INSTITUTO TECNOLÓGICO
DE
BUENOS AIRES

Ingeniería Electrónica

22.05 - Análisis de Señales y Sistemas Digitales

TRABAJO PRÁCTICO 2

Análisis y síntesis de Audio Digital

Grupo 4:	*Legajos:*
Díaz, Ian Cruz	57515
Álvarez, Lisandro	57771
Fogg, Matías	56252
Diéguez, Manuel	56273
Chiocci, Ramiro	45132

Contenido

1. Fast-Fourier Transform (FFT)

1.1. Teoría

Para aplicar el algoritmo de la *Fats-Fourier-Transform*, se utilizó el algoritmo de *Decimation-in-time*, mas conocido como el algoritmo de Cooley-Tukey, que se desarrollara a continuación[1].
Partiendo de la base que lo que se quiere obtener es,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{-kn}\,, \;\; W_N = e^{j2\pi/N} \tag{1.1}$$

Teniendo en cuenta esto, y a sabiendas de que N se puede escribir de la manera $N = 2^r$ siendo $r \in \mathbb{N}$, se puede partir esa sumatoria en partes pares e impares, quedando,

$$X(k) = \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{-kn} + W_N^{-k} \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{-kn} \tag{1.2}$$

Por lo tanto, si llamamos $x_p(n) = x(2n)$ y $x_i(n) = x(2n+1)$, podemos reescribir la expresión como,

$$X(k) = X_p(k) + W_N^{-k} X_i(k) \tag{1.3}$$

Finalmente, dada la periodicidad $X_p(k + \frac{N}{2}) = X_p(k)$ y $X_i(k + \frac{N}{2}) = X_i(k)$
Obtenemos que,

$$X\left(k + \frac{N}{2}\right) = X_p(k) - W_N^{-k} X_i(k) \tag{1.4}$$

Finalmente, de la expresión 1.4, podemos ver que guardando los valores obtenidos para la mitad del intervalo de la FFT, podemos reutilizarlos para calcular la mitad faltante. El algoritmo de Cooley-Tukey utiliza esta propiedad, subdividiendo recursivamente las partes pares e impares hasta llegar a una función con solo 1 elemento.

1.2. Implementación

La implementación del algoritmo se puede observar a continuación[2].

[1] Para mas detalle sobre este algoritmo o otras implementaciones de la FFT se puede dirigir a la pagina 362 del libro [1] y en la sección 8.1 del libro [4]

[2] Para la implementación del algoritmo de Cooley-Tukey se utilizo la lógica de implementación de [7]

```
void fft(std::complex<float>* in, std::complex<float>* out, size_t n)
    {
     vector<complex<double>> temp;
     for (size_t i = 0; i < n; i++)
         temp.push_back(in[i]);
     temp = Cooley_Tukey(temp);
     for (size_t i = 0; i < temp.size(); i++)
         out[i] = temp[i];
     return;
}

std::vector<complex<double>> Cooley_Tukey(std::vector<complex<double
    >>& samples) {
     unsigned int numSamples = samples.size();

     if (numSamples <= 1)
         return samples;
     unsigned int newSampleSize = numSamples / 2;
     std::vector<complex<double>> Xeven(newSampleSize, 0);
     std::vector<complex<double>> Xodd(newSampleSize, 0);

     for (unsigned int i = 0; i < newSampleSize; i++) {
         Xeven[i] = samples[2 * i];
         Xodd[i] = samples[2 * i + 1];
     }
     std::vector<complex<double>> Feven(newSampleSize, 0);
     std::vector<complex<double>> Fodd(newSampleSize, 0);

     Feven = Cooley_Tukey(Xeven);
     Fodd = Cooley_Tukey(Xodd);

     std::vector<complex<double>> freqBins(numSamples, 0);

     for (unsigned int i = 0; i < (numSamples / 2); i++) {
         complex<double> cmplxExponential = polar(1.0, (-2 * 3.14159 *
    i / numSamples)) * Fodd[i];
         freqBins[i] = Feven[i] + cmplxExponential;
         freqBins[i + numSamples / 2] = Feven[i] - cmplxExponential;
     }
     return freqBins;
}
```

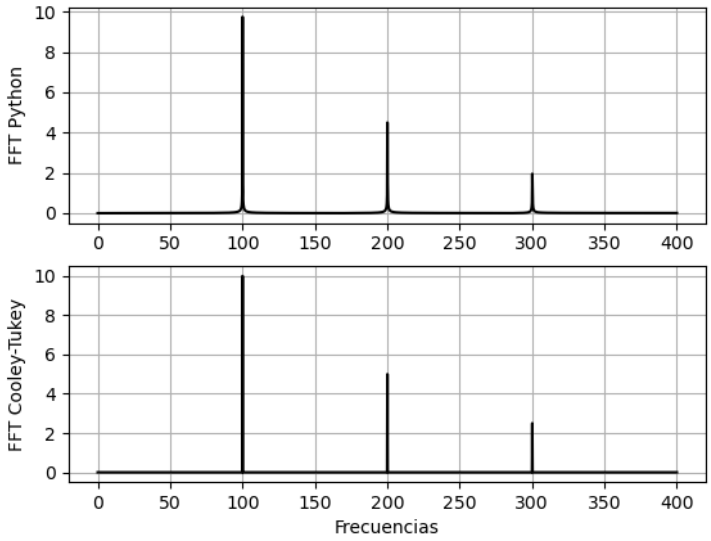


FIGURA 1.1: Resultados del algoritmo de FFT Cooley-Tukey

1.3. Resultados

Comparando los resultados para la secuencia de datos expresada en la ecuación 1.6, utilizando la función FFT de python se obtuvieron los resultados que se muestran en la figura 1.1.

$$x(n) = 10sin(2\pi 100nT) + 5sin(2\pi 200nT) + 2.5sin(2\pi 300nT) \tag{1.5}$$

$$T = \frac{1}{800} \; y \; 0 < n < 4096 \tag{1.6}$$

2. Síntesis basada en muestras

2.1. Introducción

La síntesis basada en muestras [3] es muy diferente a otras formas de síntesis como la síntesis aditiva. Como se vio anteriormente este tipo de síntesis utiliza funciones básicas como el seno, cuadrada, triangular o diente de sierra para sintetizar distintos sonidos. Sin embargo, la síntesis basada en muestras utiliza una muestra para recrear otro sonido. Esta muestra, que de ahora en mas se llamara *Sample*, puede ser un sonido o instrumento grabado. En consecuencia, esta forma de síntesis permite recrear un sonido mas realista ya que se esta trabajando sobre un sonido real. Ademas, demanda menos de la CPU que otras formas de síntesis, como la de síntesis mediante modelos físicos, ya que no se no se calcula la señal de salida en tiempo real sino que se utilizan *Samples* pre-grabadas. No obstante, este tipo de síntesis lleva asemejada una dificultad. Suponerse que se quiere sintetizar un piano. Mediante este método es posible pero se requerirían tener pre-grabadas todas las notas del piano con todas las duraciones posibles. Esto es claramente imposible ya que, la memoria es finita. Por esta razón, se utilizan algoritmos para realizar estiramientos y compresiones temporales y en frecuencia para formar a partir de una nota pre-grabada otra nota con distinta frecuencia y distinta duración. Este es el objetivo de esta sección, poder determinar un algoritmo apropiado para que, a partir de una nota grabada, se pueda crear otra nota de distinta frecuencia y duración. De esta manera, contando con unas pocas muestras, lograr sintetizar un instrumento. Teniendo todo esto en consideración, se propone lograr sintetizar un piano.

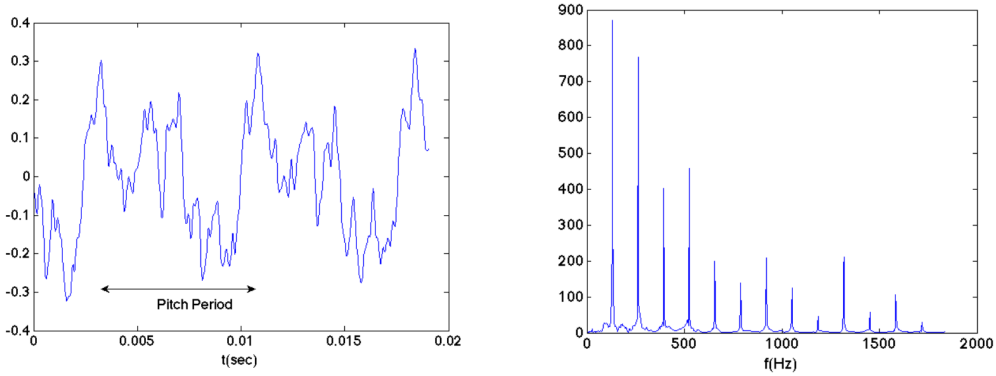
2.2. Pitch shifting

En primer lugar se analiza la posibilidad de modificar la altura de una nota. La altura o *Pitch* [6] es una de las cuatro cualidades esenciales del sonido. Dicha cualidad determina si un sonido es agudo o grave. A mayor frecuencia, más agudo será el sonido y a menor frecuencia, más grave será el mismo. El *Pitch* es determinado por la frecuencia fundamental de la señal.

Comprendiendo estas nociones básicas, se define el objetivo de esta sub-sección. A partir de un sonido con cierto *Pitch*, como una nota de un piano, se quiere modificar su frecuencia fundamental para poder generar otra nota con la misma duración que la original. Esto es posible mediante un corrimiento en frecuencia o *Pitch shifting*. Este proceso implica dos pasos: primero se debe identificar el *Pitch* (*Pitch Detection*) y en segundo lugar, se debe hacer el corrimiento o corrección (*Pitch Correction*).

[3] Más información se puede encontrar en [6]

2.2.1. *Pitch Detection*



(A) Dominio del tiempo

(B) Dominio de frecuencia

FIGURA 2.1: *Sample* de C3 de un piano

El *Pitch* de una señal se puede analizar en el dominio de la frecuencia como en el del tiempo. En la Figura 2.1 se muestra un *sample* de la nota C3 de un piano. Notese que, el grafico en el dominio de la frecuencia dice mucho mas que el otro gráfico. Se pueden ver claramente los picos de la señal. El pico mas predominante determina la frecuencia fundamental y, como se dijo anteriormente, la frecuencia fundamental es el *Pitch* de la señal. Para esta señal en particular, el *Pitch* es $130.8128Hz$. Por otro lado, ver que el gráfico en el dominio del tiempo el *Pitch* es periódico. Esto es mucho menos intuitivo que el caso anterior. Por ello, la detección del *Pitch* es mejor analizarla en el dominio de la frecuencia.

Los métodos típicos de detección incluyen métodos de autocorrelacion o por medio de promedio de diferencia de magnitud. Estos métodos son muy interesantes si no te conoce la señal a la que se va a realizar *Pitch shifting*. No obstante, como el objetivo de la sección es sintetizar notas de un piano, ya se conocen los distintos *Pitch* de antemano. Recordando conceptos de teoría musical, la frecuencia de la $k-esima$ nota de un piano esta dada por la ecuación:

$$f = 2^{\frac{k-49}{12}} \cdot 440Hz \tag{2.1}$$

Por ejemplo, la nota C3 que es la nota numero 28 tiene una frecuencia fundamental o *Pitch* de $130.8128Hz$, como se vio en la Figura 2.1. En resumen, con (2.1) se tienen todos los *Pitch* de todas las notas del piano[4]. Ademas, de la ecuacion 2.1 se desprende que cada nota esta separada por $2^{\frac{1}{12}}$. Por ejemplo, si se tiene C3, C#3 sera $130.8128Hz \cdot 2^{\frac{1}{12}}$. D3 sera $130.8128Hz \cdot 2^{\frac{2}{12}}$ y asi sucesivamente.

[4]Más información se puede encontrar en [3]

A continuación se realiza un análisis sobre un *Sample* para evidenciar la teoría descripta anteriormente. El análisis se realiza con el programa de uso gratuito *Audacity*. Dicho programa permite realizar el espectro en frecuencia de un *Sample*. En la Figura 2.2 se muestra el espectro de un *Sample* de la nota C4 de un piano electrónico clásico. El *Sample* fue adquirido mediante el programa *GrageBand*.

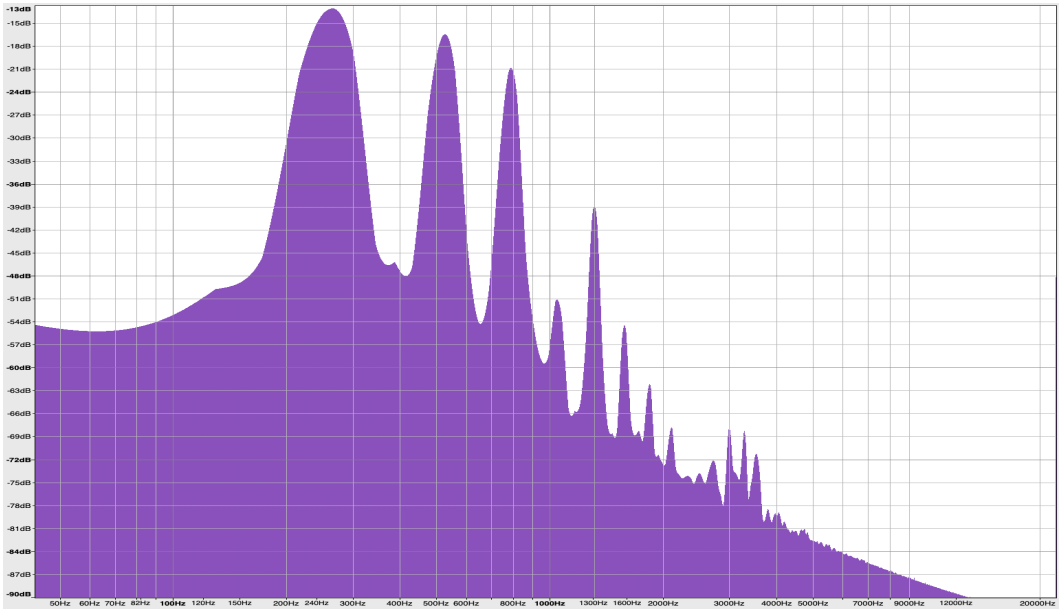


FIGURA 2.2: Espectro de C4 de Piano Electrónico Clásico. Generado en *Audacity* con ventana Hamming

Como se puede ver en la Figura 2.2, la frecuencia fundamental es $264Hz$ por lo que efectivamente es la nota C4. Este tipo de análisis es el que se va a realizar para determinar el *Pitch* de cada *Sample*.
Como ultima aclaración sobre el tema. No confundir *Pitch* que tiene unidades de Hz y se observa en el dominio de la frecuencia con el *Pitch Period* que es el reciproco del *Pitch*, tiene unidades de *segundo* y se observa en el dominio del tiempo.

