siehe Abb. 1).

Um den Algorithmus zu implementieren, entschieden wir uns für die Software *Reaktor* der Firma Native Instuments, im Prinzip ein digitaler Modular-Synthesizer auf Softwarebasis. Seit Versionsnummer 5 bietet sie mit der Core-Ebene ein mächtiges Tool, das es ermöglicht eigene Module zu programmieren. Man kann sich diese Ebene als graphische Programmieroberfläche vorstellen, in der man auf die Verarbeitung der einzelnen Audio-Samples Einfluss nimmt. Digitale Filter lassen sich so quasi direkt in der Z-Ebene erstellen.

2.2 Noise Burst X[n]

Im Gegensatz zum ursprünglichen Karplus-Strong Algorithmus, wird in seiner erweiterten Version mit einem Eingangssignal gearbeitet. Dieses besteht aus zufälligen Samples auf die nach N Werten Nullen folgen (siehe Abb. 2) Mit diesem Noise Burst wird die Wavetable bzw. die Delay-Line initialisiert. Der Vorteil dieser Herangehensweise liegt darin, dass das Eingangssignal noch gefiltert werden kann, beispielsweise um die Position und Art des Anschlags zu simulieren.

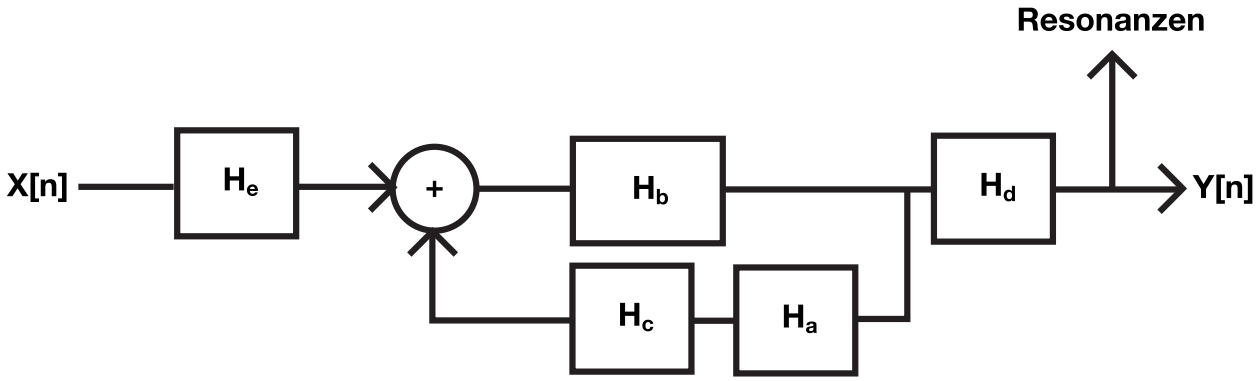


Abbildung 1: Implemtierte Version des erweiterten Karplus-Strong Algorithmus (ohne Anschlagsrichtungs- und String Stifness-Filter)

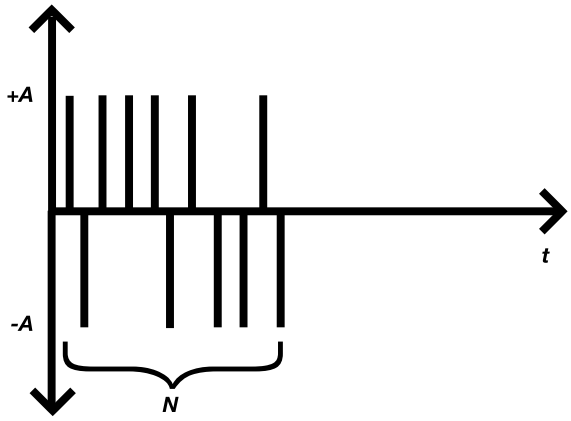


Abbildung 2: Eingangssignal X[n] - Noise Burst der Länge N, dann folgen Nullen.

