DOI: https://doi.org/10.56712/latam.v5i5.2817

Simulación del sonido de una cuerda de guitarra usando el algoritmo de Karplus-Strong extendido

Simulation of guitar string sound using the extended Karplus-Strong algorithm

Paul Freire Diaz
jpfreire@unach.edu.ec
https://orcid.org/0000-0002-0657-9717
Universidad Nacional de Chimborazo
Riobamba – Ecuador

Ximena López Mendoza
xlopez@unach.edu.ec
https://orcid.org/0000-0002-9564-6300
Universidad Nacional de Chimborazo. Universidad Nacional Mayor de San Marcos
Riobamba – Ecuador

Mónica Mazón Fierro
mmazon@unach.edu.ec
https://orcid.org/0000-0002-5303-9174
Universidad Nacional de Chimborazo. Universidad Nacional Mayor de San Marcos
Riobamba – Ecuador

Guido Mazón Fierro
guido.mazon@espoch.edu.ec
https://orcid.org/0000-0001-8745-2373
Escuela Superior Politécnica de Chimborazo (ESPOCH), Facultad de Administración de Empresas
Riobamba – Ecuador

Pamela Buñay Guisñan
pbunay@unach.edu.ec
https://orcid.org/0000-0002-4320-6899
Universidad Nacional de Chimborazo
Riobamba – Ecuador

Artículo recibido: 03 de octubre de 2024. Aceptado para publicación: 17 de octubre de 2024.
Conflictos de Interés: Ninguno que declarar.

Resumen

El objetivo de la presente investigación se centra en la implementación de un modelo de síntesis del sonido de una cuerda de guitarra en tiempo real, utilizando el entorno gráfico Simulink y basándose en el algoritmo de Karplus-Strong y sus extensiones. Este algoritmo es un método eficiente para simular cuerdas pulsadas a través de un filtro pasa bajos y un retardo, ajustando la frecuencia según la nota deseada. Las extensiones del algoritmo, conocidas como Extended Karplus-Strong (EKS), permiten generar una mayor variedad y calidad de sonidos mediante diferentes filtros digitales. Esta implementación ofrece un modelo parametrizable, con aplicaciones potenciales en juegos interactivos de guitarra, educación musical y la creación de música basada en guitarras.

Palabras clave: algoritmo de Karplus-Strong, simulink, filtros digitales

Abstract

The objective of this research focuses on the implementation of a real-time guitar string sound synthesis model using the graphical environment Simulink, based on the Karplus-Strong (KS) algorithm and its extensions. This algorithm is an efficient method for simulating plucked strings through a low-pass filter and a delay, adjusting the frequency according to the desired note. The algorithm's extensions, known as Extended Karplus-Strong (EKS), allow for generating a wider variety and higher quality of sounds through the use of different digital filters. This implementation offers a parametrizable model with potential applications in interactive guitar games, music education, and guitar-based music composition.

Keywords: Karplus-Strong algorithm, simulink, digital filters

Cómo citar: Freire Diaz, P., López Mendoza, X., Mazón Fierro, M., Mazón Fierro, G., & Buñay Guisñan, P. (2024). Simulación del sonido de una cuerda de guitarra usando el algoritmo de Karplus-Strong extendido. *LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades 5 (5), 2830 – 2844. https://doi.org/10.56712/latam.v5i5.2817*

INTRODUCCIÓN

Para producir un sonido que se asemeja al sonido de una guitarra, se necesita una técnica o algoritmo especial que pueda simular el sonido de instrumentos acústicos. (Assayag, Feichtinger, Rodrigues, 2000, pp. 257–294; Sullivan, 1990). Existen muchas técnicas actualmente utilizadas para la síntesis de música digital, incluyendo la síntesis de modulación de frecuencia (FM), la conformación de ondas, la síntesis aditiva y la síntesis sustractiva. (Karplus & Strong, 1983a)

Las cuerdas se pueden modelar de manera eficiente porque en el mundo real son uniformes, tensas y con una terminación rígida (Picquart & Jiménez, 2010). La síntesis por guías de ondas digitales (Digital Waveguide Synthesis) para modelar cuerdas es una técnica utilizada en la síntesis de sonido, específicamente en la emulación de instrumentos musicales que generan tonos a partir de vibraciones, como cuerdas y vientos, donde en lugar de modelar físicamente el objeto completo, se utilizan ecuaciones matemáticas simplificadas para calcular cómo las ondas se reflejan y se transmiten en ese medio. El retardo de propagación de la onda a lo largo de la cuerda se implementa utilizando una línea de retardo ordinaria, mientras que las características de amortiguamiento y dispersión asociadas con la propagación en la cuerda se agrupan en filtros digitales de bajo orden. (Smith III, 1996).

La implementación de un algoritmo que simule digitalmente el sonido de una cuerda de guitarra tiene un potencial enorme como base para juegos interactivos de guitarra, enseñar música y, por supuesto, componer canciones basadas en la guitarra. (Santiago et al., 2014). Para los músicos interesados principalmente en interpretar y componer música, en lugar de diseñar instrumentos, estos algoritmos ofrecen una nueva y bienvenida técnica. Para aquellos interesados en el diseño de instrumentos, abren un nuevo campo de técnicas efectivas para explorar. (Karplus & Strong, 1983a). Un ejemplo famoso de uso de esta técnica es el sintetizador Yamaha VL1 (Mignini et al., 2009).