2.2.2. *Pitch Correction*

En la sub-seccion anterior se analizo como se determina el *Pitch* de un *Sample*. Ahora se analiza como modificar o , mejor dicho, corregir el *Pitch* de un *Sample* para generar un nuevo tono. Los algoritmos de corrección pueden trabajar en el dominio del tiempo o en el dominio de la frecuencia. Generalmente los que trabajan en el dominio de la frecuencia son muy robustos pero sufren de largas latencias. Por lo cual, se trabaja con un método en el dominio del tiempo.
La gran mayoría de las técnicas que trabajan en el dominio del tiempo para realizar *Pitch Correction* están basados en el método TD-PSOLA. TD-PSOLA (*Time-Domain*

Pith Synchronous Overlap and Add method). Este es el método que se utiliza en esta sección ya que es rápido, simple y, como se vera mas adelante, tiene muy buenos resultados.
El método TD-PSOLA se puede dividir en tres pasos:

1. Descomponer la señal de entrada en una serie de señales de análisis de tiempo corto

2. Modificación de las señales de análisis de tiempo corto a señales de síntesis de tiempo corto

3. Composición de la nueva señal con las señales de síntesis de tiempo corto. Aquí esta el concepto de *overlap-add*

En el paso 3. es donde ocurre lo interesante. Al recomponer la señal, las señales de síntesis de tiempo corto se pueden superponer o alejar entre ellas. Esto hace que varié la frecuencia fundamental de la señal. Si la señales se acercan, aumenta la frecuencia fundamental (mayor *Pitch*). Por el contrario, si las señales se alejan entre si, disminuye la frecuencia fundamental (menor *Pitch*). En la Figura 2.3 se ve una ilustración del método que permite comprender su funcionamiento. Teniendo en mente los pasos del método se continua a desarrollar en detalle cada uno.

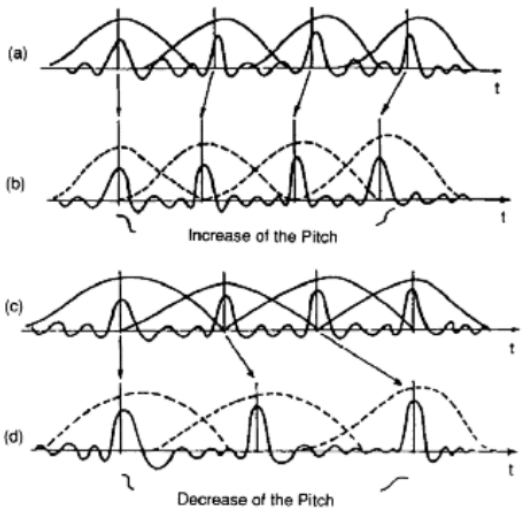


FIGURA 2.3: Ilustración de método TD-PSOLA

PASO 1. Este paso consiste en descomponer la señal de entrada $x(n)$ en una secuencia de señales de análisis de tiempo corto, llamadas $x(s, n)$. Estas señales son obtenidas multiplicando $x(n)$ por una secuencia de ventanas de análisis trasladadas en tiempo. Es decir:

$$x(s, n) = h_s(n)x(n - t_a(s)) \tag{2.2}$$

Donde $h_s(n)$ es la ventana de análisis con soporte en $n = 0$ y $t_a(s)$ es el s-esimo instante de tiempo de análisis. Estos instantes de tiempo son llamados *pitch-marks* y son los

picos en $x(n)$. En el caso de las notas de un piano, estos picos son periódicos. Este periodo es el $P(s)$ (*Pitch Period*).
Volviendo a la ventana de análisis, su longitud T es proporcional a $P(s)$. Generalmente, se suele tomar $T = \mu P(s)$ con $\mu = 2$. La forma de la ventana $h_s(n)$ es arbitraria. En este caso se utiliza la ventana Hanning por su comportamiento espectral.
PASO 2 El segundo paso consiste en transformar la secuencia de señales de análisis de tiempo corto en una secuencia de señales de síntesis de tiempo corto. Estas ultimas están sincronizadas con un nuevo conjunto de instantes de tiempo $t_s(\mu)$ llamados *synthesis pitch-marks*. La serie de $t_s(\mu)$ están determinados por la serie de instantes $t_a(s)$, dependiendo del *Pitch correction* que se quiera realizar. Este es el concepto mas importante de TD-PSOLA. Sea P_0 el *Pitch Period* original de la señal y P_1 el *Pitch Period* de la señal que se quiere sintetizar. Entonces, el corrimiento que se quiere hacer se puede expresar como $P_1 = \frac{1}{\beta} \cdot P_0$ siendo β el factor de corrimiento. Por ejemplo, si se quiere sintetizar C#4 a partir de C4, osea correr un semi-tono, $B = 2^{\frac{1}{12}}$. De esta manera, la señal sintetizada tendrá los *pitch-marks* de la señal original pero ahora espaciados P_1. Estos *synthesis pitch-marks* determinan $t_s(\mu)$. La Figura 2.4 ayuda mucho en la compresión de este paso.

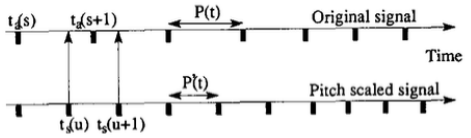


FIGURA 2.4: Configuración de los *synthesis pitch-marks* para un $\beta = \frac{3}{2}$

PASO 3 Composición de la señal sintetizada. Ya tenemos los *synthesis pitch-marks* por lo que faltaría centrar cada señal de síntesis de tiempo corto a dichos instantes de tiempo. Pero surge un pequeño problema. Si se quiere mantener la duración de la señal original puede que una señal de síntesis se deba repetir en dos o mas *synthesis pitch-marks*. Luego, para determinar que señal se repite simplemente se calcula de cual *pitch-mark* esta mas cerca el *synthesis pitch-mark* que se esta evaluando. De esta manera se va 'rellenado' la señal sintetizada. Dependiendo el valor de β puede ser que las señales de síntesis queden sobrepuestas por lo que se debe hacer *operlap-add* para formar la señal sintetizada.
En la Figura 2.5 se muestra el método sobre un *Sample* de C4 de un piano electrónico clásico para sintetizar C#4 (frecuencia fundamental un semi-tono mayor que C4). En el primer gráfico, se detectan los picos o *pitch-marks* de la señal separados aproximadamente por el *Pitch Period*. En el segundo gráfico se ponen las ventanas centradas en los *pitch-marks*. En el tercer gráfico se muestran señales de análisis de tiempo corto ya modificadas por la ventana. En el cuarto gráfico se muestra la señal sintetizada con los *synthesis pitch-marks*. Faltaría determinar si la frecuencia fundamental de la

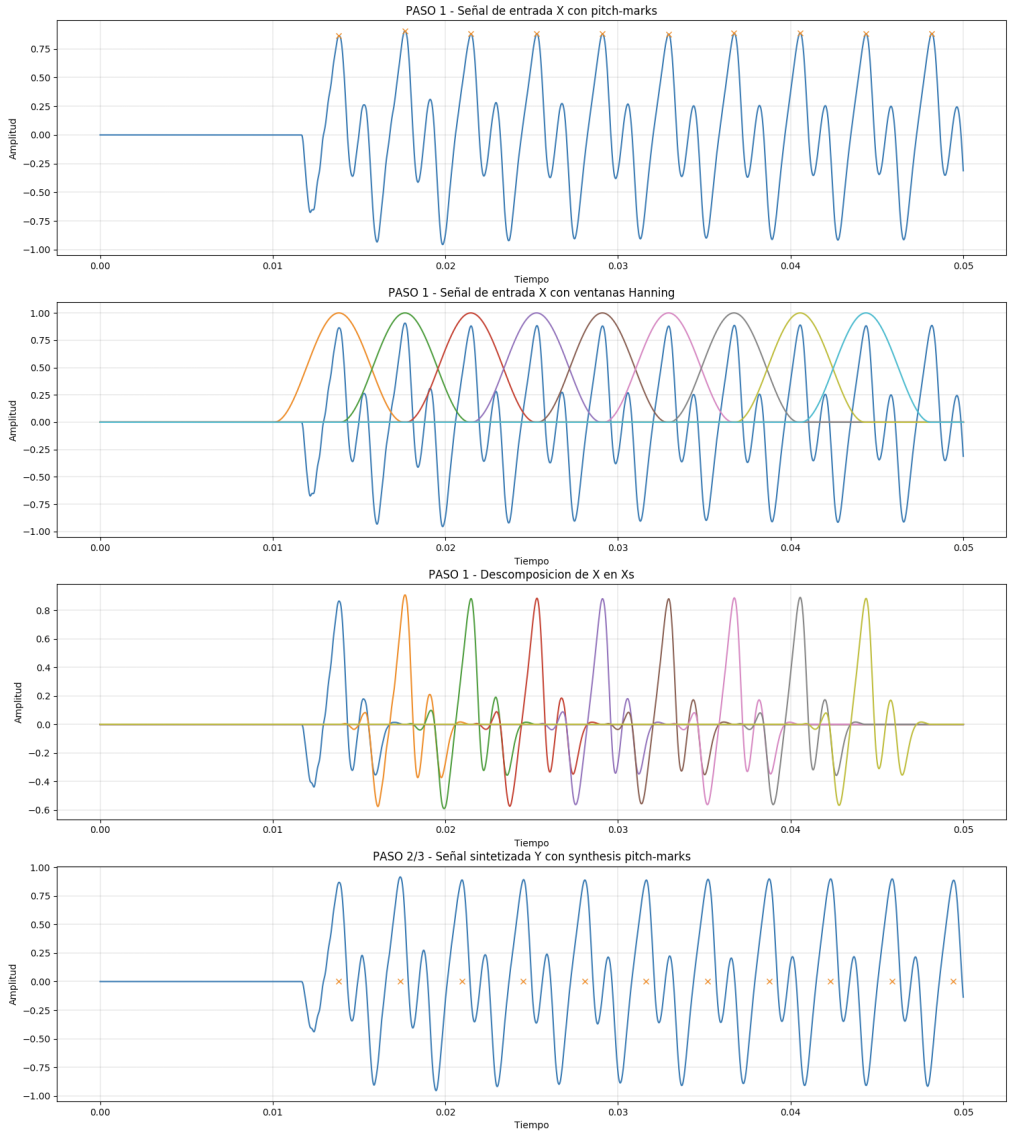


FIGURA 2.5: Método TD-PSOLA sobre un *Sample* de C4 con $\beta = 2^{\frac{1}{12}}$

señal sintetizada es efectivamente la de C#4 ($279Hz$). En la Figura 2.6 se muestra el espectro de la señal sintetizada. Efectivamente se logro un corrimiento en la frecuencia fundamental.

2.2.3. Conclusion

En conclusión, se logro hacer satisfactoriamente *Pitch shifting* sobre un *Sample*. Se pudo realizar el corrimiento de la frecuencia fundamental de la muestra manteniendo la duración de la misma. Como consideraciones adicionales se puede trabajar utilizando distintas ventanas como Barlett o similar y con distinto ancho para analizar los distintos resultados.

