Análisis de Señales y Sistemas Digitales

Primer Cuatrimestre de 2025

Trabajo Práctico N2 - Grupo 1

Análisis y Síntesis de Audio digital

Integrantes

Torino, Joaquín	63140
Pla, Juan Ignacio	63486
Caviglia, Facundo	63178
Gentile, Federico	63049
Lobova, Anna	67830
Gómez, Joaquín	62116

Profesores

Jacoby, Daniel Andrés
Paura Bersan, Martín

Fecha de Entrega: 27/05/2025

Índice

1. Introducción **2**

2. Objetivos **2**

3. Marco Teórico **3**

4. Desarrollo de los Bloques del Sistema **5**
4.1. Síntesis aditiva 5
4.2. Síntesis FM (Modulación de Frecuencia) 6
4.3. Modelado físico 8
4.4. Reproducción de muestras (Sample Playback) 9
4.5. Efectos de audio 9
4.6. Espectrograma 10
4.7. Mezcla de pistas 10
4.8. Exportación a WAV 11

5. Conclusiones y Análisis **11**

1. Introducción

En el presente trabajo se desarrolla un sistema de análisis y síntesis de señales de audio digitales. El mismo incluye módulos de síntesis aditiva, modulación de frecuencia (FM), modelado físico, reproducción por muestreo (*sample playback*), efectos de audio, generación de espectrogramas, procesamiento de eventos MIDI y exportación a formato WAV. Se emplean como señales de referencia una grabación del *Concierto de Aranjuez* de Joaquín Rodrigo, y varios archivos MIDI con diversas notas con el fin de comparar características espectrales entre la señal real y las señales sintetizadas.

2. Objetivos

Los objetivos principales del trabajo son:

- Implementar un sintetizador aditivo, capaz de generar señales musicales sumando componentes sinusoidales con frecuencias y fases definidas.
- Desarrollar un módulo de síntesis FM con uno o más osciladores moduladores, variando los índices de modulación para crear distintos timbres.
- Construir un modelo físico digital (por ejemplo de cuerda vibrante o tubo resonante) que simule el comportamiento acústico de un instrumento real.
- Incorporar la reproducción de muestras de audio (*sample playback*), cargando y controlando la reproducción de señales pregrabadas.
- Añadir efectos de audio (eco, reverberación, filtros, distorsión, etc.) simulando cadenas de efectos en tiempo real.
- Calcular y visualizar espectrogramas de las señales de audio para su análisis tiempo-frecuencia.
- Procesar archivos MIDI para extraer eventos de notas y controladores, habilitando la interpretación de partituras digitales mediante el sintetizador.
- Asignar ciertos “tracks” a partir de los archivos MIDI a varios instrumentos musicales mediante los distintos sintetizadores para luego combinar todos los tracks y resultar en una obra musical.
- Exportar la señal final a un archivo WAV de alta calidad.
- Evaluar el sistema mediante resultados experimentales: capturas de pantalla del software, gráficas generadas por código (formas de onda, espectros, etc.) y comparaciones entre la señal sintetizada y la señal real del Concierto de Aranjuez.

3. Marco Teórico

Síntesis Aditiva

La síntesis de sonido digital se basa en representar señales complejas mediante elementos simples. En la **síntesis aditiva**, se combinan varias ondas sinusoidales para formar un timbre deseado. Cada componente sinusoidal tiene frecuencia f_k, amplitud A_k y fase ϕ_k. Si la señal resultante es periódica de período $T = 1/f_0$, puede expresarse por la serie de Fourier como:

$$y(t) \;=\; \sum_{k=0}^{K} A_k \,\mathrm{sen}(2\pi k f_0 t + \phi_k).$$

En otras palabras, la síntesis aditiva es una técnica de síntesis de sonido que crea timbre sumando ondas sinusoidales. Esta idea explica cómo cualquier señal periódica de ancho de banda finito se puede reconstruir sumando sus componentes armónicas. El diseño del sintetizador aditivo implica seleccionar K armónicos relevantes (frecuencias múltiplos de la fundamental: $k \cdot f_0$) y sus amplitudes y fases según el timbre objetivo.

