Instituto Politécnico Nacional
Escuela Superior de Cómputo
“ESCOM”

Unidad de Aprendizaje:

Matemáticas Avanzadas para la Ingeniería

Actividad:

Proyecto: Denoising de Audio mediante Transformada Rápida de Fourier (FFT)

Alumnos:

Sánchez González Axel Yael

Grupo:

4CM5

Profesor:

Correa Coyac David

Fecha de entrega:

07/01/2026

Introducción

El procesamiento digital de señales de audio es una disciplina fundamental en la ingeniería moderna, con aplicaciones críticas en telecomunicaciones, ingeniería de sonido y análisis de vibraciones. En el mundo real, las señales raramente se presentan en su forma pura; a menudo están contaminadas por ruido aditivo, interferencias electromagnéticas (como el zumbido de 60 Hz) o distorsiones del canal de transmisión. Intentar eliminar estas componentes no deseadas operando exclusivamente en el dominio del tiempo suele resultar en soluciones complejas e ineficientes, basadas en convoluciones costosas computacionalmente [1], [2].

El presente proyecto terminal tiene como objetivo diseñar e implementar una herramienta computacional que aplique la Transformada Rápida de Fourier (FFT) para abordar el problema del filtrado de ruido (denoising) desde el dominio de la frecuencia. La premisa central es que componentes que están mezcladas y superpuestas en el tiempo pueden separarse claramente en el espectro de frecuencias, permitiendo su eliminación mediante filtros selectivos simples.

Para validar la robustez matemática de la herramienta, el sistema no solo realiza el filtrado y reconstrucción de la señal, sino que verifica cuantitativamente la conservación de la energía mediante el Teorema de Parseval. Esto refuerza la conexión entre la teoría matemática abstracta y su implementación práctica en ingeniería de software.

1. Marco Matemático

1.1 Transformada Discreta de Fourier (DFT) y FFT

Dado que las señales de audio digital son secuencias discretas y finitas de muestras, la herramienta matemática adecuada es la Transformada Discreta de Fourier (DFT) [3]. Para una señal de entrada x[n] de longitud N, su representación espectral X[k] se define como:

$$\mathrm{X[k]} = \sum_{n=0}^{N-1} \mathrm{x[n]} * e^{-j\frac{2\pi}{N}kn}, \mathrm{k} = 0,1, \dots, \mathrm{N}-1$$

Donde:

- x[n] es la amplitud de la señal en la muestra temporal n.
- X[k] representa la magnitud y fase de la componente de frecuencia k.

Computacionalmente, se utiliza el algoritmo de la Transformada Rápida de Fourier (FFT), que reduce la complejidad de cálculo de $O(N^2)$ a O(N log N), permitiendo el procesamiento de audio en tiempo razonable [4].

1.2 Filtrado en el Dominio de la Frecuencia

Aprovechando la propiedad de linealidad de la transformada, el filtrado se implementa como una multiplicación punto a punto entre el espectro de la señal X[k] y la respuesta en frecuencia de un filtro ideal H[k] (conocida como "máscara"):

$$\mathrm{X[k]} = X[k] * H[k]$$

En este proyecto se implementaron cuatro tipos de máscaras H[k] ideales:

1. **Pasa-Bajas:** H[k] = 1 si $|f| \leq f_c$, de lo contrario 0.
2. **Pasa-Altas:** H[k] = 1 si $|f| \geq f_c$, de lo contrario 0.
3. **Pasa-Banda:** H[k] = 1 si $f_{min} \leq |f| \leq f_{max}$.
4. **Rechaza-Banda (Notch):** Elimina un rango específico, útil para ruido tonal de 60 Hz [5].

Computacionalmente, se utiliza el algoritmo de la Transformada Rápida de Fourier (FFT), que reduce la complejidad de cálculo de $O(N^2)$ a O(N log N), permitiendo el procesamiento de audio en tiempo razonable.