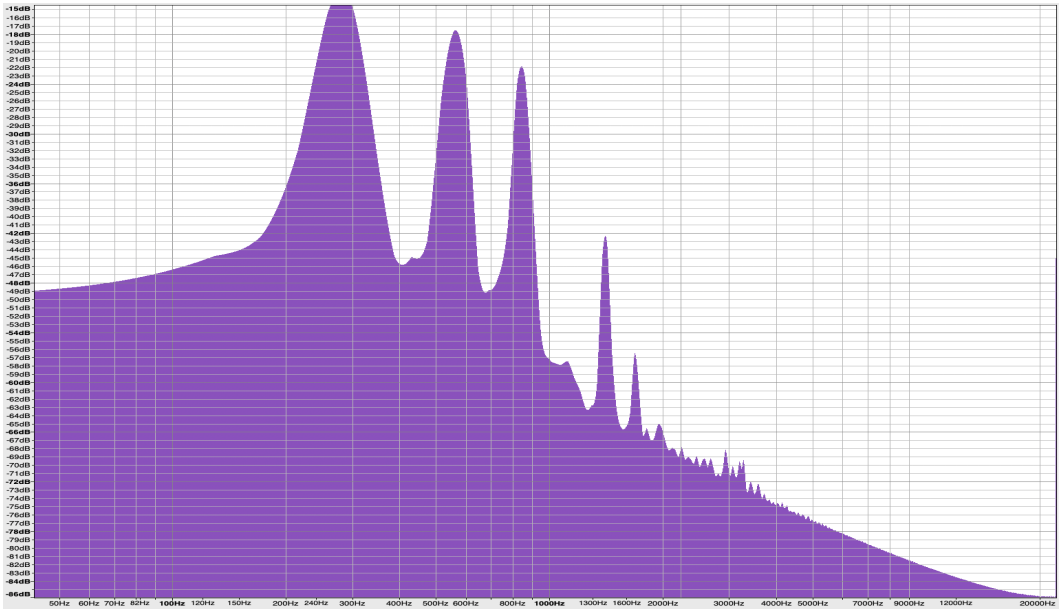


FIGURA 2.6: Espectro de la señal sintetizada

2.3. Time-scale modification

En segundo lugar, se analiza el escalamiento temporal. El método para realizar el escalamiento es muy parecido al método para realizar el *pitch shifting*. La diferencia es que ahora no interesa cambiar el *Pitch* sino que, lo que se quiere es modificar el tiempo de la señal. Para lograrlo, también se utiliza el método TD-PSOLA. Los pasos del método son iguales a los descriptos en *Pitch Correction* y otros sufren algunos cambios. El paso 1 queda exactamente igual, se descompone la señal en una serie de señales de análisis de tiempo corto. El paso 2 sufre una pequeña corrección. Recordar que ahora se desea tener el mismo *Pitch* por lo que la distancia entre cada señal de síntesis debe tener la misma distancia entre una de la otra que las señales de análisis. Esto seria $P_0 = P_1$. De aquí se desprende el problema principal. Suponerse que se quiere acortar el tiempo de la señal. Entonces, se deben quitar algunas señales de síntesis para que el tiempo total se reduzca pero que el *Pitch* se mantenga. Lo contrario sucede si se quiere aumentar el tiempo de la señal. Para lograrlo se deben repetir señales de síntesis para rellenar el tiempo. Al conocer la posición de cada señal de síntesis es posible continuar con el paso 3 y componer la señal final.
En la Figura 2.7 se ve ilustración del método de modificación temporal.

2.4. Implementación

En esta sección se explica la implementación de los métodos anteriormente descriptos para lograr la síntesis por *samples* de un piano.

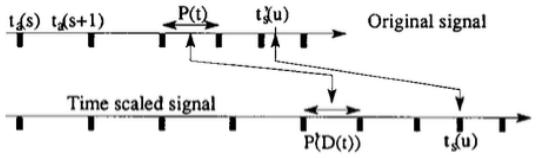


FIGURA 2.7: *Time-scale modification*

Primero se buscaron las muestras o *samples*. Para sintetizar un piano se considera utilizar las muestras de las notas C1,C2,C3,C4,C5,C6. Esto permite sintetizar todas las muestras ya que se tiene una muestra por octava. Es decir si se desea la nota D3 se va a utilizar la muestra C1 para hacele el corrimiento de frecuencia correspondiente. Si se desea A5 se usa C5 y asi sucesivamente. Como se podrá ver, cada muestra de octava podrá sintetizar 11 notas (cada octava esta separada por 12 notas). Al tener las muestras se continua a hacer un análisis espectral de cada una para determinar su frecuencia fundamental. Teniendo todo lo necesario, se continua a aplicar los métodos anteriormente explicados.

En la Figura 2.8 se ve una modificación de una nota C4. El primer gráfico muestra la señal original, el tercero muestra C4 con un corrimiento de dos semitonos y la ultima muestra un alargamiento temporal de la muestra. El segundo gráfico muestra algo interesante. Notar que se repiten fragmentos de la señal para rellenar el tiempo. Esto se implemento de esta forma ya que asi es como trabajan los sintetizadores de muestra. Siempre se le da al usuario la opción de seleccionar el fragmento de la señal que se va a repetir en caso de que se quiera sintetizar una señal con mayor duración que la señal original.

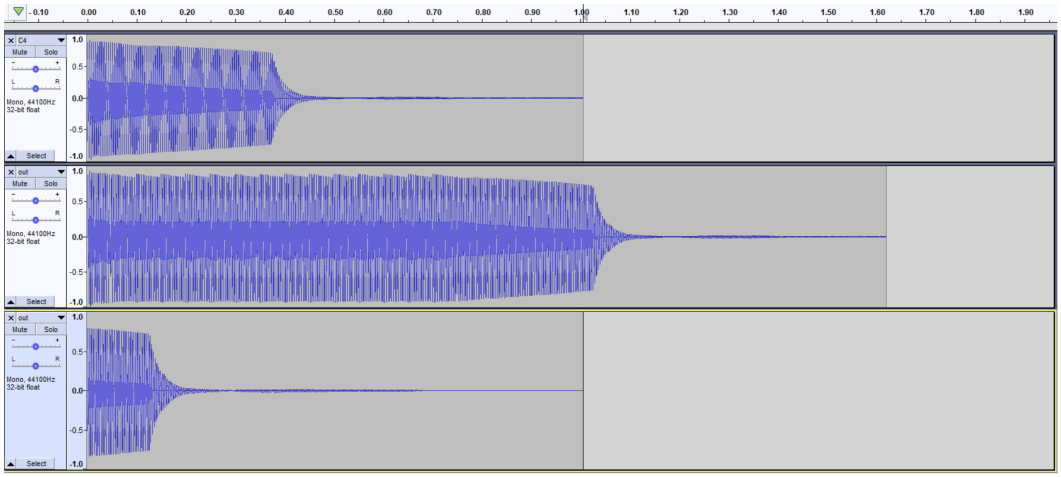


FIGURA 2.8: *Time-scale modification* y *Pitch correction* sobre una muestra C

2.5. Conclusión

Se pudo comprender las bases de síntesis por muestras, su funcionamiento e implementación. Se logro realizar el corrimiento temporal y de la frecuencia fundamental para sintetizar cualquier nota de un piano.
Con los conocimientos adquiridos se pudo sintetizar varios instrumentos como una piano steinway, un piano eléctrico, una guitarra entre otros.

2.6. Referencias

1. Non-parametric techniques for pitch-scale and time-scale modification of speech. Disponible en: https://arxiv.org/pdf/1801.06492.pdf

2. Epoch-Synchronous Overlap-Add (ESOLA) for Time- and Pitch-Scale Modification of Speech Signals. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.475.4144&rep=rep1&type=pdf

3. Phase Vocoder Implementation with FLWT and TD-PSOLA. Disponible en: https://web.stanford.edu/class/ee264/projects/EE264_w2015_final_project_kong.pdf

4. Review of Speech Synthesis Technology. Disponible en: http://research.spa.aalto.fi/publications/theses/lemmetty_mst/thesis.pdf

5. Time-Scale Modification Algorithms for Music Audio Signals. Disponible en: https://www.audiolabs-erlangen.de/content/05-fau/professor/00-mueller/01-students/2011_DriedgerJonathan_TSM_MasterThesis.pdf

6. How to make a noise: sound design and synthesiser programming

Videos de interes

1. http://www.speech.zone/td-psola-the-hard-way/

Sitios Web de interes

1. https://www.blackghostaudio.com/blog/the-beginners-guide-to-audio-synthesis

2. https://reverb.com/news/10-types-of-synthesis

3. Síntesis Aditiva

3.1. Introducción

La síntesis aditiva se basa en el concepto de las *Series de Fourier*, puntualmente en el hecho de que una señal periódica se puede escribir como un sumatorio de señales senoidales de frecuencias y amplitudes diferentes:

$$y(t) = \sum_{k=1}^{N} A_k cos\left(2\pi k f_0 + \phi_k\right) \tag{3.1}$$

La expresión 3.1 permite generar funciones periódicas uniformes, pero estas señales no representan fielmente la señal generada por un instrumento. Cada instrumento tiene un timbre determinado, es decir, su sonido tiene distribución de potencia de harmónicos característica. Además de un timbre característico, al sonido del instrumento lo caracteriza la envolvente de amplitud que modula su sonido.
Entonces, la señal de un instrumento puede ser modelada como una suma de señales de distintas frecuencias a las que llamaremos *parciales*, ya que no necesariamente deben ser múltiplos enteros de la frecuencia central de la señal como indica la expresión 3.1. Además, estos coeficientes A_k correspondientes a cada parcial, se encuentran modulados por una envolvente en tiempo, y entonces se llega a:

$$y(t) = \sum_{k=1}^{N} A_k(t) cos\left(2\pi k f_0 + \phi_k\right) \tag{3.2}$$

Considerando lo expuesto, en el presente informe se propusieron dos modelos de síntesis aditiva. La primera de ellas, a la cual denotaremos *ADSR*, propone que la envolvente que modula en amplitud es la misma para todos los parciales que componen la señal. La segunda propuesta asume, como expuesto en 3.2 que cada parcial esta modulada por una envolvente distinta $A_k(t)$. En los siguientes apartados se explica el procedimiento implementado para obtener los modelos de 4 instrumentos diferentes utilizando la segunda propuesta de síntesis.

3.2. Obtención de parámetros

Se propuso implementar la síntesis aditiva de 4 instrumentos: piano, clarinete, trombón y trompeta. Para cada uno de estos instrumentos se tomaron muestras de audio en formato .wav, y se les aplico un filtro pasa banda muy selectivo, de forma que pudiera obtenerse la señal envolvente $A_k(t)$ correspondiente al parcial de interés k. Particularmente se seleccionaron los primero 12 harmónicos de las señales.

Esta metodología se repitió para una muestra de cada octava perteneciente al conjunto de octavas que el instrumento es capaz de reproducir, dado que cuando se quiera sintetizar una nota de, por ejemplo, la 6ta octava, su síntesis sera mas fiel si sus parámetros se obtuvieron de una muestra correspondiente a esa octava y no, por ejemplo, a la 1ra octava.

3.2.1. Desarrollo parámetros de piano A5

A modo de explicar la metodología implementada, se desarrollará el método para la nota A perteneciente a la 5ta octava para un piano acústico. En primer lugar, se tomaron las curvas envolventes de una muestra de un archivo .wav para los primeros 12 harmónicos, es decir para $f = kf_0$, con $k = 1, 2, .., 12$ y $f_0 = 880.0Hz$

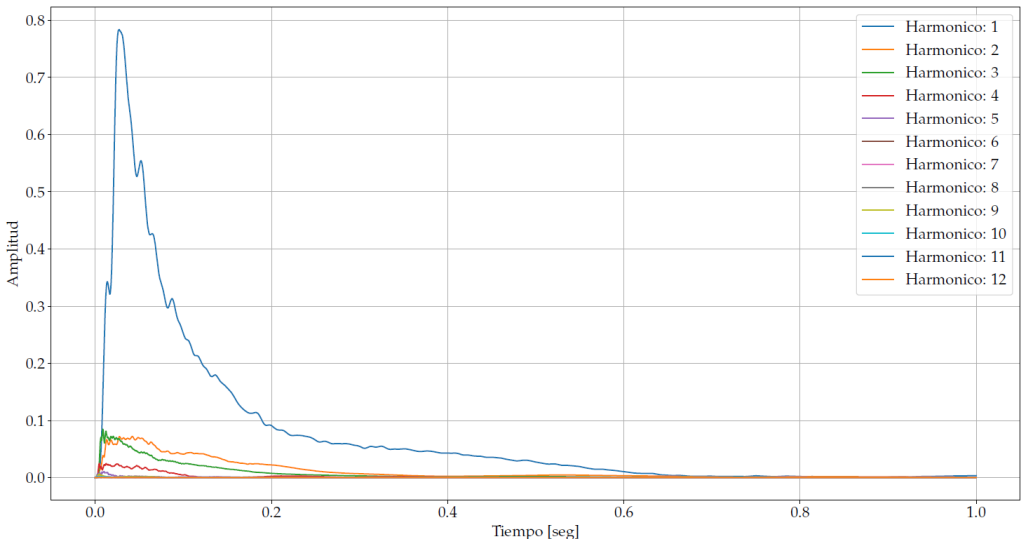


FIGURA 3.1: Envolventes para harmónicos de nota A5

La figura 3.1 muestra la distribución de las envolventes para los 12 parciales seleccionados. Luego al momento de sintetizar una nota, se aplicó la expresión 3.2 utilizando el $A_k(t)$ correspondiente a la octava de la nota a sintetizar.
A continuación se muestra el código de Python utilizado para obtener los datos mencionados:

```
import numpy as np
from numpy import sign
from scipy import signal, interpolate
import matplotlib.pyplot as plt
import math
from scipy.io import wavfile
from distutils.spawn import find_executable
```

```
import aifc
import xlsxwriter
import math
from scipy import fftpack

file = 'samples/A5-880.0.wav'
oct = 5

if find_executable('latex'):
    plt.rc('font', **{'family': 'serif', 'serif': ['Palatino'], 'size': 17})
    plt.rc('text', usetex=True)

peaks = []
datas = []
freqs = []

octave_envelopes = []
rate, data = wavfile.read(file)
d = data.astype(float)
d = (data[:,0] + d[:,1]) / 2
maxD = max(d)
data = [i/maxD for i in d[0:441000]]
t = np.arange(len(data))/rate
f0 = float(file.split('.')[0].split('-')[-1])

plt.plot(data)
plt.show()

for i in range(1,13):
    fi = i * f0
    freqs.append(fi)
    fs1 = 0.9*fi
    fs2 = 1.1*fi
    ws = [fs1, fs2]

    fp1 = 0.95*fi
    fp2 = 1.05*fi
    wp = [fp1, fp2]

    N, wn = signal.ellipord(wp, ws, 0.5, 50,fs=rate)
    sos = signal.ellip(N, 1, 100, wn, 'bandpass', output='sos',fs=rate)

    filtered = signal.sosfilt(sos, data)
```

```
51      filtered = np.array(filtered)
52
53      q_u = np.zeros(filtered.shape)
54
55      u_x = [0,]
56      u_y = [filtered[0],]
57
58      for k in range(1, len(filtered)-1):
59          if (sign(filtered[k]-filtered[k-1])==1) and (sign(filtered[k]-
    filtered[k+1])==1):
60              u_x.append(k)
61              u_y.append(filtered[k])
62      u_p = interpolate.interp1d(u_x, u_y, kind='cubic', bounds_error=
    False, fill_value=0.0)
63
64      envelope = []
65      for k in range(0, len(filtered)):
66          envelope.append(u_p(k))
67
68      del envelope[1::2]
69      octave_envelopes.append([envelope])
70      t = np.arange(len(envelope)) / rate
71      plt.plot(t[:rate],envelope[:rate], label='Harmonico: ' + str(i))
72
73  plt.xlabel('Tiempo [seg]')
74  plt.ylabel('Amplitud')
75  plt.grid('minor')
76  plt.legend()
77  plt.show()
78
79  with xlsxwriter.Workbook('OCT' + str(oct) + '.xlsx') as workbook:
80      worksheet = workbook.add_worksheet()
81      for col, data in enumerate(octave_envelopes):
82          dat = data[0]
83          max_data = max(dat)
84          dat = [i for i in dat]
85          worksheet.write_column(0, col, dat)
```

3.3. Parámetros

Los parámetros que se le presentaron al usuario de la GUI para modificar el modelo de la síntesis en cuestión fueron los coeficientes de cada uno de los 12 harmónicos que componen a la señal, es decir se le dio la posibilidad de modificar las amplitudes de sus componentes espectrales, para darle al instrumento distintos timbres.