Estudios y aplicaciones prácticas suelen reservar un número finito de armónicos para aproximar sonidos musicales, ya que gran parte de la energía de instrumentos suele concentrarse en los primeros armónicos. Además, las frecuencias del espectro audible llegan solamente hasta los 20kHz, por lo que tampoco sería práctico tomar una extensa cantidad de armónicos.

Otra cuestión importante a tener en cuenta es que las octavas de cada nota se dan a partir de la mitad o el doble de la frecuencia fundamental correspondiente a la nota que se está emitiendo. En el desarrollo de este trabajo práctico, se eligió considerar hasta 8 armónicos (subsiguientes) de la frecuencia fundamental, por lo que también se estarán valorando las octavas de la misma cada $2q \cdot f_0$ con $q \in \mathbb{N}$.

Síntesis por FM

En la **síntesis por modulación de frecuencia (FM)** se varía la frecuencia instantánea portadora mediante otro moduladora. De forma típica, la señal generada puede describirse por:

$$y(t) = A_c \cdot \mathrm{sen}\Big(2\pi f_c \cdot t + I \cdot \sin(2\pi f_m \cdot t)\Big),$$

donde f_c es la frecuencia portadora, f_m la frecuencia moduladora, I el índice de modulación, y A_c la amplitud. Este esquema puede generar sonidos armónicos e inarmónicos y, a medida que el índice de modulación aumenta, el timbre se vuelve progresivamente más complejo. Por ejemplo, cuando $f_m = nf_c$ con $n \in \mathbb{Z}$, el espectro resultante conserva una estructura más armónica; en cambio, con frecuencias no armónicas se obtienen espectros tipo campana o percusivos.

El **muestreo**, como se estudió y profundizó en anteriores trabajos prácticos, es el proceso de conversión de una señal continua en discreta (en tiempo) midiendo su valor a intervalos fijos. En la práctica, se suele usar $f_s = 44{,}1$ kHz (calidad CD) o superior, de modo que señales de audio de hasta unos 20 kHz se reproduzcan sin problema de aliasing. Para señales que no cumplen perfectamente el requisito de banda limitada, se emplean filtros antialias antes de la conversión.

Síntesis de sonidos mediante modelos físicos

El **modelado físico digital** intenta simular el comportamiento de un sistema acústico real mediante diferencias finitas o filtros digitales. Por ejemplo, una cuerda ideal satisface la ecuación de onda 1D:

$$\frac{\partial^2 y}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2},$$

donde c es la velocidad de propagación. En el dominio discreto se aproximan estas derivadas con esquemas en diferencias finitas. Una implementación clásica de cuerda pulsada es el algoritmo de Karplus-Strong, que modela la cuerda como una línea de retardo con realimentación amortiguada. De este modo, tras una excitación de ruido blanco, la señal recircula atenuada, generando un sonido de cuerda resonante.

Se muestra a continuación el diagrama en bloques de la implementación del modelo Karplus-Strong:

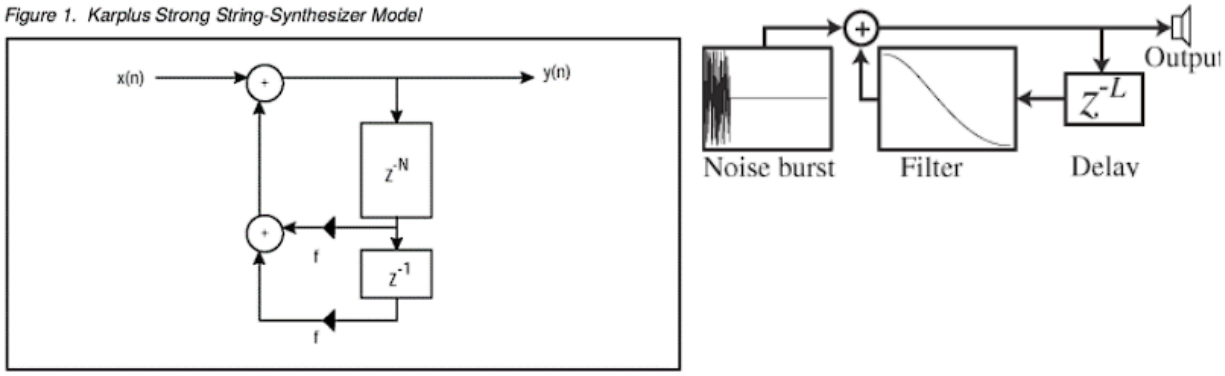

Figura 1: Modelo de Karplus-Strong

Espectrograma

En cuanto al **procesamiento de señales**, una herramienta clave es el espectrograma, que muestra cómo varía el contenido en frecuencia de una señal a lo largo del tiempo. Técnicamente, se calcula mediante la Transformada de Fourier de Tiempo Corto (STFT): la señal se divide en ventanas temporales cortas, y a cada segmento se le aplica FFT. El resultado se representa como una imagen de frecuencia vs tiempo, donde la intensidad del color del gráfico indica la amplitud de cada frecuencia.

En otras palabras, un espectrograma es una representación visual del espectro de frecuencias de una señal a medida que varía en el tiempo. Los espectrogramas son útiles, por ejemplo, para identificar las transiciones de notas, armónicos y transitorios en música.

Archivos MIDI

Con el fin de aplicarle efectos de audio o un instrumento a una serie de notas musicales, o "track", se recurre al estándar MIDI para interpretar eventos musicales digitales. Cada mensaje MIDI típicamente incluye un código de estado (por ejemplo, `Nota On`, `Nota Off`), un número de nota (pitch) y una velocidad (velocity).

En el proyecto realizado en Python, se lee un archivo `.mid` secuencialmente, y al recibir un mensaje de "Nota On" se desencadena la reproducción de la nota correspondiente en el sintetizador, y con "Nota Off" se finaliza. Se pueden manejar también mensajes de cambio de instrumento o controladores. Para la implementación se suele usar una biblioteca MIDI (por ejemplo, `mido` en Python) que abstrae estos detalles.

4. Desarrollo de los Bloques del Sistema

A continuación se detalla la implementación de cada bloque del sistema. En cada caso se justifica el enfoque elegido e incluye ecuaciones relevantes.

4.1. Síntesis aditiva

El sintetizador aditivo se implementa sumando una serie de senos discretos en el tiempo. En primer lugar, se eligió la frecuencia fundamental f_0 correspondiente a la nota deseada, y un conjunto de K armónicos kf_0. Cada armónico tiene amplitud A_k y fase ϕ_k predefinidas (por ejemplo, muestreadas de una señal real o asignadas manualmente para el timbre objetivo). La señal de síntesis se calcula como:

$$y(n) = \sum_{k=1}^{K} A_k \operatorname{sen}(2\pi k f_0 n T_s + \phi_k),$$

donde $T_s = 1/F_s$ es el periodo de muestreo.

Como ya se discutió anteriormente, en este trabajo práctico se decidió atenuar la amplitud del seno para cada señal senoidal de frecuencia armónica, por lo que la fundamental tendrá una mayor amplitud a todas y serán atenuándose a medida que se suman los armónicos.

En este proyecto, con el fin de crear distintos instrumentos virtuales manipulando los parámetros de una envolvente para cada uno de los armónicos, partimos de una señal base (una onda seno pura de frecuencia fundamental) y le aplicamos una envolvente de tipo ADSR (Attack, Decay, Sustain, Release) para controlar su evolución temporal.