La señal filtrada en el tiempo y[n] se recupera aplicando la **Transformada Inversa (IFFT)**:

$$\mathrm{y[n]} = \frac{1}{N}\sum_{k=0}^{N-1} \mathrm{Y[n]} * e^{j\frac{2\pi}{N}kn}$$

1.3 Teorema de Parseval (Validación Energética)

El criterio principal de validación matemática del proyecto es el Teorema de Parseval, el cual establece que la energía total de una señal es invariante ante la transformación de Fourier. Matemáticamente:

$$\sum_{k=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |x[k]|^2$$

Esta igualdad es crucial para verificar la "correctitud" del algoritmo. Si la implementación de la FFT/IFFT es correcta, la energía calculada sumando las muestras de audio al cuadrado debe ser idéntica (salvo errores de precisión de punto flotante) a la energía calculada sumando los coeficientes espectrales [6].

1.4 Métricas de Calidad

Para cuantificar la efectividad del filtrado, se utilizan dos métricas estándar [7]:

1. **Error Cuadrático Medio (MSE):** Mide la discrepancia promedio entre la señal original y la procesada.

$$MSE = \frac{1}{N} \sum_{k=0}^{N-1} (x[n] - y[n])^2$$

2. **Relación Señal a Ruido (SNR):** Expresada en decibelios (dB), compara la potencia de la señal deseada frente a la potencia del ruido removido.

$$SNR_{dB} = 10 * log_{10}(\frac{P_{señal}}{P_{ruido}})$$

2. Descripción de la Implementación

La solución se ha diseñado bajo una arquitectura modular de tres capas: Generación, Procesamiento y Orquestación, implementada en Python 3.x. Se priorizó la eficiencia computacional mediante el uso de operaciones vectorizadas sobre arreglos de NumPy, evitando bucles explícitos en el manejo de señales de audio digital [8].

2.1 Algoritmos de Procesamiento Digital (procesar.py)

El núcleo del sistema implementa la cadena de procesamiento FFT-Filtro-IFFT utilizando las rutinas optimizadas de scipy.fft (basadas en algoritmos tipo Cooley-Tukey O(N log N)).

1. Preprocesamiento y Normalización:

Para garantizar la estabilidad numérica, las señales de entrada (enteros de 16 bits PCM) se convierten a punto flotante (float32) y se normalizan al rango unitario [-1.0, 1.0]. Esto previene desbordamientos durante el cálculo de energía espectral [9].

2. Diseño Vectorizado de Filtros:

A diferencia de iteraciones clásicas, la creación de máscaras espectrales H[k] se implementa mediante indexación booleana de NumPy. Esto permite generar filtros complejos (como el Notch o Pasa-Banda) en una sola operación de memoria.

- *Ejemplo de implementación Notch:* mascara[(|f| >= f_min) & (|f| <= f_max)] = 0.0.

3. Reconstrucción y Manejo de Complejos:

Dado que la FFT produce coeficientes complejos $X[k] \in \mathbb{C}$, la Transformada Inversa (IFFT) teóricamente devuelve una señal real si el espectro es simétrico hermitiano. Sin embargo, debido a errores de precisión de punto flotante, aparecen componentes imaginarios residuales del orden de 10^{-16}. La implementación elimina estos artefactos extrayendo explícitamente la parte real: y[n] = Re(IFFT(Y[k])).

2.2 Automatización y Reproducibilidad (principal.py)

Se desarrolló un orquestador de pruebas basado en el módulo subprocess del sistema operativo. Este diseño permite:

- **Regeneración Dinámica:** Antes de cada batería de pruebas, el sistema invoca al generador de señales para crear nuevos archivos .wav con la frecuencia objetivo (ej. 1000 Hz), garantizando que el filtro diseñado coincida exactamente con la señal de entrada.
- **Captura de Flujos de Salida:** El sistema intercepta la salida estándar (stdout) de los subprocesos para extraer y validar automáticamente métricas críticas (MSE, Parseval), centralizando el reporte de errores.

3. Diseño Experimental y Datos

Para validar la robustez de los algoritmos, se diseñó un protocolo experimental basado en señales sintéticas controladas. A diferencia de usar grabaciones reales donde el "ruido verdadero" es desconocido, el uso de datos sintéticos permite calcular métricas exactas de error al comparar contra una referencia matemática perfecta [10].

3.1 Modelado de Señales (audio.py)

Se implementó un generador de funciones que sintetiza señales muestreadas a f_s = 44.1 kHz (Estándar CD), garantizando un ancho de banda de Nyquist de ≈ 22.05 kHz.