3.4. Resultados

A continuación se muestra el resultado obtenido mediante el método mencionado para la síntesis de una nota de piano acústico.
Se observó que las síntesis correspondientes a los instrumentos de viento resultan de una calidad mas fiel a la síntesis del piano.

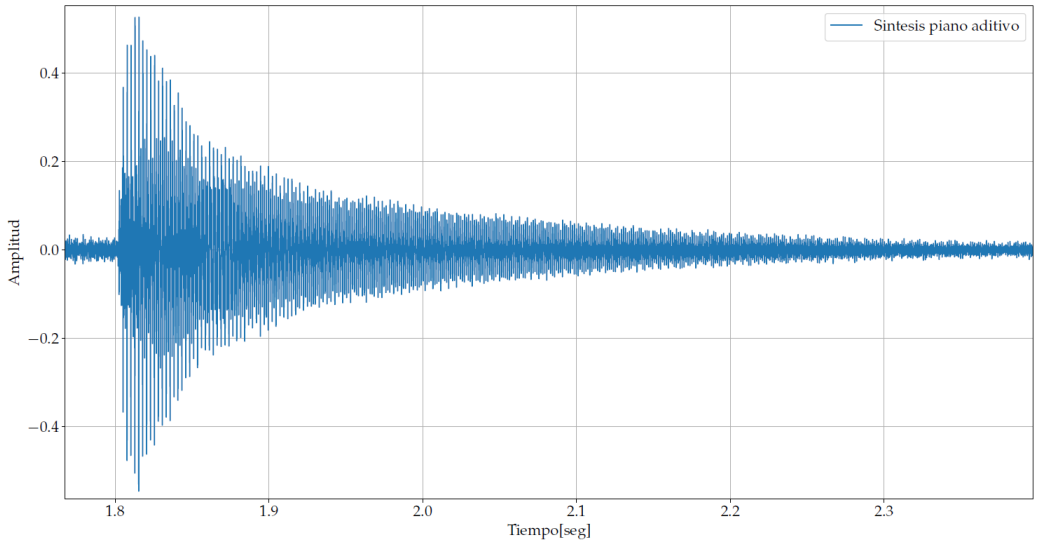


FIGURA 3.2: Forma de onda síntesis aditiva de piano acústico

4. Síntesis de sonidos mediante modelos físicos

4.1. Introducción

[2]Existen muchas técnicas de síntesis musical, las cuales incluyen síntesis de frecuencia modulada (FM), waveshaping, síntesis aditiva y substractiva pero, para llegar a sonidos de fidelidad requieren rapidez aritmética que se puede encontrar en computadoras de gran procesamiento o sintetizadores digitales, inaccesibles para la mayoría. Este tipo de síntesis permite su aplicación de forma simple y en la mayoría de las computadoras del mercado. No es solo un algoritmo de simple aplicación en software, sino también en hardware. En comparativo con la síntesis aditiva, el sonido de la cuerda pulsada es mucho más natural y fluído dado que en experimentos se precisaron 30 generador de onda senoidal para producir un timbre similar(Sleator 1981).

5. Síntesis mediante forma de onda

Este método es una de las bases para explicar la síntesis mediante modelos físicos. Consiste en repetir un número de muestras continuamente para generar una señal periódica pura.

$$y\left(n\right) = y\left(n - p\right) \tag{5.1}$$

El timbre resultante lo determina las condiciones iniciales de la relación de recurrencia. En una tabla se carga con una frecuencia de muestreo específica (f_s), una forma de onda (senoidal, triangular, cuadrada, etc.), que va a sintetizar un tono cuya frecuencia está determinada por la frecuencia de sampleo y la periodicidad (p) del sistema.

$$f_t = \frac{f_s}{p} \tag{5.2}$$

Esta técnica es muy simple pero vaga musicalmente porque los tonos son puramente periódicos y, en la realidad los instrumentos producen sonidos que varían temporalmente. Sin modificaciones en la tabla, el contenido armónico del tono es fijo, sin embargo para no perder la frecuencia del tono especificado, los cambios período a período de las muestras deberían ser pequeños. Para mantener el bajo nivel de complejidad en el procesamiento podemos modificar solo la muestra que estamos leyendo

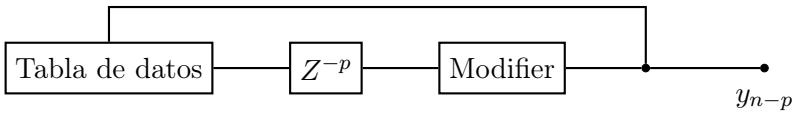


FIGURA 5.1: Diseño genérico de la síntesis mediante tabla de forma de onda

5.1. Algoritmo de la cuerda pulsada

La modificación propuesta por Alex Strong involucraba que el modificador fuera un promediador de dos muestras sucesivas.

$$y_n = \frac{y_{n-p} + y_{n-p-1}}{2} \tag{5.3}$$

El efecto que provoca esta modificación involucra un decaimiento suave de la forma de onda a una tasa que difiere según los diferentes harmónicos. Como resultado obtenemos un sonido muy natural, que independientemente del espectro original, decaerá en una forma senoidal casi pura hasta una constante que terminará en silencio. Desde el punto de vista del hardware, las operaciones necesarias para lograr este método son de baja complejidad y solo se centran en una suma y un corrimiento binario.

$$T_s = p + \frac{1}{2} \rightarrow f_t = \frac{f_s}{p + \frac{1}{2}} \tag{5.4}$$

Dada la relación de recurrencia debemos precargar la tabla con valores iniciales. Es recomendable utilizar valores aleatorios al comienzo de cada nota porque poseen la mayor parte de su contenido espectral en las altas frecuencias acústicas. Mientras las muestras se repitan periódicamente, la aleatoriedad no producirá silbidos ni chasquidos. La sutil variación espectral es suficiente para que parezca que proviene del mismo instrumento pero insuficiente para que el sonido no resuene mecánicamente.

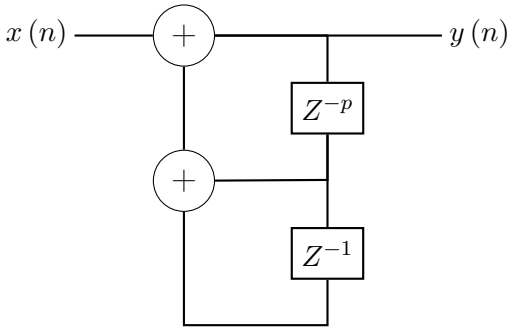


FIGURA 5.2: Modelo Karplus Strong para cuerda pulsada

Sin el algoritmo de decaimiento, una tabla de valores aleatorios tiene los harmónicos

iguales hasta la frecuencia de nyquist y suena como un órgano eléctrico (reed organ).

6. Comportamiento aleatorio

Para generar las condiciones iniciales utilizaremos muestras aleatorias de doble nivel.

$$y_t = \begin{cases} +A, prob = 0.5 \\ -A, prob = 0.5 \end{cases} - p \leq t \leq 0 \tag{6.1}$$

El valor cuadrático medio de la amplitud de la salida (A) corresponde con la mitad de la amplitud pico a pico de la señal. La utilización de este método aleatorio en reemplazo de una distribución uniforme puede originar una señal $5[db]$ más fuerte. Existen limitaciones en el valor de la periodicidad (p). Si es pequeño, la variación entre las diferentes condiciones iniciales serán relativamente largas resultando en un vago control de la amplitud. Dado que la amplitud o altura de una nota está determinado por el valor de la variable, además de que la misma debe tener un valor entero, no todas las frecuencias están disponibles. Para obtener mejor resolución en frecuencia también podemos modificar la frecuencia de sampleo. Si la variable cambia con frecuencia entre notas se producirá un efecto de glissando y vibrato. También hay que destacar que para la generación del ruido aleatorio, es recomendable utilizar una distribución cuyo espectro sea lo más extendido posible para que el efecto del filtro sintetizador se pueda apreciar en todo el rango de las mismas. [5]

6.1. Transferencia analítica del modelo original

Tenemos que separar el proceso en dos partes. Sobre la primera, tenemos un conjunto de p muestras de ruido aleatorio $x\,[-p, -1]$ y, empezamos a obtener las primeras n salidas del sistema.

$$y\,(n) = R_L \cdot \frac{x\,(n-p) + x\,(n-p-1)}{2}; \tag{6.2}$$

La segunda parte involucra el proceso recursivo donde solo interviene la salida y valores retrasados de la misma.

$$y\,(n) = R_L \cdot \frac{y\,(n-p) + y\,(n-p-1)}{2}; \tag{6.3}$$

[5] *Amid Fish - Karplus-Strong String Synthesis*, http://amid.fish/karplus-strong.

El sistema total queda representado de la siguiente manera.

$$H(z) = \frac{z^{-p} + z^{-(p+1)}}{\frac{2}{R_L} z^{p+1} - z - 1} \tag{6.4}$$

La respuesta en frecuencia del modelo analíticamente la obtenemos teniendo en cuenta la periodicidad en función de la frecuencia de la nota sintetizada.

Polos de la función

$$\frac{2}{R_L} z^{p+1} - z - 1 = 0 \rightarrow \frac{2}{R_L} z^{p+\frac{1}{2}} = z^{\frac{1}{2}} - z^{-\frac{1}{2}}$$

$$\frac{2}{R_L} a^{p+\frac{1}{2}} e^{i\omega\left(p+\frac{1}{2}\right)} = a^{\frac{1}{2}} e^{i\frac{\omega}{2}} + a^{-\frac{1}{2}} e^{-i\frac{\omega}{2}} = \sqrt{a + a^{-1} + 2cos(\omega)} \cdot e^{i\theta}$$

Mediante parte imaginaria buscamos la relación de θ con ω

$$\left.\begin{array}{c} \sqrt{a + a^{-1} + 2cos(\omega)} \cdot sen(\theta) = \left(a^{\frac{1}{2}} - a^{-\frac{1}{2}}\right) \cdot sen\left(\frac{\omega}{2}\right) \\ a = 1 - \epsilon \approx 1 \rightarrow a + a^{-1} = 2 + \epsilon^2 + \epsilon^3 ... \approx 2 \\ a^{\frac{1}{2}} - a^{-\frac{1}{2}} \approx 0 \end{array}\right\} sen(\theta) \approx 0 \rightarrow e^{j\omega\left(p+\frac{1}{2}\right)} = e^{j\theta} \approx 1$$

Frecuencia angular normalizada de cada parcial - polos $\rightarrow \omega = \frac{\theta + 2\pi n}{p + \frac{1}{2}} \approx \frac{2\pi n}{p + \frac{1}{2}}$

Módulo de los polos correspondientes $\rightarrow a \approx \left(R_L \cdot cos\left(\frac{\omega}{2}\right)\right)^{\frac{1}{p+\frac{1}{2}}} = \left(R_L \cdot cos\left(\frac{2\pi n}{2p+1}\right)\right)^{\frac{1}{p+\frac{1}{2}}}$

Condición de estabilidad para cada polo según $R_L \rightarrow R_L \leq \frac{1}{cos\left(\frac{2\pi n}{2p+1}\right)}$

6.2. Diagrama de polos y ceros

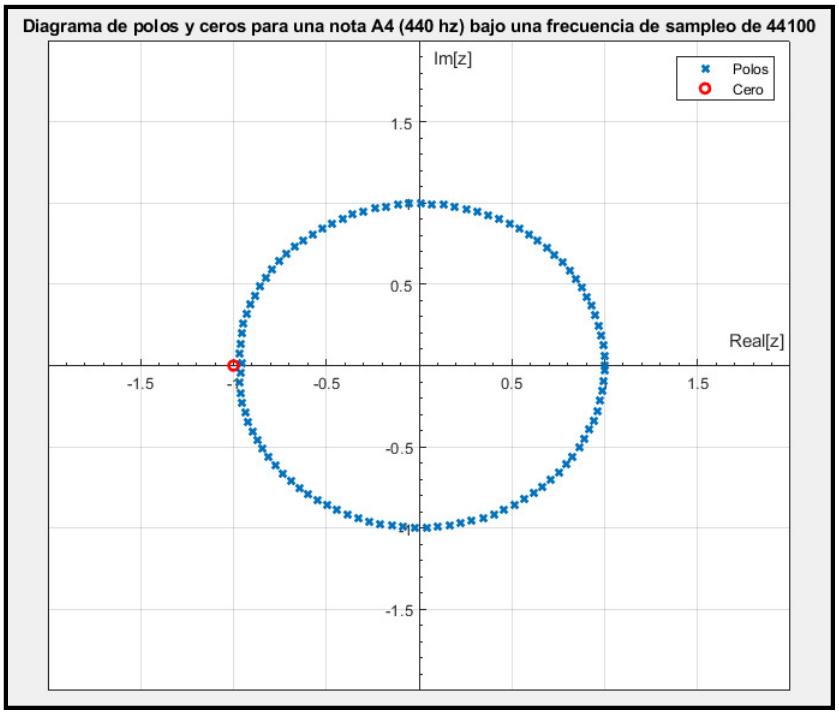


FIGURA 6.1: Diagrama de polos y ceros.