El propósito de este proyecto es implementar en el entorno de desarrollo gráfico Simulink un modelo de síntesis de guitarra eléctrica en tiempo real utilizando algunas de las extensiones propuestas encontradas en la revisión bibliográfica, desarrollando un modelo fácilmente parametrizable. Simulink tiene las herramientas necesarias para la implementación de sistemas en tiempo real, además de la capacidad de programar hardware externo para ejecutar algoritmos de forma interactiva como en (Freire Diaz, López-Mendoza, Casignia, Cisneros Barahona, & Uvidia Fassler, 2020) y (Freire, Uvidia-Fassler, López, Casignia Vásconez, & Guevara, 2022).

El algoritmo de Karplus-Strong (KS) fue descubierto en 1983 por (Karplus & Strong, 1983b) como una técnica computacional simple y eficiente para sintetizar notas musicales similares a cuerdas pulsadas (Santiago, Samuel, & Sawn, 2014), siendo un procedimiento de síntesis de tabla de ondas que consiste en hacer pasar la forma de onda corta a través de un retardo en el filtro pasa bajos para suavizar todas las demás frecuencias excepto la frecuencia del tono deseado (Abdullah, Angkasa, Hartono, & Fithra, 2019), cuya longitud depende de la frecuencia de la nota deseada, para simular el sonido de las cuerdas al ser pulsadas, como en instrumentos de percusión. El resultado de este algoritmo es una forma de onda que suena como una cuerda de guitarra idealizada, sin rigidez y con características de amortiguamiento muy específicas que resultan en el promedio de dos puntos, pulsada para un tono específico. (Miranda, 2002).

El algoritmo Karplus-Strong básico (KS) consta de una memoria de forma de onda que se lee y modifica repetidamente en cada "periodo", donde la modificación consiste en reemplazar cada muestra en la memoria por el promedio de sí misma y la muestra anterior cada vez que se lee. El algoritmo consta de un filtro digital de alto orden, que representa la cuerda; y una breve explosión de ruido, que representa el "punteo". El filtro digital está dado por la ecuación de diferencias:

$$y_n = x_n + \frac{y_{n-N} + y_{n-[N+1]}}{2}$$

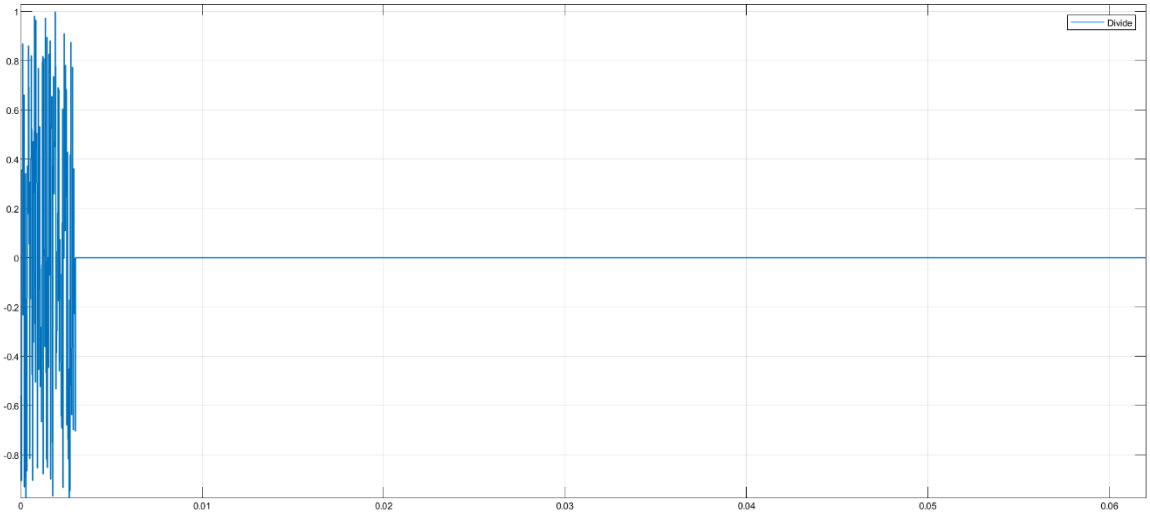
donde xn es la amplitud de la señal de entrada en la muestra n, yn es la amplitud de la salida en la muestra n, y N es el período de tono (aproximado) deseado de la nota en muestras. La explosión de ruido está definida por

$$x_n = \{Au_n, \qquad n = 0,1,2,\dots,N-1\ 0, \qquad n \geq N$$

Donde A es la amplitud deseada, y $u_n \in [-1,1]$ es la salida de un generador de números aleatorios, generando una ráfaga de ruido Figura 1 . Debido a que la línea de retardo inicialmente es llenada con números aleatorios, según (Assayag et al., 2000, pp. 262–264) este algoritmo se encuentra dentro de los "Random noise models"