Modelo Matemático

Dada una frecuencia base f_0, la señal sintetizada $y(t)$ se construye como:

$$y(t) = \sum_{k=1}^{K} ADSR_k(t) \cdot \sin(2\pi k f_0 t) \tag{1}$$

donde N es la cantidad de armónicos y $ADSR_k(t)$ es la envolvente ADSR correspondiente al armónico k.

Componentes de la Envolvente ADSR

La función ADSR modula la amplitud de cada armónico a lo largo del tiempo. Sus componentes son:

- **Attack (A):** Tiempo en segundos que tarda la señal en alcanzar su máxima amplitud desde el silencio inicial.

- **Decay (D):** Tiempo en segundos que tarda en disminuir desde la amplitud máxima hasta el nivel de sostenimiento.
- **Sustain (S):** Nivel de amplitud sostenido (valor entre 0 y 1) mientras se mantenga la nota.
- **Release (R):** Tiempo en segundos que tarda la señal en caer a cero una vez que se libera la nota.

A continuación se presenta un gráfico representativo de una envolvente ADSR. Se puede entender como una función por partes donde cada sección es una recta.

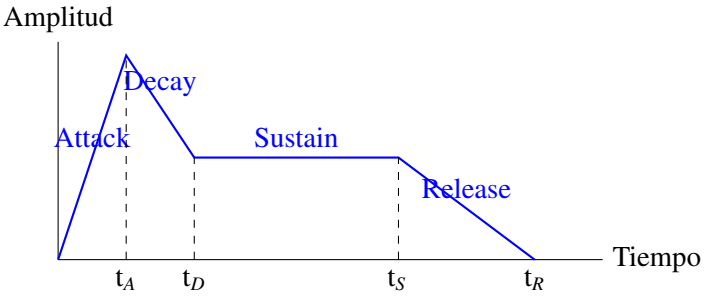

Figura 2: Forma típica de una envolvente ADSR.

Objetivo de Implementación

El objetivo de esta herramienta es permitir la creación de instrumentos personalizados, donde cada armónico tenga su propia envolvente ADSR. Esto se implementa en una interfaz interactiva en la que el usuario puede:

- Seleccionar la cantidad de armónicos (hasta 8).
- Configurar individualmente los parámetros ADSR para cada armónico.
- Visualizar las envolventes y la señal resultante.
- Escuchar el sonido generado y guardarlo como un instrumento reutilizable.

Además, se integró una funcionalidad de presets que permite cargar automáticamente configuraciones típicas de instrumentos acústicos como piano, guitarra, violín, flauta, clarinete y trompeta. Estos presets asignan valores de ADSR diferentes para cada armónico, emulando la riqueza espectral y temporal característica de cada instrumento.

4.2. Síntesis FM (Modulación de Frecuencia)

El sintetizador FM implementa dos osciladores digitales: un portador (carrier) con frecuencia f_c y un modulador con frecuencia f_m. El efecto principal es que la señal moduladora altera la fase del portador según el índice de modulación I. Siguiendo la teoría, la señal discreta resultante es:

$$y[n] = A(nT_s)\sin\big(2\pi f_c nT_s + I(nT_s)\sin(2\pi f_m nT_s)\big).$$

Para implementar esto, se actualiza la fase acumulada del portador en cada muestra sumando el término modulador. Los parámetros de $A(nT_s$ e $I(nT_s$ se selecciona según el timbre deseado.

Al generar esta señal, se observó en el dominio del tiempo que con índices bajos el output se asemeja a una onda senoidal ligeramente distorsionada, mientras que con índices altos aparecen figuras de onda más complejas. En la Fig. 3 se ilustran ejemplos de las formas de onda resultantes para distintos valores de I.