2.3 Delay-Line H_b

Die Delay-Line ersetzt beim erweiterten Karplus-Strong Algorithmus sozusagen die, mit Zufallswerten initialisierte, Wavetable des Ausgangsalgorithmus. Technisch gesehen, besteht zwischen der Wavetable mit N Samples und der Verzögerung des Signals um N Samples, rückgekoppelt auf sich selbst, kein Unterschied. Die Z-Übertragungsfunktion lautet somit logischerweise

$$H_b(z) = z^{-N}.$$

In *Reaktor* lässt sich die Größe für ein Array leider nur manuell festlegen, also nicht durch den Algorithmus selbst verändern. Da die Tonhöhe jedoch maßgeblich von der Länge der Delay-Line abhängig ist, musste hierfür eine Lösung gefunden werden. Das Problem wurde letztendlich so umgangen, dass am Anfang ein Array mit 22.050 Samples (entsprechend der tiefsten, theoretisch vorkommenden Frequenz) initialisiert wird, von dem jedoch nur ein gewisser Bereich, je nach gewünschtem Pitch genutzt wird.

2.4 Lowpass H_a

Der Standard Lowpass, der für die frequenzabhängige Dämpfung der „periodischen“ Schwingung verantwortlich ist, bleibt in einer ersten Implementierung des EKS bestehen, kann jedoch auch gegen ein anderes Filter ausgetauscht werden [5]. Das Standard-Filter ist ein so genanntes 2-point-average-Filter und hat die Z-Übertragungsfunktion (Abb. 4a)

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

Um andere Decay-Zeiten zu erreichen, so dass tiefe Pitches nicht unnatürlich lange nachschwingen und hohe Pitches nicht nur ein Klicken verursachen, verallgemeinert man die Übertragungsfunktion zu

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

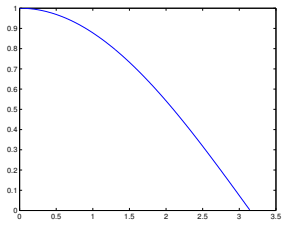
wobei gelten muss $0 < S < 1$. Für $S = \frac{1}{2}$ erhält man die ursprüngliche Form des Algorithmus zurück. Dieses Verfahren wird als Decay-Stretching bezeichnet.

In unserer Umsetzung entschieden wir uns, statt für ein Filter mit einer Nullstelle (FIR), für eines

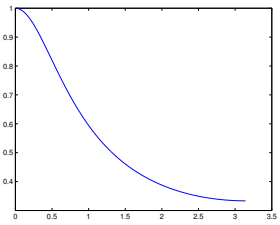
mit einer Polstelle (IIR). Die Übertragungsfunktion sieht dann wie folgt aus (Abb. 4b):

$$H_a(z, S) = \frac{1 - |S|}{1 - Sz^{-1}}.$$

Das bedeutet, das hohe Frequenzen etwas schneller gedämpft werden (siehe Abb. 3), wodurch der Klang an Schärfe verliert und vielleicht etwas eher in Richtung Konzert-Gitarre tendiert. Außerdem lässt sich so die Dämpfung realistischer einstellen, als nur mit einer Konstante ρ, wie in [1] vorgeschlagen. Für hohe S ergibt sich in unserer Variante eine starke Dämpfung hoher Frequenzen, während für $S = 0$ das Filter gar nicht wirkt, der Rauschloop also periodisch abgespielt wird. Entscheided man sich für ein FIR-Filter so kann mit variablem S das Decay nur verlängert, nicht jedoch verkürzt werden, da man nur Einfluss auf die Position der Nullstelle hat, wie aus Abbildung 4 ersichtlich wird. Zur Verkürzung der Decayzeit steht dann nur die Konstante ρ zur Verfügung.



(a) Standard-Tiefpass



(b) Mit einer Polstelle (für $S = 0.5$)

Abbildung 3: Mögliche Frequenzgänge des Tiefpasses H_a (normiert auf $\pi \triangleq f_{Nyquist}$)