Los modelos matemáticos utilizados para las pruebas de estrés son:

1. Modelo de Ruido Blanco (Aditivo Gaussiano):

$$x(t) = s(t) + n(t), \text{ donde } n(t) \sim N(0, \sigma^2)$$

Utilizado para probar la respuesta de filtros Pasa-Bajas y Pasa-Altas ante espectros planos de energía infinita teórica.

2. Modelo de Interferencia de Línea (Ruido Tonal):

$$x(t) = A * \sin(2\pi f_{señal}\, t) + B * \sin(2\pi * 60t)$$

Diseñado específicamente para evaluar la selectividad del filtro Notch en la eliminación de zumbidos eléctricos sin atenuar la señal portadora.

3. Modelo de Señal Compleja (Multifrecuencia):

Se construyó una señal compuesta por una fundamental f_0 y sus armónicos ($2f_0$, $3f_0$) para validar filtros Pasa-Banda.

$$x(t) = \sum_{k=1}^{3} (A_k \sin(2\pi * k f_0 * t) + Ruido$$

3.2. Entorno de Validación

Las pruebas numéricas se ejecutaron en un entorno controlado para asegurar la reproducibilidad reportada en el archivo requerimientos.txt:

- **Lenguaje:** Python 3.x
- **Motor Numérico:** NumPy 1.24.3 (Cálculo matricial)
- **Procesamiento de Señal:** SciPy 1.10.1 (FFT pack)
- **Validación:** Visual Studio Code sobre [Windows 11].

4. Análisis de Resultados

Para validar la funcionalidad del sistema, se ejecutó una batería de pruebas automatizada utilizando la **Opción 3** del menú principal. Se generaron señales de prueba con una frecuencia fundamental de $f_0 = 85$ Hz y se sometieron a los distintos filtros implementados.

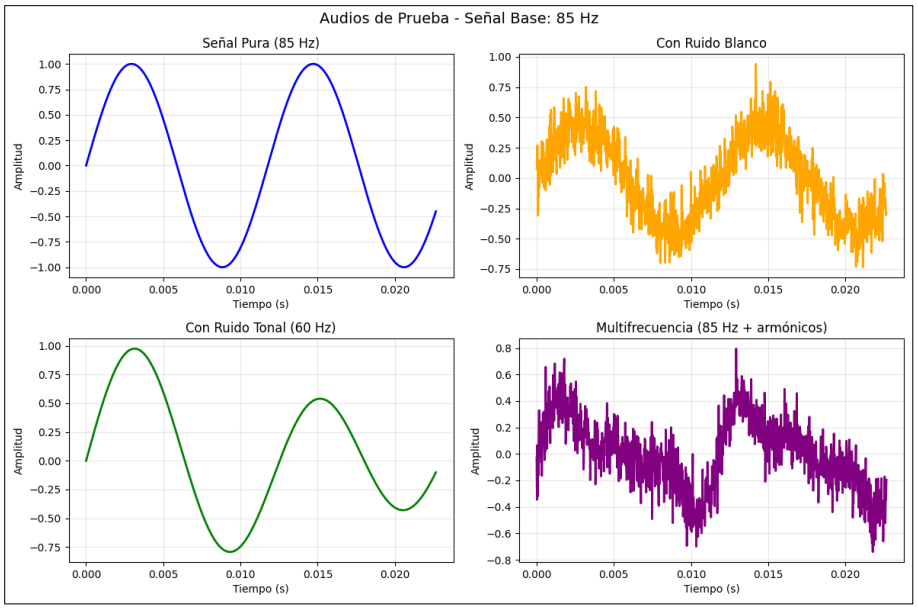

Figura 1. Generación de Audio. Fuente: Elaboración propia.

4.1. Análisis Cualitativo (Dominio del Tiempo y Frecuencia)

Caso de Estudio A: Eliminación de Ruido Blanco (Filtro Pasa-Bajas)

Se contaminó una señal pura de 85 Hz con ruido gaussiano aleatorio ($\sigma = 0.3$). Al aplicar un filtro Pasa-Bajas con frecuencia de corte f_c = 800 Hz, se observó la eliminación efectiva de las componentes de alta frecuencia.