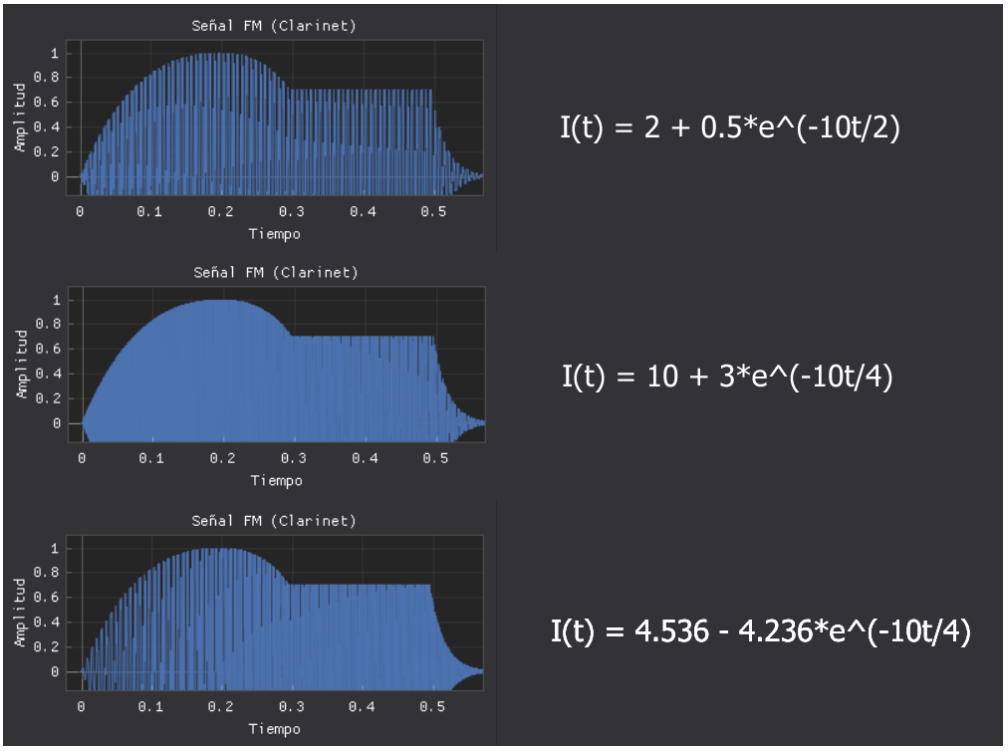

Figura 3: Síntesis FM con diferentes índices de modulación I

En el gráfico se aprecia que con $I = 2$ la señal envolvente tiene picos significativamente mayores al centro, mientras que con $I = 10$ la señal oscila rápidamente, produciendo muchas frecuencias altas. El modelo general implementado para la síntesis de estas señales se basa en dos partes: La primera refiere a la señal ADSR y su definición, y la segunda es la definición del indice.

Es importante destacar que para esta parte del trabajo práctico se decidió utilizar una función por partes de ADSR dada por exponenciales en vez de rectas para que la función final no tenga cambios abruptos y sea más suave la salida que antes. Se pudo notar en la práctica analizando el mismo sonido a través de el ADSR lineal con el exponencial y se notó una gran diferencia en la claridad de la señal de salida.

La siguiente figura da una imagen ilustrativa del programa:

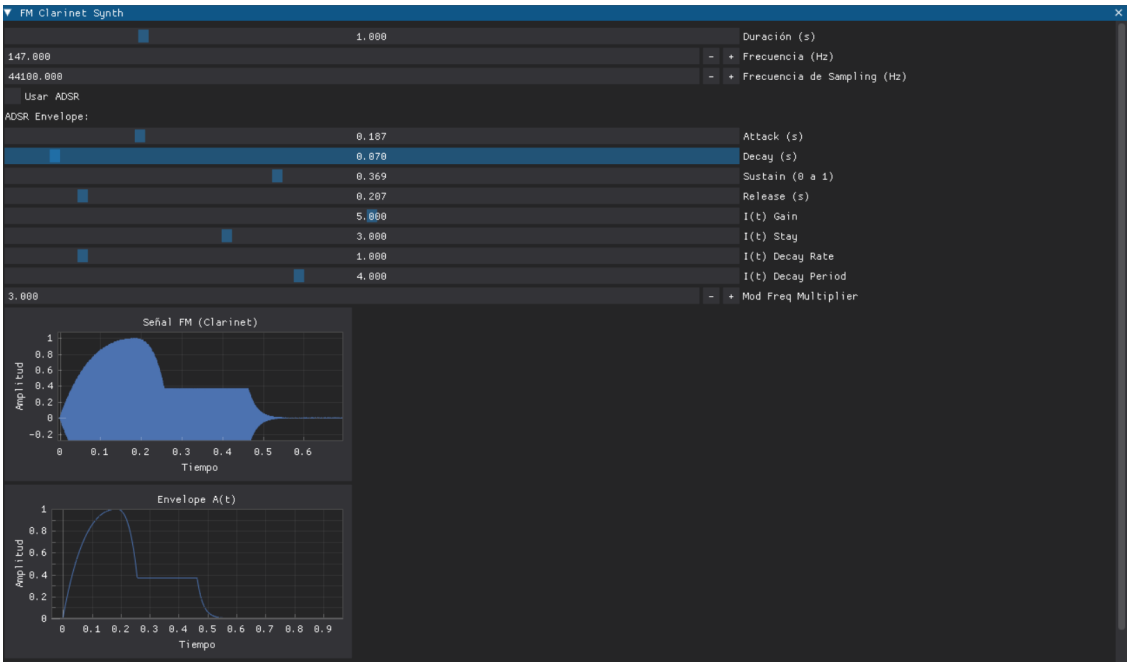

Figura 4: FM tool

El programa permite definir a la función envolvente de manera lineal o curva, chequeando la box de ADSR. Asímismo, el indice viene definido por una funcion partida: Parte lineal, parte exponencial decreciente.

Acorde a la investigación realizada, el sonido de un clarinete se corresponde más a una señal envolvente con fuerte ataque, decaimiento moderado, un sostenimiento considerable y decaimiento fuerte. También necesita de un gain menor a 2, y stay a un cuarto de este valor. Por sobre todo, la condición más importante es relación entre frecuencias de fm con fc, tal que fm sea un producto impar entero de fc, preferentemente 3.

4.3. Modelado físico

Para el módulo de modelado físico se implementó un algoritmo de cuerdas pulsadas (Karplus-Strong). La idea es simular una cuerda inicializada con ruido blanco en un buffer de retardos que corresponde a la frecuencia deseada. Sea $N = \lfloor F_s/f_0 \rfloor$ el número de muestras en una vibración completa. Primero se llena un buffer circular de tamaño N con valores aleatorios (ruido $\sim [-0{,}5, 0{,}5]$), luego se itera actualizando cada muestra como el promedio amortiguado de muestras pasadas. Un esquema simple es:

$$y[n] = (1 - \alpha)\frac{y[n - N] + y[n - N - 1]}{2},$$

donde $0 < \alpha < 1$ controla la atenuación. El siguiente pseudocódigo ilustra el proceso:

Este método genera un sonido de cuerda pulsada cuya envolvente decae exponencialmente. En el desarrollo se probaron otras variantes (filtrado lineal con diferente coeficiente) para simular distintos materiales o técnicas de ataque. También se consideró un modelo simplificado de tubo resonante usando delay con reflexión invertida. Sin embargo, para este TP se enfocó principalmente

en el algoritmo Karplus-Strong, que es sencillo y muestra claramente el concepto de modelado físico por realimentación. La frecuencia resultante se verificó ajustando N para obtener la nota deseada, y se comparó con la transformada de Fourier del sonido sintetizado para confirmar que el pico fundamental coincide con f_0.