2.5 Tuning Allpass $\mathbf{H_c}$

Der Tuning Allpass kompensiert die Ungenauigkeit in der Höhe des Pitches, die durch die Delay-Line gegeben ist: Da die Frequenz der Schwingung durch die Anzahl der Samples in der Delay-Line vorbestimmt ist (für die „normale“ Wavetablesynthese gilt $f_1 = f_s/N$, mit $f_s \triangleq Abtastfrequenz$ und $f_1 \triangleq Pitch$) und selbstverständlich nur ganzzahlige Werte für die Länge N in Frage kommen, ist der Algorithmus in seiner Originalform, besonders für hohe Grundfrequenzen, aufgrund des Rundungsfehlers verstimmt. Um auf die exakte Looplänge zu kommen, muss zum Wert N noch die Gruppenlaufzeit von H_a – im Standard-Fall $\frac{1}{2}$ – addiert werden. Mit dem Tuning-Allpass wird die Gesamtlooplänge so angepasst, dass die Frequenz f_1 jeden beliebigen Wert annehmen kann. Dies geschieht durch eine Phasenverzögerung der Grundfrequenz.

Der Allpass, der zur Korrektur der Höhe des Pitches verwendet wird, hat die Differenzengleichung:

$$y[n] = Cx[n] + x[n-1] - Cy[n-1],$$

z-transformiert ergibt sich somit die Übertragungsfunktion:

$$H_c(z) = \frac{C + z^{-1}}{1 + Cz^{-1}}.$$

Der Koeffizient C muss jetzt, abhängig von unterschiedlichen Einflüssen gewählt werden. Es gilt $|C| < 1$. Der Allpass muss nur Phasenverzögerungen zwischen 0 und T_s realisieren können. Die zu erreichende Phasenverschiebung für die Grundfrequenz $P_c(f_1)$ ist gegeben durch:

$$P_c(f_1) \triangleq P_1 - N - P_a(f_1),$$

wobei P_1 der exakte gewünschte Pitch in Hz, N die Länge der Delay-Line und $P_a(f_1)$ die durch den Tiefpass hervorgerufene Gruppenlaufzeit ist. Letztere beträgt ohne Decay-Stretching $\frac{1}{2}$ und kann ansonsten mit S angenähert werden.

Aus der gesuchten Phasenverschiebung kann nun C für tiefe Frequenzen approximiert werden:

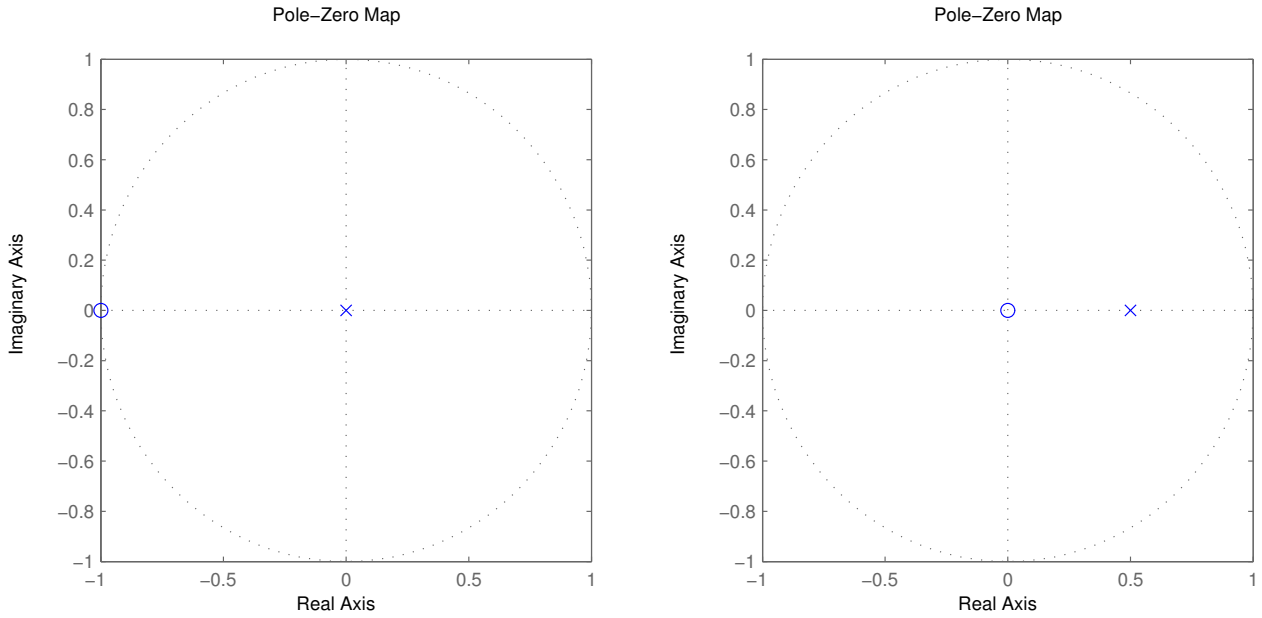
$$C \approx \frac{1 - P_c(f_1)}{1 + P_c(f_1)}.$$

