Síntesis de la pulsación de la cuerda de una guitarra.

Problema a solucionar.

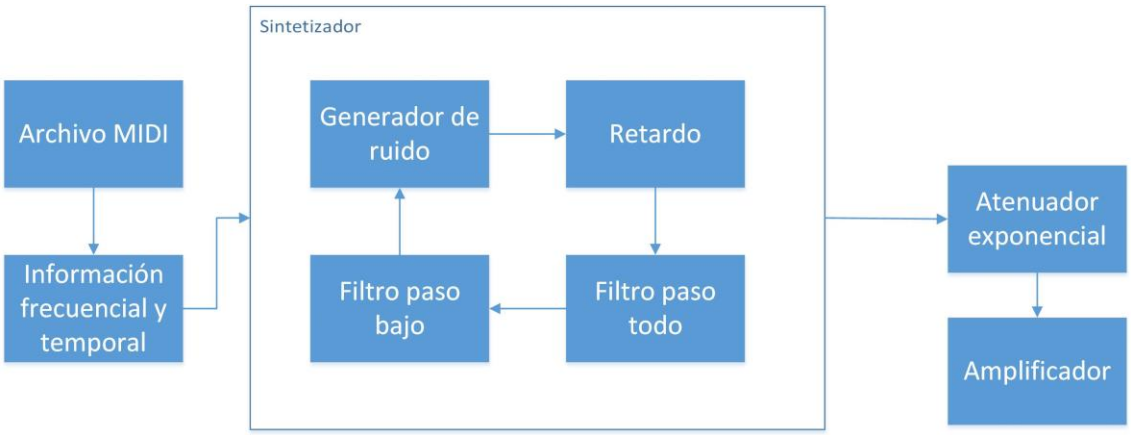
Sintetizar un sonido de una guitarra a partir del algoritmo Karplus-Strong. Además aplicaremos sobre él diversas mejoras para compensar sus carencias, buscando un sonido más fiel a la realidad.

Solución adoptada.

Desarrollaremos la solución empleando filtros digitales utilizando la herramienta Matlab®. Vamos a aplicar, además del algoritmo básico de síntesis de una cuerda las siguientes mejoras:

- Filtro paso todo de compensación de fase. Se encargará de ajustar la afinación del instrumento de forma precisa, es decir, nos permite generar una nota de frecuencia arbitraria. Es necesario ya que el sistema original solo es capaz de generar frecuencias cuantizadas.
- Ajuste del decaimiento de los armónicos. Debido a la estructura del filtro los armónicos de bajas frecuencias tienen una duración mayor que los de frecuencia superior. Por ello aplicamos primero una compresión por un factor independiente de la frecuencia y posteriormente una expansión dependiente de esta permitiéndonos ajustar la duración de todos los armónicos a un mismo valor.
- Reajuste del filtro paso todo debido a que el punto anterior introduce un efecto no deseado. Intuitivamente podemos observar que si introducimos un retardo en el bucle de realimentación dependiente de la frecuencia para compensar la duración de los armónicos, también estamos afectando a la afinación. Por ello debemos compensar esto variando los parámetros del filtro del primer punto.
- Evitar el fin abrupto del sonido generado. La generación de los sonidos tiene un fin cortante por lo que multiplicaremos las últimas muestras de la señal por una exponencial negativa para hacerlo más agradable al oído.
- Coloreo del sonido empleando un sistema distorsionador. Con el objetivo de generar un instrumento más interesante vamos a experimentar añadiendo un sistema parecido a los pedales de distorsión empleados en las guitarras eléctricas.

Esquema del proyecto por bloques



Sintetizador de guitarra

Para el sintetizador se ha optado por hallar los coeficientes del filtro que simula una cuerda en función de los parámetros dependientes de la frecuencia de la nota con la ecuación del filtro y simplificándola:

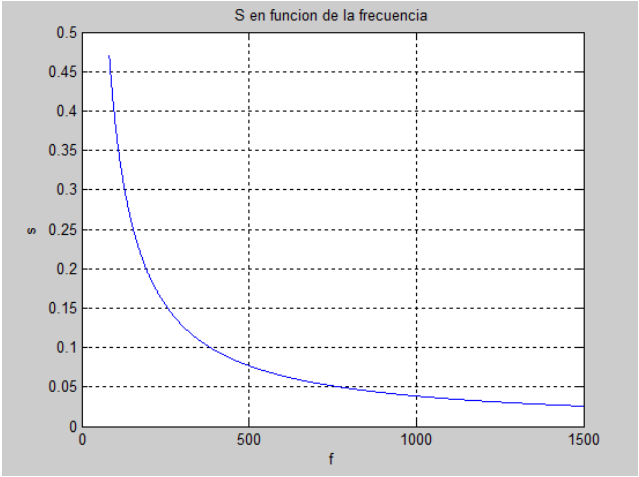
$$Ha \cdot Hb \cdot Hc = z^{-n} \cdot p \cdot (1 - S + S \cdot z^{-1}) \cdot \frac{C + z^{-1}}{1 + c * z^{-1}}$$

$$\frac{z^{-n} \cdot [p \cdot C \cdot (1 - S)] + z^{-n-1} \cdot [p \cdot (1 + C \cdot S - S)] - z^{-n-2} \cdot [p \cdot S]}{1 + C \cdot z^{-1}}$$

$$\frac{1 + C \cdot z^{-1}}{1 + C \cdot z^{-1} + z^{-n} \cdot [p \cdot C \cdot (S - 1)] + z^{-n-1} \cdot [p \cdot (S - 1 - C \cdot S)] + z^{-n-2} \cdot [p \cdot S]}$$

De aquí obtenemos los coeficientes B y A y los usamos con la función filter de Matlab para sintetizar la nota

Debido a que el decay stretching no era lo suficientemente acusado para que las notas agudas tuviesen una duración adecuada, hicimos que el parámetro S fuese adaptativo con la frecuencia siguiendo una función 1/x, es decir una hipérbola. El resultado es el siguiente:



Esto mejor significativamente el sonido de las notas agudas.