7. Efectos Digitales

A continuación se procederá a explicar y desarrollar los efectos digitales aplicados en el programa implementado y a comparar con el sonido que se observa en la figura 7.1. Además de esto, junto con el informe, se adjuntan los archivos de sonidos originales y los efectos aplicados para que se pueda apreciar mejor el efecto.

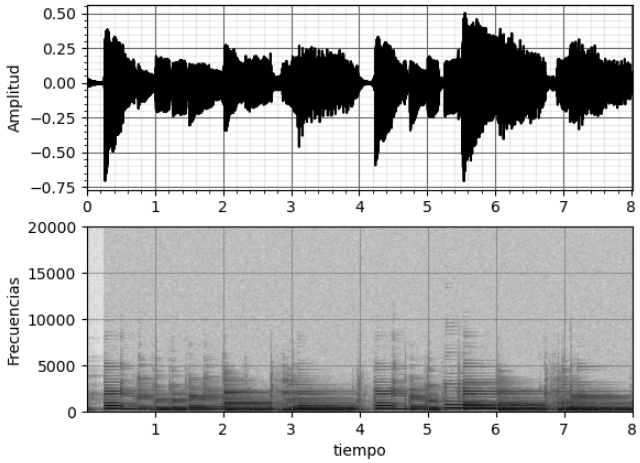


FIGURA 7.1: Sonido al cual se le aplicarán los efectos digitales

7.1. Efectos basados en Delay

Las lineas de *Delay*, son unos de los bloques mas importantes que construyen varios de los efectos mas destacados. En general, son fáciles de implementar, y pequeños cambios en sus parámetros reproducen diferentes efectos que se verán a continuación.

7.1.1. Delay

Delay Básico El *Delay* básico, reproduce una señal de audio después de un **tiempo especificado**. Este tiempo puede variar arbitrariamente desde milisegundos hasta varios segundos. En la figura 7.2 se puede observar un diagrama básico de un bloque de delay.

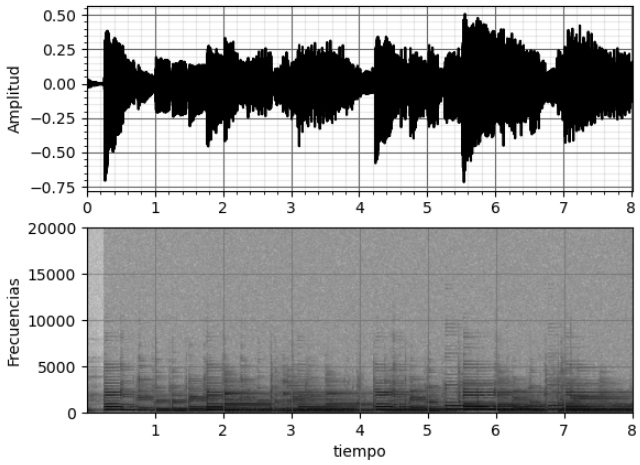


FIGURA 7.3: Delay básico aplicado al archivo de sonido

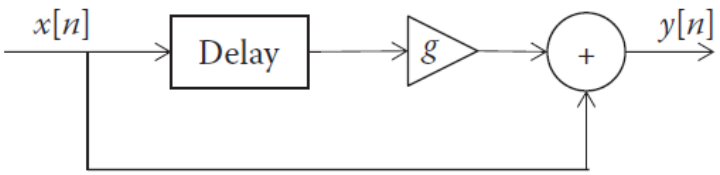


FIGURA 7.2: Diagrama del *Delay* básico

Este bloque de delay, se puede expresar con la ecuación 7.1, que se observa a continuación.

$$y(n) = x(n) + gx\,(n - N) \tag{7.1}$$

Es importante recalcar que el bloque de delay es **Linear Tiempo Invariante (LTI)**[6]. Por otra parte, puede demostrarse fácilmente que la transformada Z del bloque de delay queda determinada por la ecuación 7.2.

$$H(z) = 1 + gz^{-N} = \frac{z^N + g}{z^N} \tag{7.2}$$

Resultados En la figura 7.3 se puede ver como queda afectada la forma de onda y el espectrograma de la señal al aplicarle el efecto.

Delay con Feedback Si bien el delay básico es muy fácil de implementar, presenta algunos problemas. El principal, siendo que solo produce un solo echo a la señal de

[6]La demostración se puede observar en las ecuaciones 2.2 y 2.3 de la pagina 22 del Libro [5]

salida. Por otro lado, el feedback puede producir que el sonido a la salida tenga un eco continuo, lo cual es lo buscado la mayoría de las veces. Un diagrama en bloques del delay con feedback puede verse en la figura 7.4.

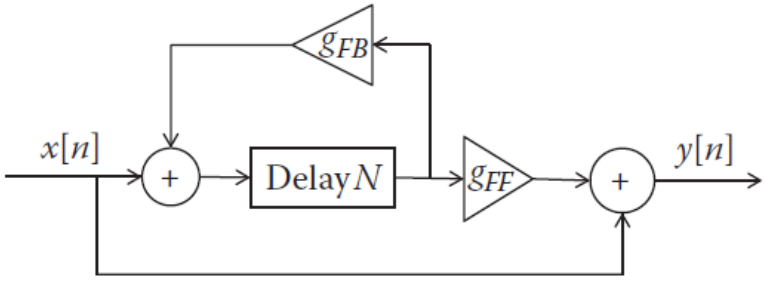


FIGURA 7.4: Delay implementado con realimentación

Fácilmente se puede llegar a la ecuación 7.3 del delay con realimentación[7].

$$y(n) = g_{FB}y(n-N) + x(n) + (g_{FF} - g_{FB})x(n-N) \tag{7.3}$$

Finalmente, aplicando la Transformada Z a la expresión anterior, se llega trivialmente a;

$$H(z) = \frac{1 + z^{-N}(g_{FF} - g_{FB})}{1 - z^{-N}g_{FB}} \tag{7.4}$$

7.1.2. Vibrato

El vibrato está definido como una variación cuasi-periódica en el tono de una nota. El vibrato esta caracterizado por su frecuencia (cuan seguído cambia el tono de una nota) y su ancho (La variación total del ton de la nota).

Teoría La implementación del vibrato está basada en un bloque de delay modulado por un oscilador de baja frecuencia (LFO por sus siglas en ingles). Un diagrama básico del vibrato puede observarse en la figura 7.5.

[7]El desarollo teórico se puede ver en las ecuaciónes 2.5 y 2.6 del libro [5]

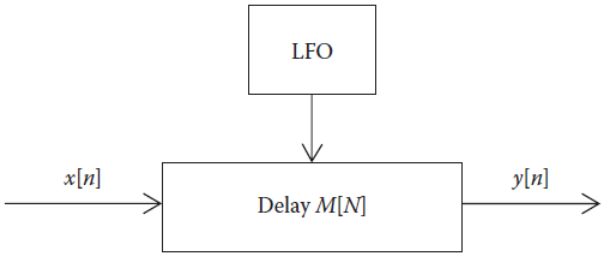


FIGURA 7.5: Caption

La expresión del vibrato queda dada por;

$$y(n) = x\left(n - M(n)\right) \tag{7.5}$$

Siendo,

$$M(n) = M_{avg} + W sin\left(2\pi n \frac{f}{f_s}\right) \tag{7.6}$$

Siendo M_{avg} el delay promedio del vibrato, W el ancho de la moudlación, f la frecuecia del LFO, y f_s la frecuencia de sampleo.

Interpolación Es importante aclarar que para que este efecto funcione es necesario tener valores de $x(n - M(n))$ para cuando $n - M(n) \notin \mathbb{Z}$. Para esto, como x no esta definida en $\mathbb{Z}$, es necesario realizar una interpolación de x. Si bien existen distintos tipos de interpolación dependiendo del caso, para los efectos de este trabajo se decidió utilizar la interpolación linear[8] que queda en la ecuación 7.7.

$$x(t) = (n + 1 - t)x(n) + (t - n)x(n + 1); n < t < n + 1 \tag{7.7}$$

Resultados En la figura 7.6 se puede ver como queda afectada la forma de onda y el espectrograma de la señal al aplicarle el efecto.Se puede observar claramente como las frecuencias no permanecen constantes, sino que siguen la forma de una senoidal muy pequeña en frecuencia.

7.1.3. Flanger

Flanger Básico El efecto del flanger está basado en el principio de interferencia constructiva y destructiva. Un flanger básico esta ampliamente relacionado con el delay

[8]Para mas información sobre distintos tipos de interpolaciones se puede leer la pagina 26 del libro [5]

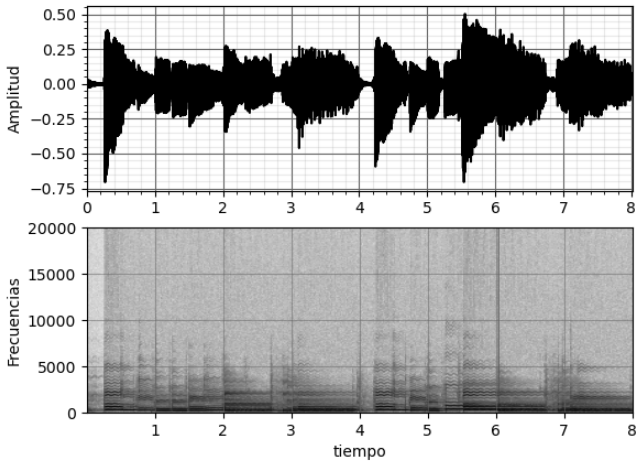


FIGURA 7.6: Vibrato aplicado al sonido original

básico y se puede ver en la ecuación 7.8.

$$y(n) = x(n) + gx(n - N) \tag{7.8}$$

El delay $M(n)$ varía respecto de un LFO de una forma similar (pero no idéntica) al del vibrato. Aplicando la Transformada Z a la ecuación del flanger básico se puede llegar a:

$$H(z) = 1 + gz^{-M(n)} \tag{7.9}$$

De esta manera, su respuesta en frecuencia quedara definida por,

$$H(f) = 1 + ge^{-i2\pi fM(n)} \tag{7.10}$$

siendo $t = nT$.

Flanger con Realimentación De la misma manera de como se explico el delay con realimentacion, el flanger obtiene su propia forma en realimentación ampliamente relacionado con su version en delay. Una figura del flanger con realimentación se observa en 7.7.

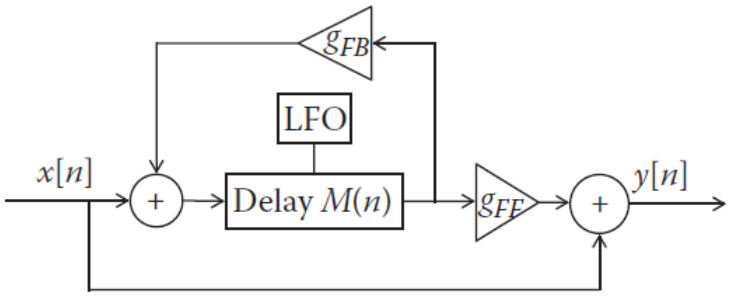


FIGURA 7.7: Flanger con realimentación

De la misma manera que con el delay, su expresión queda dada por,

$$y(n) = g_{FB}y\left(n - M(n)\right) + x(n) + (g_{FF} - g_{FB})x\left(n - M(n)\right) \tag{7.11}$$

Siendo su Transformada Z,

$$H(z) = \frac{1 + z^{-M(n)}(g_{FF} - g_{FB})}{1 - z^{-M(n)}g_{FB}} \tag{7.12}$$

Finalmente, su respuesta en frecuencia quedará dada por,

$$H(f) = \frac{1 + e^{-i2\pi fM(n)}\left(g_{ff} - g_{fb}\right)}{1 - e^{-i2\pi fM(n)}g_{fb}} \tag{7.13}$$

siendo $t = nT$.

LFO Es importante aclarar que como $M(n)$ debe variar en el tiempo, esta puede variar de diferentes maneras (senoidal, triangular, diente de sierra, etc). Sin embargo la variación mas utilizada es la del seno. La expresión de $M(n)$ queda expresada como,

$$M(n) = M_0 + \frac{M_W}{2}\left(1 + sin\left(2\pi n\frac{f_{LFO}}{f_s}\right)\right) \tag{7.14}$$

Interpolación De la misma manera que pasaba con el vibrato, para el flanger (básico o con realimentación), también se utiliza una interpolación de la señal de entrada. Para este caso, tambien se vuelve a utilizar la interpolación linear explicada en el vibrato.