4.4. Reproducción de muestras (Sample Playback)

El sistema implementa síntesis por muestras a partir de grabaciones digitales de notas musicales. Se parte de un conjunto de archivos WAV, donde cada muestra corresponde a una nota grabada a una frecuencia base f_0. La reproducción de otras notas se logra por **resampling**, ajustando la frecuencia de reproducción:

$$f = r \cdot f_0 \quad \Rightarrow \quad r = \frac{f}{f_0}$$

donde r es el factor de resampleo. Para notas no grabadas, se utiliza interpolación lineal o sinc para mantener la calidad espectral.

La envolvente de amplitud de cada nota se controla mediante un generador ADSR (Attack, Decay, Sustain, Release).

Se implementó un sistema de **mapeo MIDI**, que vincula notas y velocidades a diferentes muestras y capas. Se incluyen múltiples articulaciones y dinámicas, seleccionadas automáticamente según la entrada MIDI.

Además, se integraron efectos posteriores, como reverberación por convolución directa con una respuesta al impulso $h(n)$, o mediante suma retardada:

$$y(n) = \sum_{i=0}^{k} \alpha_i \, x(n - D_i)$$

Esto permite simular entornos acústicos con bajo costo computacional. El sistema está preparado para incorporar filtros y efectos de modulación, aplicables luego de la etapa de reproducción.

4.5. Efectos de audio

El sistema permite aplicar varios efectos a la señal sintetizada. Se implementaron, entre otros:

- **Eco/Reverberación:** Basado en retroalimentación de delays. Por ejemplo, un eco simple con atenuación α y retardo d se logra con:

  $$y[n] = x[n] + \alpha \, y[n - d].$$

  Se utilizaron vectores de con k valores de α y D, para simular reverberación plana. El valor de k es un parámetro seleccionable, y permite elegir la atenuación de cada señal desfasada D_k muestras. Un método alternativo que fue considerado para la implementación de estos efectos fue la convolución con un impulso, como se puede ver con las siguientes expresiones:

  $$y(n) = x(n) + \alpha_1 x(n - D_1) + \ldots + \alpha_k x(n - D_k) \qquad \text{Ec. del Reverberador Plano}$$

  Sea $h(n)$ la respuesta al impulso, la ecuación del reverberador plano también se puede expresar como:

$$h(n) = \delta(n) + \alpha_1\delta(n - D_1) + \ldots + \alpha_k\delta(n - D_k) \longrightarrow y(n) = x(n) * h(n)$$

Para una implementación más sencilla en el código, se decidió implementar el reverberador plano mediante la sumatoria de k términos de $\alpha_i x(n - D_i)$.

- **Filtros digitales:** Se incluyeron filtros IIR y FIR para ecualización o filtrado selectivo. Por ejemplo, un filtro paso-bajo de segundo orden (tipo biquad) filtra frecuencias altas no deseadas o genera efectos de wah-wah modulando su corte en el tiempo. Los coeficientes de los filtros se calcularon con fórmulas estándar (e.g. bilineal), pero se pueden ajustar mediante sliders para lograr el efefcto deseado.
- **Otros efectos:** Flanger, Phaser y chorus se implementaron mediante variaciones de delays de baja amplitud modulados por LFO. Cada efecto fue probado auditivamente y calibrado para evitar artefactos fuertes (por ejemplo, resonancias extrañas), aunque es posible ajustar el factor de profundidad α para reducir la ganancia de la señal desfasada en D muestras.

En la implementación de cada efecto se documentó experimentalmente su comportamiento (dentro de la interfaz de la *Flanger Tool* se encuentran los gráficos en tiempo real del efecto seleccionado, y su respuesta en frecuencia en función del tiempo).