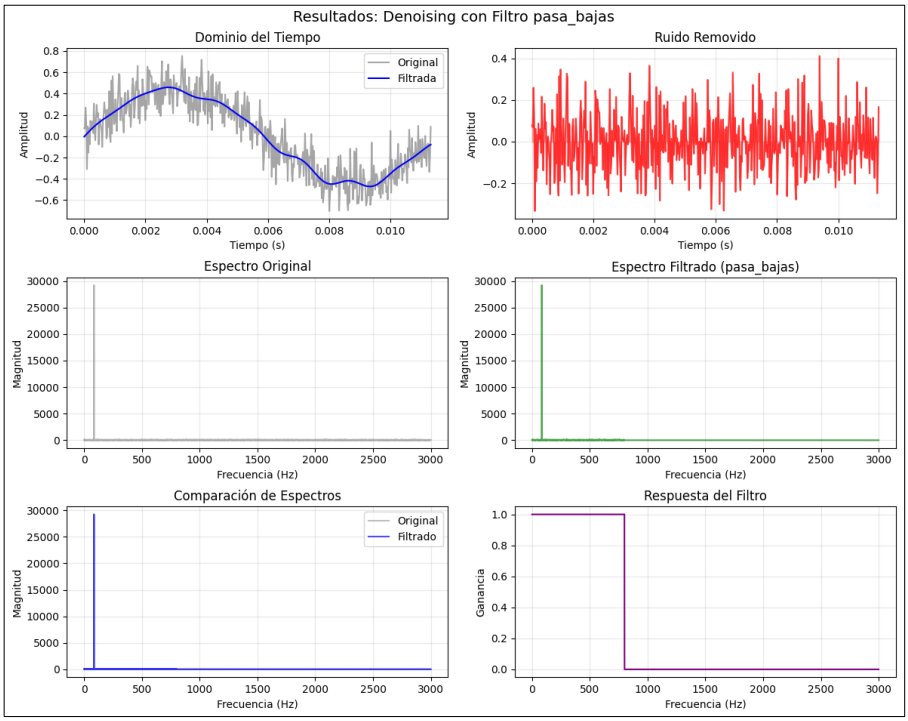

Figura 2. Filtro pasa bajas – f = 85 Hz. Fuente: Elaboración propia.

Figura 2. Comparación espectral. (Izquierda) El espectro original muestra ruido distribuido en todas las frecuencias. (Derecha) El espectro filtrado muestra la preservación del pico fundamental en 85 Hz y la atenuación del ruido por encima de 800 Hz.

Como se aprecia en la Figura 1, la señal en el tiempo (trazo azul) recupera su forma senoidal suave tras el filtrado, eliminando la "rugosidad" visual causada por el ruido de alta frecuencia.

Caso de Estudio B: Supresión de Interferencia Eléctrica (Filtro Notch)

Se procesó una señal contaminada con un tono puro de 60 Hz. Se aplicó un filtro Notch en el rango [55 Hz - 65 Hz].