Resultados En la figura 7.8 se puede ver como queda afectada la forma de onda y el espectrograma de la señal al aplicarle el efecto.

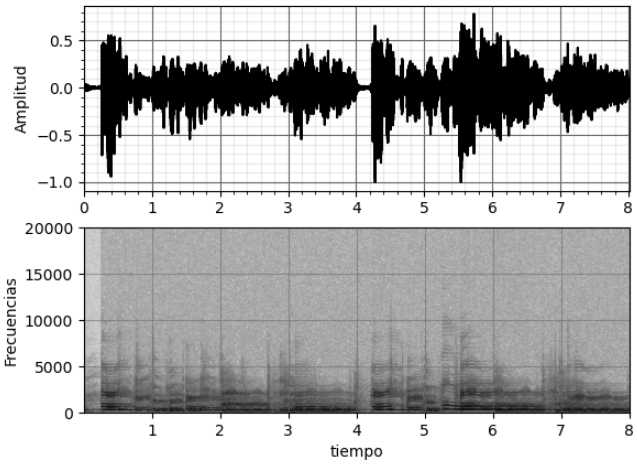


FIGURA 7.8: Flanger aplicado al sonido original

7.2. Efectos basados en filtros

Si bien en la anterior sección nos dedicamos a analizar los efectos basados en el delay, a lo largo de la siguiente sección nos centraremos en los efectos basados en filtros, ya sea, pasa-bajos, pasa-altos,pasa-banda, peaking/notch, shelving y pasa-todo.

7.2.1. Ecualización

Si bien existen muchos efectos basados en filtros, el principal y el mas utilizado es el ecualizador. El ecualizador es el efecto digital que se encarga de ïgualar"(e.g. ecualizar) las diferentes bandas de frecuencia para compensar comportamientos no ideales debido a equipos y acústicas de sala. Todos los ecualizadores utilizados hoy en día están basados en filtros, y la mayoría de los ecualizadores están compuestos de múltiples sub-filtros. Es sabido que la cantidad de bandas a ecualizar es arbitraria y elegida por la persona que diseñe el programa de ecualizador, es por esto que es necesario aclarar que a fines de mantener acotado el desarrollo de este informe, nos centraremos en desarrollar un ecualizador de 3 bandas. Estas serán los tonos bajos, los medios y los agudos, y las frecuencias centrales a cada banda se calcularán de manera similar a como se hicieron en el informe de Teoría de Circuitos (e.g. de manera logarítmica)[9].En la figura 7.9 se puede observar un diagrama en bloques de un ecualizador de 3 bandas con conexión en cascada.

Para la aplicación de un ecualizador de 3 bandas digital, se utilizaran 2 filtros tipo *shelving* y un filtro tipo *peaking/notch*. Para esto, se utilizarán las siguientes funciones

[9]Recordar que en el informe del trabajo práctico número 4 de la materia 22.01 Teoría de Circuitos se instó a implementar un ecualizador analógico.

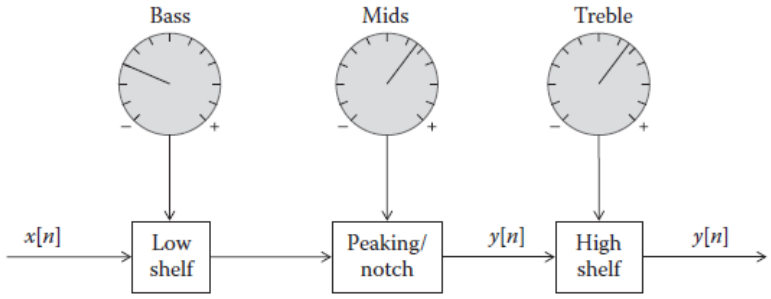


FIGURA 7.9: Diagrama en bloques de un ecualizador de 3 bandas

transferencias[10],

$$H_{LS}(z) = \frac{\left(Gtan(\omega_c/2) + \sqrt{G}\right) z + \left(Gtan(\omega_c/2) - \sqrt{G}\right)}{\left(tan(\omega_c/2) + \sqrt{G}\right) z + \left(tan(\omega_c/2) - \sqrt{G}\right)} \tag{7.15}$$

$$H_{HS}(z) = \frac{\left(\sqrt{G}tan(\omega_c/2) + G\right) z + \left(\sqrt{G}tan(\omega_c/2) - G\right)}{\left(\sqrt{G}tan(\omega_c/2) + 1\right) z + \left(\sqrt{G}tan(\omega_c/2) - 1\right)} \tag{7.16}$$

$$H_{PN}(z) = \frac{\left(\sqrt{G} + Gtan(B/2)\right) z^2 - \left(2\sqrt{G}cos(\omega_c)\right) z + \left(\sqrt{G} - Gtan(B/2)\right)}{\left(\sqrt{G} + tan(B/2)\right) z^2 - \left(2\sqrt{G}cos(\omega_c)\right) z + \left(\sqrt{G} - tan(B/2)\right)} \tag{7.17}$$

Parámetros Como se puede observar en las ecuaciones[11] 7.15, 7.16 y 7.17, las formulas son demasiado voluminosas y para nada sencillas. Es por esto, que procederemos a fijar unos cuantos parámetros con el fin de obtener una transferencia mas sencilla de implementar y mas entendible también. Los primeros parámetros a fijar serán las frecuencias centrales de cada filtro, ya que nuestro ecualizador contará con bandas fijas de ecualización. Es por esto que arbitrariamente eligiendo $\omega_{c_{LS}} = 200(Hz)$, las demás frecuencias quedarán determinadas por la escala logarítmica. Por ello, las frecuencias centrales a utilizar serán[12],

$$\omega_{c_{LS}} = 300(Hz) \tag{7.18}$$

[10]Es necesario destacar, que si bien por fines prácticos no se procederá a hacer el análisis de la obtención matemática de estas transferencias, estas están completamente detalladas en el capitulo 3 del libro [5].

[11]Es necesario remarcar, que esas ecuaciones están expresadas para frecuencias normalizadas.

[12]Estos valores fueron elegidos también, de tal manera que se encuentren dentro del rango especificado en la pagina 92 del libro [5].

$$\omega_{c_{PN}} = 700(Hz) \tag{7.19}$$

$$\omega_{c_{HS}} = 1.5(kHz) \tag{7.20}$$

Los siguientes parámetros a fijar serán los respectivos anchos de banda de cada filtro. Para esto se eligió los siguientes anchos de banda,

$$B = 400(Hz) \tag{7.21}$$

Finalmente, es sabido[13] que los controles de tono típicos tienen una variación de ganancia de $\pm 12(dB)$, es por esto que el valor de G, queda necesariamente confinado al intervalo,

$$0.25 < G < 4 \tag{7.22}$$

Interconexión de etapas Si bien existen 2 maneras de interconectar los filtros, para este caso en particular se procederá a conectar en cascada los filtros. Esto se debe a que en general, este tipo de filtros de 3 bandas se suelen conectar en cascada porque dan una mejor salida[14].

Eficiencia Como se vio a lo largo de todo el informe, la eficiencia de algoritmos es algo crucial en el desarrollo de este tipo de sistemas, y es por eso, que para este tipo de efectos, poner especial atención en la eficiencia no sea la excepción. Es por esto que se decidió no calcular los coeficientes de los filtros en cada pasada, sino calcularlos solo cuando los valores cambian[15].

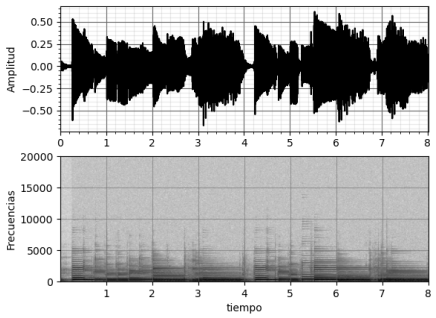
Implementación Para implementar este tipo de filtros, no alcanza con tener la transferencia en función de z, es por eso que se anti-transforma en z y se obtienen las ecuaciones de recurrencia de los filtros. Estas ecuaciones se detallan a continuación, y serán estas las que se implementarán en el sistema de efectos del programa. Para los casos de H_{LS} y H_{HS} donde,

$$H(z) = \frac{Az + B}{Cz + D} \tag{7.23}$$

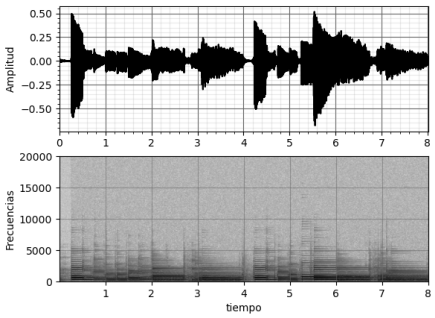
[13]Esto también se aclara en la página 92 del libro [5].

[14]Esta información se obtuvo de la página 100 del libro [5]

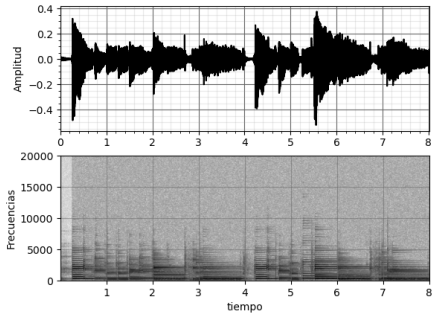
[15]Esto es especialmente necesario ya que de lo contrario, hacer los cálculos trigonométricos, y divisiones de punto flotante serían especialmente costoso a nivel de procesamiento



(A) EQ Bass



(B) EQ Mid



(C) EQ High

FIGURA 7.10: Ecualizador funcionando en 7.10a solo bajos, 7.10b solo medios, y 7.10c solo agudos

Obtenemos,

$$y(n) = \frac{Ax(n) + Bx(n-1) - Dy(n-1)}{C} \tag{7.24}$$

Por otra parte, para H_{PN} donde,

$$H(z) = \frac{Az^2 + Bz + C}{Dz^2 + Bz + E} \tag{7.25}$$

tenemos,

$$y(n) = \frac{Ax(n) + Bx(n-1) + Cx(n-2) - By(n-1) - Ey(n-2)}{D} \tag{7.26}$$

Resultados Los resultados de aplicar el ecualizador en diferentes formas se pueden observar en la figura 7.10.

7.2.2. Ecualizador de 8 Bandas

Para implementar un ecualizador de 8 bandas[16], se utilizó toda la teoría explicada previamente, con la salvedad de que en esta ocasión, todos los filtros eran del tipo *Peaking-Notch*, y se correspondían cada 1 con la transferencia descripta en la ecuación 7.17. De la misma manera, la ecuación en diferencias queda determinada por la expresión 7.26.

Determinación de frecuencias Para la determinación de las frecuencias centrales de cada filtro, se utilizo una distancia logarítmica entre ellas, siguiendo la expresión 7.27.

$$f_k = 10^{\left(2+\frac{log(20000)-log(100)}{8}k\right)} \tag{7.27}$$

Determinación de los *Bandwidth* Para la determinación de los anchos de banda de cada filtro se tomo de manera arbitraria la expresión 7.28.

$$B_k = \frac{f_k}{3} \tag{7.28}$$

Determinación del rango de ganancias Finalmente, para la determinación del rango de ganancias se tomó, de la misma manera que para los demás efectos, un intervalo de $\pm 12(dB)$, dando un intervalo de ganancias de $0.25 < G_k < 4$.

7.2.3. Wah-Wah

Por suerte para el lector, el efecto de Wah-Wah lleva 1 solo filtro digital, y por esto, las cuentas matemáticas a realizar son menores. Sin justificación teórica en este informe[17], se procederá a dar a continuación, la ecuación de transferencia de un Wah-Wah[18].

$$H(z) = \frac{b_0 z^2 + b_2}{z^2 + a_1 z + a_2} \tag{7.29}$$

Donde, $b_0 = \frac{K}{K^2Q+K+Q}$, $b_2 = -b_0$, $a_1 = \frac{2Q(K^2-1)}{K^2Q+K+Q}$ y $a_2 = \frac{K^2Q-K+Q}{K^2Q+K+Q}$. Siendo $K = tan\left(\pi\frac{f_c}{f_S}\right)$.

[16]Se Opto por implementar un ecualizador de 8 bandas para obtener más control en el programa principal, sin embargo, para el programa que aplica efectos a .wav solo se utilizo el ecualizador de 3 bandas(bajos, medios y agudos).

[17]Aunque mas detalles de porque esto es así se puede leer en las paginas 49, 50 y 67 del libro [8]

[18]Notesé que es la misma función aparente que la de un pasa-banda

Por lo tanto, su expresión en diferencias quedará determinada por,

$$y(n) = b_0x(n) + b_2x(n-2) - a_1y(n-1) - a_2y(n-2) \tag{7.30}$$

Low Frequency Oscillator (LFO) Como se ve en la ecuación 7.29, el Wah-Wah utiliza una ecuacion de un pasa-bajos, sin embargo, los especial de un efecto wah-wah es que a diferencia de un filtro pasa-bajos resonante común y corriente, este filtro cambia dependiendo de la frecuencia de corte ω_c. Para los efectos de este informe, nos centraremos en el denominado 'Auto-Wah', que genera el efecto del Wah-Wah sin necesidad de un pedal para cambiar la frecuencia. Para esto, se implementa un LFO de tal manera que la frecuencia quede determinada de la siguiente manera,

$$f_c(n) = f_{min} + \frac{W}{2}\left(1 + cos\left(2\pi n\frac{f_{LFO}}{f_s}\right)\right) \tag{7.31}$$

Los valores que tomarán estarán confinados a, $0.2(Hz) < f_{LFO} < 5(Hz)$ y $f_{min} > 250(Hz)$.En la figura 7.11 puede observarse un diagrama en bloques de un Wah-Wah[19].