Esqueleto del software

- MidiPlayer (script ejecutable)
    - Descripción:
        - Pregunta al usuario el nombre del archivo a reproducir
        - Pregunta al usuario cuantas notas quiere reproducir
        - Pregunta al usuario si quiere distorsion
        - Si aplica, pregunta por la ganancia de la distorsion
        - Obtiene las notas del archivo midi
        - Elige tantas como el usuario haya pedido
        - Las genera usando la funcion track_generator
        - Le aplica distorsión si es necesario usando la funcion Amp_Distort

- Cumpleaños feliz (cumpleanos.m) (script ejecutable)
    - Descripcion
        - Genera un listado de de las notas a tocar
        - Genera un array con la duración en pulsos de cada una de ellas
        - Decide el tiempo de pulso (650ms)
        - Genera la canción llamando a track generator y la reproduce

- Acordes (Script ejecutable)
    - Descripcion
        - Carga la matriz notas_guitarra con las notas
        - Genera los acordes de do, re y mi
        - Los hace sonar cada vez que el usuario pulsa una tecla
- Acorde
    - Entradas
        - Array de notas (frecuencia) del acorde
        - Duracion del acorde en segundos
        - FS en hercios
    - Salidas
        - Array con el acorde solicitado
    - Esquema
        - Llama a la función guitar_synthesizer por cada frecuencia del acorde
        - Forma el acorde sumando todas las notas

- Notas guitarra (archivo de matriz)
    - Contenido
        - Array bidimensional con las frecuencias que puede tocar una guitarra

        - 6 filas y 13 columnas
        - Cada fila representa su correspondiente cuerda
        - Cada columna es el número de traste +1
        - La cuerda al aire está representada por la primera columna

- Track generator
    - Entradas:
        - ~~Vector de notas en forma de strings.~~
        - Vector de notas en Hz.
        - Vector de enteros de tiempo en milisegundos de inicio de cada nota.
        - Vector de enteros de tiempo en milisegundos de final de cada nota.
        - Frecuencia de muestreo
    - Salidas:
        - Vector de amplitud de la señal en el tiempo.
        - Frecuencia de muestreo en hercios de la señal de salida.
    - Esquema:
        - ~~Paso de notas de string a frecuencia fundamental.~~
        - Paso de inicio y fin de cada nota de tiempo a número muestras.
        - Para cada nota de entrada, llamar a la Función 2 con la frecuencia. fundamental y la longitud deseada y sumar a la señal de salida la nota. generada colocada en el índice adecuado.
- Guitar synthesizer
    - Entradas:
        - Frecuencia de la nota a generar en forma de entero en hercios.
        - Longitud deseada de la nota en entero de número de muestras.
        - Frecuencia muestreo.
    - Salidas:
        - Vector con la nota.
    - Esquema:
        - Generación de ruido.
        - Implementación de la función de transferencia del generador usando la función filter de Matlab® con los coeficientes calculados.
        - Generación artificial del decaimiento de la nota en las últimas. muestras usando la funcion Exp_decay.

- Exp decay
    - Entradas
        - Vector de muestras a las que aplicar el decaimiento
        - longitud del decaimiento en muestras desde el final del vector
    - Salidas
        - Vector de muestras con decaimiento al final
    - Esquema
        - Creamos un indice para la exponencial de la longitud del decaimiento

        - lo aplicamos a una funcion exponencial decreciente
        - lo concatenamos con un vector de unos de la longitud restante de la nota para mantener la parte inicial intacta
        - multiplicamos el vector de muestras con el vector de coeficientes de decaimiento
- Amp Distort
    - Entradas
        - Vector de entrada a distorsionar
        - Ganancia de distorsion a aplicar
    - Salidas
        - Vector distorsionado
    - Esquema
        - Se obtiene el valor de saturacion del amplificador
        - se amplifica cada muestra
        - se satura si es necesario

Software externo

- Librería matlab-midi, de código libre y disponible en GitHub: https://github.com/kts/matlab-midi
- Se emplea para leer un archivo .mid y convertirlo en una estructura de datos interpretable y modificable con Matlab. Para ello se emplea la función readmidi, que toma como parámetro un archivo preparado a tal efecto (un solo canal, únicamente la pista de guitarra) y devuelve una estructura que podemos emplear.
- A partir de aquí usamos la función propia midi2freqseq, que genera una matriz de Nx3. N es el número de notas a interpretar, mientras que en cada columna se introduce de arriba a abajo:
    - Frecuencia en Hz, obtenida usando la función midi2freq de la librería.
    - Tiempo de inicio de la nota respecto al principio de la canción en ms.
    - Tiempo de fin de la nota respecto al principio de la canción en ms.

Para testar el sistema completo usamos tres archivos .mid adjuntos a la carpeta del proyecto

División de trabajo.

Todas las partes se han realizado entre todos los participantes del grupo.

Participantes.

- Amadeo de Gracia Herranz
- Gregorio Juliana Quirós
- Álvaro Rodríguez Villalba
- Carlos Santos Rancaño