2.6 Dynamics Filter $\mathbf{H_d}$

Zupft man eine Gitarrensaite leicht an, klingt sie nicht nur leiser als eine fest angezupfte Saite, sondern auch das Spektrum verändert sich, da eine hart gespielte Saite mehr Obertöne besitzt. Dieses Verhalten versucht man mit dem Dynamics-Filter zu simulieren, das letztendlich nichts weiter als ein Tiefpass mit variabler Grenzfrequenz ist. Nutzt man die Velocity einer per MIDI getriggerten Note, so entspricht die am härtesten angeschlagene Taste einer 127, die am weichsten angeschlagene einer 1.

Die Differenzengleichung des Filters lautet:

$$y[n] = (1 - R)x[n] + Ry[n-1],$$



(a) Standard-Tiefpass

(b) Mit einer Polstelle (für $S = 0.5$)

Abbildung 4: Mögliche Pol-/Nullstellen-Diagramme des Tiefpasses H_a

der Koeffizient R errechnet sich durch folgende Formel:

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

wobei G_L der gewünschte Gain und die direkte Umsetzung der MIDI-Velocity auf einen Wertebereich zwischen 0-1 ist. Es wird jeweils das Ergebnis der Berechnung gewählt, für das gilt $R < 1$, um die Stabilität zu gewährleisten. Die komplizierte Berechnung des Filterkoeffizienten R wird nötig, weil der Pitch in den Koeffizienten mit einfließen muss, da ansonsten tiefere Grundfrequenzen lauter klingen würden.

2.7 Pick Position Comb Filter H_e

Mit dem Kammfilter H_e lässt sich die Anschlagsposition auf der Saite simulieren, also ob näher an Brücke oder in der Mitte der Saite angeschlagen wird. Dafür wird abhängig von der Länge der Saite bzw. der Delay-Line N ein weiteres Delay mit der Differenzengleichung

$$y[n] = x[n] - x[n - \mu N]$$

in das Blockschaltbild eingefügt, wobei $0 < \mu \leqslant \frac{1}{2}$. Für $\mu = \frac{1}{2}$ löschen sich alle geradzahligen Harmonischen aus; die Saite wird in der Mitte gezupft. Das lässt sich im digitalen Wellenleitermodell [6, 2] sehr gut veranschaulichen. Für $\mu = \frac{1}{10}$ wird jede 10. Harmonische unterdrückt. Das bedeutet, die Saite wird auf einem Zehntel ihrer Länge über der Brücke angeschlagen.

2.8 Sympathetic Strings

Als Sympathetic Strings bezeichnet man Saiten, die mitschwingen ohne angeschlagen worden zu sein. Dies lässt sich sehr einfach durch eine weitere Instanz des EKS realisieren, indem das Ausgangssignal einer ersten Instanz als Eingangssignal statt des Noise Bursts benutzt wird, wobei die zweite Saite z.B. eine Oktave höher schwingen kann. Beide Signale können nun in einem Mixer mit beliebigen Anteilen zusammengemischt werden. Dabei führt eine höhere Dämpfung der zweiten Saite meist zu besseren Ergebnissen.