Gráfico 1

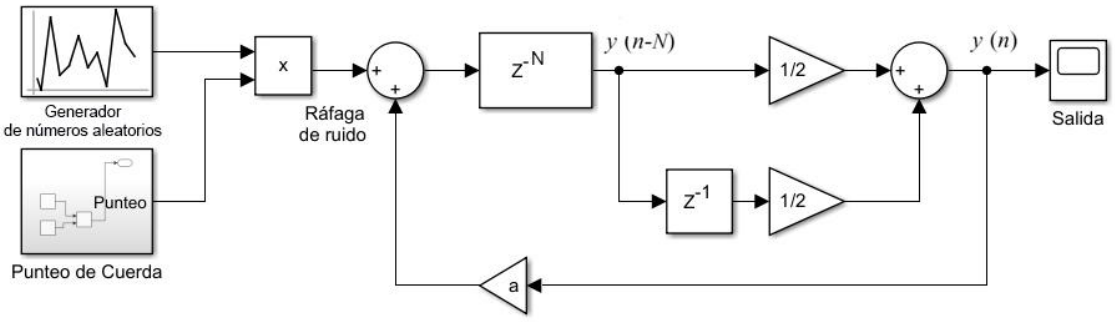
Ráfaga de ruido generada en Simulink



La salida yn es tomada empezando en el tiempo n=N en la implementación. Un análisis del simulador de cuerdas y la teoría en la que se basa se encuentra en (Jaffe David A. & Smith Julius O., 1983). El punteo de cuerda se aplica mediante una señal rectangular de amplitud 1, cuya duración depende del número de muestras de retardo requeridas. Adicionalmente se requiere una ganancia a, conocida como coeficiente de retroalimentación, la cual permite controlar el decaimiento de la señal de salida. El rango de esta ganancia es de $0 < a \leq 1$. Este algoritmo se implementa en Simulink como se presenta en la Figura 2.

Figura 2

Implementación del algoritmo básico de Karplus-Strong en Simulink, donde N es el tiempo de retardo y a es el coeficiente de retroalimentación



Se obtiene un nuevo "punteo" en el algoritmo digitar al escribir nuevos números aleatorios en la memoria de forma de onda. La frecuencia fundamental F_0 se aproxima mediante la tasa de muestreo f_s dividida por la longitud de la memoria N, o:

$$F_0 \approx \frac{f_s}{N}$$

Donde:

F_0 es la frecuencia de la cuerda (el tono que se desea generar).

f_s es la frecuencia de muestreo del sistema (por ejemplo, 44.1 kHz en audio digital).

N es el número de muestras de retardo.

Esta relación no es exacta debido a que el promedio de dos puntos agrega un retardo de fase de medio muestreo. Una fórmula más precisa según (Smith-III, 2013) es:

$$F_0 = \frac{f_s}{N + \frac{1}{2}}$$

Esta fórmula puede utilizarse como exacta para fines prácticos, pero no lo es debido al ligero decaimiento por periodo causado por el promedio de dos puntos.(Smith-III, 2013)

Si bien en un principio el algoritmo KS aparentemente no tenía nada que ver con la física (Matti Karjalainen, Välimäki, & Tolonen, 1998), (Jaffe David A. & Smith Julius O., 1983) demostraron la física de la cuerda pulsada, además de realizar extensiones o mejoras al algoritmo(Smith-III, 2013). Investigaciones adicionales plasman estas ideas en principios de síntesis más detallados e implementaciones (Matti Karjalainen, Välimäki, & Jánosy, 1993), dando como resultado síntesis muy realistas de instrumentos de cuerda pulsada, y de métodos para simular el timbre de instrumentos como la guitarra eléctrica amplificada y distorsionada. (M Karjalainen & Laine, 1991; Sullivan, 1990).

Una serie de extensiones al algoritmo original KS fueron propuestas por (Jaffe David A. & Smith Julius O., 1983), y son conocidas como Extended Karplus-Strong o EKS. Amplían el algoritmo KS original con el objetivo de mejorar la calidad y variedad de los sonidos que se podían generar, y de hacerlo más versátil para la síntesis de instrumentos musicales más complejos. Las extensiones EKS fueron motivadas por las exigencias de una composición musical y la interpretación del algoritmo KS como

un modelo de función de transferencia de una cuerda física simplificada, ilustrando cómo varios filtros digitales pueden lograr diversos efectos musicales deseados. (Smith-III, 2013)

METODOLOGÍA

La presente investigación parte de la implementación del algoritmo original de Karplus-Strong (Figura 2) incorporando extensiones o filtros, describiendo su funcionamiento y el efecto de sus parámetros en el sonido resultante. En la Tabla 1 se muestran los nombres de los filtros utilizados, junto con sus notaciones.

Tabla 1

Descripción de los bloques de filtro utilizados

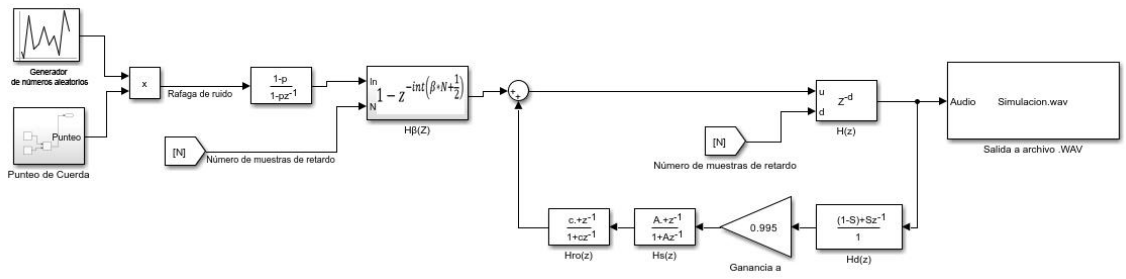
Nombre	Notación utilizada
Filtro de línea de retardo (Delay Line Filter)	$H(z)$
Filtro pasa-bajos dirección de la pica (Pick-direction lowpass filter)	$H_p(z)$
Filtro de peine de posición de pulsación (Pick-position comb filter)	$H_\beta(z)$
Filtro de retardo pasa-todo por rigidez de cuerda (String-Stiffness Allpass Filter)	$H_s(z)$
Filtro de amortiguamiento de cuerda (String-dampling filter)	$H_d(z)$
Filtro pasa-todo de afinación de cuerda (First-order string-tuning allpass filter)	$H_{ro}(z)$