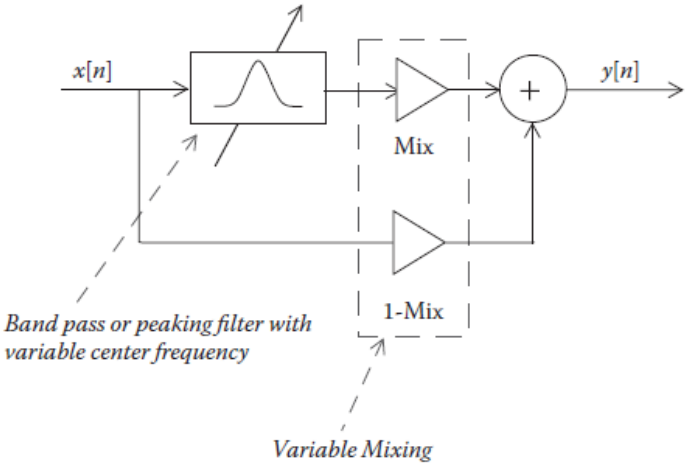


FIGURA 7.11: Diagrama en bloques de un Wah-Wah

Resultados En la figura 7.8 se puede ver como queda afectada la forma de onda y el espectrograma de la señal al aplicarle el efecto.

[19]Mas información sobre el efecto Wah-Wah puede obtenerse en la pagina 105 del libro [5].

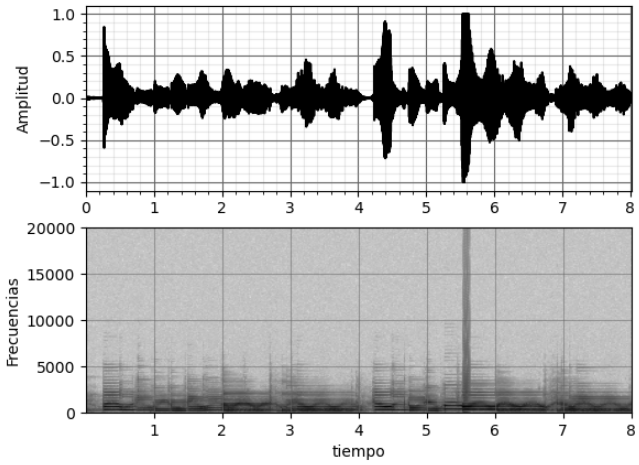


FIGURA 7.12: Efecto de Wah-Wah Aplicado a una guitarra

7.3. Efectos de reverberación

7.3.1. Reverberador de Schroeder

Para el efecto de reverberación se implemento el tipo de reverberación plana con la transferencia dada por,

$$H(z) = \frac{1}{1 - gz^{-M}} \tag{7.32}$$

Se puede observar que su ecuación en diferencias esta dada por,

$$y(n) = x(n) + gy(n - M) \tag{7.33}$$

Resultados En la figura 7.13 se puede ver como queda afectada la forma de onda y el espectrograma de la señal al aplicarle el efecto.

7.3.2. Reverberador pasa-bajos

De manera similar a como se implemento el reverberador de Schroeder, se aplico un filtro pasa-bajos simple de la forma,

$$y(n) = \frac{x(n) - x(n - 1)}{2} \tag{7.34}$$

Por lo tanto, la ecuación en diferencias del reverberador de Schroeder con el filtro

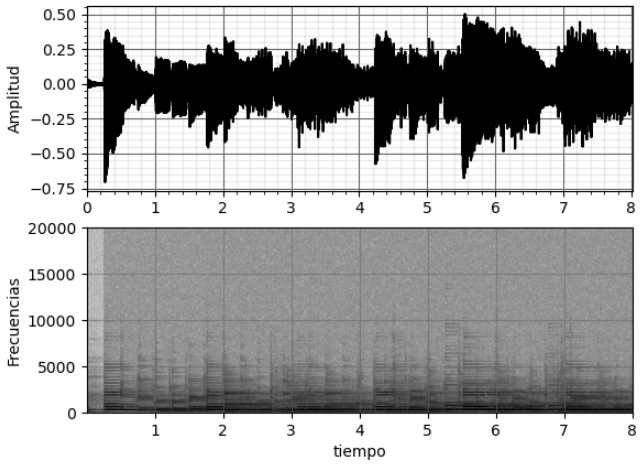


FIGURA 7.13: Efecto de reverberador plano aplicado al sonido original

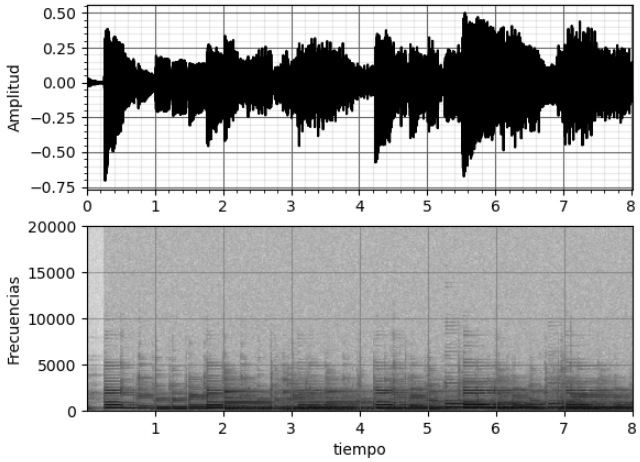


FIGURA 7.14: Efecto de reverberador pasabajos aplicado al sonido original

pasa-bajos queda de la forma,

$$y(n) = x(n) + \frac{1}{2} g \left(y(n - M) + y(n - M - 1) \right) \tag{7.35}$$

Resultados En la figura 7.14 se puede ver como queda afectada la forma de onda y el espectrograma de la señal al aplicarle el efecto.

7.4. Efectos basados en Modulación de amplitd

A continuación se porcederá a detallar uno de los efectos de modulación de amplitud implementado, el tremolo.

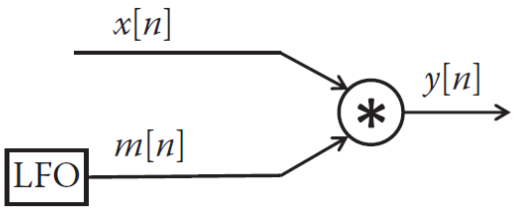


FIGURA 7.15: Diagrama en bloques del tremolo

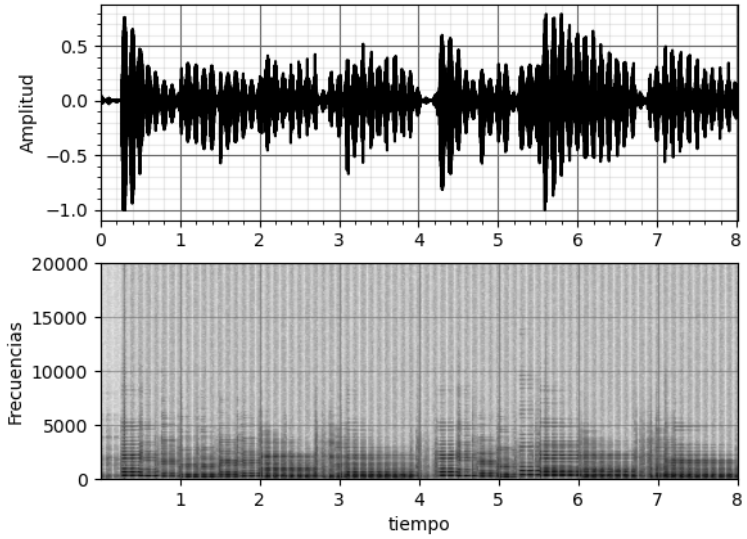


FIGURA 7.16: Efecto de tremolo aplicado al sonido original

7.4.1. Tremolo

El tremolo es el efecto que resulta de multiplicar la señal de entrada por otra señal variable en el tiempo, por lo tanto su función en el tiempo quedará dada por,

$$y(n) = m(n)x(n) \tag{7.36}$$

Siendo,

$$m(n) = 1 + \alpha cos\left(2\pi n \frac{f_{LFO}}{f_s}\right) \tag{7.37}$$

Un diagrama en bloques del sistema se puede observar en la figura 7.15.

Resultados En la figura 7.16 se puede observar como afecta al espectrograma aplicar el efecto de tremolo.

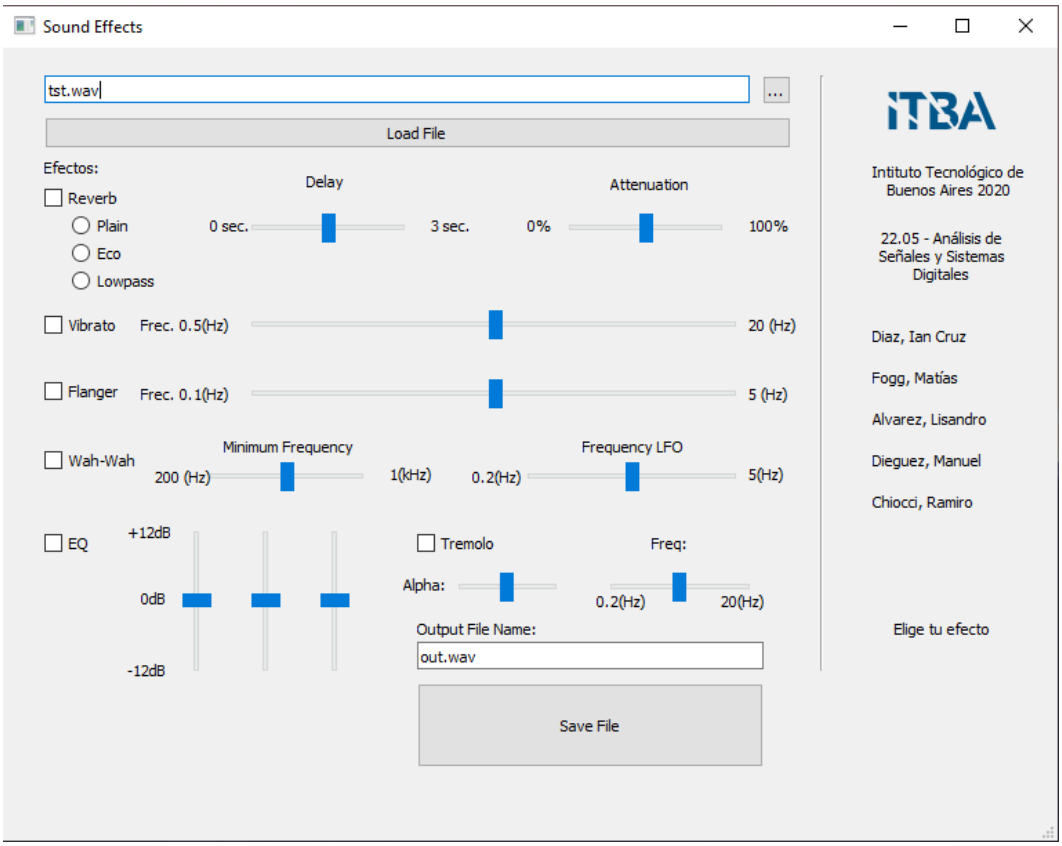


FIGURA 7.17: Software desarrollado para la aplicación de efectos de sonido digital

7.5. Implementación del programa

Para la utilización de los efectos mencionados anteriormente se implemento un programa con GUI separado del programa principal de síntesis de instrumentos, para poder así utilizar los efectos con archivos *.wav*. El diseño del programa se puede observar en la figura 7.17.
A continuación procederemos a detallas un pequeño manual de usuario de la utilización de dicho programa.

7.5.1. Manual de usuario

Cargado de archivo Para comenzar a aplicar efectos a un archivo en el programa, se debe cargar el archivo al programa. Esto se hace desde la barra superior de la ventana, insertando el path y el nombre del archivo *.wav* que se desee cargar. Si todo sale bien, un cartel diciendo *Archivo cargado con éxito* debería saltar en la esquina inferior derecha de la pantalla.

Configuración de efectos Para cargar los distintos efectos a la canción cargada en el programa, se debe tildar la cajita del efecto que se desee utilizar y mover la/s barras de configuración de efectos a gusto.

Guardado del archivo modificado Una vez cargado el archivo en el programa, y seleccionado y configurado los efectos a aplicar, se debe proceder a elegir un nombre del archivo de salida en la barra inferior derecha, y al finalizar clickear el botón *Save File*.

7.6. Conclusión

Como se pudo observar, los efectos digitales utilizados pueden generar distintos tipos de sonidos. Resulta importante aclara que, al utilizar filtros IIR en algunos de los filtros, la salida del sonido por un parlante/altavoz podría llegar a saturar/clipear el sonido. Es por esto que se debe poner especial cuidado en los coeficientes de la transferencia del filtro IIR.

8. Implementación del programa

Para el desarrolló del programa se utilizó el lenguaje C++ por su capacidad de llevar mas rápidamente a cabo tareas que otros lenguajes como por ejemplo Python. En este trabajo, nos centramos en realizar un programa capaz de sintetizar notas en tiempo real, y reproducirlas con efectos también en tiempo real. Es por esto que se requirió de un programa lo suficientemente veloz como para procesar la cantidad de datos necesaria para que el sonido no se trabe en ningún momento.