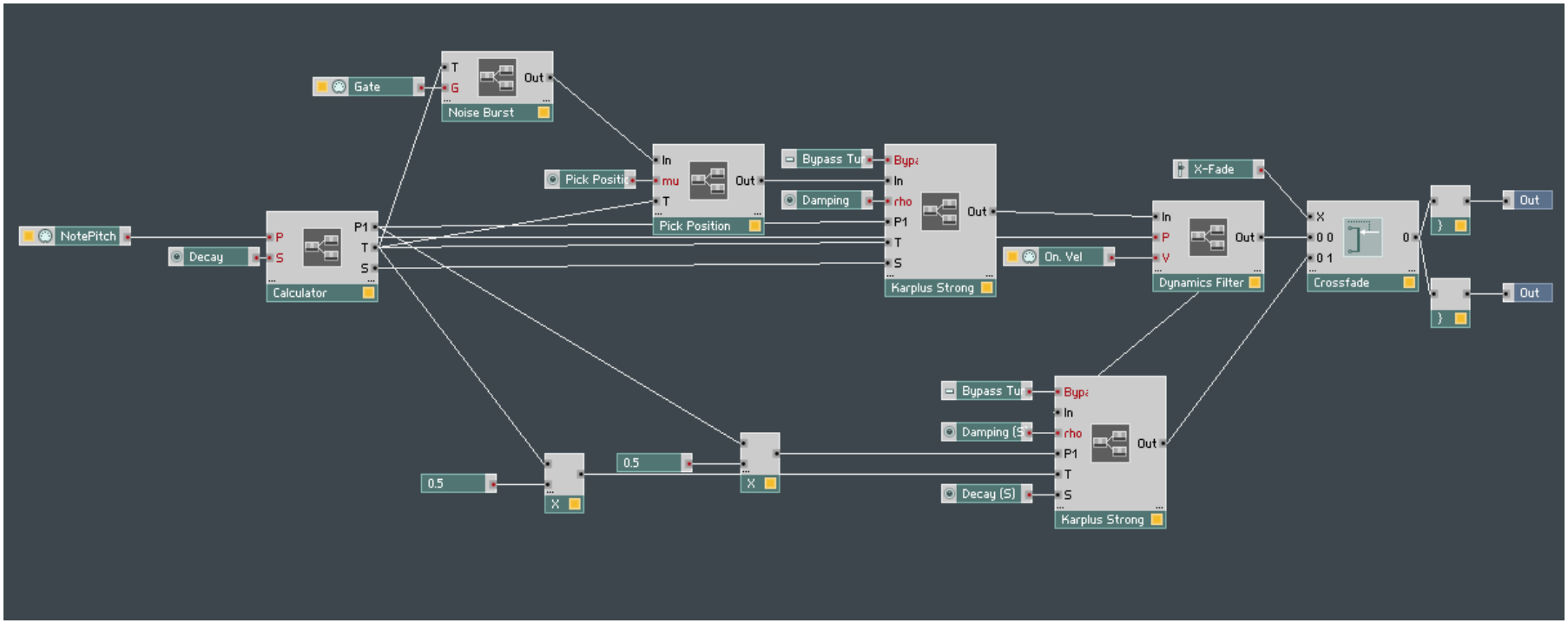


Abbildung 5: Implementierung in Reaktor. Zu sehen sind die zusammengeschalteten, auf der Core-Ebene entwickelten Module

Literaturverzeichnis

[1] Jaffe, D. A.; Smith, J. O.: Extensions of the Karplus-Strong Plucked-String Algorithm. In: *Computer Music Journal* 7 (1983), Nr. 2, S. 56–69. – URL `http://www.jstor.org/stable/3680063`. – ISSN 01489267

[2] Karjalainen, M.; Välimäki, V.; Tolonen, T.: Plucked-String Models: From the Karplus-Strong Algorithm to Digital Waveguides and beyond. In: *Computer Music Journal* 22 (1998), Nr. 3, S. 17–32. – URL `http://www.jstor.org/stable/3681155`. – ISSN 01489267

[3] Karplus, K.; Strong, A.: Digital Synthesis of Plucked-String and Drum Timbres. In: *Computer Music Journal* 7 (1983), Nr. 2, S. 43–55. – URL `http://www.jstor.org/stable/3680062`. – ISSN 01489267

[4] Roads, C.: *The Computer Music Tutorial*. The MIT Press, 1996

[5] Smith, J. O.: Physical Modeling Using Digital Waveguides. In: *Computer Music Journal* 16 (1992), Nr. 4, S. 74–91. – URL `http://www.jstor.org/stable/3680470`. – ISSN 01489267

[6] Sullivan, C. R.: Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback. In: *Computer Music Journal* 14 (1990), Nr. 3, S. 26–37. – URL `http://www.jstor.org/stable/3679957`. – ISSN 01489267

Abbildungsverzeichnis