En el presente estudio se usó herramienta de software Simulink de Matlab versión 2023a, por su capacidad de uso para modelado, simulación y análisis de sistemas dinámicos, usando bloques que permiten simular e implementar sistemas variables en el tiempo. Con esto se pretende la replicabilidad del modelo para su aplicación en investigaciones similares.

En la Figura 3 se visualiza el sistema EKS con los filtros utilizados, que se examinarán en los acápites siguientes.

Figura 3

Diagrama de bloques de la implementación del algoritmo EKS en Simulink



A continuación, se analizan los filtros utilizados en la implementación.

Filtro de línea de retardo (Delay Line Filter) $H(z)$

Este filtro se utiliza para modelar el comportamiento vibratorio de una cuerda, ya que una cuerda vibrante física tiene un retardo inherente que determina su frecuencia de oscilación.

$$H(z) = z^{-N}$$

Donde:

N es el número de muestras de retardo.

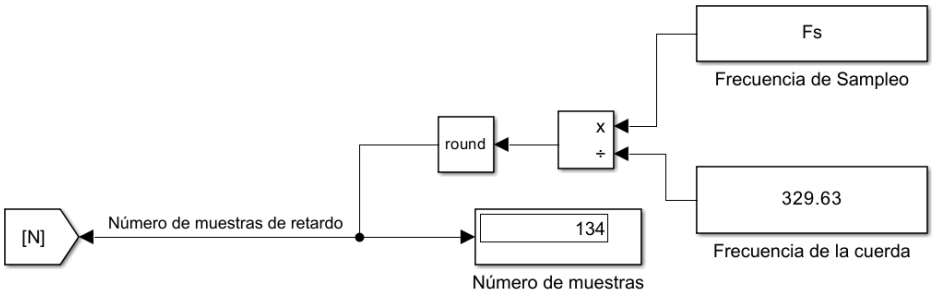
z^{-1} representa un retardo de una muestra en tiempo discreto.

z^{-N} indica que la señal de entrada se retrasa N muestras antes de salir como señal de salida.

El número N controla la longitud de la línea de retardo y, por lo tanto, determina la frecuencia de la nota que produce la cuerda sintetizada. Un valor grande de N corresponde a un tono más bajo (porque el retardo es más largo), mientras que un valor pequeño de N produce un tono más alto (retardo más corto). En la Figura 4 se muestra en bloques de Simulink el cálculo del número de muestras de retardo para simular el sonido de la primera cuerda, que vibra a una frecuencia de 329 Hz., para una frecuencia de muestreo de 44100 muestras por segundo.

Figura 4

Implementación del cálculo del número de muestras de retardo N necesario para simular el sonido de una cuerda a 329 Hz



En una cuerda real, una perturbación viaja a lo largo de la cuerda y luego se refleja en los extremos. El retardo digital z^{-N} simula este proceso, haciendo que la señal se retrase y se retroalimenta a través del sistema de síntesis para continuar la simulación de la vibración de la cuerda.

Filtro pasa-bajos dirección de la pica (Pick-direction lowpass filter) $H_p(z)$

Es un tipo de filtro que se utiliza en la síntesis de cuerdas para simular el efecto que tiene la dirección en la que se pulsa (o "pica") una cuerda en el sonido resultante. Es particularmente relevante en la síntesis física de instrumentos de cuerda, como guitarras o bajos, donde la manera en que se pulsa la cuerda afecta las características tonales y el contenido armónico.