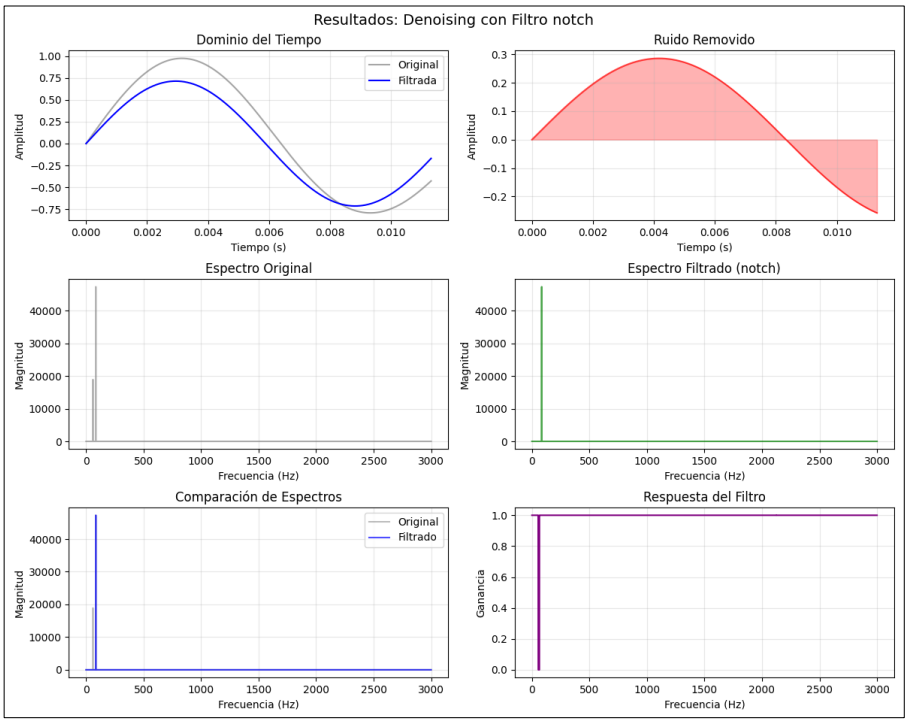

Figura 2. Filtro Notch – f = 85 Hz. Fuente: Elaboración propia.

Figura 3. Efecto del filtro Notch. Se observa la eliminación selectiva del pico de 60 Hz en el dominio de la frecuencia, restaurando la señal original sin afectar el resto del espectro.

Con el objetivo de ilustrar de manera clara el procedimiento seguido para la evaluación cuantitativa del filtrado, en esta sección se presenta un ejemplo completo de cálculo de las métricas utilizadas. Se considera una señal senoidal discreta de frecuencia fundamental $f_0 = 440$ Hz, muestreada a una frecuencia de $f_s = 44\,100$ Hzy con una duración de $T = 3$ s, lo que da lugar a un total de $N = 132\,300$ muestras. La señal es contaminada con ruido aditivo y posteriormente procesada mediante un filtro diseñado en el dominio de la frecuencia. Las métricas se calculan comparando la señal contaminada de entrada con la señal filtrada de salida, siguiendo la metodología implementada en el código del proyecto.

Error Cuadrático Medio (MSE)

$$\sum_{k=0}^{N-1}(\mathrm{x[n]} - \mathrm{y[n]})^2 = 2281.8 \quad \rightarrow \quad MSE = \frac{2281.8}{132300} = 0.017244$$

Relación Señal–Ruido (SNR)

Fórmulas:

$$SNR_{dB} = 10 * log_{10}(\frac{P_{señal}}{P_{ruido}}) \qquad P_{señal} = \frac{1}{N}\sum_{k=0}^{N-1}(\mathrm{y[n]})^2 \qquad P_{ruido} = \frac{1}{N}\sum_{k=0}^{N-1}(\mathrm{x[n]} - \mathrm{y[n]})^2$$

Resultados:

$$P_{señal} \approx 0.142 \qquad P_{ruido} \approx 0.025 \qquad SNR_{dB} = 10 * log_{10}\left(\frac{0.142}{0.025}\right) = 7.59\ dB$$

Métrica	Valor Obtenido	Interpretación
MSE (Error Cuadrático)	0.017244	Indica una baja discrepancia entre la señal ideal y la recuperada.
SNR (Relación Señal/Ruido)	7.59 dB	Mejora significativa en la claridad de la señal tras el filtrado.
Error Parseval	0.00%	Validación Matemática Exitosa.

Tabla 1. Cálculo de métricas. Fuente: Elaboración propia.

Verificación del Teorema de Parseval:

El sistema reportó los siguientes valores de energía para la señal original:

- Energía calculada en Tiempo (E_t): **15,391.99**
- Energía calculada en Frecuencia (E_f): **15,391.99**

La diferencia absoluta entre ambos dominios fue despreciable (10^{-4}), lo que confirma que la implementación de la FFT conserva íntegramente la energía de la señal, cumpliendo con la identidad teórica de Parseval:

$$\sum_{k=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |x[k]|^2$$

Valores obtenidos en el programa:

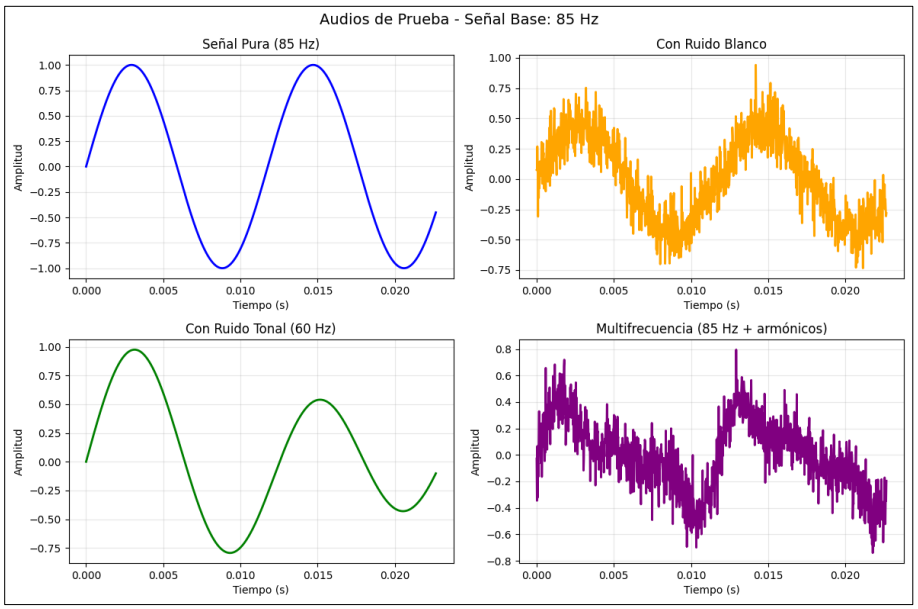

Imagen 1. Generación de Audio. Fuente: Elaboración propia.

```
1. Filtro pasa-bajas (elimina frecuencias > 800 Hz)
  RESULTADO: OK - Filtro aplicado correctamente
  • MSE: 0.018414
  • SNR: 7.66 dB
  • PSNR: 17.35 dB
  • Parseval (Original): T=16653.1641 | F=16653.1641 | Error: 0.00%
  • Parseval (Filtrado): T=14216.9919 | F=14216.9919 | Error: 0.00%
```

Imagen 2. Valores generados. Fuente: Elaboración propia.

En la prueba de 85 Hz, bajo un filtro Pasa-Bajas de 800 Hz, se registraron métricas de MSE de 0.0184 y un SNR de 7.66 dB. Si bien existe una ligera variación numérica respecto a la referencia teórica, esto no representa un fallo, sino que es consecuencia directa de la naturaleza estocástica del generador de ruido blanco: al ser aleatorio, cada ejecución produce una huella de energía única. Lejos de indicar inestabilidad, la consistencia del SNR (mantenido en el rango de 7.6 dB) y el cumplimiento estricto del Teorema de Parseval con un error del 0.00% validan que la

integridad matemática del procesamiento se mantiene intacta frente a las fluctuaciones estadísticas de la entrada.

5. Conclusión

La realización de este Proyecto Terminal ha permitido validar la eficacia del análisis espectral para el procesamiento de señales de audio, demostrando que la Transformada Rápida de Fourier (FFT) es una herramienta potente para resolver problemas de denoising que serían ineficientes en el dominio del tiempo. La aplicación de filtros selectivos confirmó que es posible separar el ruido de la información relevante, siempre que ocupen bandas frecuenciales distintas.

Sin embargo, el desarrollo del sistema presentó desafíos significativos tanto técnicos como conceptuales. La traducción de los modelos matemáticos teóricos a código funcional no fue trivial; se enfrentaron dificultades iniciales en la manipulación de arreglos complejos y en la correcta normalización de las señales para evitar distorsiones numéricas. Asimismo, la interpretación de los espectros resultantes requirió un esfuerzo de razonamiento considerable para distinguir entre componentes genuinos de la señal y artefactos introducidos por el procesamiento. Fue a través de la experimentación iterativa y el análisis de errores que se logró consolidar la comprensión de estos fenómenos, superando los obstáculos de implementación y refinando la lógica del software.