4.6. Espectrograma

Para el análisis espectral se utilizó la técnica de *Short-Time Fourier Transform* (STFT). Se segmenta la señal $x[n]$ en ventanas solapadas de longitud L (por ejemplo 1024 muestras, variables mediante un slider en el proyecto) y se aplica FFT a cada segmento con una cierta ventana, que podría ser de Hann, Hamming, Blackmann o Bartlett. Matemáticamente, si $w[n]$ es la ventana, el espectrograma se calcula como:

$$X(k, m) \;=\; \sum_{n=0}^{L-1} x[n + mH]\, w[n]\, e^{-j2\pi nk/L},$$

donde H es el salto (*hop*) entre ventanas y m índice de ventana. El resultado $|X(k, m)|$ se muestra con escala de colores. Los espectrogramas obtenidos permitieron observar la evolución de los armónicos en el tiempo, transitorios de ataque y la distribución espectral de cada síntesis. Como referencia, la teoría indica que esto produce un mapa visual en frecuencia-tiempo de la señal.

4.7. Mezcla de pistas

Para juntar las pistas, se implementó otra herramienta llamada *Mixer Tool*. Con esta herrmaienta se pueden controlar los ajustes de las pistas seleccionadas, como ganancia o retardo (offset). Una vez se tienen todas las pistas elegidas para cada instrumento utilizado, se procede a sumar las señales individuales y combinarlas en una señal única, para luego poder exportar el resultado final a un archivo de sonido.

4.8. Exportación a WAV

El último bloque consiste en escribir la señal resultante de la *Mixer Tool* en un archivo WAV. Después de generar y combinar las señales de los distintos módulos (sumando tracks aditivos, FM, muestras y aplicando efectos), la señal resultante $y[n]$ se normaliza para evitar saturación digital. Se utilizó una función estándar, por ejemplo `scipy.io.wavfile.write('output.wav', fs, y)` en Python, que crea un archivo WAV y lo guarda en la carpeta del proyecto.

5. Conclusiones y Análisis

En esta sección se discuten las observaciones sobre el desempeño del sistema, las limitaciones encontradas y las posibles mejoras:

- **Precisión espectral:** En síntesis aditiva, la precisión en la generación de los armónicos más agudos depende de la resolución en amplitud y fase usadas. Con un número limitado de armónicos (K finito) siempre habrá cierta distorsión espectral en frecuencias altas.

  Dado que para cada uno de los armónicos le correspondía variables del ADSR distintos, se decidieron utilizar 8 armónicos para encontrar el balance entre una aceptable y realista nota musical resultante y la comodidad y/o sutileza de la interfaz gráfico en el proyecto de Python.

- **Consumo computacional:** La síntesis aditiva es relativamente costosa en cálculo cuando K es grande, ya que requiere sumar K senos por muestra. La síntesis FM y los modelos de cuerdas (que usan solo retardos y promedios) resultaron más eficientes para sonidos complejos con pocas operaciones por muestra. En tiempo real, tanto para aplicar varios efectos al mismo track como para aplicarle distintos instrumentos a varios tracks, se notó una extensa cantidad de tiempo para que se termine la acción. El proyecto ha llegado a tardar más 40 segundos para compilar todos estos instrumentos y aplicarlos en el archivo final .wav de la canción final con todos los tracks juntos.

- **Calidad de los Instrumentos:** Con el fin de crear distintos instrumentos para aplicarlos a ciertos tracks para luego juntarlos todos y lograr posiblemente una armonización con varios instrumentos, se llevaron a cabo los cuatro métodos de síntesis que profundizamos en el trabajo práctico. Sin embargo, se pudo notar una gran diferencia entre cómo suena el instrumento en la realidad en comparación al sintetizado por nuestro programa.

  Si bien este es un error o diferencia lógica, ya que no se trata de un instrumento real sino uno sintetizado, se busca simularlo lo más parecido posible a uno de verdad. Para lograr esto se requiere conocer todos los valores reales y certeros de las variables de cada método de síntesis para así lograr la igualdad entre la práctica y la teoría.

  En este trabajo práctico se observó que se utilizaron valores teóricos algo distintos a los reales ya que se pudo notar una gran diferencia en como suena, por ejemplo, un Do en una guitarra real ante este sintetizador.