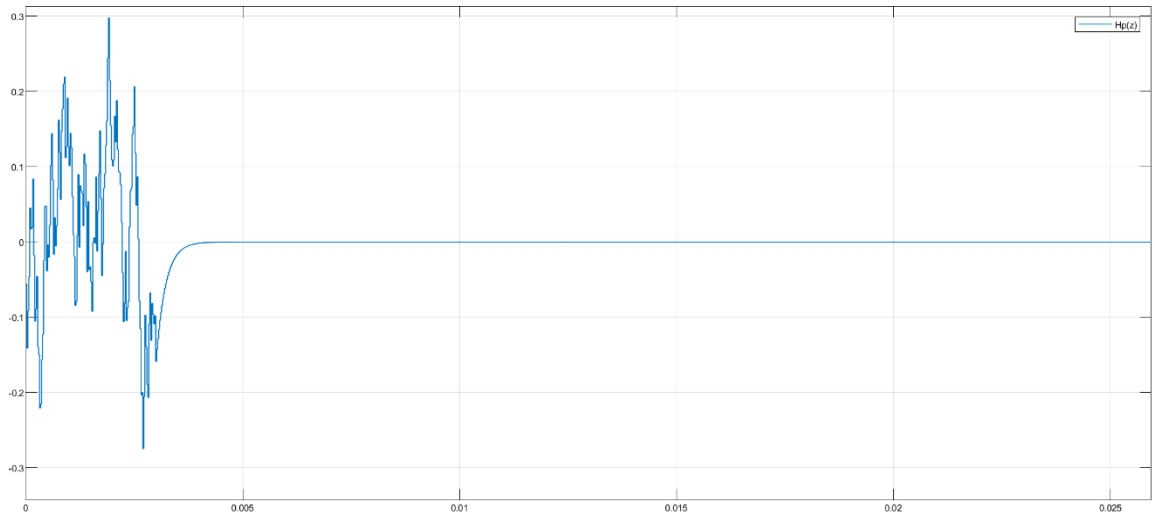
$$H_p(z) = \frac{1-p}{1-pz^{-1}}$$

En instrumentos de cuerda reales, cuando una cuerda es pulsada en diferentes direcciones (por ejemplo, hacia arriba o hacia abajo con una púa, o con los dedos), esto afecta el espectro de

frecuencias que se produce, y el contenido armónico resultante varía. Cuando una cuerda se pulsa con fuerza, especialmente en una dirección particular, se generan armónicos más agudos y ricos. En cambio, una pulsación más suave o en una dirección diferente puede resultar en un sonido con menos armónicos altos, más amortiguado o "apagado". El filtro de paso bajo $H_p(z)$ se introduce para modelar este fenómeno. Esta fórmula describe un filtro de primer orden con un comportamiento específico que depende del valor de p.

Figura 5

Salida del filtro $H_p(z)$



p es un parámetro que controla el nivel de filtrado, y su valor debe estar en el rango 0≤p<1

1−p controla la ganancia aplicada a la señal actual.

1−pz−1 introduce una retroalimentación retardada, lo que suaviza la señal y atenúa las frecuencias altas.

p cercano a 0: Permite un sonido más brillante con más armónicos altos.

p cercano a 1: Produce un sonido más suave con menos armónicos altos.

Filtro de peine de posición de pulsación (Pick-position comb filter) $H_\beta(z)$

Este filtro introduce un retardo controlado por el parámetro β, creando un filtro en peine que afecta las frecuencias específicas del espectro de la cuerda.

$$H_\beta = 1 - z^{-int\left(\beta * N + \frac{1}{2}\right)}, \beta \in (0,1)$$

La sección

$$z^{-int\left(\beta * N + \frac{1}{2}\right)}$$

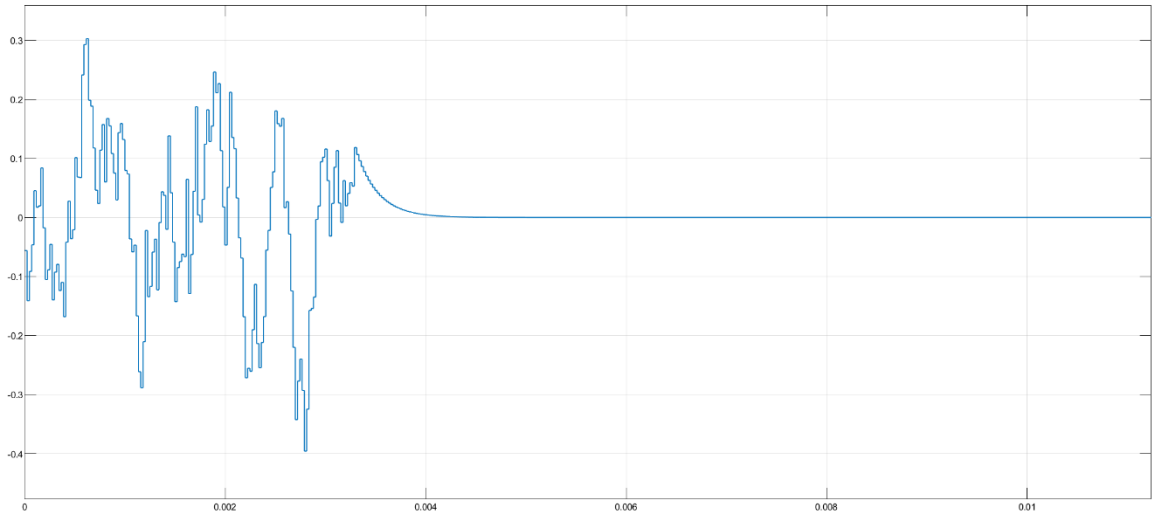
introduce un retardo de una cantidad entera de muestras, controlado por la posición de la pulsación β. La diferencia entre 1 y esta versión retardada crea cancelaciones periódicas en el espectro, lo que elimina ciertos armónicos y deja pasar otros.

Por ejemplo, si β =1/2 se obtiene la simulación de una pulsación en el medio de la cuerda, donde el filtro cancelará los armónicos pares, creando un sonido más suave o "hueco" porque estos armónicos coinciden con los nodos de la vibración de la cuerda, mientras que N es el número de muestras por período. Un valor de β cercano a 0 o 1 implica un pulso cerca de los extremos de la cuerda refuerza más armónicos y genera un sonido más brillante.