8.1. Interfaz gráfica

Para la interfaz gráfica se utilizo la librería *Open Source* Qt, por su relativo sencillo manejo y su amplio desarrollo en funciones gráficas. Algunas capturas de pantalla del programa se pueden observar en las figuras 8.1, 8.2, 8.3, 8.4 y 8.5.

8.2. Manejo de Audio

Para el manejo del audio a reproducir se utilizó la librería PortAudio, ya que contaba con funciones lo suficientemente sencillas de entender y fáciles de aplicar para poder reproducir un sonido dado. Además, con los llamados a *callbacks* que proveía esta librería, nos otorgaba un amplio abanico de posibilidades para aplicarle algoritmos a las señales que posteriormente se iban a reproducir. Es por esto, que cada síntesis, y en los efectos, se tuvieron que desarrollar callbacks necesarios para poder ser utilizados en protaudio, y siguiendo estrictas normas de programación para que los algoritmos utilizados no se tomen demasiado tiempo. Algunas de las restricciones utilizadas fueron: a) No alocar ni des-alocar memoria dentro de los callbacks, b) no tener ordenes de complejidad de algoritmos muy altos, y c) que los algoritmos no procesen información innecesaria[20].

8.3. Manejo de datos de Síntesis

Como ya fue discutido en secciones anteriores, los distintos métodos de síntesis requieren datos previos sobre instrumentos y demás parámetros, es por esto que se decidió que para nuestro programa, la carga de los datos se realice durante el comienzo del programa. Por esta razón, si bien el programa tal vez tarda algunos segundos en iniciar,

[20]Es importante aclarar, que como el tamaño del buffer que maneja portaudio es dinámico, el buffer utilizado en los callbacks también lo era, es por esto que se debía procesar hasta un tamaños especifico del buffer, y no mas allá.

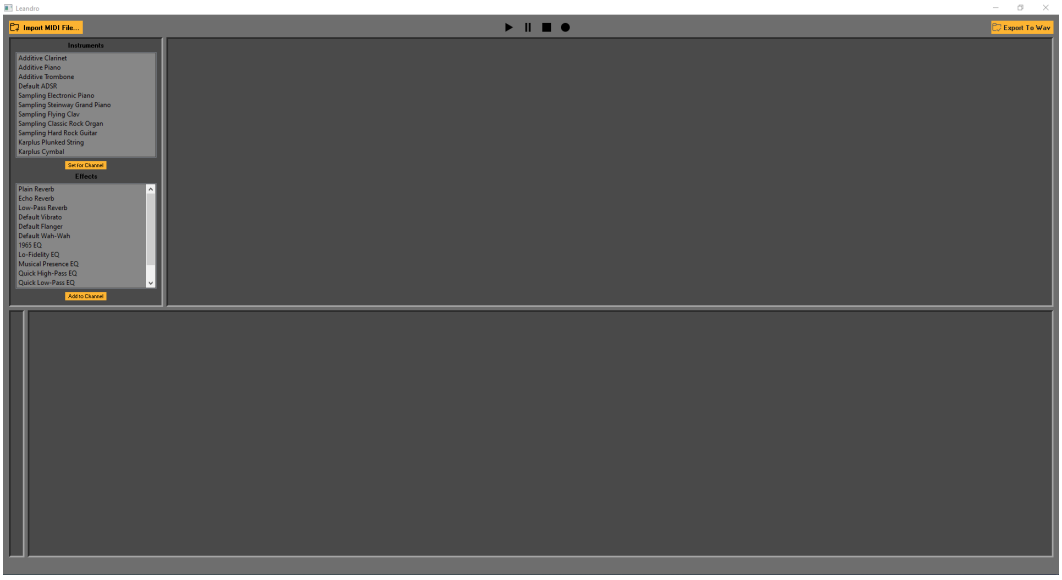


FIGURA 8.1: Inicio del programa

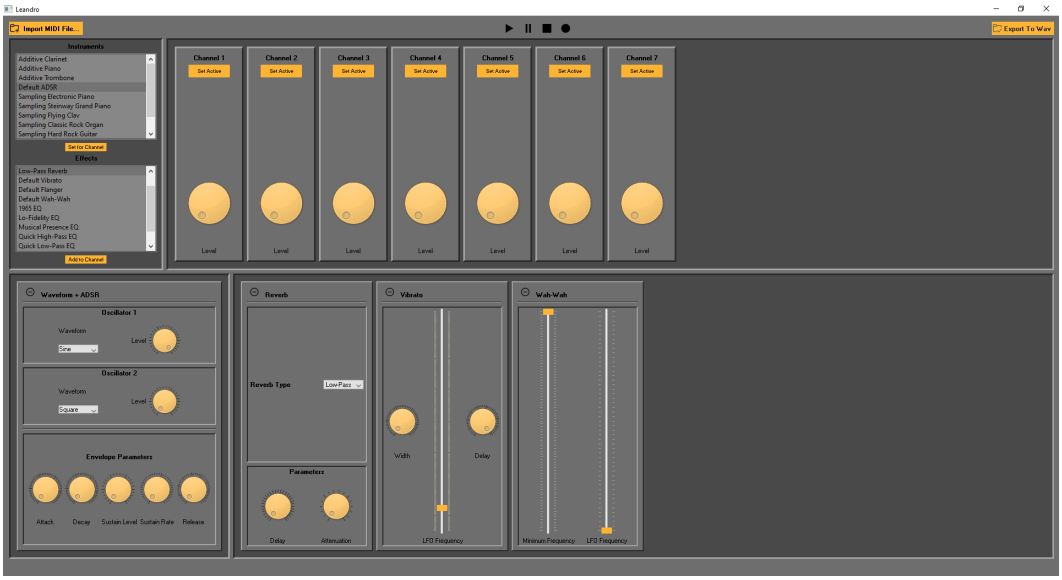


FIGURA 8.2: ADSR

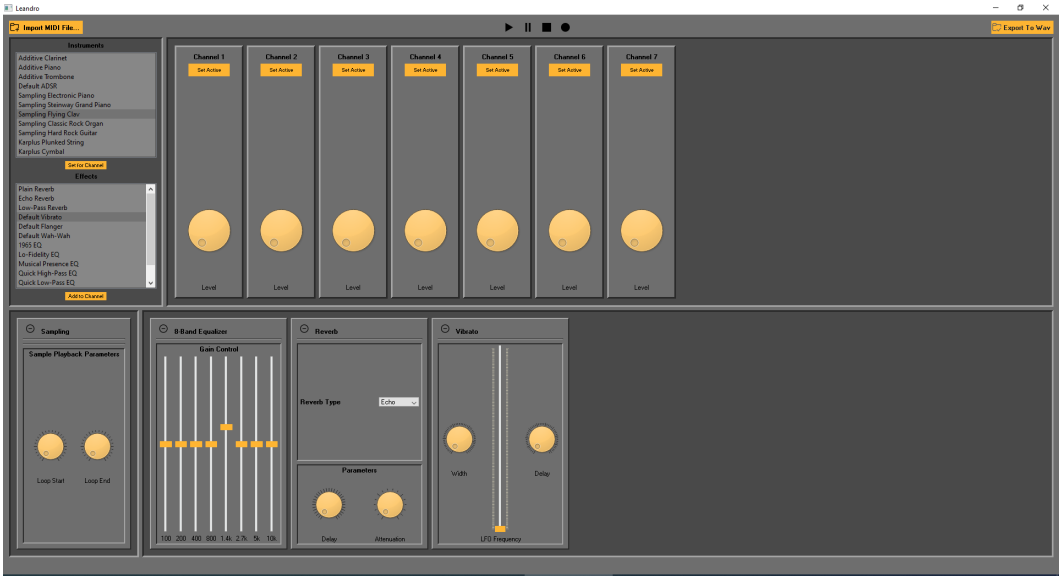


FIGURA 8.3: Sample

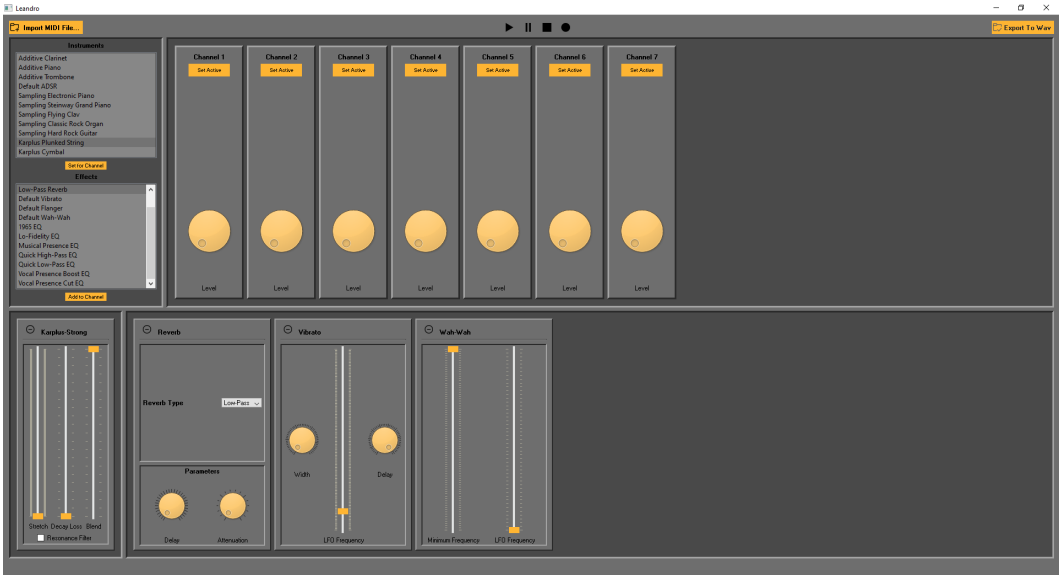


FIGURA 8.4: Karplus

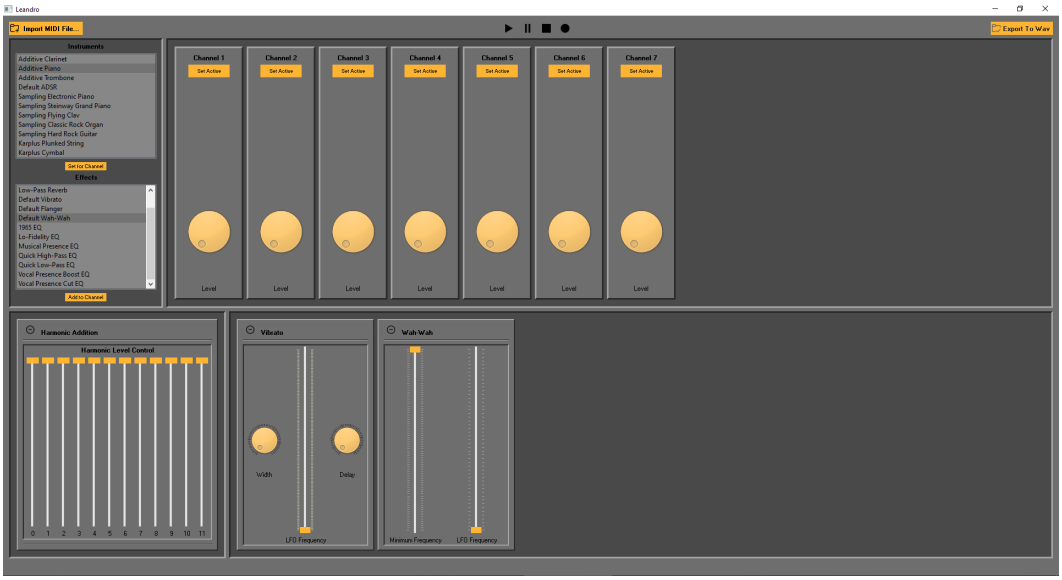


FIGURA 8.5: Additive

una vez iniciado se puede comenzar a sintetizar cualquier audio sin ninguna pantalla de carga ni demora alguna.

8.4. Presets para efectos

Para cada efecto, se le proporcionó al usuario con diferentes presets, junto con diferentes diales para ajustar los parámetros de cada efecto a gusto. De esta manera, ningún efecto es fijo, sino que se lo puede modificar, junto con los parámetros de síntesis, para generar cualquier tipo de sonido deseado, y escuchar el resultado en tiempo real.

Referencias

[1] Andreas Antoniou. *Digital Signal Processing: Signlas, Systems and Filters*. McGraw-Hill, 2006. ISBN: 0-07-145424-1.

[2] Kevin Karplus y Alex Strong. "Digital Synthesis of Plucked-String and Drum Timbres." En: *Computer Music Journal* 7.2 (1983), págs. 43-55. DOI: 10.2307/3680062.

[3] *Piano Key Frequencies*. URL: https://en.wikipedia.org/wiki/Piano_key_frequencies.

[4] John G. Proakis y Dimitris G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications Fourth Edition*. Pearson Prentice Hall, 2007. ISBN: 978-0131873742.

[5] Joshua D. Reiss y Andrew P. McPherson. *Audio Effects: Theory, Implementation and Application*. CRC Press, 2015. ISBN: 978-1-4665-6029-1.

[6] *Sample-based synthesis*. URL: https://en.wikipedia.org/wiki/Sample-based_synthesis.

[7] Simon Xu. *The FFT Algorithm - Simple Step by Step*. URL: https://www.youtube.com/watch?v=htCj9exbGo0.

[8] Udo Zölzer. *DAFX: Digital Audio Effects 2nd Edition*. Wiley, 2011. ISBN: 978-0-470-66599-2.