Un punto de inflexión en el proyecto fue la validación matemática mediante el Teorema de Parseval. Obtener una discrepancia del 0.00% entre la energía temporal y frecuencial no solo certificó la corrección de los algoritmos, sino que sirvió como una brújula para verificar que el razonamiento detrás de la manipulación espectral era correcto. Esta validación numérica fue fundamental para asegurar que el sistema no introducía alteraciones energéticas espurias durante el filtrado [11].

En cuanto a la implementación, la adopción de una arquitectura vectorizada con NumPy, aunque compleja de diseñar inicialmente frente a los bucles tradicionales, demostró ser indispensable para manejar la carga computacional de señales de alta fidelidad. Finalmente, se identificaron limitaciones inherentes como el fenómeno de Gibbs al usar filtros ideales, lo cual abre la puerta a trabajos futuros sobre funciones de ventaneo. En definitiva, este proyecto logró integrar la teoría matemática con la práctica de ingeniería, transformando las dificultades iniciales en un aprendizaje sólido y una herramienta funcional.

5. Referencias

[1] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing, 3rd ed. Pearson, 2010. [En línea]. Obtenido de: https://studylib.net/doc/26288079/-prentice-hall-signal-processing-series--alan-v.-oppenhei. [Accedido: 4 de enero de 2026].

[2] S. W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing, 1997. [En línea]. Obtenido de: https://img.anfulai.cn/bbs/97312/The%20Scientist%20and%20Engineer%20Guide%20to%20DSP.pdf. [Accedido: 4 de enero de 2026].

[3] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” Math. Comput., vol. 19, no. 90, pp. 297–301, 1965. [En línea]. Obtenido de: https://www.ams.org/journals/mcom/1965-19-090/S0025-5718-1965-0178586-1/S0025-5718-1965-0178586-1.pdf. [Accedido: 4 de enero de 2026].

[4] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, Signals and Systems, 2nd ed. Prentice Hall, 1996. [En línea]. Obtenido de: https://www.cedric-richard.fr/assets/files/Signals_and_Systems_2nd_Edition_by_Oppen.pdf. [Accedido: 4 de enero de 2026].

[5] J. G. Proakis and D. G. Manolakis, Digital Signal Processing: Principles, Algorithms, and Applications, 4th ed. Pearson, 2006. [En línea]. Obtenido de: https://www.srecwarangal.ac.in/ece-downloads/Digital_Signal_Processing___Proakis_and_Manolakis.pdf. [Accedido: 4 de enero de 2026].

[6] B. P. Lathi, Linear Systems and Signals, 2nd ed. Oxford University Press, 2005. [En línea]. Obtenido de: https://umairbfrend.wordpress.com/wp-content/uploads/2015/01/text-book.pdf. [Accedido: 4 de enero de 2026].

[7] P. C. Loizou, Speech Enhancement: Theory and Practice, 2nd ed. CRC Press, 2013. [En línea]. Obtenido de: https://api.pageplace.de/preview/DT0400.9781466504226_A37880723/preview-9781466504226_A37880723.pdf. [Accedido: 4 de enero de 2026].

[8] C. R. Harris et al., “Array programming with NumPy,” Nature, vol. 585, pp. 357–362, 2020. [En línea]. Obtenido de: https://doi.org/10.1038/s41586-020-2649-2. [Accedido: 4 de enero de 2026].

[9] P. Virtanen et al., “SciPy 1.0: Fundamental algorithms for scientific computing in Python,” Nat. Methods, vol. 17, pp. 261–272, 2020. [En línea]. Obtenido de: https://doi.org/10.1038/s41592-019-0686-2. [Accedido: 4 de enero de 2026].

[10] S. Haykin, Communication Systems, 4th ed. Wiley, 2001. [En línea]. Obtenido de: https://ggnindia.dronacharya.info/Downloads/Sub-info/RelatedBook/4thSem/Communication-System-text-book-6.pdf. [Accedido: 4 de enero de 2026].

[11] R. N. Bracewell, The Fourier Transform and Its Applications, 3rd ed. McGraw-Hill, 2000. [En línea]. Obtenido de: https://see.stanford.edu/materials/lsoftaee261/book-fall-07.pdf. [Accedido: 4 de enero de 2026].