Figura 6

Salida del filtro $H_\beta(z)$

Filtro de retardo pasa-todo por rigidez de cuerda (String-Stiffness Allpass Filter) $H_s(z)$



El filtro de retardo todo-paso por rigidez de cuerda simula el efecto de la rigidez en cuerdas reales, introduciendo el fenómeno conocido como dispersión en la propagación de las ondas a través de cuerdas reales, las cuales no son perfectamente flexibles, como las cuerdas de piano, guitarra o violín., lo que provoca que las ondas de frecuencias más altas viajen más rápido que las ondas de baja frecuencia. Esto introduce una dispersión de fase, lo que significa que los armónicos superiores no están perfectamente alineados con los múltiplos enteros de la frecuencia fundamental, lo que da al sonido un carácter diferente, más "metálico" o "brillante".

En lugar de afectar la amplitud de la señal, como lo haría un filtro paso bajo o paso alto, este filtro altera la fase de las componentes frecuenciales sin cambiar su magnitud.

El filtro todo-paso tiene la siguiente forma general:

$$H_s(z) = \frac{A + z^{-1}}{1 + Az^{-1}}, A \in [0,1)$$

Donde A es un parámetro que controla la cantidad de dispersión, relacionada con la rigidez de la cuerda y su longitud: cuanto mayor es la rigidez, mayor es la dispersión, y este parámetro se ajusta para controlar cuánto se altera la fase de las frecuencias más altas. El parámetro A generalmente debe cumplir con la condición:

$0 \leq A < 1$

A=0: No hay dispersión de fase, y el filtro no tendrá ningún efecto sobre la señal. El sonido resultante será como si no hubiera rigidez en la cuerda, y los armónicos estarán espaciados uniformemente.

0<A<1: En este rango, el filtro introduce dispersión de fase. A medida que AAA se acerca a 1, la cantidad de dispersión aumenta, y las frecuencias más altas se desplazan más hacia arriba. Esto simula un mayor grado de rigidez en la cuerda. Valores de A cercanos a 0 generan un efecto de dispersión muy leve, lo cual es adecuado para cuerdas que son casi flexibles, como las de una guitarra o un violín. Valores de A cercanos a 1 introducen más dispersión de fase y simulan cuerdas con mayor rigidez, como las cuerdas gruesas de un piano o las cuerdas de bajo.

A≥1:Si A está fuera de este rango (es decir, si es igual o mayor a 1), el filtro puede volverse inestable, lo que provocaría comportamientos no deseados en la síntesis, como distorsiones o incluso oscilaciones no controladas.

Filtro de amortiguamiento de cuerda (String-dampling filter) $H_d(z)$

El filtro de amortiguamiento de cuerda es un filtro básico de primer orden que, dentro del algoritmo EKS, modela el amortiguamiento de las vibraciones de una cuerda. Este filtro es crucial para controlar la pérdida de energía en cada oscilación de la cuerda, lo que afecta directamente el decaimiento y la suavidad del sonido generado.

$$H_d(z) = (1 - S) + Sz^{-1}, S \in [0,1]$$

El algoritmo EKS utiliza una línea de retardo (o bucle de retroalimentación) para simular la vibración de la cuerda, y el filtro de amortiguamiento $H_d(z)$ se incorpora en este bucle para controlar el decaimiento de las oscilaciones de la cuerda. Este filtro atenúa las frecuencias más altas y permite que las bajas frecuencias sigan resonando durante más tiempo.

Cuando se pulsa una cuerda en un instrumento real, las frecuencias más altas tienden a decaer más rápido que las bajas debido a la resistencia del material de la cuerda, el aire circundante y otros factores físicos. El filtro $H_d(z)$ simula este comportamiento.

El filtro atenúa más las frecuencias altas conforme el valor de S aumenta y se aproxima a 1, lo que genera un sonido más suave y menos brillante con un decaimiento más rápido de las frecuencias agudas, imitando una cuerda que pierde energía rápidamente debido al amortiguamiento. Las Frecuencias bajas tienden a persistir durante más tiempo, dado que el filtro tiene menos efecto en ellas. Si S=1 El filtro realiza un máximo suavizado, eliminando casi todas las frecuencias altas. El resultado es un sonido apagado, sin brillo, como si la cuerda estuviera fuertemente amortiguada o sofocada.

Para controlar el decaimiento, es necesario incorporar una ganancia a

$$H_d(z) = a((1 - S) + Sz^{-1})$$

Filtro pasa-todo de afinación de cuerda (First-order string-tuning allpass filter) $H_{ro}(z)$

Este filtro se utiliza en el contexto del algoritmo EKS (Extended Karplus-Strong) para ajustar de manera precisa la afinación de la cuerda sin alterar la magnitud del espectro de la señal, sino solo su fase.

$$H_{ro}(z) = \frac{C + z^{-1}}{1 + Cz^{-1}}, C \in (-1,1)$$

Dado que la longitud del ruido inicial es un número entero, se descarta la parte fraccionaria al contar, provocando inexactitudes, es decir, una cuerda desafinada. Por ejemplo, si la frecuencia de muestreo es 44100 y la longitud del ruido es 133 y 134, entonces las frecuencias de señal correspondientes son 331,57 Hz y 329,10 Hz. Y la frecuencia de las notas E de la primera octava (la primera cuerda al aire) es 329,63 Hz. Aquí la diferencia está en décimas, pero, por ejemplo, para el traste 15, la diferencia puede ser ya de varios Hz. Este filtro reduce este error en las frecuencias que produce una guitarra estándar (Tabla 2). Sin embargo, no es útil si la frecuencia de muestreo es alta (muy alta: varios cientos de miles de Hz, o incluso más) o la frecuencia fundamental es baja, como, por ejemplo, para las cuerdas de bajo.

Tabla 2

Frecuencias de las cuerdas de guitarra estándar

Cuerda Guitarra	≈Frecuencia (Hz)
6ta	82,4069
5ta	110
4ta	146,8324
3era	195,9977
2da	246,9417
1era	329,6276

El parámetro C influye directamente en la fase del filtro

Con C=0 el filtro se convierte en un pase directo (no introduce ningún cambio), porque el numerador y denominador resultan en 1. Esto significa que la señal no sufre ningún retraso adicional ni alteración en la fase, simplemente aplicando un retardo de una muestra.

Para valores positivos de C, el filtro introduce un retardo fraccionario positivo. Esto significa que la señal tiene una fase retardada, pero sin alterar su amplitud. A medida que C aumenta hacia 1, el retardo fraccional es mayor, lo que puede afinar la frecuencia de oscilación de la cuerda hacia valores más bajos.

Para C=1, el filtro introduce un retardo completo, equivalente a dos muestras. Esto significa que no hay modificación en la señal (como si fuera una señal de paso directo), pero introduce un retardo máximo en fase sin afectar la magnitud.

Para valores negativos de C, el filtro introduce un adelanto de fase (fase negativa), lo que equivale a un retardo fraccionario negativo. Esto puede ajustar la frecuencia de la cuerda ligeramente hacia valores más altos.

Para C=−1, el filtro genera un cambio en la fase que podría interpretarse como una inversión de fase o una alteración más significativa del retardo fraccional.

RESULTADOS Y DISCUSIÓN

Una vez realizado el análisis de los filtros EKS se procede a parametrizar los valores de los coeficientes de los filtros a usar, mediante la ejecución de una secuencia de comandos (script) de MATLAB (Figura 7). La determinación de los parámetros utilizados se realizó luego de pruebas de audición y variación de éstos dentro de los rangos especificados en el análisis.

Figura 7

Código de Matlab para inicializar los parámetros de los filtros a ser utilizados en la simulación

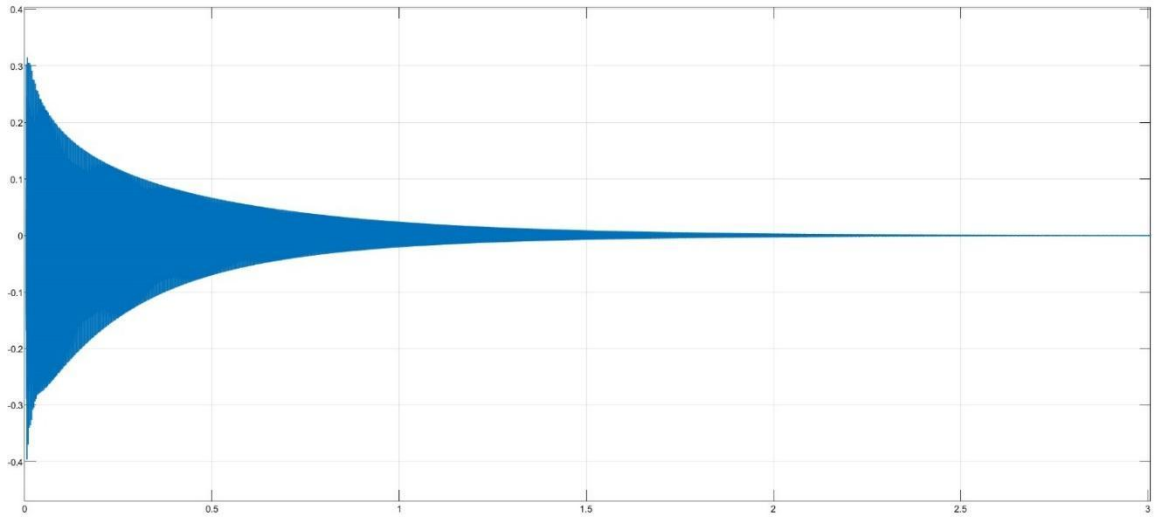
Con estos parámetros, se obtiene el sonido claro y limpio de una primera cuerda (nota mi), metálica, con la duración audible de aproximadamente de 2,5 segundos, con una onda que va decayendo paulatinamente luego de que su amplitud máxima se alcanza (Figura 8). La frecuencia de muestreo utilizada de 44100 Hz permite la generación del sonido con las características descritas, frecuencia que es usada también por (Matti Karjalainen et al., 1993) sin embargo, si se disminuye esta frecuencia a valores de 22050 Hz y 11025 Hz o menores el sonido va perdiendo realismo.

```
1  Fs =44100;%Frecuencia de muestreo
2  Ts=1/Fs;% Tiempo de muestreo
3  c=0.1; %Hro(z)
4  p=0.9; %Hp(z)
5  B=1/10; %HB(z)
6  S=0.75; %Hd(z)
7  A=0.9; %Hs(z)
```

Figura 8

Onda de salida del sistema

Se realizaron variaciones en los parámetros de los diferentes filtros y usando diferentes frecuencias de cuerdas a simular, dentro de los rangos descritos para cada uno de ellos, logrando diferentes variantes del sonido resultante, desde sonidos más o menos metalizados o similares a una cuerda de nylon, pudiendo parametrizarse además su duración con el parámetro de ganancia a. En el trabajo de (Smith-III, 2013) se programa los filtros usando el lenguaje FAUST, un lenguaje de programación de alto nivel diseñado específicamente para la creación de procesadores de audio, sintetizadores, efectos



de sonido y otros sistemas, basado en la composición de funciones matemáticas para definir cómo se procesan las señales de audio; en cambio, el programa Simulink permitió implementar de forma gráfica los filtros necesarios para llevar a cabo la simulación, cuyo resultado se guarda en un archivo WAV.

CONCLUSIÓN

El algoritmo de Karplus-Strong permite simular digitalmente el sonido de una cuerda de guitarra, pudiendo ser implementado en varias herramientas de programación. La implementación en Simulink permite la visualización en un entorno gráfico del diagrama de bloques del sistema, lo que permite su mejor comprensión y parametrización.

Existen un sinfín de filtros que permiten configurar las características del sonido obtenido, cada uno con sus propios parámetros y alcances.

Dado que Simulink tiene las herramientas necesarias para interactuar con placas de hardware externo, como por ejemplo la plataforma de software libre Arduino o Raspberry, basados en los resultados obtenidos en la presente investigación, se podría implementar físicamente el algoritmo KS en las placas mencionadas para la creación de instrumentos virtuales y físicos con fines experimentales o didácticos.

REFERENCIAS

Abdullah, D., Angkasa, K., Hartono, H., & Fithra, H. (2019). Designing Guitar Tuning Software Using Karplus Strong Algorithm. Journal of Physics: Conference Series, 1364(1). Institute of Physics Publishing. https://doi.org/10.1088/1742-6596/1364/1/012027

Assayag, G., Feichtinger, H. G., & Rodrigues, J. F. (2000). Mathematics and Music A Diderot Mathematical Forum.

Freire Diaz, P., López-Mendoza, X., Casignia, B., Cisneros Barahona, A. S., & Uvidia Fassler, M. I. (2020). Classification of Andean Chocho (Lupinus Mutabilis Sweet) by Shape and Color Using Artificial Vision. Artificial Intelligence, Computer and Software Engineering Advances, 1, 64–78. https://doi.org/https://doi.org/10.1007/978-3-030-68080-0_5

Freire, P., Uvidia-Fassler, M. I., López, X., Casignia Vásconez, B., & Guevara, D. (2022). Design and Calibration of an Arduino-Based I2C Hydraulic Flow Sensor. In H. and D. C. A. Botto-Tobar Miguel and Cruz (Ed.), Recent Advances in Electrical Engineering, Electronics and Energy (pp. 181–194). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-08280-1_13

Jaffe David A., & Smith Julius O. (1983). Extensions of the Karplus-Strong Plucked-String Algorithm. Computer Music Journal, 7(2), 56–59.

Karjalainen, M, & Laine, U. K. (1991). A model for real-time sound synthesis of guitar on a floating-point signal processor. [Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing, 3653–3656 vol.5. https://doi.org/10.1109/ICASSP.1991.151066

Karjalainen, Matti, Välimäki, V., & Jánosy, Z. (1993). Towards High-Quality Sound Synthesis of the Guitar and String Instruments. ICMC Proceedings. Retrieved from https://www.researchgate.net/publication/266316737

Karjalainen, Matti, Välimäki, V., & Tolonen, T. (1998). Plucked-String Models: From the Karplus-Strong Algorithm to Digital Waveguides and beyond. In Source: Computer Music Journal (Vol. 22).

Karplus, K., & Strong, A. (1983a). Digital Synthesis of Plucked-String and Drum Timbres. In Source: Computer Music Journal (Vol. 7). The MIT Press.

Karplus, K., & Strong, A. (1983b). Digital Synthesis of Plucked-String and Drum Timbres. In Source: Computer Music Journal (Vol. 7). The MIT Press. https://doi.org/https://doi.org/10.2307/3680062

Mignini, E. A., Miyara, F. S., García Bauza, C., Lotito, P., Parente, L., & Vénere, M. (2009). Síntesis de sonido por modelado físico de instrumentos de cuerda percutida. Mecánica Computacional, XXVIII, 101–111. Retrieved from https://amcaonline.org.ar/ojs/index.php/mc/article/view/2710

Miranda, E. R. (2002). Computer Sound Design, Synthesis techniques and programming (Second). Focal Press. Retrieved from https://www.cl72.org/150soundProc/Miranda%20computer-sound-design-synthesis-techniques-and-programming-music-technology-miranda.pdf

Picquart, M., & Jiménez, L. (2010). Estudio simplificado del timbre de cuerdas percutidas, punzadas y pulsadas. Latin-American Journal of Physics Education, 4(3), 741–752. Retrieved from https://dialnet.unirioja.es/servlet/articulo?codigo=3697960

Santiago, J. R., Samuel, S. J., & Sawn, R. (2014). Modified virtual air guitar: A concept realized using image processing techniques. Proceedings of 3rd International Conference on Context-Aware Systems and Applications, ICCASA 2014. ICST. https://doi.org/10.4108/icst.iccasa.2014.257296

Smith III, J. O. (1996). Physical Modeling Synthesis Update. The Computer Music Jounal, 20(2), 44–56.

Smith-III, J. O. (2013). Making Virtual Electric Guitars and Associated Effects Using Faust. Standford, California 94305. Retrieved from http://ccrma.stanford.edu/realsimple/faust/

Sullivan, C. R. (1990). Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback. In Source: Computer Music Journal (Vol. 14). The MIT Press